UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
Programa de Pós-Graduação em Matemática
Mestrado Profissional - PROFMAT/CCT/UFCG

Francisco das Chagas dos Santos Moura

Pensamento Algorítmico e Programação em Python no Ensino de Recorrências Lineares de 1ª e 2ª ordem

Campina Grande - PB
Agosto/2026

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
Programa de Pós-Graduação em Matemática
Mestrado Profissional - PROFMAT/CCT/UFCG

Francisco das Chagas dos Santos Moura

Pensamento Algorítmico e Programação em Python no Ensino de Recorrências Lineares de 1ª e 2ª ordem

Trabalho de Conclusão de Curso apresentado ao Corpo Docente do Programa de Pós-Graduação em Matemática - CCT - UFCG, na modalidade Mestrado Profissional, como requisito parcial para obtenção do título de Mestre.

Orientador: Prof. Dr. Marcelo Carvalho Ferreira

Campina Grande - PB
Agosto/2026

Francisco das Chagas dos Santos Moura

Pensamento Algorítmico e Programação em Python no Ensino de Recorrências Lineares de 1ª e 2ª ordem

Trabalho de Conclusão de Curso apresentado ao Corpo Docente do Programa de Pós-Graduação em Matemática - CCT - UFCG, na modalidade Mestrado Profissional, como requisito parcial para obtenção do título de Mestre.

Trabalho aprovado. Campina Grande - PB, 28 de agosto de 2026:

gov.br Documento assinado digitalmente
MARCELO CARVALHO FERREIRA
Data: 23/09/2026 21:42:59-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Marcelo Carvalho Ferreira
Orientador - UFCG

gov.br Documento assinado digitalmente
JONATHAS JERONIMO BARBOSA
Data: 24/09/2026 11:53:03-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Jonathas Jerônimo Barbosa
Examinador Interno - IFPB

gov.br Documento assinado digitalmente
ISRAEL BURITI GALVAO
Data: 24/09/2026 20:35:04-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Israel Buriti Galvão
Examinador Externo - UEPB

Campina Grande - PB
Agosto/2026

Universidade Federal de Campina Grande - UFCG
Sistema de Bibliotecas - SISTEMOTECA
Catalogação de Publicação na Fonte. UFCG - Biblioteca Central

M929p Moura, Francisco das Chagas dos Santos.

Pensamento algorítmico e programação em Python no ensino de recorrências lineares de 1ª e 2ª ordem / Francisco das Chagas dos Santos Moura. – 2026.

122 f. : il. color.

Dissertação (mestrado em Matemática) – Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, 2026.

"Orientação: Prof. Dr. Marcelo Carvalho Ferreira".

Referências.

1. Matemática. 2. Algoritmos. 3. Sequências. 4. Recorrências Lineares. 5. Python. 6. Pensamento Computacional. I. Ferreira, Marcelo Carvalho. II. Título.

UFCG/BC CDU 51(043.3)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA ITAPUANA SOARES DIAS GONÇALVES CRB-15/93

A Deus, princípio e fim.
Aos que, todos os dias, plantam, no jardim da esperança, as sementes do conhecimento e as regam com as lágrimas da fé.

Agradecimentos

A Deus, pelo ontem, hoje e amanhã.

Ao Programa de Mestrado Profissional em Matemática em Rede Nacional (PROFMAT), em particular o Campus de Campina Grande, pela oportunidade de formação acadêmica e profissional, que me proporcionou uma nova forma de compreender a Matemática e refletir sobre como posso contribuir com os estudantes da Educação Básica.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Ao meu orientador, o Prof. Dr. Marcelo Carvalho Ferreira, pelos ensinamentos compartilhados ao longo do curso e na orientação deste trabalho.

Aos professores que compartilharam seus conhecimentos durante o curso. Prof. Leomarques, Prof. Arimateia, Prof. Luiz Antônio, Prof. Jaime, Prof. Daniel, Prof. Renan. Agradeço os ensinamentos.

Aos meus colegas do mestrado, que concluíram ou que por algum motivo, decidiram trilhar outros caminhos. Que a amizade e companheirismo nesses mais de dois anos possa se estender por longos tempos.

A todos que compõem a UAMat, com especiais lembranças para Aninha e Isabela, que durante todo o curso foram mais que prestativas, acolhendo-nos todas as sextas, desde manhã cedo. Com carisma e paciência, cuidando de tudo com muito zelo se tornam a alegria que sustenta e motiva os estudantes.

A minha família.

A todos que não foram citados diretamente neste trabalho, mas que sabem que tenho admiração e carinho, agradeço todo o apoio nesta caminhada.

"Com sabedoria se constrói a casa,
e com entendimento ela se firma;
pelo conhecimento os seus cômodos se enchem
de tudo o que é precioso e agradável."
Provérbios 24,3–4

Resumo

Este trabalho investiga o potencial da programação em Python como ferramenta de apoio ao ensino de sequências e recorrências lineares no Ensino Médio. A pesquisa fundamenta-se na integração entre Matemática e Pensamento Computacional, em consonância com as orientações da Base Nacional Comum Curricular e da Política Nacional de Educação Digital. O objetivo consiste em desenvolver um produto educacional na forma de uma sequência didática que articule conceitos matemáticos, algoritmos e programação, favorecendo a compreensão de recorrências lineares de primeira e segunda ordem. Inicialmente, são apresentados aspectos históricos e conceituais das sequências, das recorrências e dos algoritmos, seguidos de uma introdução à linguagem Python e de exemplos de implementação computacional. Como resultado, é proposta uma sequência didática destinada ao Ensino Médio, com atividades que integram resolução de problemas e programação. Espera-se que o material contribua para o desenvolvimento do raciocínio lógico, do pensamento algorítmico e da aprendizagem significativa, oferecendo aos professores uma alternativa metodológica para o ensino desses conteúdos.

Palavras-chave: Sequências. Recorrências lineares. Algoritmos. Python. Pensamento Computacional.

Abstract

This study investigates the potential of Python programming as a tool to support the teaching of sequences and linear recurrences in high school. The research is based on the integration of mathematics and computational thinking, in line with the guidelines of the Base Nacional Comum Curricular and the Política Nacional de Educação Digital. The objective is to develop an educational product in the form of a teaching sequence that integrates mathematical concepts, algorithms, and programming, thereby promoting the understanding of first- and second-order linear recurrences. Initially, historical and conceptual aspects of sequences, recurrences, and algorithms are presented, followed by an introduction to the Python language and examples of computational implementation. As a result, a teaching sequence designed for high school is proposed, featuring activities that integrate problem-solving and programming. It is hoped that this material will contribute to the development of logical reasoning, algorithmic thinking, and meaningful learning, offering teachers a methodological alternative for teaching this content.

Keywords: Sequences. Linear Recurrences. Algorithms. Python. Computational Thinking.

Lista de ilustrações

Figura 1 – Representação gráfica de duas progressões aritméticas com comportamentos crescente e decrescente. 35
Figura 2 – Representação gráfica da progressão geométrica $a_n = 500 \cdot 2^n$. . . . 43
Figura 3 – Primeiros casos do Problema da Pizza de Steiner e o número máximo de regiões obtidas com cortes retos. 52
Figura 4 – Édouard Lucas (1842–1891). 55
Figura 5 – Ilustração do problema da Torre de Hanói com 7 discos. 56
Figura 6 – Sistema massa-mola discreto com força externa constante e recorrência de segunda ordem. 64
Figura 7 – Tábua Plimpton 322. 69
Figura 8 – Papiro de Rhind - Problema 79 71
Figura 9 – Problema encontrado no Papiro de Rhind (1650 a.C.) 73
Figura 10 – Euclides de Alexandria - séc III a.C 73
Figura 11 – Al-Khwarizmi (780 d.C.–850 d.C.) 75
Figura 12 – Fibonacci (1170–1240). 77
Figura 13 – Ada Lovelace (1815–1852). 78
Figura 14 – Tabela de multiplicação - MS3866 85
Figura 15 – Logomarca do Python. 89
Figura 16 – Guido van Rossum, criador da linguagem Python. 90
Figura 17 – Índice TIOBE de Comunidades de Programação. 91
Figura 18 – Interface do Python com o site como plano de funto. 95

Lista de tabelas

Tabela 1 – Competências específicas e habilidades relacionadas na BNCC. . . . 25
Tabela 2 – Desenvolvimento da sequência de Fibonacci. 86
Tabela 3 – Comparação entre abordagens recursivas e iterativas 101

Lista de quadros

1 Principais elementos da linguagem Python utilizados neste trabalho. . . 94
2 Características gerais da iteração e da recursão. 99
3 Algumas formas usuais de soluções particulares para recorrências lineares não homogêneas 122

Sumário

1 INTRODUÇÃO 13
1.1 Objetivos 16
1.1.1 Objetivo geral 16
1.1.2 Objetivos específicos 16
1.2 Organização 16

2 O ENSINO DE MATEMÁTICA NO NÍVEL MÉDIO E OS DESAFIOS CONTEMPORÂNEOS 18
2.1 Do ensino mecânico à compreensão significativa 18
2.2 Matemática e Tecnologias Digitais: o papel do pensamento computacional na promoção da aprendizagem significativa 21

3 SEQUÊNCIAS E RECORRÊNCIAS LINEARES: FUNDAMENTOS E APLICAÇÕES ALGORÍTMICAS 31
3.1 Sequências Numéricas 32
3.1.1 Tipos especiais de sequências 32
3.2 Progressões Aritméticas e Geométricas 33
3.2.1 Progressão Aritmética 34
3.2.2 Progressão Geométrica 36
3.3 Recorrências Lineares 44
3.3.1 Recorrências de 1ª ordem 46
3.3.2 Recorrências Lineares de 2ª ordem 57

4 ALGORITMOS 67
4.1 Origem e Evolução dos Algoritmos 68
4.2 Conceitos e Formas de representação 80
4.3 Algoritmos Matemáticos Clássicos 83

5 PYTHON 88
5.1 A linguagem Python 89
5.2 Estrutura Básica da Linguagem Python 93
5.3 Iteração e Recursão em Python. 98
5.4 Modelagem Computacional de Recorrências em Python 102
5.4.1 O método babilônico para aproximação de raízes quadradas 105
5.4.2 A Torre de Hanói em Python 106

5.4.3 O problema da pizza de Steiner em Python 109
5.4.4 O problema do queijo de Steiner em Python 110
5.4.5 A sequência de Fibonacci em Python 111
5.4.6 Crescimento populacional em Python 112
5.4.7 Aplicação financeira em Python 113

6 CONSIDERAÇÕES FINAIS 115

Bibliografia 118

APÊNDICES 121

APÊNDICE A – SOLUÇÕES PARTICULARES PARA RECORRÊNCIAS LINEARES NÃO HOMOGÊNEAS 122

1 Introdução

A presença das tecnologias digitais na sociedade tem modificado as formas de acesso à informação, de comunicação e de produção do conhecimento. No ambiente escolar, essas transformações têm estimulado a incorporação de computadores, dispositivos móveis, softwares educacionais, plataformas digitais e linguagens de programação no ensino e aprendizagem de diversos conteúdos, buscando a interdisciplinaridade e a contextualização daquilo que se apresenta no currículo formal. Entretanto, a introdução desses recursos não deve ser compreendida como garantia automática de inovação pedagógica.

Um recurso tecnológico pode ser utilizado para reproduzir práticas tradicionais ou para criar novas possibilidades de aprendizagem. Quando o computador serve apenas para apresentar conteúdos ou executar cálculos que seriam realizados manualmente, a dinâmica da aula pode permanecer inalterada. Por outro lado, quando possibilita investigar, simular, construir e revisar soluções, a tecnologia pode contribuir para uma participação mais ativa dos estudantes. A execução de um algoritmo em ambiente computacional não apenas evidencia seu caráter operacional, mas também permite analisar aspectos relacionados à organização lógica dos procedimentos, ao número de operações realizadas, à eficiência dos métodos e às diferentes formas de repetição, como a iteração e a recursão. Em Matemática, essas características assumem especial relevância, pois muitos conceitos e procedimentos podem ser descritos por meio de etapas sucessivas, relações entre valores e regras que determinam a construção de novos resultados.

No âmbito deste trabalho, a utilização e o aproveitamento do ensino e aprendizagem por meio da manipulação dos ambientes tecnológicos podem ser discutidos à luz da Lei nº 14.533, de 11 de janeiro de 2023, que instituiu a Política Nacional de Educação Digital e estabeleceu, entre seus eixos estruturantes, a Educação Digital Escolar. De acordo com a legislação, esse eixo tem como objetivo garantir a inserção da educação digital nos ambientes escolares, mediante o estímulo ao letramento digital, à aprendizagem de computação, à programação, à robótica e a outras competências digitais (Brasil 2023)

No ensino de Matemática, essas orientações podem ser concretizadas por meio de atividades que articulem conceitos matemáticos, algoritmos e recursos computacionais. Entre os conteúdos que apresentam grande potencial para essa integração encontram-se os padrões, as sequências e as recorrências. O reconhecimento de padrões constitui uma atividade fundamental do pensamento matemático, pois permite identificar regularidades, estabelecer relações, formular conjecturas e construir generalizações.

As sequências constituem uma forma de organizar esses padrões. Por meio delas,

é possível analisar fenômenos que evoluem de maneira ordenada, como processos de crescimento, decrescimento, acumulação ou oscilação. Uma sequência pode ser descrita por uma expressão explícita, que permite determinar diretamente cada termo, ou por uma relação recorrente, na qual o termo seguinte depende de um ou mais termos anteriores, estabelecendo-se um processo sucessivo. Essa característica aproxima as recorrências das estruturas de repetição utilizadas nas linguagens de programação, e ambas mobilizam capacidades de abstração, generalização, análise e resolução de problemas, articulando e ampliando formas de se trabalhar os conteúdos curriculares.

Considerando, por exemplo, uma sequência, a relação de recorrência de uma Progressão Aritmética, definida por

$$a_{n+1} = a_n + r. \tag{1.1}$$

Essa relação descreve um processo no qual cada novo termo é obtido adicionando-se uma constante ao termo anterior. Em linguagem computacional, o mesmo procedimento pode ser representado por uma instrução de atualização executada repetidamente:

```
a = valor_inicial

for n in range(quantidade):
    print(a)
    a = a + r
```

Nesse exemplo, a variável `a` representa o termo atual da sequência, enquanto a instrução `a = a + r` corresponde diretamente à relação matemática que determina o termo seguinte. O laço `for`, por sua vez, representa a repetição do processo. Essa correspondência permite que o estudante observe a recorrência não apenas como uma fórmula, mas como um procedimento em funcionamento.

Apresentar e estudar as recorrências lineares de primeiro e segundo graus no ensino médio, bem como as sequências e progressões, permite compreender processos que se desenvolvem de maneira sucessiva, nos quais cada novo estado depende de informações anteriores. Essa estrutura está presente em diversas situações próximas ao cotidiano dos estudantes, como o crescimento de uma população, a evolução de uma dívida ou de um investimento, a propagação de informações, a organização de parcelas, a formação de padrões geométricos e a previsão de valores ao longo do tempo. Ao reconhecer essas relações, o estudante pode perceber que as recorrências não constituem apenas um conteúdo abstrato, mas uma forma de representar fenômenos que se modificam por etapas.

Essa aproximação com situações concretas favorece a atribuição de significado aos conceitos matemáticos. Desse modo, o estudo das recorrências contribui para o desenvolvimento da capacidade de observar regularidades, formular hipóteses, construir modelos e avaliar resultados, se relacionando com os algoritmos de modo natural, conhecendo o termo inicial de uma recorrência, formulando uma estrutura básica, elaborando um pseudocódigo, e posteriormente, para uma linguagem de programação.

A implementação computacional possibilita acompanhar a produção dos termos, modificar parâmetros, comparar comportamentos e analisar diferentes estratégias de solução. Nesse processo, a programação não substitui a compreensão matemática, mas amplia suas possibilidades de representação e investigação.

Dessa forma, integrar recorrências, algoritmos e programação contribui para uma aprendizagem em que o estudante deixa de atuar apenas como receptor de fórmulas e passa a compreender, construir e testar procedimentos. Essa articulação aproxima a Matemática da cultura digital, favorece o desenvolvimento do pensamento lógico e algorítmico e amplia as condições para que os estudantes interpretem fenômenos presentes em seu contexto. Portanto, o estudo das recorrências, associado à programação em Python, pode constituir um caminho fértil para tornar o ensino mais dinâmico, significativo e conectado às demandas contemporâneas da formação matemática, tendo em vista que os códigos não são apresentados somente como produtos finalizados, mas como representações de procedimentos que precisam ser compreendidos. A programação, por sua vez, é utilizada para acompanhar a geração dos termos, comparar estratégias, analisar erros e interpretar comportamentos, preservando a centralidade do conhecimento matemático.

Os algoritmos vêm como uma forma de organizar e tornar explícitas relações existentes no cotidiano do estudante. Ao estruturar uma recorrência em etapas bem definidas, evidencia-se a lógica de produção dos termos, a importância dos valores iniciais e a necessidade de estabelecer critérios de repetição ou de parada. A passagem da relação matemática para o algoritmo contribui para que o estudante reconheça a recorrência como um procedimento e não apenas como uma fórmula.

Os códigos em Python surgem, por fim, como uma possibilidade de executar e observar esse processo. Sua função principal, nesse contexto, é permitir a visualização da evolução da sequência, a comparação entre diferentes condições iniciais e a análise do comportamento produzido pela relação recorrente. Dessa forma, a programação assume um papel complementar, enquanto o foco permanece na compreensão das recorrências, de suas propriedades e de suas aplicações matemáticas.

1.1 Objetivos

1.1.1 Objetivo geral

Discutir as potencialidades do uso de algoritmos e da programação em Python no ensino de sequências e recorrências lineares no Ensino Médio, por meio da elaboração de uma sequência didática que articule conhecimentos matemáticos, algoritmo e modelagem computacional.

1.1.2 Objetivos específicos

- Apresentar os fundamentos históricos e conceituais relacionados às sequências, às progressões e às recorrências lineares de primeira e segunda ordens.
- Analisar as relações existentes entre o raciocínio matemático, o pensamento computacional e a construção de algoritmos aplicados à resolução de problemas.
- Discutir as possibilidades pedagógicas da utilização de tecnologias digitais e da programação de computadores no ensino de Matemática na Educação Básica.
- Introduzir os principais conceitos da linguagem de programação necessários à implementação de algoritmos matemáticos, como variáveis, operadores, estruturas condicionais, estruturas de repetição, funções, iteração e recursão.
- Desenvolver algoritmos em Python capazes de representar, gerar e analisar sequências definidas por relações de recorrência.
- Explorar problemas matemáticos clássicos e situações contextualizadas que possam ser modelados por recorrências, relacionando suas representações algébricas, numéricas e computacionais.
- Incentivar o protagonismo dos estudantes na formulação de estratégias, construção de algoritmos, teste de hipóteses, identificação de erros e interpretação dos resultados obtidos.

1.2 Organização

A estrutura deste trabalho foi definida de modo a apresentar, progressivamente, os fundamentos educacionais, matemáticos e computacionais que sustentam a proposta de ensino de sequências e recorrências lineares por meio de algoritmos e da programação em Python.

O capítulo 2 discute o ensino de Matemática no Ensino Médio diante das transformações sociais, culturais e tecnológicas contemporâneas. São abordados os desafios relacionados à aprendizagem matemática, à incorporação das tecnologias digitais no ambiente escolar e ao desenvolvimento de competências associadas ao raciocínio lógico, à resolução de problemas, à cultura digital e ao Pensamento Computacional.

O capítulo 3 apresenta os fundamentos históricos e matemáticos das sequências e recorrências. Inicialmente, são discutidos o conceito de sequência, suas diferentes formas de representação e suas classificações. Em seguida, são estudadas as progressões aritméticas e geométricas, estabelecendo-se relações entre suas expressões explícitas e recursivas. Segue-se o estudo das recorrências lineares de primeira e segunda ordens, contemplando definições, propriedades, métodos de resolução e exemplos de aplicação. Busca-se, desse modo, construir a base conceitual necessária para a posterior representação algorítmica desses objetos matemáticos.

No capítulo 4 dialoga sobre os algoritmos, considerando tanto sua dimensão histórica quanto sua relação com a atividade matemática. São apresentados procedimentos desenvolvidos por diferentes povos e matemáticos ao longo da história, , destacando entre os exemplos discutidos, métodos de aproximação, decomposição, resolução de equações e problemas clássicos cuja estrutura pode ser expressa por recorrências.

O capítulo 5 aborda a linguagem de programação Python e suas possibilidades de utilização no ensino de Matemática. São apresentadas características da linguagem, sua estrutura básica e os principais recursos necessários à implementação dos algoritmos trabalhados, como variáveis, operadores, estruturas condicionais, laços de repetição, funções, iteração e recursão. O capítulo também discute o uso pedagógico da programação, destacando a importância de relacionar cada comando computacional ao processo matemático correspondente, de modo que o código não seja utilizado apenas para automatizar cálculos, mas como uma representação do raciocínio empregado na resolução do problema.

Ademais é apresentada a modelagem computacional das recorrências e a proposta didática elaborada. Nele, as relações matemáticas estudadas são traduzidas em algoritmos e implementadas em Python, permitindo explorar padrões, acompanhar a produção dos termos, analisar processos de convergência e comparar estratégias iterativas e recursivas. A sequência didática é organizada de forma progressiva, articulando compreensão conceitual, elaboração de algoritmos, implementação computacional e interpretação dos resultados.

Por fim, nas considerações finais, são discutidas as principais contribuições da articulação entre sequências, recorrências, algoritmos e programação em Python. Também são apresentadas as limitações da pesquisa, bem como possibilidades para estudos futuros, entre elas a aplicação e avaliação da sequência didática em contextos escolares.

2 O ensino de Matemática no Nível Médio e os desafios contemporâneos

2.1 Do ensino mecânico à compreensão significativa

Em diferentes períodos, aprender Matemática no Ensino Médio brasileiro tem sido uma experiência de "copiar e colar". O professor, no centro do palco, entrega fórmulas prontas; o aluno, na plateia, tenta memorizar passos que parecem não levar a lugar algum. Essa abordagem trata a Matemática como um museu de ideias acabadas, ignorando que cada conceito nasceu da curiosidade humana, de dúvidas reais e de um contexto social. Em verdade, a matemática continua uma ciência ativa. A cada dia novos fatos, relevantes ou não, são descobertos.

A Matemática tem um papel imprescindível na vida do homem e da sociedade, permitindo que, por meio dela, o homem organize o pensamento, identifique padrões e tome decisões baseadas em evidências, percorrendo desde tarefas simples, como a gestão do tempo e do orçamento doméstico, até a compreensão de riscos em escolhas complexas. Oferecendo a segurança necessária para que o indivíduo não seja enganado por dados estatísticos manipulados ou por propostas econômicas ilusórias, ela é o que chamamos de literacia[1] financeira e numérica, um pilar essencial para a liberdade pessoal no mundo contemporâneo.

O ensino de matemática, para o desenvolvimento intelectual do indivíduo, é de grande importância, e para que esse processo ocorra de modo efetivo é necessário transpor os modelos tradicionais de ensino, não se limitando exclusivamente a "decorar fórmulas pelo simples fato de utilizar em uma avaliação", e estas sendo ainda ensinadas como verdades absolutas, sem a explicitação dos critérios que as orientam, limitando o professor a ser um mero reprodutor de métodos mecanizados e conteúdos isolados. Caminhando na direção oposta, deve-se incentivar o estudante na busca pelo fortalecimento do coletivo escolar e pela sua autonomia, provocando e problematizando os conteúdos lecionados, combatendo a padronização ao substituir o foco exclusivo no desempenho cognitivo imediato por uma formação que visa ao engajamento crítico, utilizando a matemática como instrumento de leitura crítica do mundo, tal como apresenta Selbach (2010 *apud* SILVA, 2016, p. 26), que “não existem conceitos matemáticos isolados e o aluno precisa perceber que sua tarefa ao resolver um problema é construir uma rede de conceitos”, na qual cada novo conceito aprendido pelo estudante esta-

[1] conjunto de conhecimentos, habilidades e atitudes relacionados à leitura e à escrita, assim como sua prática produtiva.

belece conexões com conhecimentos anteriores, formando uma estrutura dinâmica que favorece a compreensão significativa e a aplicação em diferentes contextos. Assim, ao fazer estudo do referido campo de pesquisa deste trabalho, por exemplo (sequências, progressões e recorrências), o aluno não apenas aprende técnicas específicas, mas passa a perceber como esses conteúdos se articulam com funções, padrões e modelagem matemática. Dessa forma, a rede de conceitos possibilita uma aprendizagem mais integrada, na qual o conhecimento é construído de maneira contínua e contextualizada, promovendo o desenvolvimento do raciocínio lógico e da autonomia intelectual

Muitas vezes, trocamos a compreensão real pelo domínio técnico.[2] Ao longo da Educação Básica, o desenvolvimento do pensamento matemático ocorre de forma gradual, acompanhando o avanço cognitivo dos estudantes. No Ensino Fundamental I, o aprendizado está fortemente voltado para a construção de noções básicas, como contagem, operações fundamentais e reconhecimento de padrões simples, geralmente trabalhados de maneira concreta e contextualizada. Já no Ensino Fundamental II, há uma ampliação desses conceitos, com a introdução de ideias mais abstratas, como generalizações, expressões algébricas iniciais e sequências numéricas, exigindo dos alunos maior capacidade de interpretação e organização do raciocínio. Quando chegamos ao Ensino Médio, o nível de abstração sobe atrelado aos novos conhecimentos e domínios exigidos pelas funções, sequências e progressões, cobrando um "fôlego"de raciocínio que o ensino mecânico não preparou. Sem a base do pensamento lógico, o estudante se vê diante de um muro de algoritmos vazios. O resultado? A matemática passa a ser vista não como uma ferramenta de leitura do mundo, mas como um obstáculo a ser superado. E esses pensamentos fogem do foco do ensino de matemática, que consiste em cultivar habilidades de raciocínio lógico e estimular o pensamento de maneira independente, provocar a criatividade e a capacidade de resolução de problemas no estudante.

Para que a matemática deixe de ser vista como uma simples técnica de memorização e passe a ser compreendida, provendo uma linguagem de investigação, é fundamental que o ambiente escolar promova o diálogo e a reflexão. No contexto da Base Nacional Comum Curricular (BNCC), o foco se desloca do "saber fazer"mecânico para o "saber por quê se faz", transformando a sala de aula em um espaço de construção coletiva onde o erro e a dúvida são pontos de partida para o pensamento lógico.

Nesse sentido, o desenvolvimento do raciocínio matemático não acontece de forma isolada, mas sim através da troca de ideias e da necessidade de validar descobertas perante o outro, como destaca a BNCC:

> Assim, para o desenvolvimento de competências que envolvem o raciocinar, é necessário que os estudantes possam, em interação com seus colegas e professores, investigar, explicar e justificar os problemas

[2] Embora muitos professores não consigam perceber essa limitação, justamente porque foram assim ensinados.

> resolvidos, com ênfase nos processos de argumentação matemática. Embora todas as habilidades pressuponham a mobilização do raciocínio, nem todas se restringem ao seu desenvolvimento. Assim, por exemplo, a identificação de regularidades e padrões exige, além de raciocínio, a representação e a comunicação para expressar as generalizações, bem como a construção de uma argumentação consistente para justificar o raciocínio utilizado. (Brasil 2018, p. 519)

Fazendo uso de tais pensamentos, a Matemática ganha vida, transforma-se em algo significativo quando resolve problemas que fazem sentido. E aqui deixamos claro que nem tudo na matemática está ligado diretamente a algo prático da vida do estudante, pois como enfatiza Machado (2013)

> A redução de qualquer tema ao nível do interesse prático-utilitário estreita os horizontes e pode conduzir a desvios: no caso da Matemática, tal desvio é a Mateologia. [...] a febre do utilitarismo costuma eivar o ensino da Matemática. [...] Se não atender a tal reclamo, é como se o tema fosse mera mateologia, não merecendo ser ensinado.(p.11-12)

Não se trata de defender que a matemática deva ser sempre prática, mas sim de reconhecer que, em alguns casos, o ensino precisa estar atrelado à realidade do estudante. Quando isolamos os números do cotidiano, criamos um distanciamento que gera insegurança e desmotivação. Ensinar Progressões, por exemplo, não deveria ser apenas sobre aplicar a fórmula do Termo Geral, mas também sobre perceber padrões, ritmos e regularidades que existem na natureza, na música, na economia – enfim, em um ambiente de vivência natural para o aluno.

O percurso histórico que envolve a Matemática nos ensina que o conhecimento desta ciência foi construído por meio de tentativas, erros e muita investigação. No entanto, na sala de aula tradicional, constantemente o foco se dirige apenas para o resultado final, que pode ser classificado como certo ou errado, atribuindo ao estudante uma classificação de Aprovado(caso acerte) e Reprovado(caso erre). Isso rouba do aluno a autonomia e o prazer da descoberta. O sucesso em Matemática não deveria ser medido pela capacidade de decorar regras, mas pela coragem de investigar e pela clareza ao pensar, tendo em vista que, ao repensarmos ensino de Matemática estamos, acima de tudo, colaborando para um ato de respeito ao estudante. Ao transformar a disciplina em uma atividade humana e dinâmica, permitimos que o jovem desenvolva autonomia intelectual (capacidade de pensar por si mesmo), pensamento crítico(não aceitar respostas prontas sem entender a lógica por trás delas.), e segurança(diminuir a ansiedade e o medo diante do novo.)

Para que o ensino da Matemática não se perca em um emaranhado de abstrações vazias ou de modo que consigamos visualizar de maneira rápida e prática sua utilidade, é preciso que professores e estudantes compreendam a disciplina como uma caixa de ferramentas projetada para a ação, Machado (2013, p. 13) ilustra muito bem esta analogia definindo que

> Devemos, sim, procurar convencer nossos alunos do macrossignificado da Matemática como instrumento. Uma faca, por exemplo, é um instrumento que serve para cortar. Mas é a lâmina que corta; o cabo nada corta. [...] Muitos conteúdos de Matemática são como lâminas, cabos, parafusos. Se quisermos dispor do instrumental matemático, indiscutivelmente útil para muitas finalidades práticas, temos de aprender não somente sobre lâminas afiadas [...]

E diante disso, devemos trabalhar para que a Matemática ganhe vida e se afaste do distanciamento que gera insegurança e desmotivação, sendo importante que o ensino de conceitos transcenda a aplicação mecânica de fórmulas e se ancore na percepção de padrões e ritmos presentes na vivência do estudante. Nesse cenário, a inserção das novas tecnologias e o estudo de algoritmos surgem como pontes fundamentais entre a abstração e a realidade dentro dos desafios atuais para o ensino de matemática.

2.2 Matemática e Tecnologias Digitais: o papel do pensamento computacional na promoção da aprendizagem significativa

Dentro do contexto acadêmico, as transformações que vivemos, seja no modo como nos comunicamos, como nos divertimos ou como trabalhamos, reconfiguraram o chão da escola. Hoje, ensinar Matemática para um jovem do Ensino Médio não pode mais ser um exercício de repetição de fórmulas, pois vivemos em uma era onde a informação está na palma da mão, mas a capacidade de interpretá-la e transformá-la em solução é o que realmente define o futuro.

O desafio que se impõe à escola e aos professores é o de dar um novo sentido às aulas. Não se trata apenas de substituir o quadro negro por um aparelho digital, mas de humanizar o aprendizado, permitindo que o estudante entenda que a Matemática é uma ciência viva. É o momento de o ensino de Matemática dialogar com a criatividade, onde construir, testar e ajustar modelos deixa de ser uma tarefa abstrata para se tornar a ferramenta com a qual o jovem aprende a ler e a intervir no mundo ao seu redor. E para que aconteça essa mudança, o campo educacional precisa estar em sintonia com as tecnologias que surgem, transicionar de um ensino linear e isolado para um ecossistema de aprendizagem em rede. Com este pensamento, demonstramos enxergar a tecnologia como um catalisador de novas posturas éticas e cognitivas, transformando o estudante de um receptor passivo em um autor consciente de suas próprias descobertas. Sobre esse potencial transformador, Fagundes (2007, p.14) argumenta que:

> As tecnologias digitais estão realizando transformações profundas nos processos de aprendizagem e nas mudanças da escola. Reflete que o uso das tecnologias na educação propicia a interdisciplinaridade, uma organização heterárquica[3], estimula a participação cooperativa

[3] um conceito que descreve sistemas organizacionais ou redes onde não há um controle centralizado e vertical, como na hierarquia, mas sim uma prevalência de múltiplos centros de poder, papéis iguais ou interconectados, e inteligência distribuída, permitindo flexibilidade e adaptação.

> e solidária, promove a autonomia e a responsabilidade da autoria nos alunos.

E é este o ponto central da educação, fazer com que os estudantes possam ser autores, ou melhor, protagonistas do seu processo de ensino e aprendizagem, sendo o professor e o ambiente escolar, guias para que o objetivo seja alcançado com êxito, pois como R. B. d. Silva et al. (2016) destacam, a participação dos estudantes no processo de ensino e aprendizagem é um importante caminho para o aprendizado, e este por sua vez, se dá pela ação comunicativa e participativa. Essa mudança de postura é um sopro de renovação para o ensino de Matemática, uma área que, durante muito tempo, ficou confinada à repetição de fórmulas e à passividade das aulas expositivas.

Como forma de atingir propósitos que convirjam para um ensino de matemática mais conectado à realidade do estudante e às transformações tecnológicas contemporâneas, destaca-se a Lei nº 14.533, de 11 de janeiro de 2023, que institui a Política Nacional de Educação Digital. Tal legislação reforça a necessidade de inserção das tecnologias digitais nos processos de ensino e aprendizagem, sobretudo ao enfatizar o desenvolvimento do pensamento computacional, da programação e da resolução de problemas nos diferentes níveis da educação básica. O presente desenvolvimento deste trabalho, aproxima-se diretamente dos objetivos previstos pela referida lei, uma vez que o estudo de recorrências lineares associado à construção de algoritmos em Python favorece o raciocínio lógico, a modelagem matemática e a compreensão de padrões, competências fundamentais para a formação do estudante na sociedade digital contemporânea. Além disso, a legislação evidencia que a educação matemática não deve restringir-se à mera reprodução mecânica de procedimentos, mas contribuir para a formação crítica, criativa e tecnológica dos indivíduos, articulando conhecimentos matemáticos às demandas científicas e computacionais do século XXI.

A Lei nº 14533 torna-se um grande avanço para o sistema educacional brasileiro, integrando a cultura digital ao processo de ensino, não apenas como uma ferramenta, mas como uma parte importante da formação dos estudantes. Seu principal objetivo é garantir que a educação acompanhe as mudanças da sociedade atual, repleta de tecnologias da informação e comunicação, entendendo que não basta apenas dar acesso a dispositivos e à internet: é necessário ensinar os estudantes a usar essas tecnologias de maneira crítica, ética, responsável e criativa, para formar cidadãos capazes de atuar de forma ativa no mundo digital.

A Política Nacional de Educação Digital se baseia na ideia de que a educação digital é um direito e deve ser tratada com princípios como equidade, inclusão e qualidade da educação. Propondo ações que atendam às necessidades do século XXI, apresenta como pontos de destaque a valorização do pensamento computacional e da cultura digital como habilidades essenciais. A lei incentiva a inclusão desses elementos nos currículos

escolares, para desenvolver habilidades como resolução de problemas, raciocínio lógico, análise de dados e criação de soluções tecnológicas. Isso aproxima áreas como matemática e programação, favorecendo abordagens interdisciplinares e mais significativas para os estudantes. Para tanto, deve-se criar condições para que práticas pedagógicas inovadoras, como o uso de algoritmos, programação e modelagem matemática, sejam incorporadas ao cotidiano escolar, contribuindo para uma aprendizagem mais ativa e contextualizada.

O Programa Nacional de Educação Digital apresentado na referida lei estabelece diretrizes que ajudam a garantir que todos os estudantes tenham acesso às tecnologias digitais e as usem de forma educativa. O Art. 3º dessa lei apresenta os princípios que orientam a Política Nacional de Educação Digital. Esses princípios são como uma base para criar práticas educacionais que atendam às necessidades da sociedade de hoje.

Dois princípios são especialmente importantes: a promoção da inclusão digital com equidade e o desenvolvimento de competências digitais. A educação digital não é apenas sobre ter acesso a tecnologias, mas também sobre garantir que todos os estudantes possam usar essas tecnologias de forma significativa. Além disso, é preciso ensinar os estudantes a interagir criticamente com o mundo digital, desenvolvendo habilidades como o pensamento computacional, a resolução de problemas e a criação de conhecimentos usando ferramentas tecnológicas, como podemos perceber no artigo expresso e seus incisos:

> Art. 3º O eixo Educação Digital Escolar tem como objetivo garantir a inserção da educação digital nos ambientes escolares, em todos os níveis e modalidades, a partir do estímulo ao letramento digital e informacional e à aprendizagem de computação, de programação, de robótica e de outras competências digitais, englobando:
>
> I - pensamento computacional, que se refere à capacidade de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas soluções de forma metódica e sistemática, por meio do desenvolvimento da capacidade de criar e adaptar algoritmos, com aplicação de fundamentos da computação para alavancar e aprimorar a aprendizagem e o pensamento criativo e crítico nas diversas áreas do conhecimento;
>
> II - mundo digital, que envolve a aprendizagem sobre hardware, como computadores, celulares e tablets, e sobre o ambiente digital baseado na internet, como sua arquitetura e aplicações; [...] (Brasil 2023)

Para além da definição dos eixos estruturantes da Educação Digital Escolar, a legislação brasileira avança ao estabelecer um conjunto de estratégias prioritárias que visam garantir a efetivação dessas diretrizes no contexto educacional. Essas estratégias não se limitam à inserção pontual de tecnologias no ambiente escolar, mas propõem uma transformação mais ampla, que envolve a reconfiguração das práticas pedagógicas, a formação continuada dos profissionais da educação e a ampliação das condições de acesso e uso significativo das ferramentas digitais. Nesse sentido, a Educação Digital deixa de ser compreendida como um elemento acessório e passa a ocupar um papel

central na formação dos estudantes, alinhando-se às demandas de uma sociedade cada vez mais conectada e orientada por dados, informação e inovação.

Dessa forma, a proposta caminha junto às orientações da Base Nacional Comum Curricular, ao enfatizar o desenvolvimento de competências que ultrapassam o domínio técnico, contemplando aspectos como pensamento crítico, autonomia, resolução de problemas e participação ética no mundo digital. Ao destacar elementos como o pensamento computacional, a programação, a lógica e a cidadania digital, o documento reforça a necessidade de uma educação que prepare o estudante não apenas para consumir tecnologia, mas para compreendê-la, questioná-la e utilizá-la de maneira consciente e criativa.

Além disso, as estratégias evidenciam a importância de uma abordagem sistêmica, que envolve tanto o protagonismo discente quanto o fortalecimento da formação docente, reconhecendo que a integração efetiva das tecnologias depende de professores preparados para mediar processos de aprendizagem mais dinâmicos, investigativos e colaborativos. A valorização de práticas pedagógicas que envolvem algoritmos, modelagem e experimentação aproxima o ensino de Matemática das demandas contemporâneas, especialmente no que se refere ao desenvolvimento do raciocínio lógico e do pensamento computacional. Podemos destacar, dentro das estratégias prioritárias voltadas à consolidação da Educação Digital Escolar, evidenciando caminhos concretos para a implementação dessas diretrizes no cotidiano das instituições de ensino:

> I - desenvolvimento de competências dos alunos da educação básica para atuação responsável na sociedade conectada e nos ambientes digitais, conforme as diretrizes da base nacional comum curricular;
>
> II - promoção de projetos e práticas pedagógicas no domínio da lógica, dos algoritmos, da programação, da ética aplicada ao ambiente digital, do letramento midiático e da cidadania na era digital;
>
> [...]
>
> IV - estímulo ao interesse no desenvolvimento de competências digitais e na prossecução de carreiras de ciência, tecnologia, engenharia e matemática;
>
> [...]
>
> VI - promoção de cursos de extensão, de graduação e de pós-graduação em competências digitais aplicadas à indústria, em colaboração com setores produtivos ligados à inovação industrial;
>
> [...]
>
> IX - promoção da formação inicial de professores da educação básica e da educação superior em competências digitais ligadas à cidadania digital e à capacidade de uso de tecnologia, independentemente de sua área de formação;
>
> [...] (Brasil 2023)

Podemos perceber diante do que nos é apresentado, com base em processos de estudo e em documentos analisados até aqui, que o trabalho com algoritmos e programação facilita o processo, tendo em vista que ao escrever um código ou desenhar uma ordem,

o estudante não está apenas resolvendo um problema: está tomando decisões, testando estratégias e vendo, em tempo real, o impacto de suas escolhas, reafirmando as ideias de Papert (1985, p. 37) quando descreve que este novo conhecimento abstraído pelo aluno ao interagir com as tecnologias de programação e algoritmos "é fonte de poder e é experienciado como tal a partir do momento que começa a se formar na mente da criança". Em suma, é um convite para que o aluno saia da plateia e suba ao palco da própria aprendizagem, onde o raciocínio precisa ser organizado e as ideias são constantemente refinadas diante de cada novo desafio.

Além de ser um exercício técnico, a programação é uma forma de comunicação. Para que um programa funcione, o estudante precisa traduzir seu pensamento abstrato em uma linguagem clara e formal, que pode ser lida e discutida por outros. O código deixa de ser um dever de casa solitário e se torna um objeto de debate, uma criação compartilhada que alunos e professores analisam e reconstroem juntos. É nesse ambiente de troca e colaboração que a Matemática perde seu peso engessado e ganha a leveza da experimentação. Todo este processo ajuda a desenvolver o raciocínio e o pensamento crítico, levando o estudante a encontrar uma solução que faça sentido e responda a necessidade de determinado problema, pois como enfatiza Forbellone (2005)[2], tal caminho faz

> o uso correto das leis do pensamento da ordem da razão e de processos de raciocínio e simbolização formais na programação de computadores, objetivando a racionalidade e o desenvolvimento de técnicas que cooperem para a produção de soluções logicamente válidas e coerentes, que resolvam com qualidade os problemas que se deseja programar.

Nesse cenário, as tecnologias digitais deixam de ser um acessório e passam a ser o terreno onde competências essenciais são cultivadas. Quando o estudante lida com algoritmos e ferramentas digitais, ele é desafiado a desenvolver habilidades que o acompanharão por toda a vida: a capacidade de argumentar com base em dados, a análise crítica sobre o funcionamento das coisas e a segurança para tomar decisões fundamentadas.

Dentro das competências específicas de Matemática para o Ensino Médio e as Habilidades trazidas pela BNCC, destacam-se de maneira geral que o pensamento matemático deve transcender o puro e simples fazer mecanicamente, além de motivar a utilização de novas tecnologias que ajudem a alcançar os objetivos de provocação do conhecimento e desenvolvimento das competências de criticidade e argumentação. Temos expressas na tabela abaixo essas Competências Específicas (CE) e suas habilidades.

Tabela 1 – Competências específicas e habilidades relacionadas na BNCC.

Competência Específica 3

Utilizar estratégias, conceitos e procedimentos matemáticos, em seus campos – Aritmética, Álgebra, Grandezas e Medidas, Geometria, Probabilidade e Estatística – para interpretar, construir modelos e resolver problemas em diversos contextos, analisando a plausibilidade dos resultados e a adequação das soluções propostas, de modo a construir argumentação consistente.

Habilidades relacionadas:

(EM13MAT301) Resolver e elaborar problemas do cotidiano, da Matemática e de outras áreas do conhecimento, que envolvem equações lineares simultâneas, usando técnicas algébricas e gráficas, incluindo ou não tecnologias digitais.

(EM13MAT302) Resolver e elaborar problemas cujos modelos são as funções polinomiais de 1º e 2º graus, em contextos diversos, incluindo ou não tecnologias digitais.

(EM13MAT315) Reconhecer um problema algorítmico, enunciá-lo, procurar uma solução e expressá-la por meio de um algoritmo, com o respectivo fluxograma.

Competência Específica 4

Compreender e utilizar, com flexibilidade e fluidez, diferentes registros de representação matemáticos (algébrico, geométrico, estatístico, computacional etc.), na busca de solução e comunicação de resultados de problemas, de modo a favorecer a construção e o desenvolvimento do raciocínio matemático.

Habilidades relacionadas:

(EM13MAT401) Converter representações algébricas de funções polinomiais de 1º grau para representações geométricas no plano cartesiano, distinguindo os casos nos quais o comportamento é proporcional.

(EM13MAT402) Converter representações algébricas de funções polinomiais de 2º grau para representações geométricas no plano cartesiano.

(EM13MAT406) Utilizar os conceitos básicos de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.

Competência Específica 5

Investigar e estabelecer conjecturas a respeito de diferentes conceitos e propriedades matemáticas, empregando recursos e estratégias como observação de padrões, experimentações e tecnologias digitais.

Habilidades relacionadas:

(EM13MAT501) Investigar relações entre números expressos em tabelas para representá-los no plano cartesiano, identificando padrões e criando conjecturas.

(EM13MAT502) Investigar relações entre números expressos em tabelas para representá-los no plano cartesiano, reconhecendo quando essa representação é de função polinomial de 2º grau do tipo $y = ax^2$.

(EM13MAT507) Identificar e associar sequências numéricas (PA) a funções afins de domínios discretos para análise de propriedades.

(EM13MAT508) Identificar e associar sequências numéricas (PG) a funções exponenciais de domínios discretos para análise de propriedades.

É importante ressaltar que as competências e habilidades da BNCC destacadas aqui não encerram o debate; pelo contrário, elas abrem portas para inúmeras outras conexões no currículo de Matemática do Ensino Médio. A decisão de focar em pontos específicos nasceu da afinidade direta entre esses temas e o objetivo deste projeto: o desejo de fortalecer o raciocínio lógico, a arte de argumentar, a coragem para resolver problemas e o uso das tecnologias como pontes para o saber.

A articulação entre as competências gerais e a prática em sala de aula se materializam quando o estudante deixa de ser um executor de tarefas e assume o papel de investigador e modelador. Colocando um olhar sobre os documentos que regem a educação, vemos na BNCC as habilidades, competências, atitudes e valores definidas para o Ensino Médio e que ao concluir esta etapa, os estudantes possam tê-los garantidos para o seu exercício da cidadania e mundo do trabalho (Brasil 2018).

A BNCC não é apenas um guia técnico, mas um convite para que a escola se transforme em um espaço de descoberta e cidadania. Suas Competências Gerais funcionam como os alicerces de uma Matemática que pulsa e faz sentido. Quando olhamos para a CG[4] 2, percebemos que ela celebra a curiosidade. Ela nos lembra que o aluno deve ser incentivado a investigar e resolver problemas. Nesse contexto, trabalhar com algoritmos e modelagem matemática dentro do ensino de sequências e recorrências deixa de ser um exercício frio, meramente calculista, para se tornar uma aventura intelectual: é onde o jovem cria hipóteses, testa caminhos e aprende a validar suas próprias ideias.

A CG 5, por sua vez, traz o olhar para a ética e o propósito no mundo digital. Ela defende que usar a tecnologia não é apenas "clicar em botões", mas saber por que e como

[4] Utilizaremos a abreviatura CG para nos referirmos a Competência Geral.

fazê-lo de forma significativa. Ao aprender a programar e a criar ambientes digitais, os estudantes deixam o papel de espectadores. Eles deixam de ser consumidores de aplicativos para se tornarem autores de soluções, ganhando a autonomia necessária para não serem apenas guiados pela tecnologia, mas para serem os donos de sua própria jornada intelectual.

Por fim, a CG 7 nos fala sobre o poder da palavra e do argumento fundamentado. Em um mundo cheio de informações desencontradas, a Matemática ensina o aluno a olhar para dados e fatos com clareza. Ela dá a ele as ferramentas para justificar suas escolhas, investigar os padrões existentes em sequências, analisar resultados com rigor e defender suas ideias com firmeza e consciência.

Articular essas competências significa, em última análise, oferecer uma educação que não se contenta com o mecânico. É acreditar em um ensino de Matemática que desperta a criticidade, valoriza o protagonismo e prepara o jovem para ser um participante ativo e pensante na sociedade.

Partindo para as competências específicas e habilidades da disciplina de matemática propostas no referido documento, identificamos a ligação de algumas com este projeto. A CE3[5] convoca o estudante a interpretar e construir modelos para resolver problemas reais em diversos campos, como a Álgebra e a Aritmética. Esse processo de modelagem encontra sua ferramenta de expressão na habilidade EM13MAT315, que foca na interpretação e construção de fluxogramas. Ao desenhar um fluxograma, o aluno está, na verdade, esquematizando a lógica de um modelo matemático: ele decompõe um problema complexo em etapas finitas, analisando a plausibilidade dos resultados a cada tomada de decisão. Esse exercício de planejar o caminho da solução é o que sustenta a construção de uma argumentação consistente, pois obriga o estudante a justificar a estrutura do seu raciocínio.

Essa fundamentação lógica se aprofunda com a CE5, que enfatiza a investigação de propriedades e o estabelecimento de conjecturas por meio de tecnologias digitais. É aqui que a habilidade EM13MAT406 se torna essencial, ao propor a implementação de algoritmos em linguagens de programação. A programação funciona como um laboratório experimental de testagem de hipóteses: ao codificar uma solução, o estudante pode observar padrões em tempo real e validar suas conjecturas através da experimentação computacional. Ao analisar uma sequência, identificar sua lei de formação ou expressá-la por meio de uma relação de recorrência, o estudante mobiliza diferentes registros de representação (numérico, algébrico, gráfico e computacional) fortalecendo sua capacidade de transitar entre linguagens matemáticas e aplicá-las às tecnologias computacionais. Dessa forma, o domínio dos algoritmos e dos fluxogramas não é um fim em si mesmo, mas o meio pelo qual o aluno exerce a investigação matemática de

[5] Utilizaremos a abreviação CE para nos referirmos a Competência Específica.

forma autônoma.

As Diretrizes Curriculares do Estado da Paraíba (2025), ao organizarem os Itinerários Formativos, articulam essas competências a partir de uma compreensão do conhecimento como construção histórica e cultural, em permanente processo de reestruturação. Ao propor temas geradores como Inteligência Artificial, algoritmos e pensamento computacional, o documento busca alinhar a base teórica da BNCC às dinâmicas contemporâneas da sociedade, reconhecendo que os modos de compreender e explicar a realidade se transformam à medida que a própria sociedade reflete, se reinventa e redireciona seu modo de ser, especialmente no contexto do mundo do trabalho.

Permear toda essa estruturação, entender o conhecimento como uma construção cultural, e não como algo estático, imutável, nos permite enxergar o processo de ensino e aprendizagem e as tecnologias digitais sob uma nova luz. Ao aceitarmos que nossas explicações sobre o mundo se transformam conforme a sociedade se reinventa, admitimos que educar é um processo vivo, profundamente conectado aos dilemas e avanços do nosso tempo.

É aqui que a articulação com as tecnologias digitais e o pensamento computacional revela sua verdadeira força. Elas não entram na sala de aula apenas como ferramentas modernas, mas como uma nova forma de ler e escrever o mundo. Papert (2008) nos provoca ao enfatizar que, ao programar, o estudante não apenas usa a máquina, mas aprende a projetar seus próprios processos de pensamento. Ele se torna um criador de modelos, alguém que testa hipóteses e vê, na tela e na vida, os resultados de suas ideias. O estudo de recorrências, em particular, favorece essas práticas investigativas, pois exige que o estudante compreenda relações de dependência entre termos sucessivos, identifique padrões e justifique generalizações, contribuindo para a construção de argumentos matematicamente consistentes, baseados em regularidades observadas e em demonstrações, ainda que em níveis introdutórios no Ensino Médio. E tudo isso acontece em comunhão com a programação e os algoritmos, pois, ao ser compreendido como a materialização de uma sequência lógica, esses estudos permitem que o aluno visualize a estrutura de uma progressão não como um dado estático, mas como um processo dinâmico de construção de padrões.

Para o professor, esse caminho traz desafios que tocam a identidade profissional. O papel muda: sai o transmissor de verdades prontas e entra o mediador de investigações. Segundo Valente (1999), usar tecnologias com sentido pedagógico exige mudar o que entendemos por ensinar e aprender. Aprender passa a ser um encontro entre o sujeito, o objeto e a cultura, unindo à a formação docente, as novas práticas e o universo digital, reafirmando que ensinar Matemática é, acima de tudo, oferecer ferramentas para que o estudante compreenda e transforme sua realidade.

O uso das tecnologias digitais transforma esse estudo em algo palpável. Com algoritmos e linguagens de programação, as sequências deixam de ser listas de números parados no papel e se tornam processos dinâmicos. O estudante pode ver todo o processo acontecer através das iterações, compreendendo que a mesma recorrência que ele estuda na matemática é a estrutura que faz um software funcionar. É aqui que o pensamento matemático e o pensamento computacional se abraçam.

No horizonte da Matemática dentro do Ensino Médio, os atuais desafios para o ensino e aprendizagem nos faz refletir sobre o papel de cada um no ambiente escolar, e os documentos norteadores da educação nos convidam a enxergar as regularidades e os padrões não como fórmulas estáticas em um livro. Estudar sequências e recorrências é, em essência, aprender a ler e interpretar como as coisas crescem, como variam e como dependem umas das outras. Seja na natureza, na economia ou na tecnologia, esses padrões nos permitem modelar a vida e antecipar o futuro. A ideia central é que o estudante consiga ver um padrão, entender sua lógica e, então, aplicá-la usando diferentes linguagens, seja um gráfico, uma expressão algébrica ou um código de computador.

3 Sequências e Recorrências Lineares: Fundamentos e Aplicações Algorítmicas

Desde os primórdios da civilização, o homem, pela necessidade de compreender o mundo e tudo o que está relacionado à sua sobrevivência e desenvolvimento, buscou compreender padrões e regularidades nas situações que o cercavam. A matemática envolvida estava relacionada às aplicações práticas (Eves 2011). O estudo desses padrões, o que levou ao que atualmente denominamos sequências, era responsável pela organização da vida social, econômica e religiosa do homem. Sequências numéricas e geométricas estiveram presentes em calendários mesopotâmicos, nos estudos dos pitagóricos e nas técnicas de construção utilizada pelos egípcios. No papiro de Rhind, por exemplo, encontram-se problemas que envolvem progressões geométricas, evidenciando que a ideia de geração ordenada de termos já fazia parte do pensamento matemático egípcio, ainda que expressa de forma retórica e operacional.

Essa busca primordial por padrões transcendeu a mera contagem e medição, evoluindo para a percepção de que certos fenômenos não apenas se repetiam, mas se geravam uns aos outros em uma cadeia lógica de dependência. É aqui que o conceito de recorrência – uma regra que define cada termo a partir dos anteriores – começa a se insinuar, ainda que de forma intuitiva, no pensamento antigo. Os calendários mesopotâmicos, por exemplo, não eram apenas listas de dias; baseavam-se em ciclos astronômicos precisos, onde a posição da Lua ou do Sol em um dado momento era função de suas posições anteriores, um algoritmo primitivo inscrito nas estrelas. Da mesma forma, os construtores das pirâmides egípcias, ao trabalharem com progressões aritméticas no dimensionamento de blocos e camadas, aplicavam, na prática, uma regra de formação sequencial que garantia a estabilidade e a forma da construção. Eles operavam com a recorrência sem necessariamente formalizá-la em uma linguagem simbólica.

Do ponto de vista contemporâneo, as recorrências assumem papel central não apenas na matemática, mas também na ciência da computação, especialmente na construção e análise de algoritmos. A relação entre sequências recorrentes e algoritmos permite a modelagem de problemas de forma estruturada, favorecendo o desenvolvimento do pensamento computacional e a compreensão de processos iterativos e recursivos.

Neste capítulo apresentaremos as definições de sequências e recorrências seguidas de exemplos e problemas envolvendo os conceitos apresentados. Para tanto, tomaremos por referenciais as obras de Grimaldi (2004), Rosen (2010), Morgado e Carvalho (2022) e Gomes, Diniz e Teodoro (2024).

3.1 Sequências Numéricas

Definição 3.1. *Uma sequência de números reais é uma função $a : \mathbb{N} \to \mathbb{R}$ que associa cada número natural n a um número real $a(n) = a_n$, denominado o n-ésimo termo da sequência. Escrevemos $(a_1, a_2, ..., a_n, ...)$ ou $(a_n)_{n\in\mathbb{N}}$ ou (a_n) para representar os termos da sequência.*

Uma sequência numérica pode assumir diferentes comportamentos ao longo de sua formação, podendo apresentar padrões relacionados ao crescimento, decrescimento, repetição ou dependência entre seus termos. Em geral, esses elementos estão associados a uma lei de formação previamente estabelecida.

Exemplo 1. *Tomando a sequência (a_n), em que $a_n = 2n, \forall n \in \mathbb{N}$, obtemos a sequência dos números pares positivos.*

Exemplo 2. *Considere a sequência (a_n), em que $a_n = \frac{1}{n}, \forall n \in \mathbb{N}$. A lista de termos dessa sequência, começando por a_1, ou seja, $(a_1, a_2, a_3, ...)$, é $\left(1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \cdots\right)$.*

3.1.1 Tipos especiais de sequências

Seja (a_n) uma sequência, diz-se que (a_n) é:

- *limitada superiormente*: quando existe $c \in \mathbb{R}$ tal que $a_n \leq c$, para todo $n \in \mathbb{N}$.
- *limitada inferiormente*: quando existe $c \in \mathbb{R}$ tal que $a_n \geq c$, para todo $n \in \mathbb{N}$.
- *limitada*: quando existe $k \in \mathbb{R}$ tal que $|a_n| \leq k$, para todo $n \in \mathbb{N}$.
- *monótona*: quando se tem $a_n \leq a_{n+1}$, para todo $n \in \mathbb{N}$ ou $a_{n+1} \leq a_n$ para todo $n \in \mathbb{N}$. Nesse caso, uma sequência monótona pode ser classificada como:
  - *não decrescente*: quando $a_n \leq a_{n+1}$, para todo $n \in \mathbb{N}$;
  - *crescente*: quando $a_n < a_{n+1}$, para todo $n \in \mathbb{N}$;
  - *não crescente*: quando $a_{n+1} \leq a_n$, para todo $n \in \mathbb{N}$;
  - *decrescente*: quando $a_{n+1} < a_n$, para todo $n \in \mathbb{N}$.
- *periódica*: quando existe um número inteiro positivo p tal que $a_{n+P} = a_n$ para todo n. O menor $p > 0$ inteiro talque a condição anterior seja satisfeita é dito o *período* de (a_n).

Exemplo 3. *Seja a sequência (a_n) definida por*

$$a_n = \frac{1}{n},\ n \in \mathbb{N}^*$$

Todos os seus termos são menores ou iguais a 1, ou seja, (a_n) é limitada superiormente.

Exemplo 4. *Seja a sequência* (a_n) *dada por*

$$a_n = n, \ \forall \ n \in \mathbb{N}.$$

Todos os seus termos são maiores ou iguais a 1, isto é, (a_n) é limitada inferiormente.

Exemplo 5. *A sequência* (a_n) *definida por*

$$a_n = \frac{(-1)^n}{n}, \ \forall \ n \in \mathbb{N}^*.$$

é limitada (superior e inferiormente). De fato, temos $-1 \leq a_n \leq 1, \quad \forall n \geq 1.$

Exemplo 6. *A sequência* (a_n) *definida por*

$$a_n = 1 - \frac{1}{n}, n \in \mathbb{N}^*$$

é monótona não-decrescente. De fato, para todo $n \geq 1$*, temos*

$$a_{n+1} - a_n = \left(1 - \frac{1}{n+1}\right) - \left(1 - \frac{1}{n}\right) = \frac{1}{n} - \frac{1}{n+1} = \frac{1}{n(n+1)} > 0,$$

logo $a_{n+1} > a_n$*, para todo* $n \geq 1$*.*

Exemplo 7. *A sequência* (a_n) *definida por* $a_n = (-1)^n, n \in \mathbb{N}$*, é periódica. Tem-se o período* $p = 2$*. De fato, para todo* $n \in \mathbb{N}$*,*

$$a_{n+2} = (-1)^{n+2} = (-1)^n \cdot (-1)^2 = (-1)^n = a_n.$$

Seus termos alternam entre 1 *(para* n *par) e* -1 *(para* n *ímpar), repetindo-se a cada dois índices.*

3.2 Progressões Aritméticas e Geométricas

As progressões aritméticas e progressões geométricas são as sequências mais apresentadas no Ensino Médio, por constituírem modelos de crescimento amplamente presentes no cotidiano e em diversas áreas do conhecimento.

Exemplo 8. *Considere um modelo simplificado de crescimento populacional em um ambiente controlado. Em uma determinada situação, uma população inicial de 50 indivíduos apresenta um acréscimo constante de 10 indivíduos em determinado intervalo de tempo. Em outra situação, a mesma população inicial dobra no mesmo intervalo de tempo que a situação anterior. Com base nas informações, analisando os dois modelos de crescimento, determinamos que as sequências são, respectivamente* $(50, 60, 70, 80, ...)$ *e* $(50, 100, 200, 400, ...)$*.*

Um aspecto relevante para o estudo de sequências no ensino médio é o desenvolvimento do raciocínio lógico. Ao analisar uma sequência, o aluno precisa identificar regras de formação, testar hipóteses e justificar resultados, o que contribui para a autonomia intelectual e o pensamento crítico. Esse processo está alinhado às competências propostas pela educação contemporânea, que valoriza não apenas a memorização, mas a compreensão e a aplicação do conhecimento.

3.2.1 Progressão Aritmética

Embora classifiquemos as progressões aritméticas e geométricas como sequências, elas diferenciam-se essencialmente pela forma como os termos se relacionam. Esses dois tipos de sequências representam formas distintas de crescimento.

Definição 3.2. *Uma progressão aritmética (abreviadamente, PA) é uma sequência numérica na forma* $(\alpha, \alpha + r, \alpha + 2r, ..., \alpha + nr, ...)$*, ou seja, a diferença entre um termo e o termo anterior é uma constante. Essa constante é chamada de razão da progressão e representa por* r*.*

Seja uma progressão aritmética $(a_1, a_2, a_3, ...)$ de razão r. Pela definição, temos:

$$\begin{aligned} a_2 - a_1 &= r, \\ a_3 - a_2 &= r, \\ &\vdots \\ a_{n-1} - a_{n-2} &= r, \\ a_n - a_{n-1} &= r. \end{aligned}$$

Somando as $(n-1)$ igualdades, membro a membro, obtemos

$$\sum_{i=1}^{n} (a_i - a_{i-1}) = (n-1)r$$

Como a soma à esquerda é telescópica, obtemos

$$a_n - a_1 = (n-1) \cdot r,$$

Ou ainda, mais geralmente, é válido que

$$a_n - a_k = (n-k) \cdot r, \forall \quad 1 \leq k < n.$$

É natural buscar outras formas de caracterizar esse tipo de sequência, de modo a ampliar a compreensão sobre sua estrutura e propriedades. A ideia central esteve associada à existência de uma variação constante, à diferença fixa entre termos consecutivos. No entanto, essa não é a única maneira de descrever uma PA. Uma perspectiva

bastante útil consiste em observar o comportamento global dos termos da sequência, em vez de apenas as relações locais entre termos consecutivos. Nesse sentido, podemos investigar se existe uma expressão geral que permita calcular diretamente o termo de ordem n a partir de sua posição, sem a necessidade de recorrer aos termos anteriores. Essa abordagem conduz à noção de que as progressões aritméticas podem ser descritas por uma lei de formação linear, evidenciando uma relação direta com funções afins.

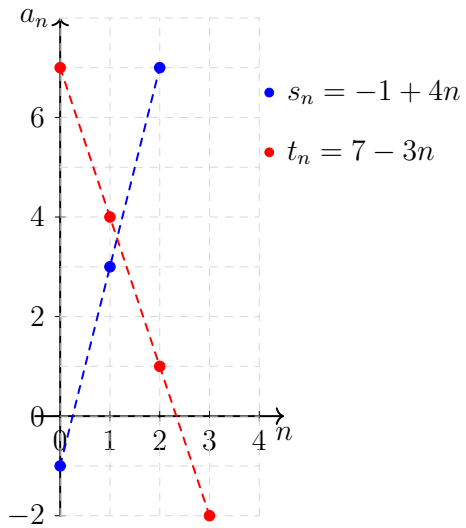
Proposição 1. *Uma sequência (a_n) é uma progressão aritmética se, e somente se, existem constantes A e B tais que $a_n = An + B$ para todo $n \in \mathbb{N}$.*

Demonstração. ($\Rightarrow$) Suponha que (a_n) é uma progressão aritmética. Daí temos, $a_n = a_1 + (n-1)r = a_1 + nr - r = rn + (a_1 - r) = An + B$, onde $A = r$ e $B = a_1 - r$.
($\Leftarrow$) Agora suponhamos que $a_n = An + B$, $\forall n \in \mathbb{N}$. Assim temos, $a_n - a_{n-1} = An + B - (A(n-1) + B) = An + B - An + A - B = A$.
Como A é constante, tem-se que a sequência (a_n) é uma progressão aritmética de razão A e primeiro termo $a_1 = A + B$. ■

Exemplo 9. *As sequências (s_n), com $s_n = -1 + 4n$ e (t_n), com $t_n = 7 - 3n$ são progressões aritméticas com termos iniciais e razões iguais a -1 e 4 e 7 e -3, respectivamente, se começarmos em $n = 0$. A lista de termos $(s_0, s_1, s_2, \dots)$ começa com $(-1, 3, 7, 11, \dots)$ e a lista de termos $(t_0, t_1, t_2, \dots)$ começa com $(7, 4, 1, -2, \dots)$.*

Esboçando o gráfico destas sequências, dispomos da seguinte representação:

Figura 1 – Representação gráfica de duas progressões aritméticas com comportamentos crescente e decrescente.



Fonte: Autor, 2026.

Com frequência, surgem problemas envolvendo progressão aritmética que pedem a soma de uma certa quantidade (finita) de termos da sequência. Temos o seguinte como exemplo:

Exemplo 10. *João decidiu economizar para uma viagem. No primeiro mês, ele guardou R$ 200,00. A cada mês subsequente, ele aumentou sua economia em R$ 50,00 em relação ao mês anterior. Quanto João economizará no 10º mês? Qual será o total acumulado após 12 meses?*

Neste exemplo, temos a sequência $(a_1, a_2, a_3, \dots)$, onde $a_1 = 200,00$ e $r = 50,00$. Utilizando a fórmula do termo geral, com $n = 10$:

$$a_{10} = 200 + (10 - 1) \cdot 50 = 200 + 9 \cdot 50 = 200 + 450 = 650$$

Portanto no 10º mês, João economizará R$ 650,00.

Para determinar o total acumulado, deduziremos primeiramente uma expressão da soma dos termos de uma Progressão Aritmética (S_n).

A soma dos n primeiros termos de uma progressão aritmética é obtida por contagem dupla. Escreve-se:

$$\begin{cases} S_n = & a_1 + a_2 + ... + a_n \\ S_n = & a_n + a_{n_1} + ... + a_1 \end{cases}$$

Adicionando membro a membro essas duas igualdades, segue-se que

$$\begin{aligned} 2 \cdot S_n =& (a_1 + a_n) + (a_2 + a_{n-1}) + ... + (a_n + a_1) \\ 2 \cdot S_n =& n(a_1 + a_n) \\ S_n =& \frac{(a_1 + a_n)}{2} n \end{aligned} \tag{3.1}$$

Agora podemos encontrar o acumulado após os 12 meses da seguinte maneira: determinamos $a_{12} = 750$ e calculamos

$$S_{12} = \frac{12 \cdot (200 + 750)}{2} = \frac{12 \cdot 950}{2} = 6 \cdot 950 = 5.700$$

Após 12 meses, João terá acumulado R$ 5.700,00.

3.2.2 Progressão Geométrica

Vimos que nas progressões aritméticas o crescimento ocorre por meio de uma diferença constante entre termos consecutivos, contudo, há situações em que a evolução de uma sequência se dá de forma "multiplicativa", ou seja, cada termo é obtido a partir

do anterior pela multiplicação por uma constante fixa. Esse tipo de comportamento aparece com frequência em fenômenos como o crescimento populacional, a propagação de epidemias e o cálculo de juros compostos.

Nesse contexto, surgem as progressões geométricas, que constituem um importante modelo para descrever processos de crescimento ou decrescimento exponencial. O estudo dessas sequências permite ao estudante compreender padrões multiplicativos e estabelecer conexões com outros conteúdos matemáticos, como potências e funções exponenciais, ampliando sua capacidade de interpretar e modelar situações do mundo real.

Definição 3.3. *Uma sequência numérica* (a_n) *é dita uma progressão geométrica quando existe uma constante real* $q \neq 0$ *tal que*

$$a_n = q \cdot a_{n-1}.$$

Denominamos essa constante q *de razão da progressão geométrica.*

Exemplo 11. *As sequências* (b_n)*, com* $b_n = (-1)^n$*,* (c_n)*, com* $c_n = 2 \cdot 5^n$ *e* (d_n)*, com* $d_n = 6\left(\frac{1}{3}\right)^n$ *são progressões geométricas com termo inicial e razão iguais a* 1 *e* $-1; 2$ *e* 5*; e* 6 *e* $\frac{1}{3}$*, respectivamente, se começarmos com* $n = 0$*.*
A lista de termos de $(b_0, b_1, b_2, ...)$ *começa com* $(1, -1, 1, -1, 1, -1, ...)$*;*
a lista de termos de $(c_0, c_1, c_2, ..)$ *começa com* $(2, 10, 50, 250, 1250, ...)$*;*
a lista de termos de $(d_0, d_1, d_2, ...)$ *começa com* $(6, 2, \frac{2}{3}, \frac{2}{9}, \frac{2}{27}, ...)$

Considere a progressão geométrica $(a_1, a_2, a_3, ..., a_n, ...)$. Por definição, temos

$$\begin{aligned} a_2 &= a_1.q \\ a_3 &= a_2.q \\ a_4 &= a_3.q \\ &\vdots \\ a_{n-1} &= a_{n-2}.q \\ a_n &= a_{n-1}.q \end{aligned}$$

Multiplicando-se membro a membro as $(n-1)$ igualdades acima,

$$a_n \cdot a_{n-1} \cdot ... \cdot a_2 = a_{n-1} \cdot a_{n-2} \cdot ... \cdot a_1.q^{n-1}$$

Obtemos a equação do termo geral da Progressão Geométrica:

$$a_n = a_1.q^{n-1}, \forall n \in \mathbb{N} \tag{3.2}$$

Mais geralmente,$a_n = a_k q^{n-k}, 1 \leq k \leq n$.

Exemplo 12. *Os três primeiros termos de uma progressão geométrica são* $a_1 = \sqrt{2}$, $a_2 = \sqrt[3]{2}$ *e* $a_3 = \sqrt[6]{2}$. *Determine o quarto termo dessa progressão.*

$$a_1 = \sqrt{2} = 2^{1/2}, \quad a_2 = \sqrt[3]{2} = 2^{1/3}, \quad a_3 = \sqrt[6]{2} = 2^{1/6}$$

Calculando a razão, obtemos:

$$q = \frac{a_2}{a_1} = \frac{2^{1/3}}{2^{1/2}} = 2^{1/3-1/2} = 2^{-1/6}$$

Aplicando agora a fórmula do termo geral, 3.2, temos, para $n = 4$:

$$a_4 = a_3 \cdot q = 2^{1/6} \cdot 2^{-1/6} = 1$$

No caso de querermos somar os termos de uma progressão geométrica, assim como fizemos com a progressão aritmética, devemos levar em consideração o fato de a qual intervalo pertence a razão q:

- Se $q = 1$, tem-se que todos os termos da PG são iguais a a_1.
- Se $q \neq 1$, seja

$$S_n = a_1 + a_2 + a_3 + ... + a_n = a_1 + a_1q + a_1q^2 + a_1q^3 + ... + a_1q^{n-1}$$

Multiplicando a igualdade acima pela razão q, obtemos

$$\begin{cases} S_n & = a_1 + a_1q + a_1q^2 + a_1q^3 + ... + a_1q^{n-1} \\ qS_n & = a_1q + a_1q^2 + a_1q^3 + ... + a_1q^n \end{cases}$$

Subtraindo membro a membro as igualdades, segue

$$(q-1)S_n = qS_n - Sn = a_1q^n - a_1 = a_1(q^n - 1)$$
$$S_n = a_1.\frac{1-q^n}{1-q}, \forall n \in \mathbb{N}. \tag{3.3}$$

Definição 3.4. *Seja* $(a_n)_{n\in\mathbb{N}}$ *uma sequência de números reais. Diz-se que o número real* L *é o limite de* (a_n) *quando, para todo* $\varepsilon > 0$, *existe* $n_0 \in \mathbb{N}$ *tal que, para todo* $n \geq n_0$, *tem-se*

$$|a_n - L| < \varepsilon.$$

Nesse caso, escreve-se

$$\lim_{n\to\infty} a_n = L.$$

Proposição 2. *Se uma sequência* $(a_n)_{n\in\mathbb{N}}$ *possui limite, então esse limite é único.*

Demonstração. Suponha que (a_n) seja uma sequência tal que

$$\lim_{n\to\infty} a_n = L \quad \text{e} \quad \lim_{n\to\infty} a_n = M.$$

Pela definição de limite, dado $\varepsilon > 0$, existem $n_1, n_2 \in \mathbb{N}$ tais que:

$$|a_n - L| < \frac{\varepsilon}{2}, \quad \forall n \geq n_1,$$

e

$$|a_n - M| < \frac{\varepsilon}{2}, \quad \forall n \geq n_2.$$

Tomando $n \geq \max\{n_1, n_2\}$, temos:

$$|L - M| = |L - a_n + a_n - M| \leq |L - a_n| + |a_n - M|.$$

Logo,

$$|L - M| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.$$

Como isso vale para todo $\varepsilon > 0$, conclui-se que $|L - M| = 0$, isto é, $L = M$. ■

Definição 3.5. *Seja $(a_n)_{n\in\mathbb{N}}$ uma sequência de números reais. Define-se série associada a essa sequência pela expressão*

$$\sum_{n=1}^{\infty} a_n,$$

a qual é entendida como o limite da sequência de somas parciais (S_n), em que

$$S_n = \sum_{k=1}^{n} a_k.$$

Diz-se que a série converge quando o limite

$$\lim_{n\to\infty} S_n$$

existe e é finito. Nesse caso, esse valor é denominado soma da série, isto é,

$$\sum_{n=1}^{\infty} a_n = \lim_{n\to\infty} \sum_{k=1}^{n} a_k.$$

Proposição 3. *Seja (a_n) uma progressão geométrica de primeiro termo a_1 e razão q, com $q \neq 1$. Então a soma dos n primeiros termos é dada por*

$$S_n = a_1 \cdot \frac{1 - q^n}{1 - q}.$$

Além disso, se $|q| < 1$, então

$$\sum_{n=1}^{\infty} a_n = \frac{a_1}{1 - q}.$$

Demonstração. Seja (a_n) uma progressão geométrica de primeiro termo a_1 e razão q. Logo,

$$S_n = a_1 \cdot \frac{1-q^n}{1-q}.$$

Agora, supondo $|q| < 1$, temos $\lim_{n\to\infty} q^n = 0$. Assim,

$$\lim{}^{1}{}_{n\to\infty} S_n = \frac{a_1}{1-q}.$$

Portanto,

$$\sum_{n=1}^{\infty} a_n = \frac{a_1}{1-q}.$$

■

Exemplo 13. *Calcule a soma da série infinita*

$$1 + 2 + \tfrac{1}{3} + \tfrac{2}{5} + \tfrac{1}{9} + \tfrac{2}{25} + \ldots + \left(\tfrac{1}{3}\right)^n + 2 \cdot \left(\tfrac{1}{5}\right)^n + \ldots$$

Solução: Observa-se que a série dada pode ser decomposta na soma de duas progressões geométricas intercaladas:

$$a_n = \left(1,\ \frac{1}{3},\ \frac{1}{9},\ \ldots\right) \quad \text{e} \quad b_n = \left(2,\ \frac{2}{5},\ \frac{2}{25},\ \ldots\right).$$

A primeira é uma progressão geométrica de primeiro termo $a_1 = 1$ e razão $q = \frac{1}{3}$, enquanto a segunda possui primeiro termo $b_1 = 2$ e razão $\tilde{q} = \frac{1}{5}$.
Assim, para a primeira progressão,

$$S = \frac{1}{1-\frac{1}{3}} = \frac{1}{\frac{2}{3}} = \frac{3}{2}.$$

Para a segunda progressão,

$$\tilde{S} = \frac{2}{1-\frac{1}{5}} = \frac{2}{\frac{4}{5}} = \frac{5}{2}.$$

A soma total pode ser obtida pela soma das respectivas séries

$$S_{\text{total}} = S + \tilde{S}.$$

Logo,

$$S_{\text{total}} = \frac{3}{2} + \frac{5}{2} = 4.$$

Exemplo 14 (UFRN-1992-ADAPTADA)**.** *Em estudos sobre a dinâmica de materiais elásticos, observa-se que uma bola de borracha, ao ser abandonada de certa altura e colidir com o solo, perde parte de sua energia a cada impacto, retornando a uma fração da altura anterior.*

[1] Diz-se que L é o limite de (a_n) se, para todo $\varepsilon > 0$, existe $N \in \mathbb{N}$ tal que $n \geq N$ implica $|a_n - L| < \varepsilon$.

Considere uma bola de borracha que é abandonada em queda livre a partir de uma altura de 2 metros e que, após cada colisão com o solo, retorna a 60% da altura anterior. Desprezando a resistência do ar e considerando que o processo se repete indefinidamente até a completa dissipação do movimento, determine a distância total percorrida pela bola até parar.

Solução: Buscando entender o movimento que a bola faz, dispomos: A bola é solta de 2 m. Ela cai do repouso até o chão, percorrendo 2 m. Ao bater, sobe até 60% da altura anterior:

$$h_1 = 0,6 \times 2 = 1,2 \text{ m}.$$

Depois cai novamente de $1,2$ m, bate e sobe até 60% de $1,2m$, ou seja,

$$h_2 = 0,6 \times 1,2 = 0,72 \text{ m},$$

e assim sucessivamente. A distância total é a soma:

$$D = \underbrace{2}_{\text{primeira queda}} + \underbrace{2 \times (1,2 + 0,72 + 0,432 + \dots)}_{\text{sobe e desce após cada batida}}.$$

As alturas após a primeira batida formam uma progressão geométrica infinita com primeiro termo $a = 1,2$ e razão $q = 0,6$:

$$(1,2 \ ; \ 0,72 \ ; \ 0,432 \ ; \ \dots)$$

A soma dos termos dessa PG infinita é

$$S = \frac{a}{1-q} = \frac{1,2}{1-0,6} = \frac{1,2}{0,4} = 3.$$

A distância total percorrida D será:

$$D = 2 + 2 \times S = 2 + 2 \times 3 = 2 + 6 = 8 \text{ metros}.$$

Portanto, a bola percorre 8 metros até parar completamente.

Uma progressão geométrica pode ser descrita tanto por meio de uma relação de recorrência quanto por uma fórmula explícita. Conhecendo-se o termo inicial e a razão da progressão, é possível determinar diretamente qualquer termo da sequência, sem a necessidade de calcular todos os termos anteriores.

Proposição 4. *Uma sequência $(a_n)_{n\geq 0}$ é uma progressão geométrica se, e somente se, existem constantes A e q, com $q \neq 0$, tais que*

$$a_n = Aq^n, \forall n \geq 0.$$

Demonstração. ($\Rightarrow$)Suponha inicialmente que $(a_n)_n \geq 0$ seja uma progressão geométrica de razão q. Tomando $a_0 = A$, obtemos

$$a_1 = qa_0 = qA,$$

$$a_2 = qa_1 = q(qA) = Aq^2,$$

$$a_3 = qa_2 = q(Aq^2) = Aq^3.$$

Prosseguindo desse modo, conclui-se, por indução, que

$$a_n = Aq^n, \forall n \geq 0$$

($\Leftarrow$)Reciprocamente, suponha que existam constantes A e q, com $q \neq 0$, tais que

$$a_n = Aq^n, n \geq 0.$$

Então,

$$a_{n+1} = Aq^{n+1} = q(Aq^n) = qa_n.$$

Logo, $(a_n)_{n\geq 0}$ é uma PG. ■

Exemplo 15. *Considere uma população de bactérias que dobra de quantidade a cada hora. Suponha que inicialmente existam* 500 *bactérias. Sejam* (a_n) *a sequência cujo termo* a_n *representa os termos que representam a quantidade de bactérias após* n *horas.*

Tem-se

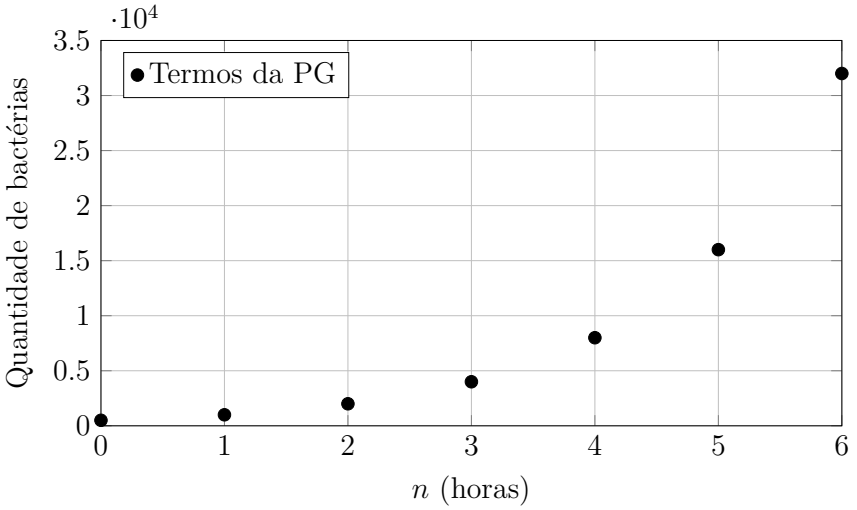
$$a_0 = 500, \quad a_1 = 1000, \quad a_2 = 2000, \quad a_3 = 4000, \ldots$$

Observe que cada termo é obtido multiplicando o anterior por 2*. Assim, a sequência é uma progressão geométrica de razão* $q = 2$*. Portanto,*

$$a_n = 500 \cdot 2^n.$$

De maneira ilustrativa, podemos acompanhar o comportamento da função exponencial no gráfico abaixo:

Figura 2 – Representação gráfica da progressão geométrica $a_n = 500 \cdot 2^n$.



Fonte: Autor, 2026.

Sequências em que os termos são expressos como diferenças entre os termos consecutivos de outra sequência, revelam uma estrutura que simplifica significativamente cálculos de somas envolvendo seus termos.

Sua importância se torna ainda mais evidente no estudo de recorrências, uma vez que diversos métodos de resolução fazem uso explícito desse tipo de cancelamento para obter expressões fechadas ou analisar o comportamento das soluções.

Definição 3.6. *Uma série é dita telescópica quando é da forma*

$$\sum_{n=1}^{\infty}(b_{n+1} - b_n)$$

ou

$$\sum_{n=1}^{\infty}(b_n - b_{n+1}),$$

para alguma sequência (b_n).

A soma dos n primeiros termos de uma série telescópica é dada por

$$\sum_{k=1}^{n}(b_k - b_{k+1}) = (b_1 - b_2) + (b_2 - b_3) + \cdots + (b_n - b_{n+1})$$

na qual observa-se o cancelamento de todos os termos intermediários, restando apenas

$$\sum_{k=1}^{n}(b_k - b_{k+1}) = b_1 - b_{n+1} \tag{3.4}$$

Exemplo 16. *Encontre o valor da soma:*

$$S = \frac{1}{1.2} + \frac{1}{2.3} + \frac{1}{3.4} + \cdots + \frac{1}{998.999} + \frac{1}{999.1000}.$$

Solução: Se observarmos, veremos que os denominadores são produtos de números consecutivos, ou seja, de modo geral

$$\frac{1}{k.(k+1)} = \frac{1}{k} - \frac{1}{k+1}$$

Daí,

$$S = \left(\frac{1}{1} - \frac{1}{2}\right) + \left(\frac{1}{2} - \frac{1}{3}\right) + \left(\frac{1}{3} - \frac{1}{4}\right) + \cdots + \left(\frac{1}{999} - \frac{1}{1000}\right)$$

e, portanto,

$$S = 1 - \frac{1}{1000} = \frac{1000-1}{1000} = \frac{999}{1000}$$

O mecanismo de cancelamento na soma acima evidencia como o comportamento global desta pode ser determinado a partir seus extremos, mesmo quando a sequência possui muitos elementos. Tal característica torna as somas telescópicas ferramentas poderosas tanto em demonstrações matemáticas quanto na análise de algoritmos iterativos.

3.3 Recorrências Lineares

Definição 3.7. (*Equação de Recorrência*) *Uma equação de recorrência é uma fórmula matemática a ser satisfeita por sequências, que exprime cada termo destas sequências a partir de um ou mais de seus termos anteriores.*

Mais precisamente, uma equação de recorrência é uma relação da forma

$$X_{n+k} = F(n, X_n, X_{n+1}, \ldots, X_{n+k-1}), \quad \forall n \in \mathbb{N},$$

em que F é uma função dada. Dizemos que tal equação tem ordem $k \in \mathbb{N}$, pois $X_s, \forall s \geq k$ depende dos k termos anteriores $X_{s-1}, \ldots, X_{s-k}$

Quando associada a condições iniciais $X_0 = a_0, \ldots X_{k-1} = a_{k-1}$, a equação de recorrência determina unicamente a sequência solução $(a_n)_{n\in\mathbb{N}}$.

No que se segue, consideraremos apenas recorrências com coeficientes constantes.

Definição 3.8 (Equação de Recorrência Linear)**.** *Uma equação de recorrência linear de ordem k, com $k \in \mathbb{N}$, é uma equação da forma*

$$X_{n+k} = C_1 X_{n+k-1} + C_2 X_{n+k-2} + \cdots + C_k X_n + b_n, \qquad \forall n \in \mathbb{N},$$

em que $C_1, C_2, \ldots, C_k$ são constantes e (b_n) é uma sequência dada.

A equação de recorrência estabelece uma relação entre termos consecutivos de uma sequência, permitindo determinar um termo a partir de termos anteriores. Entretanto, a equação, por si só, não determina uma única sequência, uma vez que diferentes

escolhas para os termos iniciais podem conduzir a diferentes sequências que satisfazem a mesma relação. Dessa forma, é necessário distinguir a relação de recorrência da sequência que a satisfaz.

Definição 3.9 (Sequências Recorrentes Lineares)**.** *Uma sequência* $(a_1, a_2, a_3, \ldots, a_n, \ldots)$ *é denominada sequência recorrente linear de ordem* k $(k \in \mathbb{N})$ *se existem constantes* $C_1, C_2, \ldots, C_k$ *e uma sequência* (b_n) *tais que*

$$a_{n+k} = \sum_{j=1}^{k} C_j \, a_{n+k-j} + b_n, \qquad \forall n \in \mathbb{N}.$$

Ou seja,

$$a_{n+k} = C_1 a_{n+k-1} + C_2 a_{n+k-2} + \cdots + C_k a_n + b_n.$$

Observe que uma tal sequência fica completamente determinada pelos seus k *primeiros termos* $a_1, a_2, \ldots, a_k$*. Observe também que sendo* (a_n) *uma sequência recorrente linear, então é solução de uma equação de recorrência da forma*

$$X_{n+k} = C_1 X_{n+k-1} + C_2 X_{n+k-2} + \cdots + C_k X_n + b_n$$

Exemplo 17. *Suponha que uma pessoa deposite R$10.000,00 em uma poupança com uma taxa de 11% ao ano, com capitalização anual. Que valor terá na poupança depois de 30 anos, não sendo feitas outras aplicações?*

Solução: Considerando P_n como a quantia na poupança depois de n anos. Como a quantia na poupança depois de n anos é igual à quantia na poupança depois de $n-1$ anos mais os juros do n-ésimo ano, podemos ver que a sequência (P_n) satisfaz

$$P_n = P_{n-1} + 0,11.P_{n-1} = (1,11) \cdot P_{n-1}$$

Daí,

$$\begin{aligned} P_1 &= (1,11)P_0 \\ P_2 &= (1,11)P_1 = (1,11)^2 P_0 \\ P_3 &= (1,11)P_2 = (1,11)^3 P_0 \\ &\vdots \end{aligned}$$

Indutivamente,

$$P_n = (1,11)P_{n-1} = (1,11)^n P_0$$

Inserindo a condição inicial $P_0 = 10.000,00$, obtemos a fórmula

$$P_n = (1,11)^n 10.000.$$

Tomando $n = 30$ na fórmula anterior, obtemos

$$P_{30} = (1,11)^{30} 10.000 = R\$228.922,97.$$

3.3.1 Recorrências de 1ª ordem

A progressão geométrica $1, 3, 9, 27, ...$ satisfaz $a_{n+1} = 3a_n, \forall n \geq 1$.

Essa relação de recorrência, no entanto, não define exclusivamente uma única progressão geométrica, tendo em vista que também pode definir as seguintes sequências, por exemplo, $2, 6, 18, 54, 162, ...$ e $4, 12, 36, 108, ...$ Para tanto, se quisermos identificar e estabelecer somente uma progressão definida por $q = 3$, devemos conhecer um dos termos da sequência. Sendo assim, se $a_{n+1} = 3a_n, \forall n \geq 0, \text{com} \quad a_1 = 1$, então temos de forma única a progressão geométrica que tomamos por referência.

Para a progressão geométrica apresentada, os primeiros 6 termos são

$$\begin{aligned} a_1 &= 1 \\ a_2 &= 3a_1 = 3 \cdot (1) \\ a_3 &= 3a_2 = 3^2 \cdot (1) \\ a_4 &= 3a_3 = 3^3 \cdot (1) \\ a_5 &= 3a_4 = 3^4 \cdot (1) \\ a_6 &= 3a_{a_5} = 3^5 \cdot (1) \end{aligned}$$

Podemos definir facilmente com base no observado que, para qualquer $n \geq 1$, temos um padrão formulado por $a_n = 1 \cdot 3^n$, sendo pois esta, solução particular da recorrência linear apresentada.

Exemplo 18. *Dada a sequência apresentada anteriormente, determine o seu décimo termo.*

Solução: Não é necessário calcular sucessivamente todos os termos anteriores para determinar o décimo termo da sequência. Observemos que a solução geral da recorrência é $a_n = 1 \cdot 3^n$, logo, é só substituirmos $n = 10$ e obtemos $a_{10} = 1 \cdot 3^{10} = 59049$.

É do nosso interesse que consigamos (quase sempre) estabelecer uma expressão explícita para o termo geral de uma sequência, denominada fórmula fechada, como a do exemplo anterior, em que possa ser possível definir qualquer que seja o termo de uma sequência somente a partir da posição n do termo e não nos tornarmos dependentes de seus termos anteriores.

Exemplo 19. *Em uma colônia de bactérias, observa-se que, a cada hora, o número de indivíduos dobra e, em seguida, 3 novas bactérias são introduzidas. No instante inicial ($t = 0$ horas), a população é de 5 bactérias. Seja P_n o número de bactérias após n horas ($n \geq 0$). A evolução da população obedece à recorrência:*

$$P_n = 2P_{n-1} + 3, \quad \textit{para } n \geq 1,$$

com $P_0 = 5$.Determine a população após 15 horas, isto é, calcule P_{15}.

Sem uma fórmula explícita, calcular o 15º termo da recorrência $P_n = 2P_{n-1} + 3$ exigiria determinar todos os termos anteriores, um a um, a partir de P_0. Esse processo iterativo demanda 15 etapas, cada uma com multiplicação por 2 e adição de 3. Como os valores crescem exponencialmente (dobram a cada passo), os números crescem rapidamente, aumentando o risco de erros aritméticos. Além disso, o método é ineficiente e trabalhoso para a determinação de termos associados a índices maiores. Já a fórmula fechada $P_n = 2^{n+3} - 3$ fornece o resultado diretamente, com uma única operação de exponenciação, sendo muito mais prática e confiável.

Solução: Expandimos a recorrência sucessivamente até alcançar P_1:

$$\begin{aligned} P_n &= 2P_{n-1} + 3 \\ &= 2(2P_{n-2} + 3) + 3 = 2^2 P_{n-2} + 2 \cdot 3 + 3 \\ &= 2^2(2P_{n-3} + 3) + 2 \cdot 3 + 3 = 2^3 P_{n-3} + 3(2^2 + 2 + 1) \\ &\vdots \\ &= 2^k P_{n-k} + 3(2^{k-1} + 2^{k-2} + \cdots + 2 + 1) \\ &= 2^k P_{n-k} + 3(2^k - 1). \end{aligned}$$

Fazemos $k = n$ para obter $P_{n-n} = P_0$:

$$P_n = 2^n P_0 + 3(2^n - 1).$$

Substituindo $P_0 = 5$:

$$P_n = 5 \cdot 2^n + 3(2^n - 1) = 8 \cdot 2^n - 3 = 2^{n+3} - 3.$$

Para $n = 15$:

$$P_{15} = 2^{18} - 3.$$

Calculando $2^{10} = 1024$, $2^{18} = 2^{10} \cdot 2^8 = 1024 \cdot 256 = 262144$. Portanto:

$$P_{15} = 262144 - 3 = 262141.$$

Portanto, após 15 horas, a população será de 262 141 bactérias.

Exemplo 20. *Resolva a recorrência $X_{n+1} = 2X_n$ com a condição inicial $X_1 = \dfrac{2}{5}$.*

Solução: Seja (a_n) uma solução da equação acima. Então

$$\begin{aligned} a_1 &= \frac{2}{5} \\ a_2 &= 2a_1 \\ &\vdots \\ a_n &= 2a_{n-1} \end{aligned}$$

Multiplicando todas as igualdades membro a membro, obtemos

$$a_n = a_1 \cdot 2^{n-1} = \frac{2}{5} \cdot 2^{n-1}.$$

Portanto, a solução da recorrência $a_{n+1} = 2a_n$ com $a_1 = \frac{2}{5}$ é

$$a_n = \frac{2}{5} \cdot 2^{n-1}.$$

Quando obtemos uma solução de uma recorrência, podemos calcular diretamente o termo desejado sem a necessidade de calcular os termos anteriores. Dadas as condições iniciais, temos uma solução particular.

As recorrências lineares de qualquer ordem distinguem-se fundamentalmente por três aspectos: homogeneidade, constância dos coeficientes e escolha do índice inicial.

- **Homogênea**: quando o termo independente é nulo, $b_n = 0$.
- **Não homogênea**: quando existe pelo menos um termo independente não nulo, $b_n \neq 0$.
- **Coeficientes constantes**: quando $c_n = c$ e $d_n = d$ para todo n, com $c, d \in \mathbb{R}$ fixos.

  $$a_n = c \cdot a_{n-1} + b, \quad n \geq 1.$$

- **Coeficientes variáveis**: quando c_n ou b_n dependem efetivamente de n.

  $$a_n = c_n a_{n-1} + b_n, \quad n \geq 1.$$

  Exemplo:

  $$a_n = n \cdot a_{n-1}, \quad a_0 = 1.$$

  Solução: $a_n = n!$ (fatorial de n).

Definição 3.10 (Solução Geral de uma Equação de Recorrência Linear)**.** *Um conjunto de sequências $(a_1, a_2, a_3, \ldots, a_n, \ldots)$ é denominado solução geral da equação de recorrência linear de ordem k, $k \in \mathbb{N}$*

$$X_{n+k} = \sum_{j=1}^{k} C_j \, X_{n+k-j} + b_n, \qquad \forall n \in \mathbb{N},$$

se existem k sequências $u_n^{(1)}, u_n^{(2)}, \ldots, u_n^{(k)}$, soluções da equação homogênea associada

$$X_{n+k} = \sum_{j=1}^{k} C_j \, X_{n+k-j},$$

e uma sequência $a_n^{(p)}$, solução particular da equação não homogênea, tais que

$$a_n = \lambda_1 u_n^{(1)} + \lambda_2 u_n^{(2)} + \cdots + \lambda_k u_n^{(k)} + a_n^{(p)}.$$

Em outras palavras,

$$a_n = \sum_{j=1}^{k} \lambda_j \, u_n^{(j)} + a_n^{(p)}.$$

Fixados os k primeiros termos $a_1, a_2, \ldots, a_k$ (condições iniciais), as constantes arbitrárias $\lambda_1, \lambda_2, \ldots, \lambda_k$ são unicamente determinadas, fornecendo assim a única sequência que satisfaz a equação e as condições iniciais dadas.

Exemplo 21.

$$a_n = 3 \cdot a_{n-1}, \quad n \geq 1, \quad a_1 = 2.$$

Solução: $a_n = 2 \cdot 3^n$. Nesse caso, cada termo é um múltiplo do anterior.

Exemplo 22.

$$a_n = 3 \cdot a_{n-1} + 5, \quad n \geq 1, \quad a_1 = 2.$$

Solução: A solução homogênea $C \cdot 3^n$ resolve a equação na ausência da constante 5, ou seja, ela foi feita para satisfazer $a_n = 3a_{n-1}$. Quando substituímos essa parcela na equação original $a_n = 3a_{n-1} + 5$, ocorre um cancelamento total entre o lado esquerdo e a parte $3a_{n-1}$ do lado direito, restando a igualdade $0 = 5$, que é um absurdo. Portanto, a solução homogênea, por si só, é incapaz de gerar esse valor constante diferente de zero. Para contornar essa impossibilidade, é necessário acrescentar uma parcela extra que não se cancele ao ser deslocada, mas que produza justamente a constante 5 como resultado dessa diferença. Denominamos tal parcela extra de solução particular da recorrência. Como a constante 5 é um valor fixo adicionado a cada passo da recorrência, procuramos uma solução particular constante A, pois a diferença $A - 3A = -2A$ é também uma constante, que pode ser ajustada para ser exatamente igual a 5. Assim, $A = -\frac{5}{2}$.

Logo, a solução geral é dada por $a_n = a_n^{(h)} + a_n^{(p)}$. Determinando a particular $a_n^{(p)} = A$ (constante), substitui-se:

$$A = 3A + 5 \quad \Longrightarrow \quad A = -\frac{5}{2}.$$

Logo, $a_n = C \cdot 3^n - \frac{5}{2}$. Usando a condição inicial $a_1 = 2$, obtém-se $C = \frac{9}{2}$, resultando em:

$$a_n = \frac{9}{2} \cdot 3^{n-1} - \frac{5}{2}.$$

Fazendo para alguns valores particulares, encontramos

$$a_2 = \frac{9}{2} \cdot 3^{2-1} - \frac{5}{2} = \frac{27}{2} - \frac{5}{2} = 11$$
$$a_3 = \frac{9}{2} \cdot 3^{3-1} - \frac{5}{2} = \frac{81}{2} - \frac{5}{2} = 38$$
$$a_4 = \frac{9}{2} \cdot 3^{4-1} - \frac{5}{2} = \frac{729}{2} - \frac{5}{2} = 362$$

Exemplo 23.

$$a_n = 4a_{n-1} - 2, \quad a_1 = 3.$$

Neste caso, a solução explícita é:

$$a_n = 4^{n-1} \cdot 3 - 2 \cdot \frac{4^{n-1} - 1}{4 - 1} = 3 \cdot 4^{n-1} - \frac{2}{3}\Big(4^{n-1} - 1\Big).$$

Podemos ainda, a partir de observações, estabelecer que toda progressão aritmética é uma recorrência linear de primeira ordem e que toda progressão geométrica é uma recorrência linear de primeira ordem homogênea.

Exemplo 24. *Um investidor aplica R$ 5.000,00 em um fundo que rende 3% de juros ao mês, sem qualquer depósito ou retirada adicional. Seja S_n o saldo (em reais) após n meses, com $n \geq 0$. A evolução do saldo é dada pela recorrência linear homogênea:*

$$S_n = 1{,}03 \cdot S_{n-1}, \quad n \geq 1,$$

com condição inicial $S_0 = 5000$. Determine quanto o investidor terá após 7 meses de aplicação.

Solução: Observe que

$$S_0 = 5000, S_1 = 1,03.S_0, S_2 = 1,03.S_1, \ldots$$

Indutivamente

$$S_n = 1,03.S_{n-1}, \forall n \geq 1.$$

Multiplicando membro a membro nas igualdades e realizando as simplificações necessárias, temos como solução geral

$$S_n = 5000.(1,03)^n.$$

Como queremos encontrar o valor que o investidor terá após 7 meses, devemos ter $n = 7$, ou seja

$$S_7 = 5000.(1,03)^7 \Rightarrow S_7 \approx 6149.37$$

Até o momento, foram estudadas recorrências lineares homogêneas, nas quais cada termo da sequência é determinado exclusivamente em função do termo anterior. Entretanto, em diversas situações práticas, o comportamento de uma sequência pode ser influenciado também por fatores externos ao processo recorrente, introduzindo termos independentes na relação de recorrência.

Essas recorrências desempenham papel fundamental na modelagem de fenômenos em que há acréscimos, perdas ou perturbações ao longo do processo, como em modelos econômicos, crescimento populacional, análise de algoritmos, etc..

Sendo a recorrência linear de primeira ordem uma equação da forma

$$X_{n+1} = dX_n + b_n, \quad \forall n \in \mathbb{N}, \quad \text{onde} \quad d \in \mathbb{R}$$

Quando $b_n \neq 0$, a recorrência é denominada não homogênea, pois a determinação de cada termo depende não apenas do termo anterior da sequência, mas também de um termo externo ao processo recorrente.

Proposição 5. *A solução (a_n) da recorrência linear de primeira ordem não homogênea*

$$X_{n+1} = r \cdot X_n + b_n, \qquad n \geq 0,$$

com condição inicial $X_0 = \alpha$, é dada por

$$a_n = \alpha r^n + \sum_{k=0}^{n-1} r^{n-1-k} b_k, \forall n \geq 1.$$

Demonstração. Partindo da recorrência

$$X_{n+1} = rX_n + b_n,$$

se (a_n) é solução, então

$$a_1 = ra_0 + b_0 = r\alpha + b_0,$$

$$a_2 = ra_1 + b_1 = r(r\alpha + b_0) + b_1 = r^2\alpha + rb_0 + b_1,$$

$$a_3 = ra_2 + b_2 = r(r^2\alpha + rb_0 + b_1) + b_2 = r^3\alpha + r^2b_0 + rb_1 + b_2.$$

Prosseguindo o processo, observa-se o padrão

$$a_n = r^n\alpha + r^{n-1}b_0 + r^{n-2}b_1 + \cdots + rb_{n-2} + b_{n-1}.$$

Em notação de somatório, obtém-se

$$a_n = r^n\alpha + \sum_{k=0}^{n-1} r^{n-1-k} b_k.$$

Portanto, esta é a solução particular da recorrência linear de primeira ordem não homogênea. ■

Nesse sentido, diversos problemas clássicos podem ser interpretados por meio de relações de recorrência não homogêneas, permitindo a obtenção de fórmulas explícitas para a determinação de seus termos. Entre esses problemas, destaca-se o Problema da Pizza de Steiner, que consiste em determinar o número máximo de regiões em que um círculo pode ser dividido a partir da realização sucessiva de cortes retos.

Exemplo 25 (Problema da Pizza de Steiner)**.** *Deseja-se determinar o número máximo de regiões em que uma pizza pode ser dividida utilizando-se n cortes retos, dadas as seguintes condições:*

i) cada novo corte intersecta todos os anteriores;

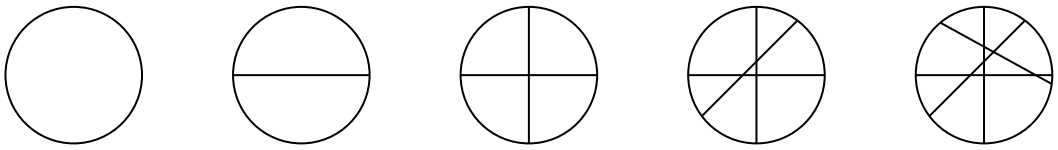
ii) não existem três cortes concorrentes em um mesmo ponto;

iii) todos os cortes atravessam completamente a pizza.

Quantas regiões podem ser obtidas com n cortes?

Solução: Inicialmente, analisemos alguns casos particulares:

Figura 3 – Primeiros casos do Problema da Pizza de Steiner e o número máximo de regiões obtidas com cortes retos.



$P_0 = 1$ $P_1 = 2$ $P_2 = 4$ $P_3 = 7$ $P_4 = 11$

Fonte: Autor, 2026.

A partir dos casos particulares ilustrados, observamos que os valores obtidos são

$$P_0 = 1, \quad P_1 = 2, \quad P_2 = 4, \quad P_3 = 7, \quad P_4 = 11.$$

Notamos que os acréscimos de uma etapa para a seguinte são:

$$P_1 - P_0 = 1, \quad P_2 - P_1 = 2, \quad P_3 - P_2 = 3, \quad P_4 - P_3 = 4.$$

Esses incrementos formam a sequência $1, 2, 3, 4, \ldots$, sugerindo que, ao passar de $n-1$ cortes para n cortes, o número de novas regiões adicionadas é exatamente n. Ao fazer o n-ésimo corte, ele intersecta os $n-1$ cortes anteriores em $n-1$ pontos distintos (condição (i) e (ii)). Esses pontos dividem o novo corte em n segmentos. Cada um desses segmentos está contido em uma região diferente e, ao ser traçado, divide essa

região em duas, aumentando em uma unidade o número total de regiões. Portanto, o n-ésimo corte acrescenta exatamente n novas regiões. Assim, o número de cortes satisfaz a recorrência

$$P_{n+1} = P_n + (n+1), n \geq 0$$

com condição inicial

$$P_0 = 1.$$

Fazendo uso da proposição 5, obtemos diretamente

$$P_n = P_0 + \sum_{k=1}^{n} k.$$

Como $P_0 = 1$ e

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2},$$

segue que

$$P_n = 1 + \frac{n(n+1)}{2}.$$

Portanto, o número máximo de regiões obtidas com n cortes é dado por

$$P_n = \frac{n^2 + n + 2}{2}.$$

Exemplo 26 (Crescimento Populacional com Migração)**.** *Considere uma cidade cuja população cresce anualmente a uma taxa de* 3%*. Além do crescimento natural, a cidade recebe, a cada ano, aproximadamente* 200 *novos habitantes provenientes de migração.*

Se P_n representa a população da cidade no n-ésimo ano, determine uma relação de recorrência capaz de modelar a evolução populacional ao longo dos anos.

Solução: Seja P_n a população da cidade no n-ésimo ano. Como a população cresce 3% ao ano, o número de habitantes é multiplicado pelo fator $1,03$. Além disso, a cidade recebe anualmente 200 novos moradores provenientes da migração.

Assim, a evolução populacional é modelada pela recorrência

$$P_{n+1} = 1,03P_n + 200, n \geq 0$$

com condição inicial P_0, que representa a população da cidade no instante inicial.

Observamos que essa é uma recorrência linear de primeira ordem não homogênea. Pela Proposição 5, sua solução é dada por

$$P_n = P_0(1,03)^n + \sum_{k=0}^{n-1} (1,03)^{n-1-k} \cdot 200.$$

Ou seja,

$$P_n = P_0(1,03)^n + 200\sum_{k=0}^{n-1}(1,03)^{n-1-k}.$$

A soma obtida acima é a soma dos n primeiros termos de uma progressão geométrica de razão $1,03$. Assim,

$$\sum_{k=0}^{n-1}(1,03)^{n-1-k} = \frac{(1,03)^n - 1}{1,03 - 1}.$$

Logo,

$$P_n = \left(P_0 + \frac{200}{0,03}\right)(1,03)^n - \frac{200}{0,03}.$$

Denotando

$$a = -\frac{200}{0,03},$$

a solução pode ser escrita de forma equivalente como

$$P_n = (P_0 - a)(1,03)^n + a.$$

Portanto,

$$P_n = (P_0 - a)(1,03)^n + a, \qquad \alpha = -\frac{200}{0,03}.$$

Essa expressão evidencia que a evolução da população é composta por duas parcelas: uma dependente da população inicial P_0, que cresce exponencialmente ao longo do tempo, e outra constante, representada por α, associada ao efeito acumulado da migração anual.

Exemplo 27 (TORRE DE HANÓI)**.** *A Torre de Hanói é um famoso quebra-cabeça criado em 1883 pelo matemático francês Édouard Lucas.*

Figura 4 – Édouard Lucas (1842–1891).



Fonte: Google Imagens, 2026.

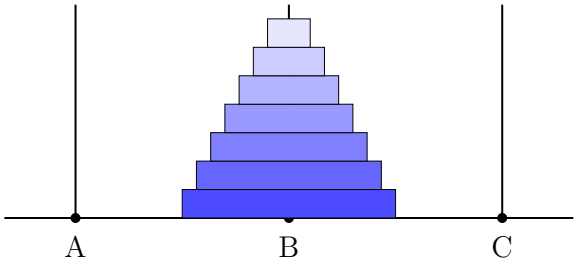
O jogo consiste em transferir todos os discos de uma haste para outra, utilizando uma haste auxiliar, obedecendo a duas regras fundamentais:

i. Mover apenas um disco por vez.

ii. Nunca posicionar um disco maior sobre um disco menor.

Na versão original proposta por Lucas, o quebra-cabeça com sete discos representava uma adaptação simplificada de uma antiga lenda indiana relacionada ao templo de Benares, cidade sagrada da Índia. Segundo essa narrativa, existia no templo uma placa de bronze com três hastes de diamante. Em uma delas, o deus Brahma teria colocado, no instante da criação do mundo, 64 discos de ouro puro organizados em ordem decrescente de tamanho, com o maior na base e o menor no topo. Os monges do templo teriam como missão transferir todos os discos para outra haste, seguindo rigorosamente as regras do jogo. Para completar a tarefa, seriam necessários exatamente $2^{64} - 1$ *movimentos, isto é,* 18.446.744.073.709.551.615 *movimentos. Considerando a realização de um movimento por segundo, a conclusão do processo demandaria bilhões de anos. De acordo com a lenda, quando o último movimento fosse executado, o mundo chegaria ao fim.*

Figura 5 – Ilustração do problema da Torre de Hanói com 7 discos.



Fonte: Autor, 2026.

Solução: Consideremos inicialmente alguns casos:

Com apenas um disco, basta movê-lo da haste de origem para a de destino, realizando apenas um movimento. Com dois discos, são necessários três movimentos para completar a tarefa: Mover o disco menor (topo) da haste A para a haste B. Mover o disco maior (base) de A para C. Mover o disco menor de B para C. Partindo para três discos, são necessários 7 movimentos, pois, movemos os 2 discos superiores (3 movimentos) para a haste B, depois o maior disco (1 movimento) para C, e novamente os 2 discos (3 movimentos) de B para C, isto é, $3 + 1 + 3 = 7$ movimentos.

Imaginemos agora o caso em que temos quatro discos. Observe que com três peças fazemos no mínimo sete movimentos, mas como ainda temos mais um disco falta agora movimentar esse disco para outra haste. Logo que fizermos esse movimento com o quarto disco ainda precisaremos de mais sete movimentos. Logo o total de movimentos será de $7 + 1 + 7$, totalizando quinze movimentos, no mínimo. Seguimos o mesmo pensamento para o caso de cinco discos. Para movimentar quatro discos são necessários 15 movimentos e ainda teremos o quinto disco para movimentar e depois trazer os outros quatro discos menores para a mesma haste novamente com 15 movimentos, então serão necessários, no mínimo, $15 + 1 + 15$, totalizando 31 movimentos.

Doravante seja a_n o número mínimo de movimentos para transferir n discos, então precisamos analisar que, com $n-1$ discos, fazemos, no mínimo, a_{n-1} movimentos. Como sobrou um disco ainda, precisaremos de mais um movimento e em seguida precisamos movimentar os $n-1$ discos, que representamos por a_{n-1}. Podemos concluir que para movimentar n discos precisamos de

$$a_n = a_{n-1} + 1 + a_{n-1}, \quad \text{isto é,} \quad a_n = 2a_{n-1} + 1$$

Observa-se que a recorrência

$$a_{n+1} = 2a_n + 1, \qquad a_1 = 1,$$

é linear de primeira ordem não homogênea. Pela Proposição 5,

$$a_n = 2^{n-1} + \sum_{k=1}^{n-1} 2^{n-1-k}.$$

Como o somatório corresponde à soma dos $n-1$ primeiros termos de uma progressão geométrica de razão 2,

$$\sum_{k=1}^{n-1} 2^{n-1-k} = 2^{n-1} - 1,$$

segue que

$$a_n = 2^{n-1} + 2^{n-1} - 1 = 2^n - 1.$$

A sequência cresce exponencialmente, o que torna o problema rapidamente inviável para valores grandes de n quando executado manualmente, o que demonstra que o número mínimo de movimentos cresce exponencialmente com o número de discos.

A passagem das recorrências lineares de primeira para as de segunda ordem amplia a estrutura matemática utilizada na definição das sequências, incorporando a dependência de dois termos anteriores.

3.3.2 Recorrências Lineares de 2ª ordem

As recorrências lineares de segunda ordem constituem uma ampliação natural das recorrências de primeira ordem, uma vez que cada termo da sequência passa a depender não apenas do termo imediatamente anterior, mas dos dois termos precedentes. Esse tipo de estrutura permite modelar fenômenos mais complexos, envolvendo relações de dependência sucessiva e padrões de crescimento que não podem ser descritos adequadamente por recorrências de ordem inferior.

Nesse contexto, torna-se fundamental compreender a estrutura dessas recorrências, bem como os métodos utilizados na determinação de suas soluções gerais, possibilitando a obtenção de expressões explícitas para os termos da sequência e a análise de seu comportamento.

Uma equação de recorrência linear de 2ª ordem com coeficientes constantes é uma equação da forma

$$X_n = -pX_{n-1} - qX_{n-2} + f_n$$

para $p, q \in \mathbb{R}$ e $f = (f_1, f_2, ...)$.

Iniciaremos estudando os casos em que $f(n) = 0$, ou seja, uma recorrência homogênea, daí escrevemos

$$X_n + pX_{n-1} + qX_{n-2} = 0$$

Portanto, supondo (a_n) solução dessa recorrência, tem-se

$$a_n + pa_{n-1} + qa_{n-2} = 0$$

Proposição 6. *Se as raízes da equação $\lambda^2 + p\lambda + q = 0$ são λ_1 e λ_2 com $\lambda \in \mathbb{C}$, então (a_n) dada por*

$$a_n = C_1\lambda_1^n + C_2\lambda_2^n, n \geq 1,$$

é solução da recorrência linear homogênea de segunda ordem $X_{n+2} + pX_{n+1} + qX_n = 0$, quaisquer que sejam as constantes C_1 e C_2.

Demonstração. De fato, substituindo $a_n = C_1\lambda_1^n + C_2\lambda_2^n$ em $a_{n+2} + pa_{n+1} + qa_n = 0$, obtemos:

$$\begin{aligned} a_{n+2} + pa_{n+1} + qa_n &= C_1\lambda_1^{n+2} + C_2\lambda_2^{n+2} + p(C_1\lambda_1^{n+1} + C_2\lambda_2^{n+1}) + q(C_1\lambda_1^n + C_2\lambda_2^n) \\ &= C_1\lambda_1^n \underbrace{(\lambda_1^2 + p\lambda_1 + q)}_{=0} + C_2\lambda_2^n \underbrace{(\lambda_2^2 + p\lambda_2 + q)}_{=0} = 0 \end{aligned}$$

■

Proposição 7. *Se p e q são números reais e não nulos, então todas as soluções (a_n) da recorrência linear homogênea de segunda ordem $X_{n+2} + pX_{n+1} + qX_n = 0$ são da forma*

$$a_n = C_1\lambda_1^n + C_2\lambda_2^n$$

em que C_1 e C_2 são constantes, se as raízes da equação $\lambda^2 + p\lambda + q = 0$, denominada equação característica da recorrência*, são distintas.*

Demonstração. Se (a_n) é uma solução da relação de recorrência $X_{n+2} + pX_{n+1} + qX_n = 0$, com $p, q \neq 0$, segue que

$$a_{n+2} + pa_{n+1} + qa_n = 0, \forall n \in \mathbb{N}.$$

Considere a sequência (b_n), definida por $b_n = a_n - (C_1\lambda_1^n + C_2\lambda_2^n)$, mostraremos que $b_n = 0, \forall n \in \mathbb{N}$, escolhendo C_1, C_2 adequadamente. De fato,

$$\begin{aligned} &b_{n+2} + pb_{n+1} + qb_n = a_{n+2} - (C_1\lambda_1^{n+2} + C_2\lambda_2^{n+2}) + p[a_{n+1} - (C_1\lambda_1^{n+1} + C_2\lambda_2^{n+1})] \\ &+ q[a_n - (C_1\lambda_1^n + C_2\lambda_2^n)] = \underbrace{(a_{n+2} + pa_{n+1} + qa_n)}_{=0} - C_1\lambda_1^n \underbrace{(\lambda_1^2 + p\lambda_1 + q)}_{=0} \\ &- C_2\lambda_2^n \underbrace{(\lambda_2^2 + p\lambda_2 + q)}_{=0} = 0 \end{aligned}$$

Ademais, $b_1 = a_1 - (C_1\lambda_1 + C_2\lambda_2)$ e $b_2 = a_2 - (C_1\lambda_1^2 + C_2\lambda_2^2)$. Afirmamos que existem constantes C_1 e C_2 tais que $b_1 = b_2 = 0$. Com efeito,

$$\begin{cases} b_1 = 0 \\ b_2 = 0 \end{cases} \iff \begin{cases} a_1 - (C_1\lambda_1 + C_2\lambda_2) = 0 \\ a_2 - (C_1\lambda_1^2 + C_2\lambda_2^2) = 0 \end{cases} \iff \begin{cases} C_1\lambda_1 + C_2\lambda_2 = a_1 \\ C_1\lambda_1^2 + C_2\lambda_2^2 = a_2 \end{cases}$$

Ora, como λ_1 e λ_2 são não nulos, podemos resolver o sistema acima para obter:

$$C_1 = \frac{\lambda_1^2 a_1 - \lambda_2 a_2}{\lambda_1\lambda_2(\lambda_2 - \lambda_1)} \qquad C_2 = \frac{\lambda_1 a_2 - \lambda_1^2 a_1}{\lambda_1\lambda_2(\lambda_2 - \lambda_1)}$$

Com essas constantes, tem-se que $b_1 = b_2 = 0$. Como $b_{n+2} + pb_{n+1} + qb_n = 0, \forall n \in \mathbb{N}$, segue que $b_n = 0, \forall n \in \mathbb{N}$. Portanto,

$$b_n = a_n - (C_1\lambda_1^n + C_2\lambda_2^n) = 0, \forall n \in \mathbb{N} \Rightarrow a_n = C_1\lambda_1^n + C_2\lambda_2^n = 0, \forall n \in \mathbb{N}.$$

■

Exemplo 28. *Considere a recorrência $X_{n+2} + X_{n+1} - 2X_n = 0$, $n \geq 0$, sendo $X_0 = 1$ e $X_1 = 4$. Qual a sua solução?*

Solução: A equação característica dessa recorrência é $\lambda^2 + \lambda - 2 = 0$. Fatorando a equação obtemos $(\lambda + 2)(\lambda - 1) = 0$. Segue que -2 e 1 são as raízes. Pela proposição 6, as soluções da recorrência são as sequências (a_n) tais que

$$a_n = C_1 + C_2(-2)^n, \forall n \in \mathbb{N},$$

onde C_1 e C_2 são constantes reais. Dadas as condições iniciais $a_0 = 1$ e $a_1 = 4$, podemos determinar os valores das constantes C_1 e C_2 por meio do seguinte sistema linear

$$\begin{cases} C_1 + C_2 & = 1 \\ C_1 - 2C_2 & = 4 \end{cases}$$

Portanto a solução particular da recorrência é

$$a_n = 2 - (-2)^n$$

Exemplo 29 (GRIMALDI)**.** *Em muitas linguagens de programação podemos considerar que as expressões aritméticas válidas, sem parênteses, estão formadas pelos dígitos $0, 1, 2, ..., 9$ e os símbolos das operações binárias $+, *, /$. Por exemplo, $3 + 4$ e $3 * 5$ são expressões aritméticas válidas; $8 + *9$ não é. Neste caso, $2 + 3 * 5 = 17$, já que há uma hierarquia de operações: a multiplicação e a divisão se realizam antes que a soma. As operações de mesmo nível se realizam em sua ordem de aparição, ao analisar as expressões da esquerda à direita. Para $n \in \mathbb{Z}^+$, seja a_n o número de expressões aritméticas válidas formadas por n símbolos. Então $a_1 = 10$, posto que as expressões aritméticas com um símbolo são os 10 dígitos. A continuação, $a_2 = 100$, com as expressões $00, 01, ...09, 10, 11, ..., 99$. Se $n \geq 3$, consideramos dois casos para obtermos uma relação de recorrência para a_n:*

- *Se x é uma expressão aritmétia de $n - 1$ símbolos, o último destes deve ser um dígito. Se adicionarmos outro dígito à direita de x, obtemos $10a_{n-1}$ expressões aritméticas de n símbolos, onde os últimos símbolos são dígitos.*

- *Seja y uma expressão de $n-2$ símbolos. Para obter uma expressão com n símbolos(que não se contaram no caso 1), adicionamos à direita de y uma das 29 expressões com dois símbolos $+1, ..., +9. + 0, *1, ... * 9, *0, /1, .../9$*[2].

Destes dois casos, obtemos a recorrência

$$a_n = 10a_{n-1} + 29a_{n-2}$$

onde $n \geq 3$ e as condições iniciais são $a_1 = 10$ e $a_2 = 100$.

Solução: A equação característica da recorrência é $\lambda^2 - 10\lambda - 29 = 0$. Encontrando suas raízes

$$\lambda_1 = \frac{-(-10) + \sqrt{10^2 - 4 \cdot 1 \cdot (-29)}}{2} = \frac{10 + \sqrt{216}}{2} = \frac{10 + 6\sqrt{6}}{2} = 5 + 3\sqrt{6},$$

$$\lambda_2 = \frac{-(-10) - \sqrt{10^2 - 4 \cdot 1 \cdot (-29)}}{2} = \frac{10 - \sqrt{216}}{2} = \frac{10 - 6\sqrt{6}}{2} = 5 - 3\sqrt{6}.$$

Logo, a solução geral é da forma $a_n = C_1 \cdot (5 + 3\sqrt{6})^n + C_2 \cdot (5 - 3\sqrt{6})^n$. Utilizando as condições iniciais para determinar as constantes C_1 e C_2, segue que

$$\begin{cases} C_1(5 + 3\sqrt{6}) + C_2(5 - 3\sqrt{6}) = 10, \\ C_1(5 + 3\sqrt{6})^2 + C_2(5 - 3\sqrt{6})^2 = 100. \end{cases}$$

Como $(5 \pm 3\sqrt{6})^2 = 79 \pm 30\sqrt{6}$, para $A = C_1 + C_2$ e $B = C_1 - C_2$, obtemos

$$\begin{cases} 5A + 3\sqrt{6}\, B = 10, \\ 79A + 30\sqrt{6}\, B = 100. \end{cases}$$

Daí segue que $A = 0$ e $B = \dfrac{5\sqrt{6}}{9}$ Assim, $C_1 = \dfrac{A + B}{2} = \dfrac{5\sqrt{6}}{18}$, e $C_2 = \dfrac{A - B}{2} = -\dfrac{5\sqrt{6}}{18}$.

Portanto,

$$\boxed{a_n = \frac{5\sqrt{6}}{18}\left[(5 + 3\sqrt{6})^n - (5 - 3\sqrt{6})^n\right], \qquad n \geq 0.}$$

No estudo das recorrências lineares homogêneas de segunda ordem com coeficientes constantes, a obtenção da solução geral está diretamente relacionada às raízes da equação característica associada. Nos casos anteriormente analisados, em que a equação característica possui duas raízes reais distintas, a solução pode ser expressa como combinação linear de potências dessas raízes.

[2] não estamos considerando /0 por não ser uma expressão válida

Entretanto, existem situações em que a equação característica apresenta duas raízes iguais. Nessa circunstância, a forma da solução geral necessita ser reformulada, pois as duas soluções obtidas a partir da raiz repetida coincidem, não fornecendo duas soluções linearmente independentes, como é necessário para uma recorrência de segunda ordem. Assim, torna-se necessário introduzir uma nova estrutura para a solução geral da recorrência, conforme apresentado na proposição a seguir.

Proposição 8. *Se a equação característica da recorrência $X_{n+2} + pX_{n+1} + qX_n = 0$, com $p, q \in \mathbb{R}^*$, possui duas raízes reais iguais $\lambda_1 = \lambda_2$, então (a_n) dada por*

$$a_n = C_1\lambda^n + C_2 n\lambda^n, \forall n \in \mathbb{N} \quad com \quad C_1, C_2 \in \mathbb{R}$$

é uma solução dessa equação, para quaisquer $C_1, C_2 \in \mathbb{R}$

Demonstração. A equação característica da recorrência $a_{n+2} + pa_{n+1} + qa_n = 0$ é $\lambda^2 + p\lambda + q = 0$. Se essa equação tem apenas uma raiz real, então $\lambda = -\frac{p}{2}$. Substituindo $a_n = C_1\lambda^n + C_2 n\lambda^n$ na recorrência $X_{n+2} + pX_{n+1} + qX_n = 0$, obtemos

$$\begin{aligned}
&a_{n+2} + pa_{n+1} + qa_n = \\
&= C_1\lambda^{n+2} + C_2(n+2)\lambda^{n+2} + p(C_1\lambda^{n+1} + C_2(n+1)\lambda^{n+1}) + q(C_1\lambda^n + C_2\lambda^n) \\
&= C_1\lambda^n \underbrace{(\lambda^2 + p\lambda + q)}_{=0} + C_2 n\lambda^n \underbrace{(\lambda^2 + p\lambda + q)}_{=0} + C_2\lambda^{n+1} \underbrace{(2\lambda + p)}_{=0} = 0
\end{aligned}$$

■

Proposição 9. *Se a equação característica da recorrência $X_{n+2} + pX_{n+1} + qX_n = 0$, com p e q reais não nulos, possui apenas uma raiz real λ, então todas as soluções (a_n) dessa recorrência são da forma*

$$a_n = C_1\lambda^n + C_2 n\lambda^n, n \in \mathbb{N},$$

para quaisquer C_1 e C_2 constantes reais.

Demonstração. Se a_n é uma solução qualquer da recorrência $X_{n+2} + pX_{n+1} + qX_n = 0$ e λ é a única solução da equação característica $r^2 + pr + q = 0$ o sistema

$$\begin{cases} C_1\lambda + C_2\lambda = a_1 \\ C_1\lambda^2 + 2C_2\lambda^2 = a_2 \end{cases}$$

possui uma única solução (C_1, C_2), sendo $C_1 = 2\frac{a_1}{\lambda} - \frac{a_2}{\lambda^2}$ e $C_2 = \frac{a_2 - \lambda a_1}{\lambda^2}$ (com $\lambda \neq 0$, pois $q \neq 0$, sendo a recorrência é de 2^{a} ordem). Para cada $n \in \mathbb{N}$, seja

$b_n = a_n - C_1\lambda^n - C_2 n\lambda^n$. Mostremos que $b_n = 0$, $\forall n \in \mathbb{N}$. Adicionando membro a membro as igualdades a seguir

$$\begin{cases} b_{n+2} = a_{n+2} - C_1\lambda^{n+2} - C_2(n+2)\lambda^{n+2} \\ b_{n+1} = pa_{n+1} - pC_1\lambda^{n+1} - C_2(n+1)p\lambda^{n+1} \\ qb_n = qa_n - qC_1\lambda^n - qC_2 n\lambda^n \end{cases}$$

Obtemos

$$b_{n+2} + pb_{n+1} + qb_n = (a_{n+2} + pa_{n+1} + qa_n) - C_1\lambda^n(\lambda^2 + p\lambda + q) - C_2 n\lambda^n(\lambda^2 + p\lambda + q) + \\ -C_2\lambda^{n+1}(2\lambda + p)$$

Tem-se que $b_{n+2} + pb_{n+1} + qb_n = 0; \forall n \in \mathbb{N}$, pois cada uma das quatro parcelas é nula. De fato, desde que a_n é uma solução da recorrência $X_{n+2} + pX_{n+1} + qX_n = 0$, tem-se que $a_{n+2} + pa_{n+1} + qa_n = 0$; as parcelas $C_1\lambda^n(\lambda^2 + p\lambda + q)$ e $C_2 n\lambda^n(\lambda^2 + p\lambda + q)$ também são nulas, pois $\lambda^2 + p\lambda + q = 0$, dado que λ é a raiz da equação característica $r^2 + pr + q = 0$; e a última parcela também é nula, pois $2\lambda + p = 0$.
Por fim,

$$C_1\lambda^1 + C_2\lambda^1 = a_1 \Rightarrow b_1 = a_1 - (C_1\lambda + C_2\lambda) = 0$$
$$C_1\lambda^2 + 2C_2\lambda^2 = a_2 \Rightarrow b_2 = a_2 - (C_1\lambda^2 + 2C_2\lambda^2) = 0$$

Ora, se $b_1 = b_2 = 0$ e $b_{n+2} + pb_{n+1} + qb_n = 0, \forall n \in \mathbb{N}$, podemos concluir que $b_n = 0, \forall n \in \mathbb{N}$ ou seja, $a_n - C_1\lambda^n - C_2 n\lambda^n = 0, \forall n \in \mathbb{N}$, o que equivale a dizer que

$$a_n = C_1\lambda^n + C_2 n\lambda^n = 0, \forall n \in \mathbb{N}.$$

■

Exemplo 30. *Considere a recorrência linear homogênea de segunda ordem:*

$$X_n = 4X_{n-1} - 4X_{n-2}, \qquad n \geq 2,$$

com condições iniciais $X_0 = 1$ e $X_1 = 2$. Determine sua solução.

Solução: A equação característica da recorrência é $\lambda^2 - 4\lambda + 4 = 0$. Fatorando, tem-se que $(\lambda - 2)^2 = 0$, sendo As raízes $\lambda_1 = \lambda_2 = 2$. Para equações que possuem raízes iguais a solução geral é da forma $a_n = C_1\lambda^n + C_2 n\lambda^n$. Substituindo $\lambda = 2$, obtemos $a_n = C_1 2^n + C_2 n 2^n$.

Das condições iniciais, podemos encontrar os valores das constantes C_1 e C_2:

$$\begin{cases} a_0 = (C_1 + C_2 \cdot 0) \cdot 2^0 = C_1 = 1, \\ a_1 = (C_1 + C_2 \cdot 1) \cdot 2^1 = (1 + C_2) \cdot 2 = 2 \implies 1 + C_2 = 1 \Rightarrow C_2 = 0. \end{cases}$$

Logo, a solução particular é:

$$a_n = 1 \cdot 2^n + 0 \cdot n \cdot 2^n = 2^n.$$

Exemplo 31. *Considere uma empresa que monitora o alcance de uma campanha publicitária em uma rede social. Suponha que a quantidade de compartilhamentos diários apresente um comportamento no qual o número de compartilhamentos em determinado dia depende do desempenho observado nos dois dias anteriores.*

Admita que, a cada novo dia, a quantidade de compartilhamentos seja obtida pelo dobro do número de compartilhamentos do dia imediatamente anterior menos a quantidade observada dois dias antes. Obtenha a relação de recorrência e estabeleça sua solução geral.

Solução: Se a_n representa o número de compartilhamentos no n-ésimo dia, então:

$$a_2 = 2a_1 - a_0, \quad a_3 = 2a_2 - a_1, \quad a_4 = 2a_3 - a_2,$$

e, de modo geral,

$$a_{n+2} = 2a_{n+1} - a_n.$$

Nessas condições, a quantidade de compartilhamentos a_n satisfaz a recorrência

$$X_{n+2} - 2X_{n+1} + X_n = 0.$$

A equação característica associada é dada por $\lambda^2 - 2\lambda + 1 = 0$. Fatorando, obtém-se $(\lambda - 1)^2 = 0$, de modo que a equação característica possui duas raiz reais iguais, isto é $\lambda_1 = \lambda_2 = 1$.

Assim, a solução geral da recorrência é da forma

$$a_n = C_1 \cdot 1^n + C_2 n \cdot 1^n = C_1 + C_2 n$$

em que C_1 e C_2 são constantes determinadas pelas condições iniciais.

Além das recorrências homogêneas de segunda ordem, existem situações em que a evolução da sequência não depende exclusivamente de seus dois termos anteriores, mas também da presença de termos externos ao processo recorrente. Tais relações são denominadas recorrências lineares de segunda ordem não homogêneas e constituem uma importante extensão dos modelos estudados anteriormente.

Proposição 10. *Se (a_n) é uma solução particular da recorrência*

$$X_{n+2} + pX_{n+1} + qX_n = f(n),$$

então a sua solução geral é dada por $z_n = a_n + y_n$, onde (y_n) é a solução geral da recorrência homogênea associada, isto é, da recorrência $x_{n+2} + px_{n+1} + qx_n = 0$.

Demonstração. Observe inicialmente que $z_n = a_n + y_n$ é uma solução. De fato,

$$\begin{aligned} z_{n+2} + pz_{n+1} + qz_n &= (a_{n+2} + y_{n+2}) + p(a_{n+1} + y_{n+1}) + q(a_n + y_n) \\ &= \underbrace{a_{n+2} + pa_{n+1} + qa_n}_{=f(n)} + \underbrace{y_{n+2} + py_{n+1} + qy_n}_{=0} \end{aligned}$$

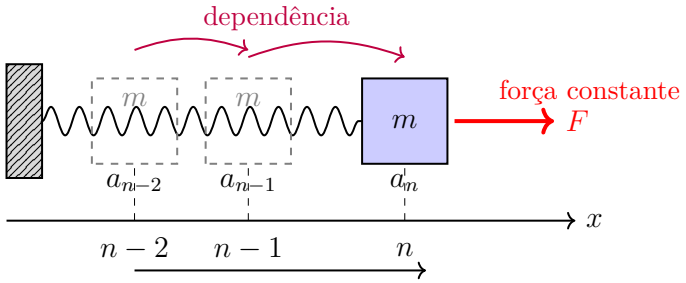
Demonstremos agora que toda solução da recorrência $X_{n+2} + pX_{n+1} + qX_n = f(n)$ é da forma $z_n = a_n + y_n$, onde a_n é uma solução particular da mesma recorrência e (y_n) é uma solução da recorrência homogênea associada.
Realmente, se (z_n) é a solução geral e (a_n) é uma solução particular da recorrência $X_{n+2} + pX_{n+1} + qX_n = f(n)$, vamos mostrar que (y_n) é dada por $y_n = z_n - a_n$ é uma solução da recorrência homogênea associada. Com efeito,

$$\begin{aligned} y_{n+2} + py_{n+1} + qy_n &= (z_{n+2} - a_{n+2}) + p(z_{n+1} - a_{n+1}) + q(z_n - a_n) = \\ &= \underbrace{(z_{n+2} + pz_{n+1} + qz_n)}_{=f(n)} - \underbrace{(a_{n+2} + pa_{n+1} + qa_n)}_{=f(n)} = f(n) - f(n) = 0 \end{aligned}$$

Assim sendo, $y_n = z_n - a_n$ é a solução geral da equação da recorrência homogênea $X_{n+2} + pX_{n+1} + qX_n = 0$, e portanto, $y_n = a_n - z_n \Rightarrow z_n = a_n + y_n$ é a solução geral da recorrência $a_{n+2} + pa_{n+1} + q_n = f(n)$. ■

Exemplo 32. *Considere um sistema mecânico discreto composto por uma massa presa a uma mola, cujo deslocamento ao longo do tempo é observado em instantes igualmente espaçados. Suponha que a posição da massa dependa de suas duas posições anteriores e também da ação de uma força externa constante aplicada ao sistema.*

Figura 6 – Sistema massa-mola discreto com força externa constante e recorrência de segunda ordem.



Fonte: Autor, 2026.

Se a_n representa o deslocamento da massa no instante n, admite-se que o sistema satisfaça a recorrência

$$X_{n+2} - 2X_{n+1} + X_n = 3,$$

Determine a solução geral da recorrência, sendo as condições iniciais $X_0 = 1$ e $X_1 = 2$.

Solução: Inicialmente, consideremos a recorrência homogênea associada, $X_{n+2} - 2X_{n+1} + X_n = 0$.. A equação característica correspondente é $\lambda^2 - 2\lambda + 1 = 0$.

Ou seja,

$$(\lambda - 1)^2 = 0.$$

Logo, a equação característica possui duas raízes iguais, isto é, $\lambda_1 = \lambda_2 = 1$. Assim, a solução geral da recorrência homogênea é

$$a_n^{(h)} = C_1 + C_2 n.$$

Agora, busquemos uma solução particular da recorrência não homogênea. Como o termo independente é uma constante, tenta-se inicialmente uma solução constante: $a_n^{(p)} = K$. Substituindo na recorrência: $K - 2K + K = 3 \Rightarrow 0 = 3$, o que mostra que essa escolha não é adequada.

Como $\lambda_1 = \lambda_2$, devemos ter que a solução particular é da forma

$$a_n^{(p)} = An^2.$$

Substituindo:

$$\begin{aligned} A(n+2)^2 - 2A(n+1)^2 + An^2 &= 3 \\ A(n^2 + 4n + 4) - 2A(n^2 + 2n + 1) + An^2 &= 3 \\ An^2 + 4An + 4A - 2An^2 - 4An - 2A + An^2 &= 3 \\ 2A &= 3 \\ A &= \frac{3}{2} \end{aligned}$$

A solução geral da recorrência é dada por

$$a_n = a_n^{(h)} + a_n^{(p)} \Rightarrow a_n = C_1 + C_2 n + \frac{3}{2}n^2.$$

Utilizando as condições iniciais para determinar C_1 e C_2, obtemos o sistema:

$$\begin{cases} a_0 = C_1 + C_2 \cdot 0 + \dfrac{3}{2} \cdot 0^2 \Rightarrow 1 = C_1, \\ a_1 = C_1 + C_2 \cdot 1 + \dfrac{3}{2} \cdot 1^2 \Rightarrow 2 = 1 + C_2 + \dfrac{3}{2}. \end{cases}$$

Resolvendo,

$$\begin{aligned} C_1 &= 1, \\ C_2 &= 2 - 1 - \frac{3}{2} = -\frac{1}{2}. \end{aligned}$$

Portanto,

$$C_1 = 1, \quad C_2 = -\frac{1}{2}$$

Logo, a solução da recorrência é

$$a_n = 1 - \frac{1}{2}n + \frac{3}{2}n^2.$$

4 Algoritmos

A resolução de problemas ocupa um papel central no processo educativo, especialmente quando se busca formar estudantes capazes de atuar de maneira crítica, criativa e autônoma diante das demandas do mundo contemporâneo. Diferentemente de abordagens tradicionais, que frequentemente conduzem a respostas únicas e previamente estabelecidas, metodologias voltadas ao desenvolvimento do pensamento computacional e algorítmico valorizam o percurso intelectual do estudante, reconhecendo o erro, a tentativa e a reformulação como partes constitutivas da aprendizagem. Ao buscar soluções para um determinado problema, o estudante não se encontra limitado a resultados pré-determinados, mas é estimulado a explorar múltiplas possibilidades. Como destacam Gontijo et al. (2019, p. 62), esse processo permite ao aluno alcançar

> uma gama de soluções por meio do pensamento divergente, algumas corretas, outras equivocadas, algumas bem elaboradas, outras em processo de estruturação, algumas tidas como válidas, outras não aceitas, e entre todas essas uma quantidade menor de respostas originais, tal como ocorre no processo de solução de problemas na vida real.

Essa perspectiva dialoga diretamente com o conceito de Pensamento Computacional, que extrapola o domínio exclusivo da Computação e se apresenta como uma competência humana ampla e transversal. Para Brackmann (2017, p. 29), o Pensamento Computacional constitui-se em uma

> capacidade criativa, crítica e estratégica humana de saber utilizar os fundamentos da Computação, nas mais diversas áreas do conhecimento, com a finalidade de identificar e resolver problemas, de maneira individual ou colaborativa, através de passos claros, de tal forma que uma pessoa ou uma máquina possam executá-los eficazmente.

Um Algoritmo é definido como uma sequência finita de instruções e ordenada de forma lógica para a execução de uma determinada tarefa ou a resolução de um determinado problema. Do ponto de vista computacional, Manzano e Oliveira (2019) o define como

> regras formais, sequenciais e bem definidas a partir do entendimento lógico de um problema a ser resolvido por um programador com o objetivo de transformá-lo em um programa que seja possível de ser tratado e executado por um computador, em que dados de entrada são transformados em dados de saída.

Nesse contexto, os algoritmos emergem como elementos fundamentais, pois representam a organização lógica e estruturada de ações necessárias para a resolução de problemas. Introduzir o estudo de algoritmos, portanto, significa oportunizar ao estudante o desenvolvimento de habilidades relacionadas à decomposição de problemas, à construção de sequências lógicas, à tomada de decisões e à validação de soluções. Assim, o ensino de algoritmos não se restringe à programação, mas configura-se como

uma estratégia pedagógica potente para aproximar a escola das práticas de resolução de problemas presentes na vida real e no contexto científico e tecnológico contemporâneo.

4.1 Origem e Evolução dos Algoritmos

A noção de algoritmo, hoje formulada resumidamente como "um conjunto finito de instruções destinadas à resolução de um problema", perpassa toda a história da matemática. Isso muito antes da era dos computadores modernos ou mesmo de se pensar na computação que temos atualmente, tendo em vista que o homem desde seus primórdios, utiliza-se de procedimentos sistemáticos, inclusive como modos de sobrevivência.

As civilizações antigas, movidas por necessidades práticas como administração, comércio, construção e astronomia, desenvolveram procedimentos sistemáticos para efetuar cálculos, resolver equações e manipular quantidades. Esses procedimentos possuíam características referenciadas por Knuth (1997), Berlinski (2002), Cormen et al. (2009), Dasgupta, Papadimitriou e Vazirani (2010), Sedgewick e Wayne (2011) para os reconhecermos como fundamentalmente algorítmicas: precisão, reprodutibilidade, generalidade e finitude. Assim, mesmo sem formalização teórica, algoritmos já estavam presentes na prática matemática desde os primórdios da civilização humana.

Um dos registros mais antigos desse pensamento algorítmico encontra-se na Mesopotâmia. A tábua Plimpton 322, Figura 7, datada de cerca de 1800 a.C., apresenta conjuntos de números que correspondem a algumas triplas pitagóricas. Embora a tábua não descreva explicitamente o procedimento utilizado para gerar esses números, estudos modernos indicam que sua construção exige regras sistemáticas baseadas em manipulações aritméticas refinadas. A existência de tais tabelas sugere que os babilônicos dominavam procedimentos gerais para produzir padrões numéricos, evidenciando uma forma embrionária de algoritmos.

Os Babilônicos também desenvolveram um método sistemático para a resolução de equações do segundo grau. Logicamente, à época Babilônica, os povos não utilizavam equações em forma algébrica, isto é, $ax^2+bx+c=0$, mas sim procedimentos numéricos e geométricos que hoje podem ser interpretados como algoritmos, pois os símbolos que determinavam os coeficientes e expoentes ainda não "existiam", além de que ainda não havia a representação do zero.

A equação $Ax^2 + Bx = C$ era interpretada como um problema de áreas: Ax^2 era a área de um quadrado de lado x (multiplicada por A), e Bx era a área de um retângulo. O método de completar quadrados envolvia construir um quadrado maior a partir dessas peças.

Figura 7 – Tábua Plimpton 322.



Fonte: Katz (1992).

A expressão final obtida,

$$L = \left(\sqrt{\left(\frac{B}{2}\right)^2 + AC} - \frac{B}{2}\right) \cdot \frac{1}{A},$$

é o valor da incógnita x, ou seja, o lado do quadrado original cuja área era representada por Ax^2. Após completar o quadrado e extrair a raiz, o valor obtido após subtrair $B/2$ e dividir por A é exatamente a raiz positiva da equação.

Dada a equação $Ax^2+Bx = C$, o procedimento pode ser descrito da seguinte forma:

1. Multiplique A por C: AC
2. Encontre metade de B:$\left(\frac{B}{2}\right)$
3. Multiplique $\frac{B}{2}$ por $\frac{B}{2}$: $\left(\frac{B}{2}\right) \cdot \left(\frac{B}{2}\right)$
4. Adicione AC e $\left(\frac{B}{2}\right)^2$: $\left(\frac{B}{2}\right)^2 + AC$
5. Extraia a raiz quadrada de $\left(\frac{B}{2}\right)^2 + AC$: $\left(\sqrt{\frac{B}{2}^2 + AC}\right)$
6. Subtraia $\left(\frac{B}{2}\right)$ do resultado anterior: $\left(\sqrt{\frac{B}{2}^2 + AC}\right) - \left(\frac{B}{2}\right)$
7. Tome o recíproco de A: $\left(\frac{1}{A}\right)$
8. Multiplique $\left(\frac{1}{A}\right)$ por $\left(\sqrt{\frac{B}{2}^2 + AC}\right) - \left(\frac{B}{2}\right)$:

9. Obtemos o lado do quadrado: $\left[\left(\sqrt{\frac{B}{2}^2 + AC}\right) - \left(\frac{B}{2}\right)\right] \cdot \left(\frac{1}{A}\right)$

É importante destacar que, no contexto da matemática babilônica, apenas soluções positivas eram consideradas válidas. Isso ocorre porque os problemas tratados pelos babilônicos estavam geralmente relacionados a situações práticas e geométricas, envolvendo medidas de comprimento, áreas e volumes. Dessa forma, uma solução negativa não possuía interpretação concreta dentro da realidade matemática da época. Assim, embora atualmente a equação $x^2+6x = 16$ admita duas soluções reais, $x = 2$ e $x = -8$, o método babilônico consideraria apenas a solução positiva $x = 2$, por representar uma quantidade geometricamente admissível.

Esse aspecto evidencia uma diferença importante entre a matemática antiga e a matemática moderna. Enquanto hoje os números negativos são plenamente aceitos como objetos matemáticos legítimos, as civilizações antigas trabalhavam predominantemente com quantidades associadas a grandezas concretas. Desse modo, o método babilônico revela não apenas um procedimento algorítmico sofisticado para sua época, mas também a forte relação histórica entre álgebra, geometria e aplicações práticas.

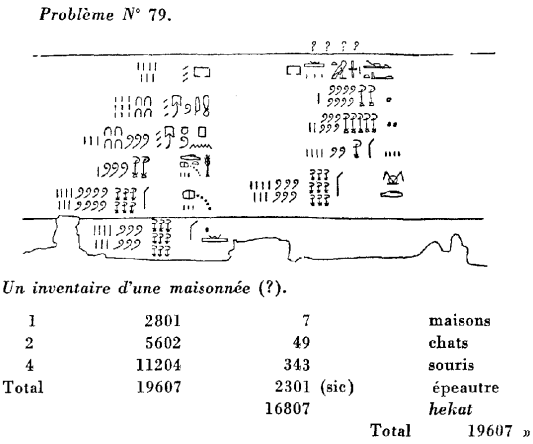
Observa-se que os babilônicos já utilizavam procedimentos equivalentes a algoritmos, evidenciando a natureza histórica e construtiva da álgebra. Diante disso, podemos interpretar o método babilônico como um algoritmo estruturado com:

- entrada: coeficientes da equação;
- processamento: transformação por completamento de quadrados;
- etapas finitas e bem definidas;
- saída: soluções da equação.

No Egito Antigo, modelos matemáticos eram expressos por meio de métodos sequenciais aplicados a problemas práticos. O Papiro de Rhind, Figura 8, por volta de 1650 a.C., nos mostra que os antigos egípcios utilizavam conjuntos ordenados de passos para resolver sistemas de equações, efetuar operações como multiplicação, divisão, cálculo de áreas e volumes, determinar impostos ou realizar partilhas, além de decomposições de frações.

Dentre os métodos utilizados pelos egípcios, por exemplo, a multiplicação era realizada por duplicação sucessiva, seguindo uma sequência rigorosa de passos, utilizando decomposições numéricas e garantindo resultados corretos, sendo, pois, um procedimento totalmente algorítmico. Este achado histórico nos mostra que os egípcios não apenas calculavam, mas operavam seguindo procedimentos estruturados que hoje reconhecemos como algoritmos.

Figura 8 – Papiro de Rhind - Problema 79



Problème N° 79.

« *Un inventaire d'une maisonnée* (?).

1	2801	7	maisons
2	5602	49	chats
4	11204	343	souris
Total	19607	2301 (sic)	épeautre
		16807	*hekat*
		Total	19607 »

Fonte: Adaptado de The Schøyen Collection (2023).

A decomposição de frações em somas de frações unitárias constitui uma das práticas matemáticas mais antigas da humanidade, estando diretamente associada à matemática desenvolvida no Egito Antigo. Esse método consistia em representar um número racional positivo como soma de frações com numerador igual a 1, chamadas de frações unitárias. Tal procedimento elucida como os egípcios compreendiam e organizavam as operações matemáticas em atividades comerciais, administrativas e de mensuração.

Este método egípcio tornou-se um importante exemplo histórico de representação numérica e de técnicas algorítmicas primitivas, pois seguia regras sistemáticas para transformar frações gerais em combinações de frações unitárias distintas. Os egípcios realizavam essa transformação por uma combinação de limitações notacionais, necessidades práticas e padronização algorítmica. Os principais motivos são: Sobre essa prática, Eves (2011, p. 73) afirma que "Os egípcios esforçaram-se para evitar algumas das dificuldades computacionais encontradas com frações representando-as, com exceção de 2/3, como soma das frações chamadas unitárias, ou seja, aquelas de numerador igual a 1."

Exemplo 33. *Considere a fração* $\frac{5}{7}$*. Deseja-se escrevê-la como soma de frações unitárias, isto é, frações com numerador igual a* 1*.*

Solução:[1] *Inicialmente, escolhe-se a maior fração unitária menor ou igual a* $\frac{5}{7}$*. Para isso, calcula-se:*

$$\frac{7}{5} = 1,4.$$

[1] É importante destacar que a técnica algorítmica utilizada não é a empregada pelos antigos egípcios. Embora o resultado final seja de fato o que caracteriza o sistema fracionário egípcio, o procedimento iterativo de calcular o teto do inverso da fração restante é uma criação matemática muito posterior.

Tomando o teto desse valor, obtém-se:

$$\left\lceil \frac{7}{5} \right\rceil = 2.$$

Logo,

$$\frac{1}{2}$$

é a primeira fração unitária da decomposição.

Subtraindo:

$$\frac{5}{7} - \frac{1}{2} = \frac{10-7}{14} = \frac{3}{14}.$$

Agora repete-se o processo para a fração restante:

$$\frac{14}{3} \approx 4,67,$$

e então

$$\left\lceil \frac{14}{3} \right\rceil = 5.$$

Assim,

$$\frac{3}{14} = \frac{1}{5} + \left(\frac{3}{14} - \frac{1}{5}\right).$$

Calculando:

$$\frac{3}{14} - \frac{1}{5} = \frac{15-14}{70} = \frac{1}{70}.$$

Portanto, a decomposição egípcia de $\frac{5}{7}$ é dada por:

$$\frac{5}{7} = \frac{1}{2} + \frac{1}{5} + \frac{1}{70}.$$

A decomposição apresentada evidencia que a representação de frações como soma de frações unitárias não se trata apenas de um resultado isolado, mas de um processo estruturado que segue etapas bem definidas e repetitivas. Em particular, observa-se que cada passo do procedimento envolve uma escolha determinística (a maior fração unitária admissível), seguida de uma subtração e da repetição do mesmo padrão sobre o restante obtido.

Um exemplo significativo do caráter algorítmico utilizados pelos egípcios encontra-se no Problema 79 [2] do Papiro de Rhind, que trata explicitamente de uma situação envolvendo o crescimento multiplicativo, aquilo que hoje reconhecemos como uma progressão geométrica. No referido problema, é descrita a seguinte situação:

[2] Posteriormente, apresentamos a resolução do problema

Figura 9 – Problema encontrado no Papiro de Rhind (1650 a.C.)

Numa aldeia egípcia há sete casas. Em cada casa, há sete gatos. Para cada gato, há sete ratos. Para cada rato, há sete espigas de trigo. Em cada espiga, há sete grãos.

- Quantos grãos há ao todo, nas sete casas?
- Casas, gatos, ratos, espigas e grãos, quantos objetos são ao todo?

Fonte: Papiro de Rhind (1650 a.C).

Este problema propõe que os escribas egípcios compreendiam e manipulavam situações que envolviam crescimento exponencial, ainda que não utilizassem a linguagem algébrica atual. Em contrapartida, recorriam a um procedimento sistemático, onde cada novo valor era calculado aplicando-se repetidamente o mesmo fator multiplicativo. Trazendo para conceitos atuais, tratamos tal problema como um algoritmo no sentido clássico: um conjunto de operações finitas, aplicadas de maneira repetitiva e ordenada, para gerar um resultado previsível.

Figura 10 – Euclides de Alexandria - séc III a.C



Fonte: https://sites.unipampa.edu.br (2015).

E esta sequência bem definida de passos é expressa também no que talvez seja um dos mais famosos algoritmos quando pensamos em matemática, que é o algoritmo de Euclides. Euclides, 10 foi um grande matemático grego do século III a.C., conhecido

como o "pai da geometria", e o autor da obra "Os Elementos", um tratado fundamental que organizou o conhecimento geométrico da época.

Quase não há informações sobre a vida de Euclides de Alexandria. O pouco que se sabe é que ele foi um dos primeiros estudiosos ativos no Museu e na Biblioteca de Alexandria, fundados por Ptolomeu I e seu sucessor, Ptolomeu II Filadelfo. Sua obra mais famosa, Os Elementos, transcendeu o tempo, sendo publicada em várias edições, mais do que qualquer outra obra, traduzida para diversas línguas e estudada intensamente. Para alguns, pode aparentar ser um livro monótono, pois seu conteúdo é composto por definições, axiomas, postulados, teoremas e demonstrações. Para outros, entretanto, Os Elementos representa uma das obras mais importantes da história da Matemática, por sistematizar conhecimentos matemáticos e estabelecer um modelo de organização lógica e dedutiva que influenciou profundamente o desenvolvimento da Matemática ao longo dos séculos. Para a história da Matemática, a obra constitui um marco na consolidação do método axiomático e na transmissão do conhecimento geométrico da Antiguidade para as gerações posteriores.

Segundo Katz (2009), a obra de Euclides foi redigida há aproximadamente 2.300 anos, porém não há registros de cópias desse texto que remontem ao período original. Os fragmentos mais antigos conhecidos incluem alguns cacos de cerâmica encontrados no Egito, datados de cerca de 225 a.C., nos quais constam anotações que parecem corresponder a duas proposições do Livro XIII, bem como pedaços de papiro contendo partes do Livro II, datados de aproximadamente 100 a.C.

Embora muitos pensem que sua obra se limita a parte geométrica, seu mais importante livro apresentam textos sobre teoria dos números e álgebra. O livro VII nos apresenta logo de início o Algoritmo de Euclides, um processo para achar o máximo divisor comum de dois ou mais números inteiros, o qual pode ser utilizado também para verificar se dois inteiros são primos entre si. De acordo com o que estamos apresentando sobre algoritmo, o proposto por Euclides nos apresenta passos bem definidos para que alcancemos sua finalidade. Dados dois números inteiros quaisquer, *"Divida o maior dos dois números inteiros positivos pelo menor e então divida o divisor pelo resto. Continue esse processo de dividir o último divisor pelo último resto, até que a divisão seja exata. O divisor final é o m.d.c. procurado."* Esta técnica, embora pareça simples, é um dos algoritmos fundamentais da teoria dos números e atesta sua importância e eficácia em sua longevidade, mostrando procedimentos sistematizados mesmo antes dos computadores existirem.

Partindo para o mundo islâmico, entre os séculos VIII e IX, destaca-se a figura de Abu Abdullah Muhammad ibn Musa al-Khwarizmi (780 d.C.–850 d.C.). A forma medieval de seu nome, Algorismus, é a raiz etimológica que, por corruptela linguística, gerou o vocábulo 'algarismo', empregado para designar os símbolos numéricos do

sistema decimal. Paralelamente, também deu origem ao termo 'algoritmo', associado aos procedimentos sistemáticos de cálculo que ele próprio sistematizou. Al-Khwarizmi atuou na Casa da Sabedoria, em Bagdá, um dos mais importantes centros de produção científica da Idade de Ouro Islâmica[3]. Nesse ambiente, teve acesso a conhecimentos oriundos das tradições grega, persa e indiana, contribuindo para sua sistematização e difusão.

Figura 11 – Al-Khwarizmi (780 d.C.–850 d.C.)



Fonte: https://google.com (2025).

Al-Khwarizmi escreveu, dentre tantos documentos, o Tratado sobre o cálculo de al-jabr e al-muqabala. Suas contribuições foram significativas para diversas áreas, como astronomia, geografia, trigonometria e, especialmente, matemática. Sua obra sobre álgebra é considerada um marco na história da matemática, sendo responsável pela popularização de métodos gerais para a resolução de equações lineares e quadráticas. Além disso, seus estudos sobre os numerais indo-arábicos contribuíram para a disseminação de um sistema de numeração mais eficiente do que os numerais romanos então utilizados em grande parte da Europa.

[3] A Idade de Ouro Islâmica (séc. VIII–XIV) foi um período de intenso florescimento científico e cultural no mundo islâmico, especialmente sob o Califado Abássida. Em Bagdá, a *Casa da Sabedoria* (*Bayt al-Hikma*) reuniu estudiosos de diversas origens para traduzir e ampliar obras da antiguidade grega, indiana e persa. Esse movimento preservou e difundiu conhecimentos fundamentais para a matemática, astronomia, medicina e filosofia, que posteriormente influenciariam o Renascimento europeu.

Seu método consistia em apresentar um conjunto de regras claras e sequenciadas para resolver equações, manipular grandezas e calcular resultados. As traduções latinas de suas obras exerceram profunda influência sobre o desenvolvimento da matemática ocidental e fizeram com que seu nome, latinizado para Algoritmi, passasse a ser associado aos procedimentos de cálculo. Com o passar do tempo, essa associação deu origem à palavra "algoritmo", atualmente utilizada para designar uma sequência finita e ordenada de instruções destinadas à resolução de um problema.

Embora a computação moderna ainda estivesse distante, os princípios fundamentais dos algoritmos — sequência, clareza, finitude e exatidão, os quais serão conceituados adiante — já estavam presentes em suas obras. Em seus escritos sobre álgebra, Al-Khwarizmi descreve procedimentos gerais que independem do contexto específico do problema, aproximando-se significativamente da concepção moderna de algoritmos determinísticos. Dessa forma, pode-se afirmar que Al-Khwarizmi não apenas introduziu novos conceitos matemáticos, mas também formalizou uma maneira sistemática de organizar o raciocínio matemático por meio de procedimentos, estabelecendo bases que influenciariam o desenvolvimento da matemática, da computação e da própria noção contemporânea de algoritmo.

Durante a Idade Média, por volta do século XII, com a difusão do sistema numérico hindu-arábico, o jovem Leonardo Fibonacci, 12, nas diversas viagens que realizou pelo Mediterrâneo com seu pai comerciante e das conversas que teve com os vários estudiosos islâmicos, absorveu o conhecimento matemático do mundo islâmico e mais tarde veio a se tornar um importante matemático, contribuindo também para que os algoritmos ganhassem uma nova camada de formalização e disseminação.

Leonardo, filho de Bonaccio, nasceu em Pisa, importante centro comercial, seu pai era negociante mercantil. Desde cedo, Fibonacci foi influenciado pelas atividades comerciais e matemáticas de seu pai, desenvolvendo um interesse particular pela aritmética. Posteriormente, suas viagens por diferentes regiões do Mediterrâneo, incluindo Egito, Sicília, Grécia e Síria, possibilitaram o contato direto com os conhecimentos matemáticos orientais e árabes, exercendo papel fundamental em sua formação intelectual e na difusão desses saberes para a Europa ocidental. Dentre suas obras, se destacam o Practica geometriae (1220), o Liber quadratorum (1225) e o mais famoso, Liber abbaci*(livro de ábaco)* (1202, 1228). Este introduziu na Europa ocidental métodos sistemáticos de cálculo herdados das tradições árabe e indiana, incluindo procedimentos para multiplicação, divisão, proporções, problemas de comércio e progressões.

Nesse mesmo texto surge a famosa sequência que hoje leva seu nome, cuja definição é naturalmente recursiva e cuja compreensão depende diretamente de uma perspectiva algorítmica: cada termo é obtido aplicando-se uma regra aos termos anteriores. Isso demonstra que algoritmos recursivos, longe de serem uma invenção da compu-

Figura 12 – Fibonacci (1170–1240).



Fonte: Google Imagens, 2026.

tação moderna, constituem uma forma de pensamento presente na matemática há séculos. Estudar sequências numéricas permite compreender como processos podem ser descritos por regras bem definidas, característica essencial da noção de algoritmo. Entre os exemplos mais clássicos desse tipo de estrutura está a sequência de Fibonacci, cuja lei de formação evidencia como uma regra simples pode ser aplicada repetidamente, gerando uma sucessão infinita de valores. E esse tipo de construção antecipa ideias fundamentais da computação, uma vez que cada termo da sequência depende de uma instrução previamente estabelecida, podendo ser executada de forma sistemática. Nesse contexto, a análise de processos recursivos como o de Fibonacci abre espaço para uma discussão mais ampla sobre a formalização de procedimentos algorítmicos e sua evolução histórica.

É nesse cenário que surge a contribuição de Ada Lovelace, cuja visão pioneira sobre a possibilidade de máquinas executarem sequências de instruções matemáticas consolida a ligação entre recorrências, algoritmos e programação, estabelecendo as bases conceituais que serão exploradas a seguir. Ada Lovelace, nome pelo qual ficou conhecida Augusta Ada King, Condessa de Lovelace, nasceu em 10 de dezembro de 1815, em Londres, sendo filha do poeta romântico Lord Byron e de Anne Isabella Milbanke, uma mulher com forte formação em matemática e ciências.

Por intermédio de Mary Somerville(1780–1872), cientista grande influente, Ada conheceu Charles Babbage(1791–1871), criador da Máquina Diferencial e da Máquina Analítica, projetos considerados precursores dos computadores modernos. A relação

Figura 13 – Ada Lovelace (1815–1852).



Fonte: www.britannica.com, 2026.

entre Ada Lovelace e Charles Babbage tornou-se fundamental para a história da computação. Fascinada pelas possibilidades matemáticas da Máquina Analítica, Ada percebeu que o dispositivo poderia ir muito além de simples cálculos numéricos. Enquanto muitos estudiosos da época viam a máquina apenas como um instrumento mecânico de cálculo, Ada compreendeu que ela poderia manipular símbolos, executar sequências de instruções e resolver problemas de maneira automatizada.

Em 1843, Ada traduziu um artigo do matemático italiano Luigi Federico Menabrea sobre a Máquina Analítica, porém, suas contribuições mais importantes apareceram nas notas adicionais que anexou ao texto. Essas notas eram mais extensas que o próprio artigo original e continham descrições detalhadas do funcionamento lógico da máquina. Entre elas, destacou-se um procedimento destinado ao cálculo dos números de Bernoulli, considerado por muitos como o primeiro algoritmo desenvolvido especificamente para ser executado por uma máquina. Frequentemente reconhecida como a primeira programadora da história, sua contribuição não se limitou apenas à elaboração de instruções matemáticas, mas também pela antecipação de conceitos fundamentais da computação moderna, por exemplo, ao afirmar que máquinas poderiam manipular não apenas números, mas também símbolos, sons e informações em geral.

Muitos desafios tiveram que ser enfrentados durante seu percurso na ciência, além

do próprio preconceito e exclusão que as mulheres da época sofriam com relação a serem cientistas. Outro desafio importante estava relacionado à própria inexistência de computadores reais em funcionamento. A Máquina Analítica de Babbage nunca foi concluída fisicamente, o que significa que Ada desenvolveu ideias algorítmicas para uma máquina que existia apenas em projetos mecânicos e conceituais. Ainda assim, ela conseguiu compreender princípios fundamentais de programação, estruturas sequenciais e automação de cálculos.

O reconhecimento de sua importância permanece vivo na contemporaneidade por meio de diversas iniciativas que homenageiam seu legado. Entre elas destaca-se o Ada Lovelace Day, celebrado anualmente na segunda terça-feira de outubro, com o objetivo de valorizar e dar visibilidade às contribuições de mulheres nas áreas de ciência, tecnologia, engenharia e matemática. O evento tornou-se uma referência internacional na promoção da participação feminina nesses campos do conhecimento, inspirando novas gerações de pesquisadoras e profissionais. Outra homenagem significativa é o Ada Lovelace Award, premiação concedida pela Association for Women in Computing a pessoas que se destacam por suas contribuições à computação e pelo incentivo à participação das mulheres na área tecnológica. A existência de iniciativas como essa evidencia o impacto duradouro de Ada Lovelace e o reconhecimento de sua importância para a evolução da ciência da computação.

A partir do Renascimento e especialmente com o desenvolvimento da álgebra simbólica, os algoritmos tornam-se cada vez mais associados ao processo de resolução sistemática de problemas gerais. Esse movimento culmina, no século XX, na formalização do conceito de algoritmo por matemáticos como Alan Turing, Alonzo Church e, posteriormente, Donald Knuth, que estabelecem os fundamentos teóricos da ciência da computação e classificam algoritmos de acordo com suas características e eficiências.

Ao longo de toda a evolução histórica, observa-se que, embora o termo “algoritmo” seja relativamente recente, a ideia que ele representa atravessa todo um percurso histórico. Desde tabelas babilônicas até a álgebra de Al-Khwarizmi, passando por métodos chineses, indianos e europeus, o elemento comum é uma forma de organizar o pensamento matemático por meio de regras gerais e procedimentos repetíveis. Essa perspectiva histórica é essencial para compreender o desenvolvimento dos algoritmos na matemática moderna, especialmente quando se considera o papel central que eles desempenham na definição de recorrências lineares, na construção de algoritmos recursivos e na implementação computacional.

Mais do que ferramentas, algoritmos são formas de pensar matematicamente, articulando abstração, generalização e execução, permitindo que problemas sejam decompostos em etapas bem definidas e reprodutíveis.

4.2 Conceitos e Formas de representação

Embora os algoritmos estejam presentes desde a antiguidade, a formalização do conceito é relativamente recente e surge principalmente no século XX, influenciada pelo desenvolvimento da lógica matemática, da computação e do estudo da eficiência dos procedimentos de cálculo. A evolução histórica cria um cenário no qual se torna necessário definir com precisão o que significa seguir um conjunto de regras para resolver um problema, e, sobretudo, o que distingue um procedimento matemático comum de um algoritmo propriamente dito. Esse esforço de formalização culmina nas definições clássicas adotadas hoje na ciência da computação e na matemática.

As definições de algoritmo sofreram mudanças ao longo do tempo até chegar ao conceito que utilizamos atualmente. Podemos definir Algoritmo, de modo informal[4], como "uma receita de bolo", na qual devemos seguir todo um procedimento para obtermos um resultado: o bolo. De um modo mais até mais poético, Berlinski (2002) pondera que

> No mundo de onde surge a matemática e para o qual o matemático, como nós, deve voltar, um algoritmo, por assim dizer, é um conjunto de regras, uma receita, uma prescrição para a ação, um guia, uma diretiva concatenada e controlada, uma intimação, um código, um esforço feito para jogar um complexo xale verbal sobre o caos inarticulado da vida.

Vários momentos da história mostram que o homem utiliza uma sequência bem definida de procedimentos para obter um resultado. Podemos notar que antes da formalização do conceito de algoritmo e da introdução da computação moderna, ou mesmo anteriormente a criação dos computadores, procedimentos sistemáticos eram utilizados para realizar determinadas tarefas, envolvessem elas matemática ou não. No entanto, o conceito de algoritmo, tal como é entendido na atualidade, ganhou rigor formal apenas no século XX, especialmente com os avanços da lógica matemática, da teoria da computação e das contribuições de estudiosos como Donald Knuth e Thomas H. Cormen.

Knuth (1997) em seu livro *"The Art of Computer Programming"* pondera que um algoritmo possui cinco características importantes que ele nomeia de:

1. FINITUDE: *(Finiteness)* Um algoritmo deve sempre terminar após um número finito de passos.

2. DETERMINISMO: *(Definiteness)* Cada passo de um algoritmo deve ser precisamente definido; as ações a serem executadas devem ser especificadas de forma rigorosa e não ambígua para cada caso.

[4] Chamamos atenção para o fato de que conceituar algoritmo como uma receita é incorreto. Trataremos disso mais a frente.

3. ENTRADA: *(Input)* Um algoritmo possui zero ou mais entradas: quantidades que lhe são fornecidas inicialmente, antes do algoritmo começar, ou dinamicamente durante sua execução.

4. SAÍDA: *(Output)* Um algoritmo possui uma ou mais saídas: quantidades que têm uma relação especificada com as entradas.

5. EFETIVIDADE: *(Effectiveness)* Um algoritmo deve ser efetivo no sentido de que todas as suas operações devem ser suficientemente básicas para que possam, em princípio, ser efetivamente realizadas e em um tempo finito por alguém usando lápis e papel.

O autor ainda atribui como falha a ideia de comparar algoritmo a uma receita de bolo pelo fato de que esta não possui em muitos casos o determinismo, pois muitas instruções de cozinheiros são indefinidas quando citam, por exemplo, "adicione uma pitada de sal", pois apesar da quantidade poder ser definida[5], o local da adição não é determinado (podendo ser em cima, em baixo, ao lado). Portanto, um algoritmo dever ser o mais específico e direto possível, ao ponto de que qualquer pessoa ou máquina possa realizar a tarefa que se quer.

As definições podem variar em suas nomenclaturas, porém sua mensagem é que um algoritmo é algo bem definido para se chegar a um fim. Mas como afirmar que determinado algoritmo está correto ou não? O que analisar em seus passos para não produzir um algoritmo errado? Cormen et al. (2009) afirmam que se um algoritmo está correto para cada instância de entrada, ele termina com a saída correta, entre outras palavras, quando resolve um problema. Em muitos casos, a prova de correção envolve indução matemática, uma técnica intimamente associada ao pensamento algorítmico, especialmente em procedimentos recursivos. Contudo, ainda que um algoritmo esteja incorreto, ele pode ser útil, desde que possamos controlar sua taxa de erro[6].

Alternativamente, algoritmos são descritos como procedimentos que tomam valores de entrada, realizam uma sequência de operações e produzem valores de saída. Para esses autores, a ênfase recai tanto na correção quanto na eficiência, dimensões que se tornaram cruciais com o avanço da computação. A correção refere-se ao fato de que o algoritmo deve produzir sempre o resultado desejado para todas as entradas válidas, já a eficiência diz respeito ao uso de recursos, como tempo de processamento e memória, aspectos que se tornaram relevantes à medida que problemas matemáticos passaram a ser resolvidos por máquinas. A terminação garante que o algoritmo não executa um ciclo infinito, retornando sempre um resultado. Essas duas propriedades, correção

[5] Uma pitada equivale a aproximadamente um valor entre $\frac{1}{8}$ e $\frac{1}{16}$ de uma colher de chá.

[6] Um exemplo é o teste de primalidade de Miller Rabin explicitado adiante.

e terminação, estão diretamente ligadas à robustez lógica da definição moderna de algoritmo.

Outro aspecto importante da formalização contemporânea é a distinção entre algoritmos determinísticos e algoritmos probabilísticos. Um algoritmo determinístico produz sempre o mesmo resultado para a mesma entrada, seguindo exatamente os mesmos passos. É o caso do algoritmo de Euclides[7] para o cálculo do máximo divisor comum. Já os algoritmos probabilísticos utilizam escolhas aleatórias durante sua execução e, portanto, podem produzir resultados diferentes em instâncias distintas, embora, em geral, possuam garantias estatísticas de correção. Um exemplo amplamente utilizado é o teste de primalidade de Miller–Rabin, que pode errar ao afirmar que um número composto é primo. Embora o algoritmo seja formalmente incorreto, é possível torná-lo tão confiável quanto desejado simplesmente aumentando suas interações. Por exemplo, se realizarmos o teste 10 vezes, a chance de ele estar errado ao garantir que um número é primo é menor que 1 em 1 milhão:

Seja n um número ímpar $n \geq 3$ e $b \in \{1, ..., n-1\}$. Seja $n-1 = 2^k q$ onde $k \geq 1$ e q é ímpar. Se n for primo, então:
ou

$$b^q \equiv 1 \ (mod\ n) \tag{4.1}$$

ou

$$b^{2^i q} \equiv -1 \ (mod\ n) \tag{4.2}$$

para algum $i = 0, ...k-1$.

Se o desenvolvimento dos algoritmos antigos se baseava na prática e na experiência acumulada pelas civilizações, as definições modernas incorporam elementos lógicos, computacionais e matemáticos que permitem analisar, comparar e implementar algoritmos de maneira sistemática. A noção de algoritmo passa, então, de uma técnica prática de resolução de problemas para um objeto matemático estruturado, capaz de ser estudado rigorosamente e aplicado tanto em teorias quanto em aplicações computacionais.

A reflexão sobre algoritmos na Matemática evidencia que a disciplina opera constantemente com procedimentos que se constroem etapa por etapa, a partir de regras bem definidas. E essa relação se estreita ainda mais quando falamos de algoritmos recursivos, visto que eles ocupam lugar de destaque tanto na matemática quando na computação.

A recursão é uma forma de definir processos em que um problema é resolvido por meio da resolução de versões menores de si mesmo. Em outras palavras, um algoritmo recursivo é aquele que chama a si próprio com parâmetros reduzidos, até alcançar um

[7] Trabalharemos tal algoritmo nos próximos capítulos.

caso básico que encerra o processo. Esse tipo de estrutura está profundamente ligado ao pensamento matemático, especialmente porque diversas definições formais da disciplina dependem de regras que operam justamente nesse mecanismo de repetição.

Do ponto de vista conceitual, um algoritmo recursivo é composto por dois elementos fundamentais:

1. caso base, que representa a forma mais simples do problema e pode ser resolvida diretamente;

2. passo recursivo, que estabelece como o problema pode ser reduzido a uma instância menor do mesmo problema.

Essa estrutura permite que problemas complexos sejam resolvidos de maneira organizada, decompondo-os em etapas que repetem a mesma lógica até que se alcance o caso trivial.

Historicamente, procedimentos recursivos aparecem em diversas práticas matemáticas muito antes da formalização moderna. O método de duplicações sucessivas utilizado pelos egípcios no Papiro de Rhind, por exemplo, pode ser interpretado como um algoritmo inicialmente iterativo, mas cuja lógica permite uma interpretação recursiva clara: multiplicar sucessivamente um valor por 2 nada mais é do que aplicar repetidamente a mesma regra de transformação. De maneira semelhante, o processo descrito no Problema 79 do Papiro de Rhind, em que um valor é multiplicado repetidas vezes por um mesmo fator, é uma ilustração perfeita de uma definição recursiva de sequência geométrica. Estes processos estão permeados em diversos episódios e aplicações ao longo da história.

4.3 Algoritmos Matemáticos Clássicos

Conceituar algoritmo nos permite compreender com maior clareza procedimentos matemáticos, que, apesar de conhecidos há séculos, só recentemente passaram a ser analisados sob a ótica algorítmica, algo que aparece naturalmente. Em álgebra, resolver uma equação implica seguir uma sequência lógica de transformações permitidas; em Geometria, construções com régua e compasso dependem de passos sequenciais rigorosos; em Cálculo, métodos numéricos como os de Newton ou Euler são algoritmos iterativos aplicados a funções; em Teoria dos Números, métodos como o Algoritmo de Euclides continuam fundamentais.

Tomando por referência Euclides, o procedimento exposto no Livro VII dos Elementos consiste basicamente em aplicar repetidas divisões sucessivas até que o resto seja zero. O divisor encontrado na última etapa é o máximo divisor comum. Dados

dois inteiros positivos a e b, o algoritmo procede por divisões sucessivas:

$$\begin{aligned} a &= bq_1 + r_1, \quad 0 \leq r_1 < b \\ b &= r_1 q_2 + r_2, \quad 0 \leq r_2 < r_1 \\ r_1 &= r_2 q_3 + r_3, \quad 0 \leq r_3 < r_2 \\ &\vdots \\ r_{n-1} &= r_n q_{n+1} + 0 \end{aligned} \tag{4.3}$$

O último resto não nulo r_n é o máximo divisor comum: $(a, b) = r_n$.

A prova de correção deste algoritmo, por exemplo, fundamenta-se na propriedade de que números que dividem a e b também dividem o resto da divisão de a por b. Esta afirmação evidencia como a lógica interna de um algoritmo pode ser demonstrada por argumentos puramente matemáticos, em muitos casos utilizando indução.

Muitas demonstrações matemáticas podem ser vistas como processos estruturados, nos quais cada passo deriva logicamente do anterior. Isso evidencia que a Matemática não é apenas um corpo de conhecimentos abstratos, mas também uma ciência de procedimentos envolvida por um campo histórico, repleta de métodos sistemáticos que buscam resolver problemas por meio de regras gerais e executáveis, muitos dos quais se tornaram indispensáveis tanto para o desenvolvimento teórico quanto para aplicações práticas.

Esse desenvolvimento prático advém da consolidação dos algoritmos que incorporam explicitamente estruturas recursivas e um algoritmo bastante visual e de comum utilização está envolvido no cálculo do fatorial, definido por

$$n! = \prod_{k=1}^{n} k = 1 \times 2 \times 3 \times \cdots \times n \tag{4.4}$$

sendo pois uma expressão simples, porém elegante. Sua definição já contém o algoritmo: *para calcular* $n!$*, basta calcular* $(n-1)!$ *e multiplicar por* n. Essa forma de descrição, na qual um problema é reduzido a uma versão menor de si mesmo, é típica de algoritmos recursivos e desempenha um papel central tanto na matemática quanto na computação. Além disso, podemos utilizar várias definições para o cálculo do fatorial, inclusive uma definição própria de recursividade,

$$n! = \begin{cases} 1 & \text{se } n = 0 \\ \displaystyle\prod_{k=1}^{n} k & \text{se } n \geq 1 \end{cases} \tag{4.5}$$

Outro processo amplamente utilizado, agora desde as civilizações antigas é a chamada multiplicação egípcia por duplicação, encontrada tanto no Papiro de Rhind

quanto em documentos babilônicos, dentre outros na MS3866, figura 14, um manuscrito em argila, provavelmente Larsa (antes da Babilônia), datado do século XIX a.C..

Figura 14 – Tabela de multiplicação - MS3866



Fonte: The Schøyen Collection (2023).

Esse algoritmo consiste em decompor um dos fatores em potências de dois e somar apenas as duplicações correspondentes. Embora rudimentar em aparência, esse procedimento é eficiente e utiliza apenas operações elementares, o que demonstra a engenhosidade dos algoritmos antigos, pois evitava contagem direta para números grandes e só era necessário conhecer os quadrados.

Em essência, trata-se de uma forma primitiva de exponenciação binária, utilizada até hoje para multiplicações rápidas em computadores, evidenciando como a matemática antiga já antecipava ideias que se tornaram fundamentais em algoritmos modernos.

A sequência de Fibonacci, difundida na Europa por Leonardo Fibonacci em seu Liber Abaci, oferece outro exemplo marcante dessa ideia. Nela, cada termo é definido como a soma dos dois anteriores:

$$F_0 = 0, \quad F_1 = 1 \quad \text{e} \quad F_n = F_{n-1} + F_{n-2} \text{ para } n \geq 2 \tag{4.6}$$

Partindo das condições iniciais é possível gerar sucessivamente os demais elementos da sequência mediante a aplicação repetida da regra de formação. A Tabela 2 apre-

senta o desenvolvimento dos dez primeiros termos da sequência, evidenciando o caráter iterativo e algorítmico desse processo.

Tabela 2 – Desenvolvimento da sequência de Fibonacci.

Termo	**Cálculo**	**Valor**
F_0	Condição inicial	0
F_1	Condição inicial	1
F_2	$F_1 + F_0 = 1 + 0$	1
F_3	$F_2 + F_1 = 1 + 1$	2
F_4	$F_3 + F_2 = 2 + 1$	3
F_5	$F_4 + F_3 = 3 + 2$	5
F_6	$F_5 + F_4 = 5 + 3$	8
F_7	$F_6 + F_5 = 8 + 5$	13
F_8	$F_7 + F_6 = 13 + 8$	21
F_9	$F_8 + F_7 = 21 + 13$	34
F_{10}	$F_9 + F_8 = 34 + 21$	55

Fonte: Elaborado pelo autor (2026).

Mais do que uma simples descrição numérica, essa relação define um algoritmo recursivo para construir a sequência, ilustrando como conceitos matemáticos podem naturalmente se traduzir em processos computacionais.

Esses algoritmos clássicos provenientes de diversas épocas e tradições matemáticas ilustram que a matemática não é apenas um conjunto de resultados, mas também de processos. A execução ordenada de passos, a transformação sistemática de problemas e a busca por métodos gerais são elementos estruturantes tanto da matemática quanto da ciência da computação. Assim, o estudo desses procedimentos fornece uma ponte natural entre a história, a formalização contemporânea e a implementação computacional, preparando o terreno para a análise das recorrências que trabalharemos.

Portanto, compreender algoritmos não significa apenas aprender técnicas computacionais, mas apropriar-se de uma forma de raciocínio profundamente conectada ao funcionamento interno da Matemática, perfazendo uma das bases estruturantes não apenas da computação moderna, mas da própria natureza da atividade matemática.

E essa relação torna-se ainda mais evidente quando se considera a eficiência dos processos. Uma recorrência pode ser vista como a versão matemática condensada de um algoritmo, descrevendo o raciocínio necessário, passo a passo, para obter os termos de uma sequência, expressando essa dinâmica por meio de uma lei estrutural, compacta e simbólica. Assim, algoritmo e recorrência não constituem áreas independentes; formam

um par conceitual complementar, no qual a matemática fornece a estrutura interna e a computação oferece o procedimento operacional.

Ao expressar métodos matemáticos como algoritmos, abre-se espaço para sua implementação em linguagens de programação, onde esses procedimentos podem ser executados de forma rápida, precisa e repetida. Nesse sentido, a linguagem Python surge como uma ferramenta particularmente adequada: possui sintaxe acessível, ampla difusão educacional e um conjunto robusto de bibliotecas que facilitam tanto a experimentação matemática quanto a análise computacional.

5 Python

A execução de um algoritmo em ambiente computacional não apenas evidencia seu caráter operacional, mas também permite discutir aspectos relacionados à eficiência, à recursão, ao número de operações realizadas e às diferentes estratégias empregadas na resolução de um problema. Entre as linguagens de programação contemporâneas, a linguagem Python tem alcançado ampla aceitação tanto no campo educacional quanto na pesquisa, sobretudo em razão de sua sintaxe relativamente simples, de sua legibilidade e da ampla comunidade científica que contribui para o desenvolvimento de bibliotecas, ambientes e aplicações.

Este capítulo apresenta conceitos fundamentais e algumas aplicações tecnológicas relacionadas à linguagem Python, bem como possibilidades pedagógicas de sua utilização em sala de aula. A discussão fundamenta-se nos estudos de Valente (1999), Lutz e Ascher (2007) e Downey (2016). Valente contribui para a compreensão do computador como recurso de aprendizagem, destacando a importância de práticas nas quais o estudante não apenas receba informações, mas construa, teste e revise suas próprias soluções. Lutz oferece uma base técnica para o estudo da linguagem Python, abordando sua estrutura, seus principais recursos e sua lógica de funcionamento. Downey, por sua vez, aproxima a aprendizagem da programação do desenvolvimento de formas organizadas de pensar, resolver problemas e elaborar procedimentos.

A partir desses referenciais, o Python é compreendido neste trabalho não apenas como uma ferramenta destinada à automatização de cálculos, mas como uma linguagem capaz de representar procedimentos matemáticos, tornar explícitas as etapas de um algoritmo e favorecer a análise de processos iterativos e recursivos. Ao traduzir uma relação matemática para um programa, o estudante precisa identificar os dados envolvidos, organizar as operações, estabelecer condições, definir repetições e interpretar os resultados produzidos.

Entretanto, a utilização da linguagem Python no ensino de Matemática não deve se limitar à apresentação de códigos prontos ou à obtenção imediata de respostas. Para que o recurso computacional assuma uma função pedagógica, é necessário que cada instrução do programa seja relacionada ao procedimento matemático correspondente. Dessa forma, o código deixa de ser apenas um produto técnico e passa a constituir uma representação do raciocínio desenvolvido na resolução do problema.

Assim, ao longo deste capítulo, serão apresentados os principais elementos da linguagem Python necessários à implementação dos algoritmos estudados, com destaque para variáveis, operadores, estruturas condicionais, laços de repetição, funções, iteração e recursão. Também serão discutidas aplicações que evidenciam as possibilidades de

integração entre programação e Matemática, preservando a centralidade dos conceitos e utilizando o ambiente computacional como espaço de exploração, experimentação e análise.

5.1 A linguagem Python

A linguagem Python foi criada por Guido Van Rossum, cientista da computação nascido nos Países Baixos, cuja trajetória acadêmica e profissional esteve fortemente ligada ao desenvolvimento de ferramentas computacionais voltadas para a pesquisa e a educação. Graduado em Matemática e Ciência da Computação pela Universidade de Amsterdã, Van Rossum atuou em diferentes centros de pesquisa ao longo de sua carreira, dedicando-se ao desenvolvimento de linguagens de programação, sistemas distribuídos e ambientes computacionais voltados à produção científica.

Figura 15 – Logomarca do Python.



Fonte: Disponível em: <https://www.python.org/>. Acesso em: 18 dez. 2025.

Segundo o próprio Van Rossum, o desenvolvimento do Python teve início no final da década de 1980, quando trabalhava no Centrum Wiskunde & Informatica (CWI), instituto nacional de pesquisa em matemática e ciência da computação dos Países Baixos. O projeto surgiu a partir da necessidade de uma linguagem que fosse ao mesmo tempo poderosa e simples, capaz de favorecer a produtividade dos programadores sem sacrificar a clareza do código. A primeira versão pública da linguagem foi disponibilizada em 1991, incorporando conceitos de modularidade, tratamento de exceções e estruturas de dados de alto nível, características que contribuíram para sua rápida aceitação na comunidade computacional (Van Rossum 2025).

O nome *Python* não foi escolhido em referência à serpente homônima, mas ao programa humorístico britânico *Monty Python's Flying Circus*[1], do qual Van Rossum era admirador. Essa escolha reflete uma das características marcantes da comunidade

[1] O grupo de comédia britânico Monty Python foi formado por Graham Chapman, John Cleese, Terry Gilliam, Eric Idle, Terry Jones e Michael Palin. Informações disponíveis em: <https://www.montypython.com/.>

Figura 16 – Guido van Rossum, criador da linguagem Python.



Fonte: Disponível em: <https://gvanrossum.github.io/bio.html>. Acesso em: 08 jun. 2026.

Python: a valorização de um ambiente colaborativo, acessível e menos formal do que aquele encontrado em muitas linguagens de programação tradicionais.

Ao longo das décadas seguintes, Python passou por contínuas evoluções, tornando-se uma das linguagens mais utilizadas em todo o mundo. Sua popularidade decorre, em grande medida, da combinação entre sintaxe simples, legibilidade e versatilidade, fatores que possibilitam sua utilização em áreas tão diversas quanto desenvolvimento de software, inteligência artificial, ciência de dados, modelagem matemática, computação científica e educação.

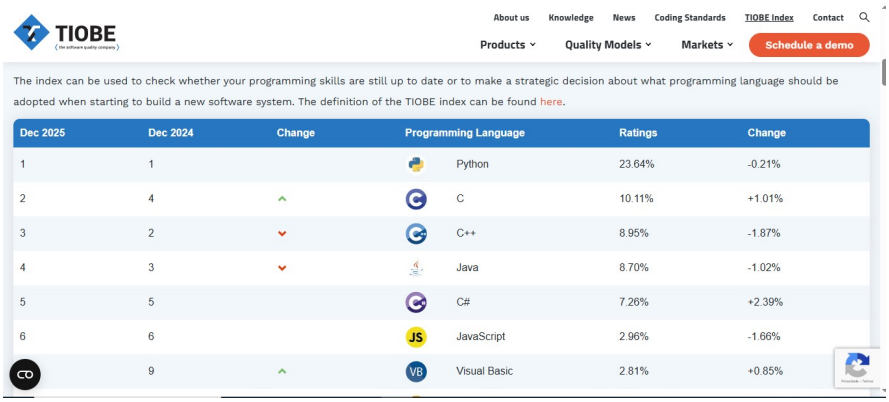
Um dos princípios fundamentais da linguagem encontra-se sintetizado no documento conhecido como *The Zen of Python*, elaborado por Tim Peters, no qual se destacam ideias como simplicidade, clareza e legibilidade. Esses princípios aproximam a programação do raciocínio matemático, permitindo que algoritmos sejam descritos de forma organizada e compreensível. Nesse sentido, Python apresenta-se como uma ferramenta particularmente adequada para o estudo de recorrências e algoritmos, pois possibilita a tradução direta de definições matemáticas para procedimentos computacionais executáveis.

No contexto desta pesquisa, a escolha da linguagem Python está diretamente relacionada à sua capacidade de representar processos iterativos e recursivos de maneira intuitiva. Como as recorrências lineares de primeira e segunda ordem são definidas por relações entre termos sucessivos de uma sequência, sua implementação computacional exige mecanismos capazes de reproduzir tais dependências. As estruturas de repeti-

ção e as funções recursivas disponíveis em Python permitem representar essas relações de forma próxima à notação matemática, favorecendo a compreensão dos conceitos envolvidos e a experimentação computacional dos modelos estudados.

O Python, 15, foi escolhido por ser uma linguagem generalista de alta legibilidade, com sintaxe inspirada em princípios de clareza e objetividade na sua aplicação. Ao apresentar essa linguagem, estamos dispondo de um dialeto popular e de grande aceitabilidade entre as linguagens de programação, visto que desde o ano de 2024 ele ocupa a 1ª posição no Índice TIOBE[2].

Figura 17 – Índice TIOBE de Comunidades de Programação.



Fonte: https://www.tiobe.com Acesso: 18 dez. 2025.

Isso reduz barreiras iniciais de aprendizagem e permite que o foco esteja sobre os conceitos matemáticos e algorítmicos, e não sobre a complexidade que outras linguagens possam exigir. Ao mesmo tempo, sua versatilidade possibilita construir desde implementações elementares, como o cálculo de termos de uma sequência definida por recorrência, até algoritmos avançados utilizados em áreas como criptografia, modelagem computacional e análise numérica.

Essas características tornam o Python particularmente adequado para a implementação e experimentação de algoritmos matemáticos, conforme discutido neste trabalho. Como afirmado no próprio tutorial oficial do Python, produzido pela Python Software Foundation (2025),

> Python é uma linguagem fácil de aprender e poderosa. Ela tem estruturas de dados de alto nível eficientes e uma abordagem simples mas efetiva de programação orientada a objetos. A sintaxe elegante e a tipagem dinâmica do Python, aliadas com sua natureza interpretativa, o tornam uma linguagem ideal para fazer scripts e desenvolvimento de aplicações rápidas em diversas áreas e na maioria das plataformas.

[2] O índice TIOBE de Comunidades de Programação é um indicador da popularidade de linguagens de programação. O índice é atualizado mensalmente. As classificações são baseadas no número de engenheiros qualificados em todo o mundo, em cursos e em fornecedores terceirizados.Ver: <www.tiobe.com/tiobe-index/>

Essa definição evidencia que Python combina simplicidade e poder de forma rara entre as linguagens de programação. A facilidade de aprendizado reduz a curva de entrada para estudantes que estão iniciando seus estudos em programação, enquanto o conjunto de recursos avançados, como estruturas de dados de alto nível e suporte a paradigmas variados, permite que a mesma linguagem seja utilizada para construir soluções complexas, incluindo implementações de algoritmos matemáticos, visualização de dados e resolução de problemas via programação.

Pretende-se apresentar a aproximação que existe entre a matemática e a experimentação computacional, algo que não é recente. Nesse sentido, o uso da linguagem Python ultrapassa o caráter instrumental, configurando-se como um ambiente de experimentação matemática. Papert (1985, p. 23), destaca que "o computador pode ser um instrumento para pensar, para explorar ideias e para aprender fazendo, e não apenas uma máquina para transmitir informações". Assim, a programação passa a atuar como mediadora do pensamento matemático, favorecendo a construção ativa do conhecimento. Essa perspectiva encontra eco em iniciativas de ensino atuais, por exemplo, Valente (1999) destaca que a programação pode funcionar como meio para promover pensamento lógico e resolução estruturada de problemas, elementos intrinsecamente associados ao raciocínio matemático.

No contexto da Educação Matemática, ela oferece meios de ilustrar propriedades de funções, de simular modelos iterativos, de resolver sistemas, de representar graficamente fenômenos matemáticos e de analisar recorrências de maneira dinâmica. Em vez de lidar apenas com a abstração simbólica, o estudante passa a observar o comportamento dos processos matemáticos em tempo real, o que amplia sua compreensão conceitual.

Um aspecto particularmente relevante para este trabalho é a capacidade do Python de representar processos iterativos e recursivos de forma clara e intuitiva. Como as recorrências lineares são definidas por relações entre termos sucessivos de uma sequência, sua implementação computacional exige mecanismos capazes de reproduzir tais dependências. Nesse contexto, estruturas de repetição e funções recursivas presentes na linguagem Python oferecem recursos adequados para transformar definições matemáticas abstratas em procedimentos executáveis, permitindo observar, testar e analisar o comportamento das sequências geradas.

Além da implementação direta das recorrências, a utilização do Python possibilita comparar diferentes estratégias algorítmicas para um mesmo problema. Por exemplo, uma sequência pode ser calculada por meio de abordagens iterativas ou recursivas, cada uma apresentando características próprias relacionadas ao número de operações realizadas, ao consumo de memória e ao tempo de execução. Essa perspectiva amplia a compreensão do estudante sobre a relação existente entre matemática, algoritmos e computação, evidenciando que diferentes métodos podem conduzir ao mesmo resultado,

mas com desempenhos distintos.

Ao transformar ideias matemáticas em código executável, a linguagem em Python reforça o caráter ativo do conhecimento matemático, permitindo ao estudante assumir o papel de investigador como é proposto pelos documentos que regem a educação no Brasil (BNCC, PCNs), e não apenas de receptor de métodos prontos. O interesse maior não é formar profissionais na área da programação, mas proporcionar ao estudante experiências que favoreçam o desenvolvimento do pensamento lógico, da resolução de problemas e da modelagem matemática., como aprender a pensar, aprender a resolver problemas complexos, subdividir os problemas, tornando-os menores para uma melhor clareza e visualização, preparando seu pensamento para formar um bom profissional em qualquer área.

Dessa forma, o estudo da linguagem Python neste trabalho não se limita à apresentação de conceitos básicos de programação. O interesse está em compreender como estruturas computacionais podem ser utilizadas para representar objetos matemáticos definidos por recorrências, permitindo a experimentação, a visualização de padrões e a análise de algoritmos. Nos tópicos seguintes serão apresentados os elementos da linguagem necessários para a implementação dos modelos matemáticos desenvolvidos ao longo desta pesquisa.

5.2 Estrutura Básica da Linguagem Python

A linguagem Python destaca-se no cenário atual por sua sintaxe simples, legível e próxima da linguagem natural, o que favorece tanto o aprendizado inicial de programação quanto a aplicação de algoritmos em contextos matemáticos. Python foi criado por Guido van Rossum, um programador holandês, no final dos anos 80 e início dos anos 90. A primeira versão do Python, a 0.9.0, foi lançada em 1991. Guido van Rossum nomeou a linguagem em homenagem ao grupo de comédia britânico Monty Python, do qual era um grande fã. Seu desenho minimalista, fundamentado na clareza e na expressividade, contribui para que o estudante concentre sua atenção no raciocínio lógico e nos procedimentos necessários à resolução de problemas, sem que a complexidade da linguagem se torne um obstáculo.

Essa aproximação entre pensamento algorítmico e prática matemática é amplamente reconhecida na literatura brasileira sobre tecnologias digitais e educação. Valente (1999, p. 32), ao tratar do papel do computador como instrumento cognitivo, afirma explicitamente que “a atividade de programar coloca o aluno diante da necessidade de explicitar o que sabe, de organizar seu pensamento e de refletir sobre os procedimentos que utiliza para resolver um problema”. Essa perspectiva dialoga diretamente com os objetivos da educação matemática, especialmente no que se refere à formalização de

estratégias, à análise de processos e à construção de relações entre conceitos.

Barrére e Souza (2010, p. 48) reforçam essa articulação entre programação e pensamento matemático ao destacar que "programar significa decompor problemas, formular estratégias e organizar sequências lógicas de ações, articulando conhecimentos matemáticos, computacionais e de raciocínio abstrato". Assim, a introdução de linguagens como Python no estudo de algoritmos e recorrências não apenas operacionaliza procedimentos, mas também mobiliza capacidades cognitivas centrais para a aprendizagem matemática contemporânea.

Embora a linguagem Python possua uma ampla variedade de recursos e bibliotecas, a implementação dos algoritmos e recorrências abordados neste trabalho requer apenas um conjunto reduzido de elementos fundamentais. Entre eles destacam-se as variáveis, as estruturas condicionais, os mecanismos de repetição, as funções e as estruturas de dados básicas. O Quadro 1 apresenta uma síntese desses componentes e de suas principais finalidades no contexto da programação.

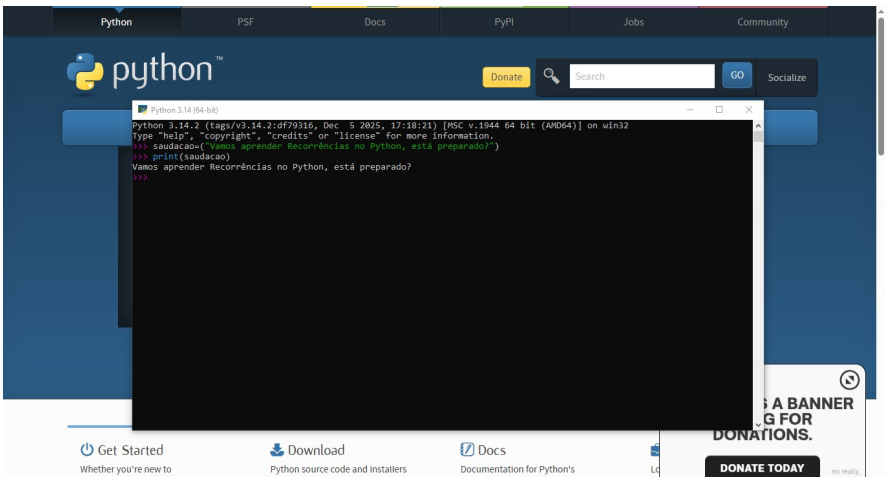
Quadro 1 – Principais elementos da linguagem Python utilizados neste trabalho.

Elemento	**Sintaxe básica**	**Finalidade**
Variável	`x = 10`	Armazenar valores durante a execução do programa.
Condicional	`if`	Permitir a tomada de decisões com base em condições lógicas.
Laço de repetição	`for` e `while`	Executar instruções repetidamente de forma controlada.
Função	`def`	Organizar e reutilizar procedimentos computacionais.
Lista	`[]`	Armazenar e manipular coleções de valores.
Recursão	Função chamando a si própria	Representar definições recursivas e recorrências matemáticas.

Fonte: Elaborado pelo autor (2026).

Dentre os elementos fundamentais da linguagem, um dos mais importantes é o conceito de variável, que representa uma posição de memória capaz de armazenar valores durante a execução do programa. Em Python, a atribuição é dinâmica: não é necessário declarar previamente o tipo da variável, e o próprio interpretador identifica o tipo com base no valor atribuído, além de que são necessárias menos linhas de comando do que outras linguagens de programação. Essas características simplificam a transcrição de ideias matemáticas para o ambiente computacional. Por exemplo, ao implementar

Figura 18 – Interface do Python com o site como plano de funto.



Fonte: Autor(2025)

uma sequência definida por recorrência, o estudante pode atribuir diretamente valores iniciais sem lidar com especificações de tipos complexos.

Exemplo 34. *A atribuição de valores em Python pode ser realizada de forma simples:*

```
1  a = 1
2  b = 1
```

Nesse exemplo, as variáveis `a` *e* `b` *armazenam os valores iniciais da sequência de Fibonacci.*

Outro aspecto estruturante da linguagem são as expressões aritméticas, que seguem uma ordem de precedência compatível com a notação matemática usual. Expressões como `"(3*x + 5)**2"` mantêm uma correspondência natural com a escrita algébrica presente no ensino médio e superior, o que reduz a barreira entre a abstração matemática e sua implementação computacional. Essa proximidade torna Python uma ferramenta apropriada para a experimentação numérica, permitindo verificar conjecturas, simular processos e analisar comportamentos de funções ou sequências.

A lógica condicional, expressa principalmente pelo comando `"if"`, possibilita que o programa tome decisões com base em propriedades matemáticas dos valores envolvidos. Testes como paridade, divisibilidade ou comparações entre números inteiros são diretamente traduzíveis na sintaxe da linguagem. Essa estrutura é essencial para algoritmos clássicos, como o de Euclides para o cálculo do máximo divisor comum, cujas etapas dependem de condições específicas.

Exemplo 35. *A estrutura condicional permite que o programa tome decisões com base em uma condição lógica.*

```
n = 10

if n % 2 == 0:
    print("Número par")
```

Nesse exemplo, o operador `%` *calcula o resto da divisão de* `n` *por 2. Caso o resto seja zero, o programa identifica que o número é par.*

As estruturas de repetição, como os laços `"for"` e `"while"`, permitem a execução sequencial e controlada de blocos de código, representando matematicamente processos iterativos. Essas estruturas são fundamentais para o cálculo de termos sucessivos de uma sequência definida por regra de formação, para a implementação de métodos numéricos e para a execução repetida de algoritmos que dependem de ciclos de refinamento. A clareza da sintaxe dos laços em Python torna evidente o encadeamento lógico das etapas envolvidas na iteração.

Exemplo 36. *O laço* `for` *é utilizado quando o número de repetições é conhecido previamente.*

```
for i in range(5):
    print(i)
```

O comando imprime os números de 0 a 4, executando o bloco de código cinco vezes. Estruturas desse tipo são frequentemente utilizadas para calcular termos sucessivos de uma sequência numérica.

Exemplo 37. *O laço* `while` *executa um bloco de código enquanto uma determinada condição for verdadeira.*

```
n = 1

while n <= 5:
    print(n)
    n = n + 1
```

Nesse caso, os números de 1 a 5 são exibidos na tela. A variável `n` *é atualizada a cada iteração até que a condição deixe de ser satisfeita.*

Além disso, Python oferece um modelo simples e poderoso para definição de funções, que encapsulam procedimentos e permitem modularizar o código. A palavra-chave `"def"` estabelece a estrutura básica de uma função, seguida por parâmetros que representam suas entradas. Essa organização reflete a definição formal de algoritmo discutida por Cormen et al. (2009), segundo a qual um algoritmo é um procedimento bem definido que recebe entradas, processa-as por meio de etapas claras e produz uma saída. Funções em Python podem representar desde funções matemáticas simples, como $f(x) = x^2 + 1$, até processos recursivos complexos, como a geração de sequências definidas por recorrências lineares ou algoritmos baseados em divisão e conquista

Exemplo 38. *Funções permitem organizar algoritmos em blocos reutilizáveis.*

```
1  def quadrado(x):
2      return x**2
3
4  print(quadrado(4))
```

A função `quadrado` *recebe um valor como entrada e retorna seu quadrado. Ao ser chamada com o valor 4, produz como saída o número 16.*

A recursão, em particular, desempenha papel crucial na ligação entre matemática e programação. Python permite que uma função invoque a si própria, possibilitando representar de maneira direta definições matemáticas recursivas, como a sequência de Fibonacci ou o próprio algoritmo de Euclides. A correspondência entre definição formal e implementação recursiva favorece a compreensão das estruturas matemáticas subjacentes.

Por fim, as estruturas de dados, em especial as listas, constituem um recurso indispensável para organizar e armazenar conjuntos de valores. Listas permitem representar sequências numéricas, manter históricos de iterações ou armazenar resultados intermediários de algoritmos. Sua flexibilidade e facilidade de manipulação tornam-nas particularmente úteis em experimentações matemáticas e simulações computacionais.

Exemplo 39. *As listas permitem armazenar diversos valores em uma única estrutura.*

```
1  fibonacci = [1, 1, 2, 3, 5]
2
3  print(fibonacci[3])
```

Nesse exemplo, a lista armazena os primeiros termos da sequência de Fibonacci. O comando imprime o elemento de índice 3, que corresponde ao valor 3.

Entre os conceitos apresentados, destacam-se especialmente as variáveis, os laços de repetição e as funções, uma vez que esses elementos constituem a base para a implementação computacional de recorrências lineares. Em particular, as estruturas iterativas permitem calcular sucessivamente os termos de uma sequência, enquanto as funções recursivas possibilitam representar diretamente definições matemáticas nas quais cada termo depende de valores anteriores. Essa correspondência entre a formulação matemática e sua implementação em Python será explorada nos tópicos seguintes.

A compreensão desses elementos básicos nos coloca em frente à implementação de algoritmos matemáticos em Python, permitindo articular a lógica sintática da linguagem com alguns dos princípios algorítmicos. Nos tópicos seguintes, essa integração será aprofundada por meio do estudo da recursão, da iteração e de algoritmos clássicos aplicados à matemática, enfatizando sua relação com as recorrências lineares analisadas anteriormente.

5.3 Iteração e Recursão em Python.

A construção de algoritmos matemáticos frequentemente envolve a repetição de procedimentos ou a definição de etapas sucessivas que dependem de resultados previamente calculados. Em Python, essas duas formas de organização computacional são representadas principalmente por meio da iteração e da recursão, mecanismos fundamentais para a implementação de algoritmos clássicos e de recorrências lineares.

Enquanto a iteração baseia-se na execução repetida de um conjunto de instruções por meio de estruturas de repetição, a recursão consiste na definição de funções que invocam a si próprias para resolver versões menores de um mesmo problema. Ambas as abordagens permitem traduzir processos matemáticos em algoritmos executáveis, embora apresentem características distintas quanto à clareza, eficiência e consumo de recursos computacionais.

Python disponibiliza duas estruturas principais de repetição: o laço `for`, adequado para repetições baseadas em contagem, e o laço `while`, apropriado para repetições condicionais. Essas estruturas permitem representar de forma clara algoritmos matemáticos iterativos e constituem ferramentas essenciais para a implementação de sequências, recorrências e métodos numéricos.

Quadro 2 – Características gerais da iteração e da recursão.

Iteração	**Recursão**
Utiliza laços de repetição.	Utiliza chamadas da própria função.
Consome menos memória.	Consome mais memória devido à pilha de execução.
Geralmente apresenta melhor desempenho.	Frequentemente produz códigos mais próximos da definição matemática.
Adequada para processos sequenciais.	Adequada para estruturas recursivas e recorrências.

Fonte: Elaborado pelo autor (2026).

No exemplo abaixo,40, o valor final é construído passo a passo. Essa estrutura reflete a própria definição combinatória do fatorial, que envolve multiplicações sucessivas. A clareza da implementação permite que o estudante visualize o processo e compreenda o comportamento multiplicativo da função.

Exemplo 40 (Cálculo Iterativo do Fatorial.)**.** *O fatorial é uma das funções matemáticas mais frequentemente utilizadas para ilustrar processos iterativos.*

```
1  def fatorial_iter(n):
2      resultado = 1
3
4      for i in range(1, n + 1):
5          resultado *= i
6
7      return resultado
```

No exemplo 40, o valor final é construído progressivamente por meio de multiplicações sucessivas. Essa implementação reflete a definição matemática do fatorial e ilustra como estruturas iterativas podem representar procedimentos algorítmicos compostos por etapas repetitivas. A clareza dessa abordagem permite que o estudante visualize o processo de construção do resultado e compreenda a lógica subjacente ao algoritmo.

Por outro lado, a recursão consiste na definição de uma função que chama a si própria, exemplo 41, de modo que a solução de um problema seja construída com base em instâncias menores do mesmo problema. Essa ideia é fundamental na matemática moderna, especialmente na formulação de recorrências, nas definições de sequências e em algoritmos de divisão e conquista. A recursão permite que definições matemáticas sejam transcritas para Python quase literalmente. Uma função recursiva deve conter:

- um caso base, que encerra a recursão;
- uma chamada recursiva, que reduz o problema.

As recorrências lineares constituem um exemplo natural de aplicação da iteração. Considere a sequência definida por

$$a_n = 3a_{n-1} - 2$$

A obtenção de cada novo termo depende apenas do termo imediatamente anterior, o que permite implementar o cálculo por meio de estruturas iterativas simples.

```
def recorrencia_iter(a0, n):
    a = a0
    for _ in range(n):
        a = 3*a - 2
    return a
```

Esse tipo de implementação é particularmente útil para análises empíricas de comportamento assintótico, identificação de padrões e verificação de conjecturas. Embora a iteração seja frequentemente a alternativa mais eficiente para o cálculo de recorrências, existe outra abordagem igualmente importante do ponto de vista matemático: a recursão. Nesse paradigma, a solução de um problema é construída a partir de soluções de versões menores do próprio problema, reproduzindo de forma bastante fiel diversas definições matemáticas.

Exemplo 41 (Função recursiva para soma)**.** *O código abaixo apresenta uma função recursiva que calcula a soma dos primeiros n números naturais.*

```
def soma_recursiva(num):
    if num == 1:  # Caso base
        return num
    return num + soma_recursiva(num - 1)  # Caso recursivo

print(soma_recursiva(4))  # 4 + 3 + 2 + 1 = 10
```

A recursão, neste caso, é uma representação natural da ideia matemática. Tanto a recursão quanto a iteração podem resolver muitos dos mesmos problemas, mas cada uma tem seus prós e contras. Para que possamos escolher melhor o que mais se adequa a necessidade de usuário, observemos o que podemos diferenciar ambos e aplicar ao nosso propósito:

Tabela 3 – Comparação entre abordagens recursivas e iterativas

Quando usar RECURSÃO	**Quando usar ITERAÇÃO**
Quando o problema pode ser naturalmente dividido em subproblemas menores.	Para problemas que exigem tarefas repetitivas sem dividir o problema em subproblemas menores.
Para problemas que envolvem estruturas de dados hierárquicas, como árvores.	Para tarefas de desempenho crítico, em que a recursão profunda pode resultar em alto uso de memória.
Quando a solução em forma recursiva é mais intuitiva e leva a um código mais limpo.	Quando você trabalha com grandes conjuntos de dados que podem exceder o limite de recursão.
Quando a legibilidade do código é priorizada em relação à otimização do desempenho.	Quando o uso da memória é a principal preocupação.
Ao trabalhar com dados de profundidade desconhecida (como analisar JSON ou XML aninhados).	Para problemas que são naturalmente sequenciais em vez de hierárquicos.

Exemplo 42 (Implementação recursiva da sequência de Fibonacci)**.** *A sequência de Fibonacci pode ser implementada diretamente a partir de sua definição recursiva.*

```
1  def fibonacci(n):
2      if n <= 2:
3          return 1
4      return fibonacci(n-1) + fibonacci(n-2)
```

A implementação apresentada reproduz fielmente a definição matemática da sequência de Fibonacci, na qual cada termo é obtido pela soma dos dois termos anteriores. Essa proximidade entre a formulação matemática e sua implementação computacional constitui uma das principais vantagens pedagógicas da recursão.

Do ponto de vista computacional, a iteração geralmente apresenta maior eficiência em Python, especialmente para valores elevados, uma vez que evita o custo associado ao empilhamento de chamadas recursivas e às limitações impostas pela profundidade máxima de recursão da linguagem. Entretanto, a recursão possui elevado valor conceitual e pedagógico, pois permite representar de maneira natural definições matemáticas baseadas em recorrências.

Assim, iteração e recursão não devem ser entendidas como abordagens concorrentes, mas como estratégias complementares para a resolução de problemas. A escolha entre

uma e outra depende da natureza do algoritmo, da estrutura matemática envolvida e dos objetivos didáticos da atividade proposta. A possibilidade de analisar um mesmo problema sob diferentes perspectivas de resolução favorece o desenvolvimento do pensamento matemático e computacional. Nesse sentido, Gontijo et al. (2019, p. 44) destaca que a flexibilidade de pensamento está relacionada com "a facilidade e a liberdade para mudar de uma operação mental para outra ou, ainda, com a habilidade de analisar um problema sob diferentes pontos de vista, observando características específicas e identificando semelhanças e divergências entre os elementos envolvidos". Sob essa perspectiva, a compreensão das abordagens iterativa e recursiva amplia o repertório de estratégias disponíveis ao estudante, permitindo-lhe escolher a forma mais adequada para modelar e resolver problemas matemáticos e computacionais.

5.4 Modelagem Computacional de Recorrências em Python

A modelagem computacional pode ser compreendida como um processo de representação, implementação e investigação de situações matemáticas por meio de recursos computacionais. Nesse processo, parte-se de um problema ou fenômeno, identificam-se as variáveis envolvidas, estabelecem-se relações matemáticas entre elas e, posteriormente, essas relações são traduzidas para uma estrutura algorítmica capaz de ser executada pelo computador. Assim, o modelo computacional não se reduz ao código produzido, pois compreende também a formulação matemática, a escolha do algoritmo, a execução, a interpretação dos resultados e a validação da solução.

De maneira esquemática, esse processo pode ser representado por

situação-problema
↓
modelo matemático
↓
algoritmo
↓
programa
↓
resultados
↓
interpretação

No contexto das recorrências, a modelagem computacional assume particular relevância, uma vez que uma relação recorrente descreve um processo que se desenvolve em

etapas sucessivas. Cada novo valor é determinado a partir de um ou mais valores anteriormente calculados, característica que encontra correspondência direta com as estruturas iterativas e recursivas presentes nas linguagens de programação. Desse modo, ao implementar uma recorrência em Python, não se realiza apenas a automatização de cálculos: torna-se possível observar a evolução da sequência, analisar sua convergência, comparar diferentes valores iniciais, investigar regularidades e verificar computacionalmente propriedades previamente estudadas.

A utilização de Python no ensino de Matemática tem sido defendida por sua possibilidade de aproximar os conteúdos matemáticos das tecnologias digitais e de favorecer a construção de formas organizadas de raciocínio. Para Borges (2021), a implementação de problemas matemáticos nessa linguagem pode contribuir para o desenvolvimento do raciocínio lógico estruturado, ao mesmo tempo que possibilita uma releitura computacional de conteúdos da Educação Básica. Nessa perspectiva, o estudante precisa identificar os dados do problema, estabelecer as operações necessárias, organizar as etapas de resolução e transformá-las em instruções que possam ser executadas pelo computador. Entretanto, o potencial pedagógico da programação não se encontra apenas na rapidez com que o computador apresenta um resultado.

Conforme discutem Santos e Menezes (2025, p. 3), a utilização indiscriminada de funções e bibliotecas prontas pode reduzir as oportunidades de compreensão dos procedimentos matemáticos envolvidos. Segundo os autores, "o uso de funções prontas pode limitar a exploração dos processos implícitos no desenvolvimento de um algoritmo", e essa limitação ocorre porque o estudante pode obter uma resposta correta sem necessariamente compreender a sequência de operações, as condições consideradas e as relações matemáticas que fundamentam o resultado. Por essa razão, sobretudo nas atividades introdutórias, é importante que a construção dos programas ocorra de maneira progressiva, explicitando-se a correspondência entre cada etapa do procedimento matemático e cada comando utilizado no código. Desse modo, o recurso computacional deixa de funcionar apenas como meio de automatização e passa a contribuir para a compreensão do processo matemático representado.

Essa abordagem também se relaciona ao Pensamento Computacional. De acordo com Wing (2006, p. 33), esse conceito "baseia-se no poder e nos limites de processos de computação, quer eles sejam executados por um ser humano ou por uma máquina". Essa compreensão evidencia que o Pensamento Computacional não se restringe ao uso de computadores, mas envolve a formulação de problemas e a organização de procedimentos que possam ser executados de maneira clara e sistemática. Em perspectiva semelhante, Brackmann (2017, p. 29) afirma que o Pensamento Computacional constitui-se em uma

> capacidade criativa, crítica e estratégica humana de saber utilizar os

> fundamentos da Computação, nas mais diversas áreas do conhecimento, com a finalidade de identificar e resolver problemas, de maneira individual ou colaborativa, através de passos claros, de tal forma que uma pessoa ou uma máquina possam executá-los eficazmente.

Na implementação de recorrências, esses elementos podem ser identificados quando o estudante decompõe o problema em etapas, reconhece a regularidade que relaciona os termos, seleciona as informações essenciais e organiza um procedimento para gerar os valores seguintes. Assim, o estudo das recorrências favorece a articulação entre pensamento matemático e Pensamento Computacional, uma vez que exige a identificação de padrões, a formulação de relações e a construção de algoritmos capazes de representar o processo de geração dos termos.

Além disso, as atividades de programação podem assumir uma dimensão investigativa. Ao escrever, executar e revisar um programa, o estudante acompanha o desenvolvimento da solução e pode confrontar o resultado obtido com aquilo que havia previsto. Nesse sentido, Binotto, Maltempi e R. A. B. d. Silva (2023, p. 18) afirmam que

> ao codificar e testar um programa, o autor acompanha todo o processo executado pela máquina. Isso permite a simulação de situações, depuração, reflexão sobre a solução, incremento e inovação, além de proporcionar respostas interativas e rápidas.

O erro, nesse contexto, não deve ser entendido exclusivamente como uma falha, mas como parte do processo de construção do conhecimento. Quando o programa não produz o resultado esperado, o estudante precisa revisar tanto as instruções do código quanto o raciocínio matemático que orientou sua elaboração. A depuração transforma-se, portanto, em uma oportunidade para verificar hipóteses, identificar inconsistências, reorganizar procedimentos e compreender com maior profundidade as relações matemáticas empregadas.

Essa dinâmica também pode favorecer uma postura mais persistente e investigativa diante dos problemas. Na experiência analisada pelos autores, os participantes demonstraram motivação, envolvimento na resolução da atividade e familiaridade com a abordagem de tentativa e erro Binotto, Maltempi e R. A. B. d. Silva 2023. Desse modo, a programação pode criar um ambiente em que testar, analisar, corrigir e reformular deixam de ser ações secundárias e passam a integrar o próprio processo de aprendizagem matemática.

Dessa forma, a implementação computacional de uma recorrência pode ser organizada em três níveis complementares:

1. **formulação matemática:** identificação da recorrência, das condições iniciais e das propriedades que serão investigadas;

2. **formulação algorítmica:** organização das etapas necessárias para calcular os termos, definir a condição de parada e armazenar os resultados;

3. **implementação computacional:** tradução do algoritmo para Python, seguida da execução, análise e validação dos valores produzidos.

Essa organização impede que o código seja apresentado como uma sequência pronta de comandos sem significado. O estudante deve compreender que as variáveis utilizadas no programa representam objetos matemáticos, que as estruturas de repetição correspondem à produção sucessiva dos termos e que as condições de parada estão associadas ao objetivo matemático da investigação.

5.4.1 O método babilônico para aproximação de raízes quadradas

O método babilônico é um procedimento clássico para obter aproximações da raiz quadrada de um número positivo. Sua estrutura é recorrente, pois cada nova aproximação é calculada a partir do valor obtido na etapa anterior. Dado $S > 0$ e escolhida uma aproximação inicial $x_0 > 0$, define-se

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right), \qquad n \geq 0. \tag{5.1}$$

A relação (5.1) constitui uma recorrência não linear de primeira ordem, uma vez que cada termo depende apenas do termo imediatamente anterior. A repetição do procedimento produz aproximações sucessivamente mais próximas de $\sqrt{S}$.

Para representar esse processo computacionalmente, é necessário definir o valor inicial, aplicar repetidamente a relação de recorrência e estabelecer uma condição de parada. Uma implementação em Python pode ser feita da seguinte maneira:

```
1  def raiz_babilonica(S, tolerancia=1e-10):
2      if S < 0:
3          raise ValueError("O número deve ser não negativo.")
4
5      if S == 0:
6          return 0.0
7
8      x = S / 2
9
10     while True:
11         proximo = 0.5 * (x + S / x)
12
13         if abs(proximo - x) < tolerancia:
```

```
            return proximo

        x = proximo


resultado = raiz_babilonica(10)
print(resultado)
```

Nesse código, a variável `x` representa a aproximação atual, enquanto a instrução

```
proximo = 0.5 * (x + S / x)
```

corresponde diretamente à recorrência matemática. A repetição é encerrada quando a diferença entre duas aproximações consecutivas se torna menor do que a tolerância estabelecida.

Para $S = 10$, o programa fornece aproximadamente

$$\sqrt{10} \approx 3{,}162277660168. \tag{5.2}$$

A aplicação permite observar como uma relação recorrente pode ser organizada na forma de algoritmo e executada em Python. O código torna visível o processo de geração das aproximações, preservando a compreensão matemática do método em vez de apenas fornecer o resultado por meio de uma função pronta.

5.4.2 A Torre de Hanói em Python

A Torre de Hanói é um problema clássico formado por três hastes e um conjunto de discos de tamanhos distintos, inicialmente organizados em uma única haste, do maior para o menor. O objetivo consiste em transferir todos os discos para outra haste, movimentando apenas um disco por vez e sem colocar um disco maior sobre um menor.

Para transferir uma torre com n discos, é necessário mover inicialmente os $n-1$ discos superiores para uma haste auxiliar, transferir o maior disco para a haste de destino e, por fim, mover novamente os $n-1$ discos. Assim, o número mínimo de movimentos é definido pela recorrência

$$T_n = 2T_{n-1} + 1, \qquad T_0 = 0. \tag{5.3}$$

A solução dessa recorrência é

$$T_n = 2^n - 1. \tag{5.4}$$

Em Python, o procedimento pode ser representado por uma função recursiva:

```python
def torre_de_hanoi(n, origem, destino, auxiliar):
    """Versão iterativa da Torre de Hanói.

    A lógica da recursão é simulada explicitamente por meio
    de uma pilha, evitando chamadas recursivas da função.
    Retorna o número de movimentos e imprime cada passo.
    """
    # Pilha: (n, origem, destino, auxiliar, estado)
    # estado = 0 -> mover n-1 para auxiliar
    # estado = 1 -> mover o maior disco
    # estado = 2 -> mover n-1 de auxiliar para destino
    pilha = [(n, origem, destino, auxiliar, 0)]
    movimentos = 0

    while pilha:
        n, orig, dest, aux, estado = pilha.pop()

        if n == 1:
            print(f"Mover disco 1 de {orig} para {dest}")
            movimentos += 1
        else:
            if estado == 0:
                # Empilha o estado 1 para processar o disco maior
                pilha.append((n, orig, dest, aux, 1))

                # Simula a primeira chamada recursiva:
                # mover n-1 discos de orig para aux
                pilha.append((n-1, orig, aux, dest, 0))

            elif estado == 1:
                # Move o maior disco
                print(f"Mover disco {n} de {orig} para {dest}")
                movimentos += 1

                # Simula a segunda chamada recursiva:
                # mover n-1 discos de aux para dest
                pilha.append((n-1, aux, dest, orig, 0))

    return movimentos
```

```
40
41
42  try:
43      discos = int(input("Digite o número de discos: "))
44
45      if discos <= 0:
46          print("O número de discos deve ser um inteiro positivo.")
47      else:
48          print("\nSequência de movimentos:\n")
49
50          total = torre_de_hanoi(discos, "A", "C", "B")
51
52          print(f"\nQuantidade de movimentos realizados: {total}")
53          print(f"Quantidade mínima pela fórmula: {2**discos - 1}")
54
55  except ValueError:
56      print("Entrada inválida. Digite apenas um número inteiro.")
```

Nesse programa, o caso `n == 1` representa a situação inicial mais simples. Para $n > 1$, a função resolve duas vezes o mesmo problema com $n - 1$ discos, reproduzindo computacionalmente a relação recorrente (5.3).

Para três discos, são necessários

$$T_3 = 2^3 - 1 = 7$$

movimentos.

Podemos, ainda, explorar a mesma recorrência de um modo mais simples

```
1   def torre_de_hanoi(n, origem, destino, auxiliar):
2       if n == 1:
3           print(f"Mover disco 1 de {origem} para {destino}")
4       else:
5           torre_de_hanoi(n-1, origem, auxiliar, destino)
6           print(f"Mover disco {n} de {origem} para {destino}")
7           torre_de_hanoi(n-1, auxiliar, destino, origem)
8
9
10  discos = int(input("Digite o número de discos: "))
11
12  print("\nSequência de movimentos:\n")
13  torre_de_hanoi(discos, "A", "C", "B")
```

```
14
15  print(f"\nQuantidade mínima de movimentos: {2**discos - 1}")
```

Essa aplicação permite observar a correspondência entre a definição recorrente do problema, o algoritmo recursivo e sua implementação em Python.

5.4.3 O problema da pizza de Steiner em Python

O problema da pizza de Steiner consiste em determinar o número máximo de regiões em que uma pizza pode ser dividida por n cortes retos. Para que essa quantidade seja máxima, cada novo corte deve intersectar todos os cortes anteriores em pontos distintos e não pode passar por um ponto de interseção já existente.

Ao realizar o n-ésimo corte, são acrescentadas n novas regiões. Assim, a quantidade máxima de partes pode ser definida pela recorrência

$$P_n = P_{n-1} + n, \qquad P_0 = 1. \tag{5.5}$$

A solução dessa recorrência é

$$P_n = 1 + \frac{n(n+1)}{2}. \tag{5.6}$$

Em Python, a quantidade máxima de regiões pode ser calculada iterativamente:

```
1   def pizza_de_steiner(cortes):
2       """
3       Calcula o número máximo de regiões de uma pizza (círculo)
4       ao fazer 'cortes' cortes retos.
5       Fórmula: 1 + n*(n+1)/2
6       """
7       regioes = 1
8
9       for n in range(1, cortes + 1):
10          regioes = regioes + n
11
12      return regioes
13
14  try:
15      n = int(input("Digite o número de cortes: "))
16      if n < 0:
17          print("Digite um número não negativo.")
18      else:
```

```
        print(f"Regiões máximas: {pizza_de_steiner(n)}")
except ValueError:
    print("Entrada inválida. Digite apenas um número inteiro.")
```

Nesse código, a variável `regioes` representa a quantidade de partes obtidas até a etapa atual. A instrução

```
regioes = regioes + n
```

corresponde diretamente à relação recorrente (5.5).

Para cinco cortes, o programa fornece

$$P_5 = 1 + \frac{5 \cdot 6}{2} = 16.$$

A implementação permite acompanhar como cada novo corte acrescenta regiões à quantidade anterior, evidenciando a relação entre a recorrência matemática, o algoritmo iterativo e o código em Python.

5.4.4 O problema do queijo de Steiner em Python

O problema do queijo de Steiner consiste em determinar o número máximo de partes em que um sólido pode ser dividido por n cortes planos. Para alcançar essa quantidade máxima, cada novo plano deve intersectar os planos anteriores de modo a formar o maior número possível de novas regiões.

O n-ésimo corte acrescenta uma quantidade de partes correspondente ao número máximo de regiões em que os $n-1$ planos anteriores dividem o novo plano. Assim, a recorrência pode ser escrita como

$$Q_n = Q_{n-1} + 1 + \frac{n(n-1)}{2}, \qquad Q_0 = 1. \tag{5.7}$$

A solução geral dessa recorrência é

$$Q_n = 1 + n + \binom{n}{2} + \binom{n}{3} = \frac{n^3 + 5n + 6}{6}. \tag{5.8}$$

Em Python, o número máximo de partes pode ser calculado iterativamente:

```
def queijo_de_steiner(cortes):
    partes = 1

    for n in range(1, cortes + 1):
        novas_partes = 1 + n * (n - 1) // 2
```

```
        partes = partes + novas_partes

    return partes


print(queijo_de_steiner(5))
```

Nesse código, a variável `partes` representa a quantidade acumulada de regiões, enquanto `novas_partes` indica quantas regiões são acrescentadas pelo novo corte. Para cinco cortes, obtém-se

$$Q_5 = \frac{5^3 + 5 \cdot 5 + 6}{6} = 26.$$

A implementação evidencia como a quantidade de partes produzida em cada etapa depende dos cortes anteriormente realizados, estabelecendo a relação entre o modelo recorrente, o algoritmo e sua representação em Python.

5.4.5 A sequência de Fibonacci em Python

A sequência de Fibonacci constitui um dos exemplos clássicos de recorrência de segunda ordem. Seus termos são obtidos pela soma dos dois termos imediatamente anteriores, sendo definida por

$$F_0 = 0, \qquad F_1 = 1$$

e

$$F_n = F_{n-1} + F_{n-2}, \qquad n \geq 2.$$

Os primeiros termos da sequência são

$$0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21, \ldots \tag{5.9}$$

Em Python, a sequência pode ser gerada de maneira iterativa:

```
def fibonacci(quantidade):
    if quantidade <= 0:
        return []

    termos = []
    anterior, atual = 0, 1

```

```
    for _ in range(quantidade):
        termos.append(anterior)
        anterior, atual = atual, anterior + atual

    return termos

try:
    n = int(input("Digite a quantidade de termos: "))
    print(f"Fibonacci: {fibonacci(n)}")
except ValueError:
    print("Entrada inválida. Digite um número inteiro.")
```

A execução do programa fornece

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Nesse código, as variáveis `anterior` e `atual` armazenam os dois termos necessários para calcular o valor seguinte. A instrução

```
anterior, atual = atual, anterior + atual
```

corresponde diretamente à recorrência (5.4.5), pois atualiza simultaneamente os termos utilizados na próxima etapa.

A implementação evidencia que, diferentemente das recorrências de primeira ordem, o cálculo de cada termo da sequência de Fibonacci depende de dois valores anteriores. O código permite acompanhar a geração dos termos e observar o crescimento da sequência a partir de suas condições iniciais.

5.4.6 Crescimento populacional em Python

Uma população que cresce a uma taxa percentual constante e recebe, a cada período, uma quantidade fixa de novos indivíduos pode ser representada por uma recorrência linear de primeira ordem. Suponha que uma população inicial seja de P_0 indivíduos, cresça 3% ao ano e receba mais 200 indivíduos em cada período. Nesse caso,

$$P_{n+1} = 1{,}03P_n + 200, \qquad P_0 = P_{\text{inicial}}. \tag{5.10}$$

Em Python, esse processo pode ser implementado da seguinte maneira:

```
def crescimento_populacional(pop_inicial, anos, taxa=1.03,
↪   imigracao=200):
```

```
    pop = pop_inicial
    valores = [round(pop, 2)]  # arredonda para 2 casas

    for _ in range(anos):
        pop = taxa * pop + imigracao
        valores.append(round(pop, 2))

    return valores

# Exemplo de uso
print(crescimento_populacional(10000, 5))
```

A execução do programa fornece valores aproximados para a população em cada ano. A instrução

```
populacao = 1.03 * populacao + 200
```

corresponde diretamente à recorrência (5.10). O fator 1,03 representa o crescimento percentual, enquanto o valor 200 corresponde ao acréscimo fixo anual.

Essa aplicação permite relacionar as recorrências a situações de crescimento ao longo do tempo. Além disso, possibilita alterar a população inicial, a taxa de crescimento e o acréscimo anual, analisando como essas mudanças interferem nos valores gerados pelo modelo.

5.4.7 Aplicação financeira em Python

Situações envolvendo juros e depósitos periódicos também podem ser representadas por recorrências. Considere um capital inicial C_0 aplicado a uma taxa de juros i por período, com um depósito fixo d realizado ao final de cada etapa. O saldo pode ser descrito por

$$C_{n+1} = (1+i)C_n + d, \qquad C_0 = C_{\text{inicial}}. \tag{5.11}$$

Em Python, a evolução do capital pode ser calculada da seguinte maneira:

```
def evolucao_capital(capital_inicial, taxa,
                     deposito, periodos):
    """
    Calcula a evolução do capital com depósitos periódicos.
    Retorna uma lista com os saldos arredondados para 2 casas.
    """
```

```
    capital = capital_inicial
    valores = [round(capital, 2)]

    for _ in range(periodos):
        capital = (1 + taxa) * capital + deposito
        valores.append(round(capital, 2))

    return valores

saldos = evolucao_capital(1000, 0.01, 200, 12)
print(saldos)
```

Nesse exemplo, considera-se um capital inicial de `R$ 1.000,00`, uma taxa de juros de 1% ao mês e depósitos mensais de `R$ 200,00`. A instrução

```
capital = (1 + taxa) * capital + deposito
```

corresponde diretamente à recorrência (5.11). Em cada período, os juros incidem sobre o saldo anterior e, em seguida, acrescenta-se o depósito fixo.

Essa aplicação permite analisar situações próximas ao cotidiano dos estudantes, como poupança, investimentos, empréstimos e planejamento financeiro. Também possibilita modificar a taxa, o valor dos depósitos e a quantidade de períodos, observando como esses parâmetros interferem na evolução do capital.

6 Considerações Finais

Com o presente trabalho, espera-se que as contribuições estabelecidas possam fazer com que a utilização das Tecnologias de Informação e Comunicação sejam aplicadas com o propósito de articulação de conhecimentos matemáticos, pensamento algorítmico e modelagem computacional por meio da linguagem Python. A proposta foi construída a partir da compreensão de que o estudo de recorrências pode ultrapassar a apresentação de fórmulas e procedimentos isolados, constituindo-se em um campo fértil para o desenvolvimento do raciocínio lógico, da identificação de padrões, da organização de estratégias de resolução e da interpretação de processos que evoluem de forma sucessiva.

Ao longo do trabalho, buscou-se apresentar uma fundamentação histórica, matemática e educacional, perpassando diversos períodos históricos e os matemáticos que tomaram o campo matemático e computacional como objetos de estudo. O estudo das sequências, das progressões e das recorrências evidenciou que muitos processos matemáticos podem ser compreendidos por meio de relações entre termos consecutivos, o que favorece a aproximação com a lógica dos algoritmos. Nesse sentido, a recorrência não foi tratada apenas como uma forma de definir sequências, mas como uma estrutura matemática capaz de representar processos iterativos, fenômenos de crescimento, problemas clássicos e situações modeladas computacionalmente.

A abordagem histórica dos algoritmos mostrou que a ideia de organizar procedimentos em etapas antecede o surgimento dos computadores. Métodos desenvolvidos por diferentes povos e matemáticos, como o método babilônico para aproximação de raízes, os procedimentos associados a Euclides, Fibonacci e outros algoritmos clássicos, revelam que a Matemática sempre recorreu a processos sistemáticos de resolução. A programação, nesse contexto, não representa uma ruptura com a tradição matemática, mas uma forma contemporânea de expressar, executar e analisar procedimentos que já possuem uma estrutura algorítmica. Além disso, colocando uma linguagem de programação, como a Python, pode-se mostrar a possibilidade de representar algorítimos clássicos ou atuais com a utilização de poucas instruções, favorecendo assim uma aproximação entre a linguagem simbólica da Matemática e a linguagem computacional, permitindo que o estudante identifique, no código, as etapas do procedimento que está sendo executado. Variáveis, estruturas condicionais, laços de repetição, funções, iteração e recursão podem ser relacionadas diretamente a conceitos matemáticos presentes no estudo de sequências e recorrências.

A discussão desenvolvida neste trabalho também evidenciou que a simples utilização do Python não garante, por si só, uma aprendizagem matemática mais significativa, ou

a efetiva utilização de meios de programação e tecnologias. O potencial pedagógico da programação depende da maneira como as atividades são organizadas e dos objetivos que orientam seu uso, isto é, quando o código é apresentado apenas como uma sequência pronta de comandos, o estudante pode limitar-se a reproduzi-lo, sem compreender as relações matemáticas envolvidas. De modo semelhante, o uso excessivo de funções prontas e bibliotecas pode conduzir à obtenção de resultados sem que os processos internos sejam compreendidos.

Por essa razão, defendemos uma abordagem em que a construção do algoritmo preceda ou acompanhe sua implementação computacional. O estudante deve ser levado a identificar os dados de entrada, compreender a relação de recorrência, definir as condições iniciais, estabelecer a condição de parada, organizar as etapas do cálculo e interpretar os resultados obtidos. Dessa forma, o programa deixa de ser apenas um instrumento de automatização e passa a atuar como uma representação executável do raciocínio matemático.

Outro aspecto importante é a relação estabelecida entre programação e Pensamento Computacional. A decomposição de problemas, o reconhecimento de padrões, a abstração e a elaboração de algoritmos estão presentes tanto na atividade matemática quanto na atividade computacional, devendo serem levados para as atividades desenvolvidas no ambiente escolar. No estudo de recorrências, esses elementos podem ser explorados quando o estudante identifica a regularidade entre os termos, separa o problema em etapas, seleciona as informações essenciais e organiza um procedimento que possa ser executado de maneira sistemática, aproximando-se de uma perspectiva construcionista, na medida em que o estudante é incentivado a construir objetos computacionais, testar soluções, identificar erros e reformular estratégias.

A depuração de um programa pode assumir valor pedagógico quando o erro é compreendido como parte do processo de aprendizagem. Ao corrigir o código, o estudante precisa revisar não apenas a sintaxe utilizada, mas também o raciocínio matemático que orientou sua construção. Assim, a programação pode favorecer uma postura mais investigativa, na qual testar, conjecturar, comparar e revisar tornam-se ações centrais.

Como sugestão para continuidade, faz-se pertinente investigar a comparação entre abordagens iterativas e recursivas, o papel da depuração na aprendizagem matemática, o impacto do uso de diferentes ambientes computacionais e as adaptações necessárias para contextos com acesso limitado a computadores. Outra possibilidade consiste em ampliar a proposta para outros conteúdos matemáticos que apresentem estrutura algorítmica, como funções, matrizes, combinatória, probabilidade, métodos numéricos e Matemática Financeira. As recorrências também podem ser relacionadas a modelos populacionais, problemas de crescimento, juros compostos, sistemas dinâmicos discretos e outros fenômenos, fortalecendo a articulação entre Matemática, tecnologia e realidade.

Em suma, este trabalho reafirma que o uso de algoritmos e da programação em Python pode constituir uma possibilidade relevante para o ensino de sequências e recorrências lineares, desde que sua utilização seja orientada pela compreensão dos conceitos matemáticos e pela construção consciente dos procedimentos. O principal valor da programação não está apenas em acelerar cálculos ou apresentar respostas, mas em tornar visíveis as etapas do raciocínio, permitir a experimentação e oferecer novas formas de representar e investigar ideias matemáticas.

Espera-se, portanto, que a proposta elaborada contribua para ampliar as possibilidades metodológicas no ensino de Matemática e incentive professores e estudantes a compreenderem a programação não como um conteúdo isolado, mas como uma linguagem capaz de dialogar com o pensamento matemático. Ao aproximar recorrências, algoritmos e Python, o trabalho procura colaborar para uma prática pedagógica em que o estudante participe ativamente da construção, da análise e da interpretação dos processos matemáticos.

Bibliografia

Barrére, Eduardo e Danilo Medeiros de Souza (2010). "Pensamento computacional e ensino de programação". Em: *Revista Brasileira de Informática na Educação* 18.2, pp. 45–56.

Berlinski, David (2002). *O advento do algoritmo: a ideia que governa o mundo*. Tradução de Leila Ferreira de Souza Mendes. Título original: *The Advent of the Algorithm*. São Paulo: Globo. ISBN: 85-250-3539-4.

Binotto, Rosane Rossato, Marcus Vinicius Maltempi e Rogério Aparecido Batista da Silva (2023). "Potencialidades da programação em Python para o desenvolvimento do pensamento criativo em Matemática". Em: *Zetetiké* 31, pp. 1–22. DOI: <10.20396/zet.v31i00.8672180>.

Borges, Brunno de Oliveira (2021). "A linguagem Python como ferramenta no ensino básico". Dissertação (Mestrado Profissional em Matemática em Rede Nacional). Uberaba: Universidade Federal do Triângulo Mineiro, p. 69.

Brackmann, Christian P. (2017). "Desenvolvimento do Pensamento Computacional Através de Atividades Desplugadas na Educação Básica". Tese de Doutorado em Informática na Educação – Cinted. Tese de Doutorado. Porto Alegre: Universidade Federal do Rio Grande do Sul.

Brasil (2018). *Base Nacional Comum Curricular*. Disponível em: <http://basenacionalcomum.mec.gov.br>. Acesso em: 19 jan. 2026. Brasília: Ministério da Educação.

— (2023). *Lei nº 14.533, de 11 de janeiro de 2023. Institui a Política Nacional de Educação Digital*. <https://www.planalto.gov.br/ccivil_03/_ato2023-2026/2023/lei/L14533.htm>. Acesso em: 19 mar. 2026.

Cormen, Thomas H. et al. (2009). *Introduction to Algorithms*. 3ª ed. Cambridge, MA: MIT Press. ISBN: 978-0-262-03384-8.

Dasgupta, Sanjoy, Christos Papadimitriou e Umesh Vazirani (2010). *Algoritmos*. Trad. por Guilherme Albuquerque Pinto. E-book. Recurso eletrônico. Versão impressa publicada em 2009. Porto Alegre: AMGH. ISBN: 9788563308535.

Downey, Allen B. (2016). *Pense em Python: Pense como um cientista da computação*. 2ª ed. Tradução de: Think Python: how to think like a computer scientist. São Paulo: Novatec. ISBN: 978-85-7522-508-0.

Eves, Howard (2011). *Introdução à História da Matemática*. Trad. por Hygino H. Domingues. 5ª ed. Campinas, SP: Editora da Unicamp.

Fagundes, Léa (2007). *O professor deve tornar-se um construtor de inovações: entrevista*. Entrevista. Midiativa. URL: <http://www.midiativa.org.br/index.php/educadores/layout/set/print/content/view/full/1053/> (acesso em 19/01/2026).

Forbellone, A. L. (2005). *Lógica de programação: a construção de algoritmos e estruturas de dados*. São Paulo: Pearson.

Gomes, Carlos A., Iesus C. Diniz e Roberto Teodoro (2024). *Matemática discreta: conjuntos, recorrências, combinatória e probabilidade*. 3ª ed. Vol. 1. São Paulo: LF Editorial. ISBN: 978-65-5563-517-1.

Gontijo, C. H. et al. (2019). *Criatividade em matemática: conceitos, metodologias e avaliação*. Brasília, DF: Editora da UnB.

Grimaldi, Ralph P. (2004). *Matemática Discreta e Combinatória: Uma Introdução*. Português. 5ª ed. Tradução de: Discrete and Combinatorial Mathematics, 5th ed., 1999. Rio de Janeiro: Editora Campus. 644 pp. ISBN: 85-352-1017-4.

Katz, Victor J. (2009). *A History of Mathematics: An Introduction*. 3ª ed. Boston: Addison-Wesley.

Knuth, Donald Ervin (1997). *The Art of Computer Programming: Fundamental Algorithms*. 3ª ed. Vol. 1. Reading, MA: Addison-Wesley. ISBN: 0-201-89683-4.

Lutz, Mark e David Ascher (2007). *Aprendendo Python*. 2ª ed. Tradução de: Learning Python. Porto Alegre: Bookman. ISBN: 978-85-7780-013-1.

Machado, Nílson José (2013). *Matemática e realidade: das concepções às ações docentes*. 8ª ed. São Paulo: Cortez.

Manzano, José Augusto N. G. e Jayr Figueiredo de Oliveira (2019). *Algoritmos: lógica para desenvolvimento de programação de computadores*. 29ª ed. São Paulo: Érica, p. 368. ISBN: 978-85-365-3146-5.

Morgado, Augusto César e João Bosco Pitombeira de Carvalho (2022). *Análise Combinatória e Probabilidade*. 9ª ed. Rio de Janeiro: SBM - Sociedade Brasileira de Matemática. ISBN: 978-85-85818-92-5.

Papert, Seymour (1985). *Logo: computadores e educação*. Tradução de José Armando Félix da Silva. São Paulo: Brasiliense. ISBN: 85-11-01026-4.

— (2008). *A máquina das crianças: repensando a escola na era da informática*. Porto Alegre: Artmed.

Python Software Foundation (2025). *Tutorial do Python*. Acesso em: 18 dez. 2025. URL: <https://docs.python.org/pt-br/3/tutorial/>.

Rosen, Kenneth H. (2010). *Matemática Discreta e Suas Aplicações*. Português. Trad. por Carlos A. de Oliveira e Marlene S. de Oliveira. 6ª ed. Tradução de: Discrete Mathematics and Its Applications, 6th ed., 2007. Porto Alegre, RS: AMGH Editora Ltda. 1016 pp. ISBN: 978-85-8055-110-0.

Santos, Jailson França dos e Ilton Ferreira de Menezes (set. de 2025). "Entre facilidades e aprendizagens: conectando habilidades matemáticas através da programação em Python". Em: *Revista Internacional de Pesquisa em Educação Matemática* 15.3, pp. 1–19. DOI: <10.37001/ripem.v15i3.4528>.

Sedgewick, Robert e Kevin Wayne (2011). *Algorithms*. 4th. Upper Saddle River, NJ: Addison-Wesley, p. 955. ISBN: 9780321573513.

Silva, R. B. da et al. (2016). "Educação, linguagem e novas tecnologias da comunicação: por uma aprendizagem significativa". Em: *Anais do Encontro Virtual de Documentação em Software Livre, 13. e Congresso Internacional de Linguagem e Tecnologia On-line, 10.* Disponível em: <http://evidosol.textolivre.org/>. Acesso em: 19 jan. 2026. Belo Horizonte: UFMG. URL: <http://evidosol.textolivre.org/>.

Valente, José Armando (1999). *O Computador na Sociedade do Conhecimento*. Campinas: UNICAMP.

Van Rossum, Guido (2025). *Biography*. URL: <https://gvanrossum.github.io/bio.html> (acesso em 08/06/2026).

Wing, Jeannette M. (2006). "Computational Thinking". Em: *Communications of the ACM* 49.3, pp. 33–35.

Apêndices

APÊNDICE A – Soluções particulares para recorrências lineares não homogêneas

Quadro 3 – Algumas formas usuais de soluções particulares para recorrências lineares não homogêneas

Termo não homogêneo $f(n)$	**Forma sugerida para a solução particular**
c (constante)	$a_n^{(p)} = A$
$kn + b$	$a_n^{(p)} = An + B$
$an^2 + bn + c$	$a_n^{(p)} = An^2 + Bn + C$
r^n	$a_n^{(p)} = Ar^n$
nr^n	$a_n^{(p)} = (An + B)r^n$
n^2r^n	$a_n^{(p)} = (An^2 + Bn + C)r^n$
$\text{sen}(\alpha n)$ ou $\cos(\alpha n)$	$a_n^{(p)} = A\,\text{sen}(\alpha n) + B\cos(\alpha n)$
$P_m(n)r^n$	$a_n^{(p)} = Q_m(n)r^n$

Em que $P_m(n)$ e $Q_m(n)$ representam polinômios de grau m.