ISEP

DEPARTAMENTO DE MATEMÁTICA
Instituto Superior de Engenharia do Porto

Sebenta de

Matemática Discreta

Ano letivo 2024/2025

do curso de

Licenciatura em Engenharia Informática

Ana Moura
aim@isep.ipp.pt

Departamento de Matemática
Instituto Superior de Engenharia do Porto

O Hotel Infinito, situado em Infinitópolis, tem um número infinito de quartos, um por cada número natural, estando numerados de 1 até ao infinito. Um dia, estando os quartos todos ocupados, chegou um viajante que pretendia pernoitar nesse hotel.

Empregado: "Desculpe, mas vai ser impossível hospedá-lo. Estamos lotados."

O viajante pediu para falar com o gerente do hotel, para tentar arranjar uma solução, visto que não existia mais nenhum hotel em Infinitópolis.

Gerente: "Realmente, não temos vagas, mas eu vou arranjar-lhe um quarto."

Mandou mudar todos os hóspedes de cada quarto para o quarto com o número seguinte, deixando assim o quarto 1 vago para o viajante. No dia seguinte, apareceram 5 casais em lua-de-mel pedindo quartos. O empregado pediu aos hóspedes para mudarem de cada quarto para o quarto com um número superior em 5 unidades, deixando os primeiro 5 quartos vagos para os 5 casais recém-chegados. No dia seguinte, chegou uma excursão com um número infinito de turistas, tantos quantos os números naturais, pedindo alojamento. O empregado ficou atrapalhado:

Empregado: "Sr. Gerente, já percebi como é que no Hotel Infinito se resolve o problema sempre que há um número finito de novos hóspedes, mas será possível arranjar espaço para um número infinito deles, isto é, tantos quantos os quartos que temos já ocupados?"

Gerente: "Claro! Mudamos os hóspedes de cada quarto para outro com um número duas vezes superior!"

Empregado: "Claro! Desse modo mudamos os hóspedes todos para os quartos com número par, o que deixa vagos todos os quartos com número ímpar, que são em número infinito, tantos quantos os turistas!"

O Hotel de Hilbert.[1]

E perguntamos ao leitor: Qual foi a sugestão do gerente à chegada de mais um número infinito de turistas, mas, desta vez, tantos quantos os números racionais? E se fossem tantos quantos os números reais? Neste último caso, o gerente já não encontrou uma solução.

[1] *O Hotel de Hilbert*, adaptado de http://www.educ.fc.ul.pt/docentes/opombo/seminario/cantor/hotelinf.htm (página atualmente indisponível).

Conteúdo

Introdução

Estes apontamentos, baseados nos capítulos correspondentes das referências bibliográficas, têm como objetivo apoiar os estudantes da Licenciatura em Engenharia Informática do ISEP, na unidade curricular de Matemática Discreta. Esta unidade curricular insere-se no 2º semestre do 1º ano do plano curricular do curso de Licenciatura em Engenharia Informática. Pretende introduzir aos alunos as estruturas discretas, desenvolver o raciocínio lógico e matemático e o pensamento algorítmico, fundamentos matemáticos essenciais para a continuação dos estudos na área de Ciência da Computação e Informática.

Devido às imensas aplicações em várias áreas das ciências, engenharia e indústria, a Matemática Discreta tem-se revelado, nas últimas décadas, uma das áreas da Matemática de interesse crescente, e os tópicos que a constituem são muito vastos. Tornou-se, portanto, necessária a escolha dos temas a abordar neste curso. Assim, a unidade curricular está dividida em quatro capítulos: Linguagem Matemática e Lógica, Introdução às Estratégias de Demonstração, Análise de Algoritmos e Grafos. No primeiro capítulo, será feita uma breve revisão à Teoria de Conjuntos e a sua tradução na linguagem da Lógica Proposicional e da Lógica de Predicados, que servirão de alicerces para o capítulo seguinte de métodos de prova, dos quais se destaca o método de indução. O primeiro capítulo termina ainda com a introdução das estruturas discretas: as relações binárias. Será dado especial destaque aos Grafos, constituindo o quarto capítulo deste texto, começando por se introduzir alguns conceitos dessa teoria, e passando para o estudo dos principais algoritmos em grafos. Finalmente, que por questões de planificação e avaliação na unidade curricular ocupa o terceiro tema a ser abordado, será introduzida a noção de Complexidade Algorítmica, com o estudo da complexidade temporal do pior caso. E porque a Análise Combinatória é crucial para resolver muitos e diferentes problemas, como exemplo, na determinação da complexidade de alguns algoritmos, optou-se por colocar, em apêndice, uma breve revisão das técnicas de contagem fundamentais. Também em apêndice, é feita uma breve visita às relações de recorrência, que servem de suporte ao estudo da complexidade de algoritmos recursivos.

A preparação desta unidade curricular, cuja regência iniciei no ano letivo de 2011/2012, tem contado com as sugestões e as ajudas dos restantes colegas que comigo trabalharam ou têm trabalhado, João Matos, Ivone Amorim, Manuel Cruz, Stella Abreu, Amélia Caldeira, Teresa Araújo, Alexandra Gavina, Luís Roque e Tiago Paiva, a quem devo o meu agradecimento. Ainda à Alexandra, o meu muito obrigada pela incansável ajuda a redesenhar algumas das imagens em LaTeX, num *sprint* final para a submissão destas notas. É muito gratificante quando trabalhamos em equipas colaborativas. E à Elsa Gomes, diretora do curso, pelo apoio, reforço positivo e incentivo na progressão e inovação dos métodos de ensino da unidade curricular, tornando-a, ano após ano, mais desafiante.

Aos alunos, a quem destino estes apontamentos, espero que encontrem aqui a motivação necessária para que a aprendizagem da Matemática Discreta seja uma tarefa cativante.

Bom trabalho!

CAPíTULO 1

Linguagem Matemática e Lógica

Qual o valor lógico da proposição "Esta proposição é falsa."?
E o que acontece ao nariz do Pinóquio quando ele diz "O meu nariz está a crescer."?

1. Noções básicas de Teoria de Conjuntos

Um *conjunto* é uma coleção de objetos, de qualquer natureza, que se designam por *elementos* do conjunto. Denotamos um conjunto por uma letra maiúscula, A, um elemento de um conjunto por uma letra minúscula, a, e escrevemos $a \in A$ e $a \notin A$ para representar que, respetivamente, "a pertence a A"e "a não pertence a A". Usamos as chavetas para explicitar todos os elementos de um conjunto, por exemplo, $A = \{a, b, c\}$ apresenta os três elementos, a, b e c do conjunto A. Quando não é conveniente explicitá-los a todos, podemos usar as reticências, $A = \{1, 2, \ldots, 100\}$ ou $B = \{1, 2, \ldots\}$, respetivamente, quando o número de elementos é finito ou não.

O conjunto sem elementos designa-se por *conjunto vazio* e denota-se por $\emptyset$ (ou por $\{\ \}$).

Dados dois conjuntos A e B, dizemos que A é *igual a* B, e escrevemos $A = B$, se A e B têm exatamente os mesmos elementos. Dizemos que A é *subconjunto* de B, ou que A *está contido* em B, e escrevemos $A \subseteq B$, se todos os elementos de A pertencerem a B. Caso contrário, escrevemos $A \nsubseteq B$, se pelo menos um elemento de A não pertence a B. Dizemos ainda que A é subconjunto *próprio* de B (ou está contido *estritamente*), e escrevemos $A \subsetneq B$, se A é um subconjunto de$B$ diferente de B.

Exemplo 1.1. Sejam $A = \{1, 2, 3, 4, 5\}$, $B = \{1, 3, 5\}$ e $C = \{1, 2, 3\}$. Temos

$$2 \in A, \text{ mas } 2 \notin B;$$
$$B \subseteq A \text{ e } B \subsetneq A;$$
$$B \nsubseteq C \text{ e } C \nsubseteq B.$$

Num dado contexto, designamos um conjunto U de conjunto *universal*, se todos os conjuntos considerados nesse contexto forem subconjuntos de U.

Os conjuntos utilizados frequentemente em Matemática, que em vários contextos serão conjuntos universais, são os listados em seguida:

$$\mathbb{N} = \{\text{números naturais}\} = \{0, 1, 2, 3, \ldots\},$$
$$\mathbb{Z} = \{\text{números inteiros}\},$$
$$\mathbb{Q} = \{\text{números racionais}\},$$
$$\mathbb{R} = \{\text{números reais}\},$$
$$\mathbb{C} = \{\text{números complexos}\}.$$

Há duas formas de descrever os elementos de um conjunto: em *extensão*, com a indicação exaustiva de todos os elementos; em *compreensão*, com a indicação de uma propriedade satisfeita por todos os elementos do conjunto, e apenas esses.

EXEMPLO 1.2. O conjunto $A = \{n \in \mathbb{N} : n < 5\}$, representado em compreensão, pode ser representado em extensão por $A = \{0, 1, 2, 3, 4\}$.

O conjunto $B = \{x \in \mathbb{R} : x^2 - 1 = 0\}$, representado em compreensão, pode ser representado em extensão por $B = \{-1, 1\}$.

O conjunto $C = \{x \in \mathbb{Z} : x \text{ é um número par positivo}\}$, representado em compreensão, pode ser representado em extensão por $C = \{2, 4, 6, \ldots\}$.

Num conjunto, a ordem pela qual os elementos são apresentados é irrelevante, bem como o número de vezes que cada elemento ocorre. No entanto, por vezes pretendemos mesmo que a ordem seja tida em conta.

Chamamos *par ordenado* (a, b) a uma sequência de dois elementos com uma determinada ordem, sendo a o primeiro elemento da sequência e b o segundo elemento. Dados dois pares ordenados, (a, b) e (c, d), dizemos que estes elementos são iguais se e só se $a = c$ e $b = d$.

Dados dois conjuntos A e B, definimos o *produto cartesiano* de A por B, e denotamos por $A \times B$, pelo conjunto formado por todos os pares ordenados (a, b), com $a \in A$ e $b \in B$, ou seja,

$$A \times B = \{(a, b) : a \in A \text{ e } b \in B\}.$$

O produto cartesiano pode estender-se, naturalmente, a um número finito de fatores. Assim, o produto cartesiano dos conjuntos $A_1, A_2, \ldots, A_n$, denotado por $A_1 \times A_2 \times \ldots \times A_n$, é o conjunto de todos os n-uplos ordenados $(a_1, a_2, \ldots, a_n)$ tais que $a_i \in A_i$, para todo o $i = 1, 2, \ldots, n$.

EXEMPLO 1.3. Sejam $A = \{1, 2\}$ e $B = \{1, 3, 5\}$. Temos:

$$A \times B = \{(1,1), (1,3), (1,5), (2,1), (2,3), (2,5)\};$$
$$B \times A = \{(1,1), (1,2), (3,1), (3,2), (5,1), (5,2)\};$$
$$(1,3) \neq (3,1),\ (1,3) \in A \times B \text{ e } (1,3) \notin B \times A.$$

Dado um conjunto A, ao conjunto de todos os subconjuntos de A denominamos de *conjunto das partes* de A e denotamos por $\mathcal{P}(A)$. Em linguagem matemática escrevemos

$$\mathcal{P}(A) = \{x : x \subseteq A\}.$$

EXEMPLO 1.4. Seja $A = \{1, 2, 3\}$. Então:

$$\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, A\}.$$

Ao número de elementos distintos de um conjunto A damos o nome de *cardinal* de A e denotamos por $|A|$. Obviamente, $|A| \geq 0$. Se o cardinal de A for um número natural, então dizemos que A é um *conjunto finito*. Caso contrário, diz-se um *conjunto infinito*.

LEMA 1.1. *Sejam A e B conjuntos. Tem-se:*

1. $|A \times B| = |A| \times |B|$;
2. $|\mathcal{P}(A)| = 2^{|A|}$.

É usual representar os conjuntos pictoricamente, usando os *diagramas de Venn* (ver Figura 1).

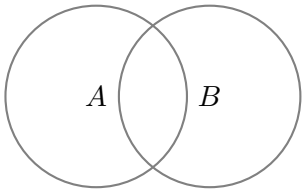


FIGURA 1. Diagrama de Venn.

Operações sobre conjuntos. Sejam U o conjunto universal e A e B dois conjuntos arbitrários. Em seguida, definimos as principais operações sobre conjuntos: a interseção, a união, a diferença e a complementação.

A *interseção* de A e B denota-se por $A \cap B$ e é definida pelo conjunto:

$$A \cap B = \{x \in U : x \in A \text{ e } x \in B\}.$$

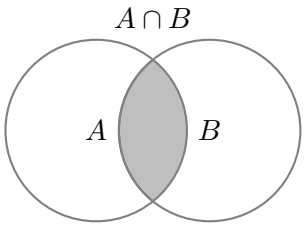


FIGURA 2. Intersecção de dois conjuntos.

A *união* de A e B denota-se por $A \cup B$ e é definida pelo conjunto:

$$A \cup B = \{x \in U : x \in A \text{ ou } x \in B\}.$$

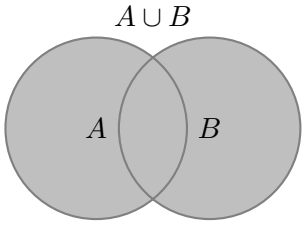


FIGURA 3. União de dois conjuntos.

A *diferença* entre A e B (ou o *complementar* de B em A) denota-se por $A \backslash B$ (ou por $A - B$) e é definida pelo conjunto:

$$A \backslash B = \{x \in U : x \in A \text{ e } x \notin B\}.$$

O *complementar* de A denota-se por A^c e é definido pelo conjunto:

$$A^c = \{x \in U : x \notin A\}.$$

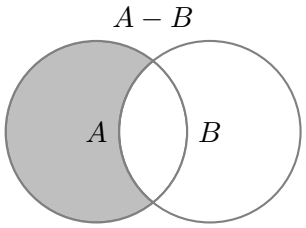


FIGURA 4. Diferença de dois conjuntos.

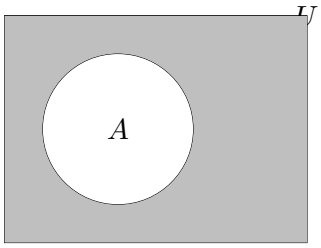


FIGURA 5. Complementar de um conjunto.

EXEMPLO 1.5. Sejam $A = \{1, 2, 3, 5\}$, $B = \{1, 3\}$ e $C = \{2, 4, 5\}$. Temos:

$$A \cap B = \{1, 3\};$$
$$B \cap C = \emptyset;$$
$$A \cup B = A;$$
$$A \cup C = \{1, 2, 3, 4, 5\} = B \cup C;$$
$$A \backslash B = \{2, 5\};$$
$$A \backslash C = B;$$
$$B \backslash C = B.$$

Dizemos que dois conjuntos, A e B, são *disjuntos* se não tiverem elementos comuns, ou seja, se $A \cap B = \emptyset$. Por exemplo, qualquer que seja o conjunto A, A e A^c são disjuntos.

Terminamos esta secção, apresentando algumas propriedades das operações sobre conjuntos. É feita a demonstração da Propriedade 7 , sendo as restantes deixadas ao cuidado do leitor.

LEMA 1.2. *Sejam A, B e C conjuntos arbitrários de um universo U. As propriedades seguintes verificam-se:*

1. *(Elemento neutro da união e absorvente da interseção) $A \cup \emptyset = A$ e $A \cap \emptyset = \emptyset$;*
2. *(Elemento neutro da interseção e absorvente da união) $A \cap U = A$ e $A \cup U = U$;*
3. *(Idempotência) $A \cup A = A$ e $A \cap A = A$;*
4. *(Associatividade) $A \cup (B \cup C) = (A \cup B) \cup C$ e $A \cap (B \cap C) = (A \cap B) \cap C$;*
5. *(Comutatividade) $A \cup B = B \cup A$ e $A \cap B = B \cap A$;*
6. *(Distributividade) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ e $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$;*
7. *(Absorção) $A \cap (A \cup B) = A$ e $A \cup (A \cap B) = A$;*
8. *$A \backslash B = A \cap B^c$;*
9. *$(A \backslash B) \cap (B \backslash A) = \emptyset$;*
10. *(Dupla complementaridade) $(A^c)^c = A$;*

11. *(Leis de De Morgan)* $(A \cup B)^c = A^c \cap B^c$ *e* $(A \cap B)^c = A^c \cup B^c$.

DEMONSTRAÇÃO. (7) Dados os conjuntos arbitrários A e B, temos:

$$\begin{aligned} A \cap (A \cup B) &= \{x \in U : x \in A \text{ e } x \in A \cup B\} \\ &\subseteq \{x \in U : x \in A\} \\ &= A \end{aligned}$$

e, reciprocamente, como $A \subseteq A$ e $A \subseteq A \cup B$, então $A \subseteq A \cap (A \cup B)$. Ora, se dois conjuntos estão simultaneamente contidos no outro, então eles são iguais.

A demonstração da segunda igualdade é análoga. □

2. Lógica Proposicional

A palavra *Lógica* deriva do grego e significa pensamento, ideia, argumento, sendo, portanto, entendida como a *Ciência do Raciocínio*. É a Aristóleles (384 a.C. - 322 a.C.) que se deve o primeiro estudo formal do raciocínio. A Lógica Simbólica trata da estrutura do raciocínio, estudando a relação entre conceitos e fornecendo um meio de compor provas de declarações. A Lógica Matemática, onde nos iremos debruçar, usa a lógica simbólica para estudar o raciocínio matemático. Tem como objetivo estabelecer a validade de um argumento, evitar erros de raciocínio e saber se uma conclusão é consequência de certas premissas.

Ao combinarmos sinais elementares da linguagem matemática, letras, números, sinais operatórios, etc., construímos expressões que poderão ter ou não significado. No que se segue, usaremos unicamente expressões com significado. Os elementos elementares da Lógica são as proposições (ou sentenças).

Uma *designação* é uma expressão que nomeia um objeto. A uma expressão acerca da qual podemos referir se é verdadeira ou falsa damos o nome de *proposição*.

EXEMPLO 1.6. São designações as expressões seguintes:

Matemática, Porto, computador, 6, $\frac{1+\sqrt{5}}{2}$.

São proposições as expressões seguintes:

Matemática Discreta é uma u.c. da LEI, Porto é a capital de Portugal,
$1 + 1 \neq 2$, π é um número transcendente.

Notemos que frases exclamativas ou interrogativas não são proposições. Há ainda expressões para as quais não conseguimos referir se são verdadeiras ou falsas, denominadas de *ambíguas* (por exemplo, *Aquele automóvel é bonito.*, ou *A Matemática é divertida!*). Não trabalharemos com este tipo de expressões.

A Lógica Aristotélica tem por base os dois princípios seguintes:

Princípio da Não Contradição: Uma proposição não pode ser simultaneamente verdadeira e falsa.

Princípio do Terceiro Excluído: Uma proposição é verdadeira ou é falsa (não havendo uma terceira opção).

Se uma proposição é verdadeira, dizemos que tem o *valor lógico verdade*, e representamos pelo símbolo V ou 1. Se uma proposição é falsa, dizemos que tem o *valor lógico falsidade*, e representamos pelo símbolo F ou 0. Denotamos o universo dos valores lógicos por $\mathcal{L} = \{V, F\}$ e o valor lógico de uma proposição p por $V(p)$.

Duas *designações* dizem-se *equivalentes* se nomeiam o mesmo objeto. Usamos o símbolo $=$ para ligar duas designações equivalentes.

EXEMPLO 1.7. Dadas as designações *Porto* e *cidade invicta*, escrevemos *Porto* $=$ *cidade invicta* para indicar que designam a mesma cidade.

Duas *proposições* dizem-se *equivalentes* se têm o mesmo valor lógico. Usamos o símbolo$\Leftrightarrow$ para ligar duas proposições equivalentes.

EXEMPLO 1.8. Dadas as proposições 7 *é um número primo* e 7 *tem exatamente dois divisores inteiros positivos*, escrevemos 7 *é um número primo* $\Leftrightarrow$ 7 *tem exatamente dois divisores inteiros positivos* para indicar que têm o mesmo valor lógico, neste caso, verdade.

Dadas as proposições $1 + 1 \neq 2$ e π *é um número racional*, escrevemos $1 + 1 \neq 2 \Leftrightarrow \pi$ *é um número racional* para indicar que têm o mesmo valor lógico, neste caso, falsidade.

Operações lógicas. Podemos obter proposições à custa de outras usando certas operações, a que chamamos *operações lógicas* (ou *conetivos lógicos*). Uma proposição que não pode ser decomposta noutras proposições diz-se *simples* (ou *atómica*). Caso contrário, diz-se *composta*.

EXEMPLO 1.9. Consideremos as proposições seguintes:

1. O Joaquim vai estudar Matemática Discreta.
2. O Joaquim vai ao cinema.

Podemos construir com estas duas proposições simples as proposições compostas que se seguem:

(3) O Joaquim não vai ao cinema.
(4) O Joaquim vai estudar Matemática Discreta e vai ao cinema.
(5) O Joaquim vai estudar Matemática Discreta ou vai ao cinema.

As proposições (3), (4) e (5) do exemplo anterior resultam da aplicação das operações lógicas negação, conjunção e disjunção, respetivamente, e que estudaremos em seguida.

São seis as operções lógicas, ou conetivos lógicos, que vamos estudar: *negação* ($\sim$), *conjunção* ($\wedge$), *disjunção* ($\vee$), *disjunção exclusiva* ($\dot{\vee}$), *implicação* ($\Rightarrow$ ou $\rightarrow$) e *equivalência* ($\Leftrightarrow$ ou $\equiv$). Uma proposição composta caracteriza-se completamente à custa da tabela dos valores lógicos que adquire, verdadeiro (V) ou falso (F), quando se percorrem todas as combinações possíveis dos valores lógicos das proposições simples que a constituem. Esta tabela designa-se por *tabela de verdade*.

Negação ($\sim$). No universo das proposições, definimos a operação *negação* como a operação que a cada proposição p faz corresponder uma e uma só proposição $\sim p$ (lê-se *não p* ou *não é verdade que p*), sendo verdadeira, se p for falsa, e falsa, se p for verdadeira.

Esta operação é totalmente apresentada na tabela de verdade seguinte:

p	$\sim p$
V	F
F	V

EXEMPLO 1.10. A negação da proposição (falsa) $\pi < 1$ é a proposição (verdadeira) $\sim (\pi < 1)$, que pode escrever-se na forma $\pi \geq 1$.

Conjunção ($\wedge$). No universo das proposições, definimos a operação *conjunção* como a operação que a cada par de proposições (p, q) faz corresponder uma e uma só proposição $p \wedge q$ (lê-se p *e* q), sendo verdadeira apenas no caso em que p e q são ambas verdadeiras.

Esta operação é totalmente apresentada na tabela de verdade seguinte:

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

EXEMPLO 1.11. A conjunção das proposições $\pi < 1$ (falsa) e $e \geq 2$ (verdadeira) é a proposição (falsa) $\pi < 1 \wedge e \geq 2$.

Disjunção ($\vee$). No universo das proposições, definimos a operação *disjunção* como a operação que a cada par de proposições (p, q) faz corresponder uma e uma só proposição $p \vee q$ (lê-se p *ou* q), sendo falsa apenas no caso em que p e q são ambas falsas.

Esta operação é totalmente apresentada na tabela de verdade seguinte:

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

EXEMPLO 1.12. A disjunção das proposições $\pi < 1$ (falsa) e $e \geq 2$ (verdadeira) é a proposição (verdadeira) $\pi < 1 \vee e \geq 2$.

Disjunção exclusiva ($\dot{\vee}$). No universo das proposições, definimos a operação *disjunção exclusiva* como a operação que a cada par de proposições (p, q) faz corresponder uma e uma só proposição $p \dot{\vee} q$ (lê-se *ou p, ou q*), que é verdadeira apenas quando uma das proposições é verdadeira e a outra falsa.

Esta operação é totalmente apresentada na tabela de verdade seguinte:

p	q	$p \dot{\vee} q$
V	V	F
V	F	V
F	V	V
F	F	F

EXEMPLO 1.13. A disjunção exclusiva das proposições $\pi < 1$ (falsa) e $e \geq 2$ (verdadeira) é a proposição (verdadeira) $\pi < 1 \dot{\vee} e \geq 2$.

Implicação ($\Rightarrow$ ou $\rightarrow$). No universo das proposições, definimos a operação *implicação* como a operação que a cada par de proposições (p, q) faz corresponder uma e uma só proposição $p \Rightarrow q$ (lê-se p *implica* q ou *se* p *então* q), sendo falsa apenas no caso em que p é verdadeira e q é falsa.

Esta operação é totalmente apresentada na tabela de verdade seguinte:

p	q	$p \Rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

Seja $p \Rightarrow q$ uma proposição verdadeira. Podemos observar pela tabela de verdade que é suficiente que a proposição p seja verdadeira para que a proposição q o seja. Dizemos que

p é condição suficiente para q.

Por outro lado, para p ser verdadeira é necessário que q seja verdadeira. Dizemos que

q é condição necessária para p.

EXEMPLO 1.14. A implicação

se m e n são números pares, então $m + n$ é um número par

é verdadeira. Ou seja, é suficiente que m e n sejam números pares para que $m + n$ seja um número par. No entanto, esta condição não é necessária, visto que a soma de números ímpares é também um número par. Por outro lado, é necessário que $m + n$ seja um número par para que m e n sejam ambos números pares (se $m + n$ é um número ímpar, então um dos números é par e o outro é ímpar!).

Dada uma implicação $p \Rightarrow q$, designamos as proposições seguintes:

Implicação recíproca: $q \Rightarrow p$.
Implicação contrária: $\sim p \Rightarrow \sim q$.

EXEMPLO 1.15. Para a proposição do exemplo anterior temos, respetivamente, a implicação recíproca e a implicação contrária como se seguem:
Implicação recíproca: *se $m + n$ é um número par, então m e n são números pares.*
Implicação contrária: *se m e n não são ambos números pares, então $m + n$ não é um número par.*

Equivalência ($\Leftrightarrow$ ou $\equiv$). No universo das proposições, definimos a operação *equivalência* como a operação que a cada par de proposições (p, q) faz corresponder uma e uma só proposição $p \Leftrightarrow q$ (lê-se p *é equivalente a* q ou p *se e só se* q), que só é verdadeira no caso em que p e q têm o mesmo valor lógico.

Esta operação é totalmente apresentada na tabela de verdade seguinte:

p	q	$p \Leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

Se $p \Leftrightarrow q$ for uma proposição verdadeira, então dizemos que

p é condição necessária e suficiente para q

e

q é condição necessária e suficiente para p.

EXEMPLO 1.16. A equivalência

n é um número primo se e só se n tem exatamente dois divisores inteiros positivos

é verdadeira. Ambas as condições são necessárias e suficientes para que se tenha a outra condição.

A partir das tabelas de verdade das operações definidas podemos agora determinar as tabelas de verdade de proposições compostas mais complexas. O procedimento baseia-se em determinar-se os valores lógicos de subexpressões sucessivamente mais complexas constituintes da proposição composta. Vejamos o exemplo seguinte.

EXEMPLO 1.17. A tabela de verdade da proposição

$$((p \wedge q) \vee r) \wedge (\sim (p \wedge r))$$

pode ser determinada com os cálculos intermédios seguintes:

p	q	r	$p \wedge q$	$(p \wedge q) \vee r$	$p \wedge r$	$\sim (p \wedge r)$	$((p \wedge q) \vee r) \wedge (\sim (p \wedge r))$
V	V	V	V	V	V	F	F
V	V	F	V	V	F	V	V
V	F	V	F	V	V	F	F
V	F	F	F	F	F	V	F
F	V	V	F	V	F	V	V
F	V	F	F	F	F	V	F
F	F	V	F	V	F	V	V
F	F	F	F	F	F	V	F

De reparar que a tabela de verdade de uma proposição composta p tem 2^n linhas, onde n é o número de proposições atómicas que a constituem (2^n é o número de todas as combinações diferentes dos valores lógicos das proposições atómicas).

Na proposição composta do exemplo anterior foi necessário recorrer ao uso dos parênteses para indicar a prioridade das operações. De modo a reduzir o número de parênteses a apresentar numa proposição, introduzimos as regras de prioridade seguintes:

conetivo	$\sim$	$\wedge$	$\vee$	$\Rightarrow$	$\Leftrightarrow$
prioridade	1	2	3	4	5

EXEMPLO 1.18. A proposição $p \vee q \wedge \sim r \Rightarrow q$ é igual (por supressão dos parênteses) à proposição $(p \vee (q \wedge (\sim r))) \Rightarrow q$.

Designamos por *tautologia* uma proposição composta cujo valor lógico é sempre verdade, quaisquer que sejam os valores lógicos das proposições simples que a compõem.

Designamos por *contradição* uma proposição composta cujo valor lógico é sempre falsidade, quaisquer que sejam os valores lógicos das proposições simples que a compõem.

Dizemos que uma proposição é *satisfazível* se existir alguma combinação de valores lógicos atribuídos às proposições simples que tornem verdadeira a proposição composta.

Propriedades das operações lógicas. As operações lógicas apresentadas gozam de diversas propriedades. Apresentamos, em seguida, as mais importantes. Tais propriedades podem ser facilmente demonstradas recorrendo às tabelas de verdade e mostrando que são tautologias.

Propriedades da Conjunção e Propriedades da Disjunção

Princípios da Lógica Aristotélica	
Não Contradição	$(p \wedge \sim p) \Leftrightarrow F$
Terceiro Excluído	$(p \vee \sim p) \Leftrightarrow V$

Propriedades	Conjunção	Disjunção
Comutatividade	$(p \wedge q) \Leftrightarrow (q \wedge p)$	$(p \vee q) \Leftrightarrow (q \vee p)$
Associatividade	$[(p \wedge q) \wedge r] \Leftrightarrow [p \wedge (q \wedge r)]$	$[(p \vee q) \vee r] \Leftrightarrow [p \vee (q \vee r)]$
Idempotência	$(p \wedge p) \Leftrightarrow p$	$(p \vee p) \Leftrightarrow p$
Existência de elemento neutro	$(p \wedge V) \Leftrightarrow p \Leftrightarrow (V \wedge p)$	$(p \vee F) \Leftrightarrow p \Leftrightarrow (F \vee p)$
Existência de elemento absorvente	$(p \wedge F) \Leftrightarrow F \Leftrightarrow (F \wedge p)$	$(p \vee V) \Leftrightarrow V \Leftrightarrow (V \vee p)$

Propriedades envolvendo a Conjunção e a Disjunção

	Distributividade da conjunção em relação à disjunção
à esquerda	$[p \wedge (q \vee r)] \Leftrightarrow [(p \wedge q) \vee (p \wedge r)]$
à direita	$[(q \vee r) \wedge p] \Leftrightarrow [(q \wedge p) \vee (r \wedge p)]$
	Distributividade da disjunção em relação à conjunção
à esquerda	$[p \vee (q \wedge r)] \Leftrightarrow [(p \vee q) \wedge (p \vee r)]$
à direita	$[(q \wedge r) \vee p] \Leftrightarrow [(q \vee p) \wedge (r \vee p)]$
	Absorção
	$[p \wedge (p \vee q)] \Leftrightarrow p$
	$[p \vee (p \wedge q)] \Leftrightarrow p$

Propriedades da Negação

Dupla negação
$\sim\sim p \Leftrightarrow p$

Leis de De Morgan	
Negação da conjunção	$\sim (p \wedge q) \Leftrightarrow (\sim p \vee \sim q)$
Negação da disjunção	$\sim (p \vee q) \Leftrightarrow (\sim p \wedge \sim q)$

Propriedades da Implicação

Relação da implicação com a disjunção e a negação
$(p \Rightarrow q) \Leftrightarrow (\sim p \vee q)$
Lei da conversão
$(p \Rightarrow q) \Leftrightarrow (\sim q \Rightarrow \sim p)$
Transitividade
$[(p \Rightarrow q) \wedge (q \Rightarrow r)] \Rightarrow (p \Rightarrow r)$

Propriedades da Equivalência

A equivalência como conjunção de implicações
$(p \Leftrightarrow q) \Leftrightarrow [(p \Rightarrow q) \wedge (q \Rightarrow p)]$

Demonstração. Demonstramos uma das propriedades da absorção e deixamos as restantes para o leitor:

p	q	$p \vee q$	$[p \wedge (p \vee q)]$	$[p \wedge (p \vee q)] \Leftrightarrow p$
V	V	V	V	V
V	F	V	V	V
F	V	V	F	V
F	F	F	F	V

□

Proposições logicamente equivalentes. Duas proposições que têm o mesmo valor lógico para toda a combinação de valores lógicos atribuídos às proposições simples dizem-se *logicamente equivalentes*.

Podemos usar as tabelas de verdade para verificar se duas proposições são logicamente equivalentes, tal como o fizemos para demonstrar as propriedades das operações lógicas. Contudo, quando as proposições são muito longas, tal método não é prático. Recorrendo às propriedades das operações lógicas, podemos simplificar algumas proposições que são subexpressões de, pelo menos, uma das proposições compostas totais, de modo a obter proposições logicamente equivalentes menos longas.

Exemplo 1.19. Usando as propriedades das operações lógicas, vamos mostrar que

$$(\sim p \wedge q) \vee \sim (p \vee q) \equiv \sim p.$$

DEMONSTRAÇÃO.

$$
\begin{array}{lll}
(\sim p \wedge q) \vee \sim (p \vee q) & \equiv (\sim p \wedge q) \vee (\sim p \wedge \sim q) & \text{Lei de De Morgan} \\
& \equiv \sim p \wedge (q \vee \sim q) & \text{Distributividade} \\
& \equiv \sim p \wedge V & \text{Princípio do Terceiro Excluído} \\
& \equiv \sim p & \text{Elemento neutro}
\end{array}
$$

□

Simplesmente para distinguirmos o conetivo *equivalência* da propriedade *logicamente equivalentes* usamos, respetivamente, os símbolos $\Leftrightarrow$ e $\equiv$.

3. Lógica de predicados

Predicados. Um *predicado* é uma propriedade de um ou vários objetos ou indivíduos. Na Lógica Simbólica, um *predicado* (ou *expressão proposicional*) é uma expressão com uma ou mais variáveis que se transforma numa proposição quando se concretizam as variáveis. Denotamos um predicado por $P(x)$, onde x é a *variável preditiva*.

Uma proposição é então constituída por um sujeito e por um predicado. Usualmente, representam-se os sujeitos por letras minúsculas e os predicados por letras maiúsculas. Se s for o sujeito e$P(x)$ o predicado, escrevemos $P(s)$ para representar a proposição *s possui a propriedade $P(x)$*.

EXEMPLO 1.20. Consideremos os sujeitos $p =$ *o professor* e $a =$ *os alunos*, e os predicados $P(x) = x$ *passa os alunos* e $E(x) = x$ *estudam muito*. São proposições as expressões seguintes:

Os alunos estudam muito.

O professor passa os alunos.

Se os alunos estudam muito, então o professor passa os alunos.

e podemos escrevê-las simbolicamente como $E(a)$, $P(p)$ e $E(a) \Rightarrow P(p)$, respetivamente.

Notemos que as proposições simples têm um sujeito e um predicado, enquanto as proposições compostas têm mais do que um sujeito ou predicado.

Definimos o *domínio* de uma variável preditiva como o conjunto dos valores que a variável pode tomar, de modo a que a expressão tenha significado.

Uma expressão proposicional (ou predicado) diz-se:

- *possível* quando existe pelo menos uma concretização das variáveis que a tornam numa proposição verdadeira.
- *universal* quando para qualquer concretização das variáveis se torna numa proposição verdadeira.
- *impossível* quando para qualquer concretização das variáveis se torna numa proposição falsa.

No exemplo anterior, como nem todos os alunos estudam muito, não podemos aferir se a proposição $E(a) \Rightarrow P(p)$ é verdadeira ou falsa. Existem vários exemplos, tais como os que apresentamos em seguida, em que se torna necessário quantificar a variável x no seu domínio, justificando o uso dos chamados *quantificadores*.

EXEMPLO 1.21. Consideremos o predicado $P(x) = x$ *tem coração*. Ora, P(Joaquim), P(Matilde), etc., expressam que um conjunto particular de humanos têm coração. Como podemos exprimir que *Todos os humanos têm coração*?

EXEMPLO 1.22. Sejam $x \in \mathbb{Z}$ e $P(x) : x < 0$. Dependendo do valor que x assumir em $P(x)$, o predicado torna-se, ou não, numa proposição verdadeira. Como podemos expressar esta situação?

Quantificadores. Já vimos que, se atribuirmos à variável de um predicado um dos valores do seu domínio, obtemos uma proposição. Outra forma de obter proposições é utilizando quantificadores.

Quantificador universal. Designamos de *quantificador universal* o símbolo $\forall$ (e lemos *para todo* ou *qualquer que seja*) que, aplicado a uma variável de um predicado (nessa única variável), transforma-o numa proposição verdadeira se e só se esse predicado for universal.

EXEMPLO 1.23. No universo H de todos os seres humanos, o predicado $P(x) = x$ *tem coração* é universal. Portanto, a proposição

$$\forall x \in H,\, P(x)$$

é verdadeira. No entanto, no universo S de todos os seres vivos, o mesmo predicado não é condição universal. Logo,

$$\forall x \in S,\, P(x)$$

é uma proposição falsa.

Quantificador existencial. Designamos de *quantificador existencial* o símbolo $\exists$ (e lemos *existe um* ou *existe pelo menos um*) que, aplicado a uma variável de um predicado (nessa única variável), transforma-o numa proposição verdadeira se e só se esse predicado for possível.

EXEMPLO 1.24. Em $\mathbb{Z}$, o predicado $P(x) : x < 0$ é uma condição possível. Portanto, a proposição

$$\exists x \in \mathbb{Z},\, P(x)$$

é verdadeira. No entanto, em $\mathbb{N}$, o mesmo predicado é condição impossível. Logo,

$$\exists x \in \mathbb{N},\, P(x)$$

é uma proposição falsa.

EXEMPLO 1.25. Consideremos o predicado $P(x) : x^2 = 1$. O predicado é possível em $\mathbb{R}$, basta tomar $x = -1$ ou $x = 1$, pelo que é verdadeira a proposição

$$\exists x \in \mathbb{R},\, x^2 = 1.$$

Mais ainda, o predicado é também possível em $\mathbb{N}$, basta tomar $x = 1$, pelo que é também verdadeira a proposição

$$\exists x \in \mathbb{N},\, x^2 = 1.$$

Neste caso, só existe um valor que transforma o predicado numa proposição verdadeira. Dizemos que *existe um, e um só,* $x \in \mathbb{N}$ tal que $x^2 = 1$, e escrevemos,

$$\exists^1 x \in \mathbb{N},\, x^2 = 1.$$

Quantificador de existência e unicidade. Designamos de *quantificador de existência e unicidade* o símbolo $\exists^1$ (e lemos *existe um e um só*) que, aplicado a uma variável de um predicado (nessa única variável), transforma-o numa proposição verdadeira apenas no caso em que o predicado tem uma única solução.

É agora importante perceber como se negam estes quantificadores. Apresentamos então as denominadas *Segundas Leis de De Morgan*:

A negação transforma o quantificador universal no quantificador existencial seguido de negação:

$$\sim [\forall x,\, P(x)] \Leftrightarrow \exists x,\, \sim P(x).$$

A negação transforma o quantificador existencial no quantificador universal seguido de negação:

$$\sim [\exists x,\, P(x)] \Leftrightarrow \forall x,\, \sim P(x).$$

EXEMPLO 1.26. A negação da proposição (falsa)

$$\forall x \in \{\text{seres vivos}\},\, x \text{ tem coração}$$

é a proposição (verdadeira)

$$\exists x \in \{\text{seres vivos}\},\, x \text{ não tem coração.}$$

EXEMPLO 1.27. A negação da proposição (falsa)

$$\exists x \in \mathbb{N},\, x < 0$$

é a proposição (verdadeira)

$$\forall x \in \mathbb{N},\, x \geq 0.$$

Até aqui consideramos unicamente predicados numa única variável. No entanto, um predicado pode ter mais do que uma variável. Neste caso, ele só se transforma numa proposição quando todas as variáveis estiverem quantificadas. No caso de termos mais do que um quantificador, eles devem ser lidos da esquerda para a direita.

EXEMPLO 1.28. A proposição

$$\forall x \in \mathbb{R}\, \forall y \in \mathbb{R},\, x + y = y + x$$

é uma proposição verdadeira que traduz a propriedade comutativa da adição no corpo dos números reais. Também pode ser escrita na forma

$$\forall y \in \mathbb{R}\, \forall x \in \mathbb{R},\, x + y = y + x$$

ou, ainda, na forma

$$\forall x, y \in \mathbb{R},\, x + y = y + x.$$

De notar que, no exemplo anterior, a troca da ordem dos quantificadores produziu proposições equivalentes. Mais geralmente, a troca da ordem de um qualquer número de quantificadores do mesmo tipo, ou todos universais, ou todos existenciais, produz sempre proposições equivalentes. O mesmo já não acontece se os quantificadores forem de diferentes tipos, como podemos ver no exemplo seguinte.

Exemplo 1.29. Consideremos as proposições

$$\forall x \in \mathbb{R}\, \exists y \in \mathbb{R},\ x + y = 0 = y + x$$

e

$$\exists y \in \mathbb{R}\, \forall x \in \mathbb{R},\ x + y = 0 = y + x.$$

A primeira proposição traduz a existência de um simétrico para cada número real, que varia consoante o número, sendo uma proposição verdadeira. A segunda proposição afirma a existência de um número real que é simétrico de qualquer número real, o que é falso.

Numa expressão, uma variável diz-se *muda* quando ela está quantificada. Caso contrário, a variável diz-se *livre*.

Terminamos com um exemplo de negação de uma proposição com quantificação múltipla.

Exemplo 1.30. Consideremos a proposição

$$\forall x, y \in \mathbb{R}\, \exists z \in \mathbb{R},\ x = yz$$

que é uma proposição falsa (fazendo $x = 1$, $y = 0$, não existe z que satisfaça $1 = 0z$). A negação desta proposição é a proposição verdadeira

$$\begin{aligned} \sim (\forall x, y \in \mathbb{R}\, \exists z \in \mathbb{R},\ x = yz) &\equiv \exists x, y \in \mathbb{R} \sim (\exists z \in \mathbb{R},\ x = yz) \\ &\equiv \exists x, y \in \mathbb{R}\, \forall z \in \mathbb{R},\ \sim (x = yz) \\ &\equiv \exists x, y \in \mathbb{R}\, \forall z \in \mathbb{R},\ x \neq yz. \end{aligned}$$

4. Relações binárias

Relembramos o conceito de par ordenado e de produto cartesiano de dois conjuntos, introduzido na Secção 1.

Um *par ordenado* (a, b) é uma sequência de dois elementos com uma determinada ordem, sendo a o primeiro elemento e b o segundo elemento da sequência.

Dados dois conjuntos A e B, o *produto cartesiano* de A por B, que se denota por $A \times B$, é o conjunto formado por todos os pares ordenados (a, b), com $a \in A$ e $b \in B$, ou seja,

$$A \times B = \{(a, b) : a \in A \wedge b \in B\}.$$

Uma *relação binária* R entre os conjuntos A e B é um subconjunto não vazio do produto cartesiano $A \times B$. Dado $(a, b) \in A \times B$, escrevemos aRb e dizemos que *a está em relação com b por R* se e só se $(a, b) \in R$. Se, em particular, $A = B$, dizemos que R é uma relação binária definida em A. Neste caso, podemos escrever o produto cartesiano $A \times A$ na forma A^2.

Exemplo 1.31. A relação menor ou igual, denotada por $\leq$, é uma relação binária sobre o conjunto dos números inteiros. Com efeito, temos

$$\leq = \{(a, b) \in \mathbb{Z}^2 : a \leq b\} \subseteq \mathbb{Z}^2.$$

Notemos que, embora se possa escrever $(2, 3) \in \leq$ e $(3, 2) \notin \leq$, usualmente escreve-se $2 \leq 3$ e $3 \not\leq 2$ (ou $3 > 2$).

EXEMPLO 1.32. Sejam $A = \{1, 2, 3\}$, $B = \{1, 2, 3, 4, 5, 6\}$ e R a relação binária definida por

$$\{(x, y) \in A \times B : x \text{ divide } y\}.$$

Então R pode ser escrito em extensão por

$$R = \{(1,1), (1,2), (1,3), (1,4), (1,5), (1,6), (2,2), (2,4), (2,6), (3,3), (3,6)\}.$$

Sejam R uma relação binária entre os conjuntos A e B, a um elemento de A e b um elemento de B. Designamos por:

- *domínio* de R, e denotamos por $\text{Dom}(R)$, o conjunto
$$\text{Dom}(R) = \{x \in A : \exists y \in B,\ (x, y) \in R\}.$$
- *contradomínio* de R (ou imagem de R), e denotamos por $\text{CD}(R)$ (ou $\text{Im}(R)$), o conjunto
$$\text{CD}(R) = \{y \in B : \exists x \in A,\ (x, y) \in R\}.$$
- *imagem* de a por R, e denotamos por $R(a)$, o conjunto
$$R(a) = \{y \in B : (a, y) \in R\}.$$
- *imagem recíproca* de b por R, e denotamos por $R^{-1}(b)$, o conjunto
$$R^{-1}(b) = \{x \in A : (x, b) \in R\}.$$
- *relação inversa* de R, e denotamos por R^{-1}, o subconjunto de $B \times A$ definido por
$$R^{-1} = \{(y, x) \in B \times A : (x, y) \in R\}.$$

EXEMPLO 1.33. Para o Exemplo 1.32, temos:

$$\begin{array}{ll} \text{Dom}(R) = \{1, 2, 3\} = A, & \text{CD}(R) = \{1, 2, 3, 4, 5, 6\} = B, \\ R(3) = \{3, 6\}, & R^{-1}(5) = \{1\} \end{array}$$

e

$$R^{-1} = \{(1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (2,2), (4,2), (6,2), (3,3), (6,3)\}.$$

Podemos ainda definir a união, a interseção, e o complementar de relações tais como definimos as operações sobre conjuntos.

Finalmente, definimos a *composição* de duas relações S, entre B e C, e R, entre A e B, e denotamos por $S \circ R$, o conjunto

$$S \circ R = \{(a, c) \in A \times C : \exists b \in B,\ (a, b) \in R \wedge (b, c) \in S\}.$$

EXEMPLO 1.34. Sejam R a relação definida no Exemplo 1.32 e S a relação entre B eA definida por

$$S = \{(x, y) \in B \times A : x + y \leq 3\}.$$

Então, temos

$$\begin{array}{l} R = \{(1,1), (1,2), (1,3), (1,4), (1,5), (1,6), (2,2), (2,4), (2,6), (3,3), (3,6)\}, \\ S = \{(1,1), (1,2), (2,1)\} \\ \text{e } S \circ R = \{(1,1), (1,2), (2,1)\} \end{array}$$

é uma relação em A.

Representações. Existem várias formas de representar graficamente relações. Apresentamos aqui duas: a *matricial* e *por grafos*.

Representação matricial. Seja R uma relação binária entre $A = \{a_1, a_2, \ldots, a_m\}$ e $B = \{b_1, b_2, \ldots, b_n\}$. Esta relação pode ser representada por uma matriz de dimensão $m \times n$ com as entradas $c_{i,j}$, $i = 1, \ldots, m$ e $j = 1, \ldots, n$, determinadas pela condição:

$$\begin{cases} \text{se } (a_i, b_j) \in R, \text{ então } c_{i,j} = 1 \\ \text{se } (a_i, b_j) \notin R, \text{ então } c_{i,j} = 0 \end{cases}$$

Exemplo 1.35. A relação R do Exemplo 1.32 representa-se matricialmente por:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Lema 1.3. *Sejam R uma relação binária entre A e B e S uma relação binária entre B e C, representadas matricialmente pelas matrizes M e N, respetivamente. Então:*

1. *A relação inversa de R, R^{-1}, é representada matricialmente pela matriz transposta de M, M^T.*
2. *A composição de S com R, $S \circ R$, é representada matricialmente pelo produto Booleano das matrizes M e N, $M \otimes N$. Explicando de uma forma simplificada, na álgebra de Boole $\mathbb{B} = \langle \{0,1\}, \vee, \wedge, \sim, 0, 1 \rangle$, as operações binárias de adição e multiplicação são, respetivamente, as operações de disjunção e conjunção sobre o conjunto dos valores lógicos $\{0,1\}$.*

Exemplo 1.36. A relação $S \circ R$ do Exemplo 1.34 pode ser determinada pelo cálculo matricial seguinte:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} =$$

$$= \begin{pmatrix} (1 \wedge 1) \vee (1 \wedge 1) \vee (1 \wedge 0) \vee (1 \wedge 0) \vee (1 \wedge 0) \vee (1 \wedge 0) & 1 & 0 \\ (0 \wedge 1) \vee (1 \wedge 1) \vee (0 \wedge 0) \vee (1 \wedge 0) \vee (0 \wedge 0) \vee (1 \wedge 0) & 0 & 0 \\ (0 \wedge 1) \vee (0 \wedge 1) \vee (1 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) \vee (1 \wedge 0) & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

ou seja, $S \circ R$ é uma relação sobre $A = \{1, 2, 3\}$ dada por $S \circ R = \{(1,1), (1,2), (2,1)\}$.

Representação por grafos. Um *grafo (não orientado)* é um par $G = (V, E)$, onde V é um conjunto não vazio, designado por conjunto dos *vértices*, e E é um subconjunto de V^2, onde não é considerada nenhuma ordem nos pares de V^2, designado por conjunto das *arestas*.

Um *grafo orientado* (ou *digrafo*) é um par $\vec{G} = (V, E)$, em que E é um subconjunto de V^2, onde os pares são ordenados, designado por conjunto das *arestas orientadas*.

Nota: Na literatura há várias definições para grafo e grafo orientado, sendo a que adotamos uma das mais simples, mas suficiente para os objetivos desta unidade curricular.

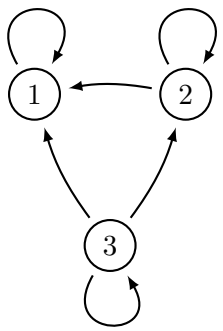
Seja R uma relação binária em $A = \{a_1, a_2, \ldots, a_m\}$. Então, R pode ser representada por um grafo orientado $\vec{G} = (V, E)$ onde:

- $V = A = \{a_1, a_2, \ldots, a_m\}$;
- $E = \{(a_i, a_j) \in V^2 : (a_i, a_j) \in R\}$, onde (a_i, a_j) é uma aresta orientada do vértice a_i para o vértice a_j.

EXEMPLO 1.37. Consideremos a relação R em $A = \{1, 2, 3\}$ definida por

$$\{(x, y) \in A^2 : x \geq y\}.$$

Então o grafo orientado que a representa é:



LEMA 1.4. *Seja R uma relação binária em A, representada pelo grafo orientado $\vec{G} = (V, E)$. Então a relação inversa de R, R^{-1}, é representada pelo grafo $\vec{G}^{-1}$, que resulta da troca da orientação das arestas de $\vec{G}$.*

Propriedades das relações binárias. Seja R uma relação binária definida em A. Dizemos que R é:

- *reflexiva*, se
$$\forall x \in A,\ (x, x) \in R.$$
- *simétrica*, se
$$\forall x, y \in A,\ (x, y) \in R \Rightarrow (y, x) \in R.$$
- *anti-simétrica*, se
$$\forall x, y \in A,\ [(x, y) \in R \wedge (y, x) \in R] \Rightarrow x = y.$$
- *transitiva*, se
$$\forall x, y, z \in A,\ [(x, y) \in R \wedge (y, z) \in R] \Rightarrow (x, z) \in R.$$

De outro modo, negando as condições acima, dizemos que R:

- *não é reflexiva*, se
$$\exists x \in A,\ (x, x) \notin R.$$

- *não é simétrica*, se

$$\exists x, y \in A,\ (x, y) \in R \wedge (y, x) \notin R.$$

- *não é anti-simétrica*, se

$$\exists x, y \in A,\ (x, y) \in R \wedge (y, x) \in R \wedge x \neq y.$$

- *não é transitiva*, se

$$\exists x, y, z \in A,\ (x, y) \in R \wedge (y, z) \in R \wedge (x, z) \notin R.$$

EXEMPLO 1.38. Consideremos a relação R em $A = \{1, 2, 3, 4\}$ representada em extensão por

$$R = \{(1, 1), (1, 2), (2, 1), (2, 2), (2, 4), (3, 3), (4, 2), (4, 4)\}.$$

Então:

- R é reflexiva, pois $(1, 1), (2, 2), (3, 3), (4, 4) \in R$.
- R é simétrica, pois $(1, 2) \in R \Rightarrow (2, 1) \in R$ e $(2, 4) \in R \Rightarrow (4, 2) \in R$.
- R não é transitiva, pois $(1, 2) \in R \wedge (2, 4) \in R$, mas $(1, 4) \notin R$.

Estas propriedades observam-se facilmente nas representações das relações, quer matricialmente, quer por grafos. Assim, na representação matricial de uma relação, ela é:

- *reflexiva*, se a diagonal principal tiver todas as entradas com 1.
- *simétrica*, se a matriz for simétrica.
- *anti-simétrica*, se as entradas da matriz satisfizerem a condição $c_{i,j} = 1 \Rightarrow c_{j,i} = 0$, para todo o par (i, j), $i \neq j$.
- *transitiva*, se sempre que a matriz M, que representa a relação, tiver uma entrada nula, então o seu elemento homólogo na matriz M^2 também é nulo.

Na representação por grafos de uma relação, ela é:

- *reflexiva*, se todos os vértices tiverem um lacete (aresta com saída e entrada nesse vértice).
- *simétrica*, se, para todo o par de vértices (a_i, a_j), se existir uma aresta (a_i, a_j), então também existe a aresta (a_j, a_i).
- *anti-simétrica*, se não existir nenhum par de vértices (a_i, a_j), com $a_i \neq a_j$, que satisfaça a condição anterior.
- *transitiva*, se sempre que existir um passeio[1] de comprimento dois entre dois vértices a_i e a_j, então também existe a aresta (a_i, a_j).

EXERCÍCIO 1.39. Observe as propriedades da relação do Exemplo 1.38 na representação matricial e na representação por grafos.

[1] Para saber mais sobre passeios em grafos, consulte a Subsecção Passeios.

Relações de ordem. Uma relação binária R em A diz-se uma *relação de ordem*, se for reflexiva, anti-simétrica e transitiva.

EXEMPLO 1.40. Seja A o conjunto dos divisores positivos de 18, $A = \{1, 2, 3, 6, 9, 18\}$ e R a relação definida por

$$R = \{(x, y) \in A^2 : x \text{ divide } y\}.$$

Podemos representar R por extensão como se segue

$$\begin{aligned} R = \quad & \{(1,1),(1,2),(1,3),(1,6),(1,9),(1,18),(2,2),(2,6),(2,18),\\ & (3,3),(3,6),(3,9),(3,18),(6,6),(6,18),(9,9),(9,18),(18,18)\}. \end{aligned}$$

É fácil observar que esta relação é reflexiva, anti-simétrica e transitiva e, portanto, R é uma relação de ordem em A.

EXEMPLO 1.41. O exemplo anterior pode ser estendido ao conjunto dos números inteiros positivos, $\mathbb{Z}^+$. A relação x *divide* y é uma relação de ordem em $\mathbb{Z}^+$.

Seja R uma relação de ordem num conjunto A. Dizemos que R é uma *relação de ordem total* se satisfizer a condição seguinte (propriedade de *dicotomia*):

$$\forall a, b \in A,\ (a, b) \in R \vee (b, a) \in R.$$

Neste caso, dizemos que o par (A, R) é um *conjunto totalmente ordenado*.

Se R for uma relação de ordem sobre um conjunto A e não for de ordem total, dizemos ainda que R é uma *relação de ordem parcial*.

EXEMPLO 1.42. A relação R do Exemplo 1.40 é uma relação de ordem parcial. Basta observar que $(6, 9) \notin R$ e $(9, 6) \notin R$.

EXEMPLO 1.43. A relação $R : x$ *divide* y é uma relação de ordem total sobre o conjunto dos divisores de 32, $D_{32} = \{1, 2, 4, 8, 16, 32\}$. Verifique.

Relações de equivalência. Uma relação binária R sobre A diz-se uma *relação de equivalência*, se for reflexiva, simétrica e transitiva.

EXEMPLO 1.44. Seja R a relação sobre $A = \{1, 2, 3, 4\}$ representada em extensão por

$$R = \{(1,1),(1,2),(1,3),(2,2),(2,1),(2,3),(3,3),(3,1),(3,2),(4,4)\}.$$

R é uma relação de equivalência. Verifique.

Seja R uma relação de equivalência sobre um conjunto A. Denominamos de *classe de equivalência* de um elemento $x \in A$ ao conjunto de todos os elementos de A que estão relacionados com x pela relação R, ou seja,

$$[x]_R = \{y \in A : (x, y) \in R\}.$$

Quando não existem dúvidas em relação a R, a classe de equivalência de x também se denota por $[x]$. Ao elemento x damos o nome de *representante da classe*. Notemos que este representante não é único: qualquer outro elemento da mesma classe pode ser um representante dela.

EXEMPLO 1.45. Sejam $A = \{\text{Ana, Bernardo, Camila, Diogo, Eurico}\}$ um subconjunto de alunos do ISEP e R a relação definida por $xRy \Leftrightarrow x$ e y estudam no mesmo curso. Suponhamos que a Ana estuda Engenharia Química, o Bernardo estuda Engenharia Informática, a Camila estuda Engenharia Informática, o Diogo estuda Engenharia Civil e o Eurico estuda Engenharia Informática.

É evidente que R é uma relação de equivalência e existem três classes de equivalência em A, que são:

$$[\text{Ana}] = \{\text{Ana}\}$$
$$[\text{Bernardo}] = \{\text{Bernardo, Camila, Eurico}\}$$
$$[\text{Diogo}] = \{\text{Diogo}\}.$$

Notemos que [Bernardo] = [Camila], pelo que, tanto Bernardo, como Camila, e ainda Eurico, podem ser representantes desta classe de equivalência.

TEOREMA 1.5. *Seja R uma relação de equivalência sobre um conjunto A. As condições seguintes são verificadas:*

1. $\forall a \in A,\ a \in [a]$ *e, portanto,* $\forall a \in A,\ [a] \neq \emptyset$.
2. $\forall a, b \in A,\ aRb \Leftrightarrow [a] = [b]$.
3. $\forall a, b \in A,\ [a] \neq [b] \Rightarrow [a] \cap [b] = \emptyset$.
4. $A = \bigcup_{a \in A} [a]$.

Seja A um conjunto não vazio e seja $P \subseteq \mathcal{P}(A)$ um conjunto que satisfaz as condições seguintes:

1. $\forall S \in P,\ S \neq \emptyset$;
2. $\forall S_1, S_2 \in P,\ S_1 \neq S_2 \Rightarrow S_1 \cap S_2 = \emptyset$;
3. $A = \bigcup_{S \in P} S$.

Então o conjunto P é designado de *partição do conjunto A*.

COROLÁRIO 1.6. *Seja R uma relação de equivalência sobre um conjunto A. Então a família de classes de equivalência formam uma partição de A.*

EXEMPLO 1.46. As classes de equivalência do Exemplo 1.45 formam a seguinte partição de A:

$$\begin{aligned} P &= \{[\text{Ana}], [\text{Bernardo}], [\text{Diogo}]\} \\ &= \{\{\text{Ana}\}, \{\text{Bernardo, Camila, Eurico}\}, \{\text{Diogo}\}\}. \end{aligned}$$

EXEMPLO 1.47. Consideremos a relação binária sobre o conjunto dos números inteiros, $\mathbb{Z}$, definida por $x \equiv y\,(mod\ 5)$ *se e só se* 5 *divide* $x - y$ ou, de modo equivalente, $x \equiv y\,(mod\ 5)$ *se e só se x e y divididos por* 5 *têm o mesmo resto.* Mostramos que $\equiv \pmod 5$, denominada de *congruência módulo* 5, é uma relação de equivalência. Com efeito, temos:

Reflexividade: Para todo o $z \in \mathbb{Z}$, temos que 5 divide $z - z = 0$, logo $z \equiv z \pmod 5$.

Simetria: Se $u \equiv v \pmod 5$, então 5 divide $u - v$, ou seja, existe $z \in \mathbb{Z}$ tal que $u - v = 5z$. Mas então, também se tem $v - u = 5(-z)$, com $-z \in \mathbb{Z}$. Resulta que 5 divide $v - u$ e, portanto, $v \equiv u \pmod 5$.

Transitividade: Se $u \equiv v \pmod 5$ e $v \equiv w \pmod 5$, então $u - v = 5z$ e $v - w = 5t$, com $z, t \in \mathbb{Z}$. Resulta que $u - w = 5(z + t)$ e, portanto, $u \equiv w \pmod 5$.

As classes de equivalência podem ser representadas pelos restos da divisão por 5, sendo as seguintes:

$$[0] = \{x \in \mathbb{Z} : x \equiv 0 \,(\text{mod } 5)\} = \{\ldots, -10, -5, 0, 5, 10, \ldots\},$$
$$[1] = \{x \in \mathbb{Z} : x \equiv 1 \,(\text{mod } 5)\} = \{\ldots, -9, -4, 1, 6, 11, \ldots\},$$
$$[2] = \{x \in \mathbb{Z} : x \equiv 2 \,(\text{mod } 5)\} = \{\ldots, -2, -3, 2, 7, 12, \ldots\},$$
$$[3] = \{x \in \mathbb{Z} : x \equiv 3 \,(\text{mod } 5)\} = \{\ldots, -7, -2, 3, 8, 13, \ldots\},$$
$$[4] = \{x \in \mathbb{Z} : x \equiv 4 \,(\text{mod } 5)\} = \{\ldots, -6, -1, 4, 9, 14, \ldots\}.$$

Fecho de relações. Seja R uma relação binária sobre um conjunto A. Definimos o:

Fecho reflexivo: de R em A, como a menor relação reflexiva S sobre A que contémR, ou seja,

$$S = R \cup \{(x, x) : x \in A\}.$$

Fecho simétrico: de R em A, como a menor relação simétrica S sobre A que contémR, ou seja,

$$S = R \cup \{(y, x) : (x, y) \in R\}.$$

Fecho transitivo: de R em A, como a menor relação transitiva S sobre A que contém R, ou seja,

$$S = \bigcup_{i=1}^{\infty} R^i,$$

onde $R^i = R \circ \cdots \circ R$ (i vezes). Prova-se ainda que a igualdade seguinte é válida

$$S = R \cup R^2 \cup R^3 \cup \cdots \cup R^n,$$

onde $n = |A|$. Assim, se R for representada matricialmente pela matriz M, então o fecho transitivo de R é representado matricialmente por

$$M \oplus M^2 \oplus M^3 \oplus \cdots \oplus M^n,$$

onde $\oplus$ é a soma Booleana de matrizes.

Assim, dada uma relação binária R sobre um conjunto A, é possível calcular a menor relação de equivalência sobre A que contém R, como podemos ver no exemplo seguinte:

EXEMPLO 1.48. Consideremos a relação R do Exemplo 1.38, em que $A = \{1, 2, 3, 4\}$ e R é representada em extensão por

$$R = \{(1, 1), (1, 2), (2, 1), (2, 2), (2, 4), (3, 3), (4, 2), (4, 4)\}.$$

Vimos que esta relação é reflexiva, simétrica e não é transitiva. É fácil verificar que o fecho transitivo de R é a relação

$$S = \{(1, 1), (1, 2), (2, 1), (1, 4), (2, 2), (2, 4), (3, 3), (4, 1), (4, 2), (4, 4)\}$$

que é uma relação de equivalência.

Exercícios propostos

1. Descreva os conjuntos seguintes por extensão:
   a) $A = \{x \in \mathbb{R} : x^2 = 1\}$.
   b) $B = \{x \in \mathbb{Q} : x^2 = 2\}$.
   c) $C = \{x \in \mathbb{Z} : 0 \leq x < 5\}$.
   d) $D = \{x \in \mathbb{Z} : x \text{ é divisível por 3 e } x \in [-5, 16[\}$.
   e) $E = \{x \in \mathbb{N} : x \text{ é o quadrado de um número inteiro e } x \leq 100\}$.
   f) $F = \{x \in \mathbb{R} : x \text{ é um número primo e é par}\}$.

2. Descreva os conjuntos seguintes por compreensão:
   a) $A = \{2, 4, 6, 8, 10\}$.
   b) $B = \{3, 5, 7, 11, 13, 17, 19\}$.
   c) $C = \{7, 14, 21, 28, 35, 42, 49, 56, 63\}$.

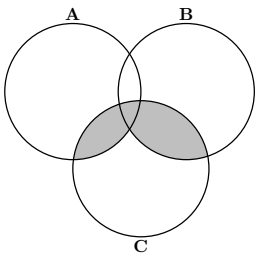
3. Considere os conjuntos seguintes:

$$A = \{x \in \mathbb{R} : x \text{ é um número par}\},$$
$$B = \{x \in \mathbb{Z} : x \text{ é múltiplo de } 6\},$$
$$C = \{x \in \mathbb{Z} : x \text{ é múltiplo de } 3\},$$
$$D = \{x \in \mathbb{N} : 2x \leq 11\},$$
$$E = \{x \in \mathbb{N} : x^2 < 30\}.$$

   Complete as afirmações seguintes, usando os símbolos $=$, $\subsetneq$ e $\supsetneq$:
   a) $A \ldots B$.
   b) $B \ldots C$.
   c) $D \ldots E$.

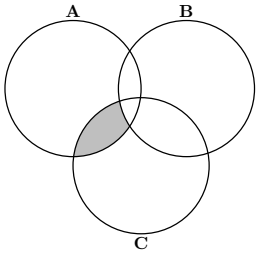
4. Determine a união e a interseção dos conjuntos $\mathbb{N}$ e $\{x \in \mathbb{Z} : x \text{ é par}\}$.

5. Para cada um dos conjuntos sombreados, escolha a expressão que o representa:

a)


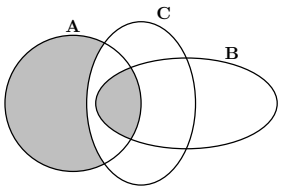
(i) $(A \cap B) \cap C$.
(ii) $A \cup (B \cup C)$.
(iii) $(A \cap B) \cup C$.
(iv) $(A \cup B) \cap C$.

b)



(i) $A \cap (B \backslash C)$.
(ii) $A \backslash (A \cap C)$.
(iii) $C \backslash (A \cup B)$.
(iv) $(A \cap C) \backslash B$.

c)



(i) $(A \backslash C) \cup (A \cap B)$.
(ii) $A \backslash C$.
(iii) $(A \backslash C) \cup B$.
(iv) $(A \cup B) \backslash C$.

6. Em 415 alunos inscritos nas unidades curriculares de Matemática Discreta (MDISC) e de Linguagens e Programação (LPROG), 275 alunos frequentam as aulas de MDISC, 110 alunos frequentam as aulas de LPROG e 55 alunos não frequentam nenhuma delas. Determine quantos alunos frequentam:
   a) só Matemática Discreta.
   b) Matemática Discreta ou Linguagens e Programação, mas não as duas.
   c) Matemática Discreta e Linguagens e Programação.
   d) pelo menos uma das duas.
   e) só Linguagens e Programação.

7. Escreva a expressão $|A \cup B|$ em termos de $|A|$, $|B|$ e $|A \cap B|$. E para $|A \cup B \cup C|$?

8. Sejam A, B e C conjuntos tais que $|A| = 50$, $|B| = 30$, $|C| = 85$, $|A \cap B| = 22$, $|A \cap B \cap C| = 18$, $|B \cap C| = 20$ e $|A \cup C| = 100$. Determine as cardinalidades seguintes:
   a) $|A \cap C|$.
   b) $|C \backslash B|$.
   c) $|(B \cap C) \backslash (A \cap B \cap C)|$.
   d) $|A \cup B \cup C|$.

9. Sejam $X = \{\text{André, Bernardo, Cecília}\}$, $Y = \{\text{ananás, banana, cereja, damasco}\}$ e $Z = \{\text{iogurte, leite, sumo}\}$.
   a) Descreva o conjunto $X \times Y$ por extensão.
   b) O Bernardo lancha sempre uma bebida e uma peça de fruta. Se, em casa dele, existirem exatamente os alimentos do conjunto $Y \cup Z$, de quantas formas diferentes o Bernardo poderá elaborar o lanche? Justifique, usando o conceito de cardinal de um conjunto.

10. Sejam A e B conjuntos tais que $A \times B = \emptyset$. O que pode concluir acerca dos conjuntos A e B?

11. Para cada um dos conjuntos seguintes, calcule o conjunto das suas partes:
    a) $\{a, b\}$.
    b) $\emptyset$.
    c) $\{\emptyset, A\}$.

12. Sem descrever os conjuntos por extensão, calcule o número de elementos de cada um dos conjuntos seguintes:
    a) $\mathcal{P}(\{a, b, \{a, b\}\})$.
    b) $\mathcal{P}(\{\emptyset, a, \{a\}, \{\{a\}\}\})$.
    c) $\mathcal{P}(\mathcal{P}(\emptyset))$.

13. Usando as propriedades das operações sobre os conjuntos, mostre que $(A \cup B) \cap (A \cup \emptyset) = A$.

14. Sejam $A = \{1, 2, 8, 9\}$ e $B = \{1, 2\}$. Calcule $\Big|\mathcal{P}\big((A \times B) \setminus (B \times A)\big)\Big|$.

15. Seja A um conjunto tal que $|A| = 4$. Sabendo que $\big\{\emptyset, \{1, 2\}, \{1, 2, 5\}, \{\emptyset\}\big\} \subset \mathcal{P}(A)$, determine o conjunto A.

16. Determine, se possível, um conjunto A tal que: $|\mathcal{P}(A)| = 16$ e $A \cap \mathbb{N} = \{2, 3, 5\}$. Se responder que não existe um tal conjunto, justifique a sua resposta.

17. Considere as expressões seguintes:
    (i) $(-1)^2$;
    (ii) O Rio Douro nasce em Espanha;
    (iii) $|1 - 5| = 6$;
    (iv) O Sol gira à volta da Terra;
    (v) $7 + 2 \times (-3)$;
    (vi) $x^2 = 1$;
    (vii) Se n é primo ímpar, então n é divisível por 2;
    (viii) Cuidado!;
    (ix) Vinho do Porto;
    (x) $|-5|$.
    a) Distinga as designações, das proposições e das expressões ambíguas. Indique o valor lógico das proposições.
    b) Determine se existem designações ou proposições equivalentes e, em caso afirmativo, indique-as.

18. Sejam p, q e r as proposições seguintes:

$p =$ Eu recebi o louvor na disciplina
$q =$ Eu resolvi todos os exercícios
$r =$ Eu obtive uma classificação excelente a Matemática Discreta

Escreva as proposições seguintes em linguagem simbólica, ou seja, usando p, q, r e os conetivos lógicos.

a) Eu obtive uma classificação excelente a Matemática Discreta, mas eu não resolvi todos os exercícios.

b) Eu resolvi todos os exercícios, recebi o louvor na disciplina e obtive uma classificação excelente a Matemática Discreta.

c) Para eu receber o louvor na disciplina é necessário que obtenha uma classificação excelente a Matemática Discreta.

d) Eu recebi o louvor na disciplina, mas eu não resolvi todos os exercícios; mesmo assim, eu obtive uma classificação excelente a Matemática Discreta.

e) Obter uma classificação excelente a Matemática Discreta e resolver todos os exercícios é suficiente para eu receber o louvor na disciplina.

f) Eu receberei o louvor na disciplina se e só se resolver todos os exercícios ou obtiver uma classificação excelente a Matemática Discreta.

19. Sejam p, q e r as proposições seguintes:

$p =$ Eu não estudo
$q =$ A Matemática é fácil
$r =$ A Matemática é interessante

Escreva as proposições seguintes em linguagem corrente.

a) $q \wedge r$.

b) $\sim p \Rightarrow q$.

c) $(r \wedge q) \Rightarrow \sim p$.

20. Negue cada uma das proposições seguintes e simplifique:

a) $\sim p \wedge q$.

b) $\sim\sim p \vee \sim q$.

c) $p \Rightarrow \sim q$.

21. Sejam p, q e r proposições com valores lógicos, respetivamente, V, F e V. Determine o valor lógico das proposições seguintes:

a) $(\sim p \wedge q) \wedge r$.

b) $(\sim p \wedge \sim r) \wedge (\sim (q \vee r) \wedge \sim (r \wedge p))$.

c) $\sim (p \Rightarrow q) \vee (q \Rightarrow \sim p)$.

22. Sabendo que a proposição

$$\sim (p \vee q) \Rightarrow r$$

é falsa, determine, se possível, o valor lógico de cada uma das proposições:

a) $p \Leftrightarrow q$.

b) $r \Rightarrow \sim q$.

c) $p \wedge r$.

23. Sabendo que as proposições

$$\sim a \vee b, \quad c \Rightarrow \sim b, \quad \sim c \Rightarrow d, \quad a$$

são simultaneamente verdadeiras, determine, se possível, os valores lógicos das proposições b, c e d.

24. Sabendo que a proposição q tem valor lógico *verdade*, determine, se possível, os valores lógicos de p, r e s que tornam a expressão

$$\Big[q \Rightarrow \big((\sim p \vee r) \wedge \sim s\big)\Big] \wedge \big[\sim s \Rightarrow (\sim r \wedge q)\big]$$

numa proposição verdadeira.

25. Construa uma tabela de verdade para cada uma das proposições:

a) $(p \wedge q) \vee (\sim p \vee q)$.

b) $(p \Rightarrow q) \Rightarrow [(p \vee \sim q) \Rightarrow (p \vee q)]$.

c) $(p \vee q) \Leftrightarrow [p \Rightarrow (q \wedge \sim p)]$.

d) $p \vee (q \wedge r) \Leftrightarrow [(\sim p \Rightarrow q) \wedge (\sim p \Rightarrow r)]$.

26. Utilizando as propriedades das operações lógicas, simplifique as proposições compostas seguintes:

a) $\sim [\sim (q \Rightarrow \sim p) \vee ((p \vee q) \wedge p)]$.

b) $\sim [[p \wedge (p \Rightarrow q)] \Rightarrow (p \wedge q)]$.

c) $(p \vee q) \Rightarrow [(p \wedge q) \vee (p \wedge \sim q) \vee (\sim p \wedge q)]$.

27. Utilizando as propriedades das operações lógicas, mostre que as proposições compostas são logicamente equivalentes.

a) $p \Leftrightarrow q$ e $(p \wedge q) \vee (\sim p \wedge \sim q)$.

b) $(p \wedge q) \vee (\sim p \wedge q) \vee (\sim p \wedge \sim q)$ e $q \vee \sim p$.

c) $(\sim p \vee q) \wedge \sim [(p \vee \sim q) \Rightarrow (p \vee q)]$ e $\sim p \wedge \sim q$.

28. Verifique se as proposições compostas seguintes são tautologias ou contradições. No caso de não ser nenhuma delas, justifique-o apresentando uma combinação de valores lógicos a atribuir a cada uma das proposições simples que contrarie cada um dos conceitos.

a) $[(p \wedge q) \vee (p \vee \sim q)] \Rightarrow q$.

b) $[(p \wedge q) \Rightarrow r] \Leftrightarrow [(p \Rightarrow r) \vee (q \Rightarrow r)]$.

c) $[p \vee (q \wedge r)] \Rightarrow [(q \Rightarrow p) \Rightarrow r]$.

d) $[(p \vee q) \wedge \sim (p \vee r)] \wedge \sim q$.

29. Sejam $P(x, y)$ o predicado *x gosta de y* e o domínio das variáveis o conjunto de todas as pessoas do mundo. Usando quantificadores, escreva em linguagem matemática cada uma das proposições seguintes:
    a) Toda a gente gosta da Maria.
    b) Toda a gente gosta de alguém.
    c) Existe alguém de quem toda a gente gosta.
    d) Ninguém gosta de toda a gente.

30. Usando quantificadores, escreva em linguagem matemática cada uma das proposições seguintes e determine o seu valor lógico:
    a) Há pelo menos um número inteiro que é igual ao seu triplo.
    b) Todos os números naturais são não negativos.
    c) Todo o número real, se é menor do que 3, então é menor do que π.
    d) A cada número real corresponde outro que é o seu dobro.

31. Sejam $P(x)$ o predicado *x estuda mais de* 10 *horas semanais* e o domínio da variável o universo de todos os estudantes da Licenciatura em Engenharia Informática. Traduza para linguagem corrente as proposições seguintes:
    a) $\exists x$, $P(x)$.
    b) $\forall x$, $P(x)$.
    c) $\exists x$, $\sim P(x)$.
    d) $\forall x$, $\sim P(x)$.

32. Considere o predicado $x \leq x^2$. Quantifique-o e defina um domínio das variáveis de modo a transformá-lo numa proposição:
    a) verdadeira.
    b) falsa.

33. Negue cada uma das proposições seguintes, apresentando o resultado sem o símbolo$\sim$:
    a) $\exists n \in \mathbb{N}$, $n > 1 \wedge n \leq 4$.
    b) $\forall x \in \mathbb{R}$, $(x > 2 \wedge x < 4) \vee x > 3$.
    c) $\forall z \in \mathbb{Z}$, $z \geq -2 \wedge z < 3 \Rightarrow |z| < 3$.
    d) $\exists x, y \in \mathbb{R}$, $x > 2 \Rightarrow x + y \leq 2$.

34. Indique, justificando, o valor lógico das proposições seguintes:
    a) $\forall n \in \mathbb{N}$, $n^2 > n$.
    b) $\exists n \in \mathbb{N}$, $2n = 3n$.

c) $\forall n \in \mathbb{N}\, \exists m \in \mathbb{N},\ n^2 < m$.

d) $\exists n \in \mathbb{N}\, \forall m \in \mathbb{N},\ n < m^2$.

e) $\exists n \in \mathbb{N}\, \forall m \in \mathbb{N},\ nm = m$.

f) $\exists n \in \mathbb{N}\, \forall m \in \mathbb{N},\ n^2 + m^2 = 5$.

35. Considere os predicados seguintes:

$$P(x) : x^2 - x - 2 = 0;$$
$$Q(x) : x \text{ é par};$$
$$R(x) : x > 0.$$

Determine o valor lógico das proposições que se seguem.

a) $\forall x \in \mathbb{Z},\ P(x) \Rightarrow Q(x)$.

b) $\exists x \in \mathbb{Z},\ R(x) \wedge P(x)$.

c) $\exists x \in \mathbb{Z},\ P(x) \wedge Q(x)$.

d) $\forall x \in \mathbb{Z},\ Q(x) \Rightarrow P(x)$.

36. a) Indique, justificando, o valor lógico das seguintes proposições:

(i) $\forall x \in \mathbb{R}, \exists y \in \mathbb{R},\ xy = 1$;

(ii) $\exists z \in \mathbb{R}, \forall x, y \in \mathbb{R},\ x + y = z$;

(iii) $\forall x \in \mathbb{R}, \exists y \in \mathbb{R},\ x + y > 0$;

(iv) $\exists x \in \mathbb{R}, \forall y \in \mathbb{R},\ xy \leq 0$.

b) Negue as proposições da alínea anterior, apresentando o resultado sem o símbolo $\sim$.

37. Indique os pares ordenados pertencentes à relação R entre $A = \{0, 1, 2, 3\}$ e $B = \{0, 1, 2\}$, onde R é definida por:

a) $(a, b) \in R$ se e só se $a = b$.

b) $(a, b) \in R$ se e só se $a + b = 2$.

c) $(a, b) \in R$ se e só se $a + b > 3$.

d) $(a, b) \in R$ se e só se $a \leq b$.

e) $(a, b) \in R$ se e só se b é múltiplo de a.

38. Indique os pares ordenados da relação R entre $A = \{1, 2, 3, 4\}$ e $B = \{a, 1, h, 3\}$ representada na forma matricial pela matriz

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

39. Represente as relações no conjunto $\{a, b, c, d\}$ que se seguem sob a forma matricial e por grafos:
    a) $R = \{(b, b), (b, c), (b, d), (c, b), (c, c), (c, d)\}$.
    b) $S = \{(a, a), (a, b), (b, a), (b, b), (c, c), (d, d)\}$.
    c) $T = \{(b, d), (d, b)\}$.
    d) $U = \{(a, b), (b, c), (c, d)\}$.

40. Das relações do exercício anterior, indique as que são:
    a) Reflexivas.
    b) Simétricas.
    c) Anti-simétricas.
    d) Transitivas.

41. Considere as relações sobre o conjunto dos estudantes do ISEP:

$$R = \{(a, b) : a \text{ é mais alto que } b\}$$
$$S = \{(a, b) : a \text{ nasceu no mesmo dia que } b\}$$
$$T = \{(a, b) : a \text{ tem o mesmo nome que } b\}$$
$$U = \{(a, b) : a \text{ tem um avô comum a } b\}$$

    Diga quais das relações são:
    a) Reflexivas.
    b) Simétricas.
    c) Transitivas.

42. Para cada uma das matrizes seguintes que representam a relação R sobre o conjunto $A = \{a, b, c, d\}$, classifique a relação quanto à reflexividade, simetria, anti-simetria e transitividade.

    a) $\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$ b) $\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$ c) $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$

43. Considere a relação binária R sobre o conjunto $A = \{1, 2, 3, 4\}$ definida por xRy *se e só se* $xy \geq 3$.
    a) Represente a matriz da relação R.
    b) Represente o grafo orientado da relação R.
    c) Usando o grafo orientado da alínea anterior, classifique a relação quanto à reflexividade, simetria, anti-simetria e transitividade.

44. Considere cada uma das relações binárias R que se seguem sobre o conjunto dos números naturais e classifique-as quanto à reflexividade, simetria, anti-simetria e transitividade:
    a) $(x, y) \in R$ se e só se $x \neq y$.
    b) $(x, y) \in R$ se e só se $x = y + 1 \vee x = y - 1$.
    c) $(x, y) \in R$ se e só se x é múltiplo de y.
    d) $(x, y) \in R$ se e só se $x = y^2$.
    e) $(x, y) \in R$ se e só se $x \geq y^2$.

45. Dê exemplo de relações binárias que sejam:
    a) Simétrica e anti-simétrica.
    b) Simétrica e não anti-simétrica.
    c) Não simétrica e anti-simétrica.
    d) Nem simétrica nem anti-simétrica.

46. Quantas relações simétricas e anti-simétricas sobre um conjunto A de cardinal 4 existem?

47. Determine o número de relações:
    a) Reflexivas de cardinal n, para $n = 1, 2, 3$.
    b) Simétricas de cardinal n, para $n = 1, 2, 3$.
    c) Anti-simétricas de cardinal n, para $n = 1, 2, 3$.
    d) Generalize os resultados anteriores (n arbitrário).

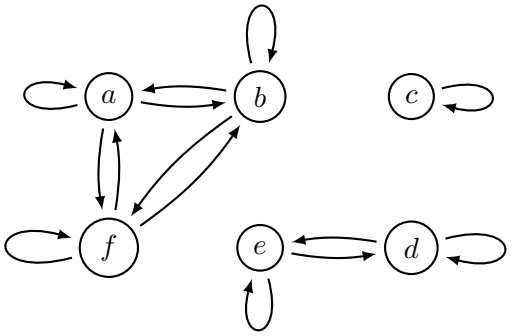
48. Determine quais das seguintes relações sobre o conjunto $A = \{a, b, c, d\}$ são relações de equivalência e/ou de ordem. Caso não o sejam, indique as propriedades que não são verificadas:
    a) $R = \{(a, a), (b, b), (c, c), (d, d)\}$.
    b) $R = \{(a, a), (a, c), (c, a), (c, c), (c, d), (d, c), (d, d)\}$.
    c) $R = \{(a, a), (b, b), (b, c), (c, b), (c, c), (d, d)\}$.
    d) $R = \{(a, a), (b, b), (b, d), (c, c), (c, d), (d, b), (d, c), (d, d)\}$.
    e) $R = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, c), (d, d)\}$.

49. Para as relações de equivalência do exercício anterior, construa o grafo orientado e calcule as classes de equivalência.

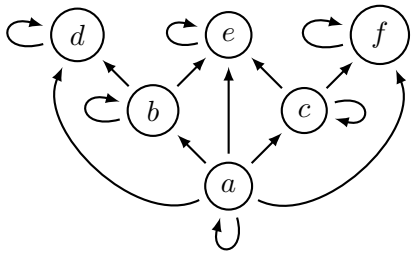
50. Considere o conjunto $A = \{1, 2, 3, 4\}$ e a partição $P = \{\{1, 2, 3\}, \{4\}\}$ de A. Determine a relação de equivalência sobre A cujas classes de equivalência formam a partição P.

51. Considere a relação $R = \{(m, n) : m \text{ é múltiplo de } n\}$, definida no conjunto dos números naturais. Verifique se R é uma relação de ordem total.

52. Considere a relação R, definida no conjunto $A = \{a, b, c, d, e, f\}$, representada pelo digrafo abaixo indicado.
    a) Determine a matriz representativa de R.
    b) Mostre que R é uma relação de equivalência.
    c) Determine a partição de A formada pelas classes de equivalência de R.



53. Considere a relação R, definida no conjunto $A = \{a, b, c, d, e, f\}$, representada pelo digrafo abaixo indicado.
    a) Determine a matriz representativa de R.
    b) Mostre que R é uma relação de ordem.
    c) Diga, justificando convenientemente, se R é uma relação de ordem parcial ou total.



54. Sejam R a relação binária em $\{1, 2, 3, 4\}$, $R = \{(1, 1), (1, 4), (2, 3), (3, 1), (3, 4)\}$, e S a relação binária de $\{1, 2, 3, 4\}$ para $\{0, 1, 2\}$, $S = \{(1, 0), (2, 0), (3, 1), (3, 2), (4, 1)\}$. Determine a relação $S \circ R$.

55. Seja R a relação binária representada pela matriz $\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$. Escreva a matriz representativa da relação:
    a) R^{-1}.
    b) R^2.

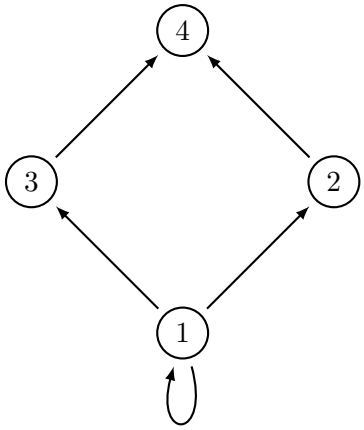
56. Sejam R e S as relações binárias no conjunto das pessoas definidas por:

$$R = \{(a, b)|\ a \text{ é progenitor de } b\};$$

$$S = \{(a, b)|\ a \text{ e } b \text{ são irmãos}\}.$$

Escreva as relações $S \circ R$ e $R \circ S$.

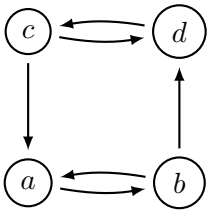
57. Considere as relações R e S definidas no conjunto $A = \{1, 2, 3, 4\}$ e representadas, respetivamente, pelos digrafos abaixo. Represente o digrafo da relação $S \circ R$.

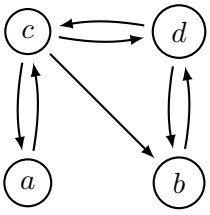


58. Seja R a relação no conjunto $\{0, 1, 2, 3\}$, $R = \{(0, 1), (1, 1), (1, 2), (2, 0), (2, 2), (3, 0)\}$. Determine:
    a) O fecho reflexivo de R.

    b) O fecho simétrico de R.

59. Desenhe o digrafo do fecho reflexivo e do fecho simétrico de cada uma das relações binárias representadas pelos digrafos seguintes:

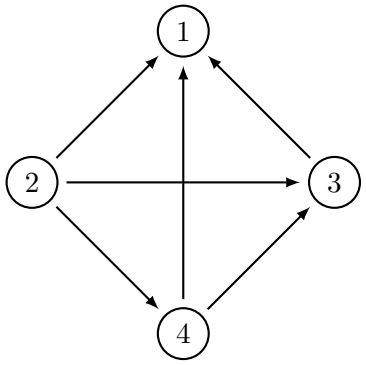
a)


b)


60. Determine o fecho transitivo das relações em $\{1, 2, 3, 4\}$ seguintes:
    a) $\{(2, 1), (2, 3), (3, 1), (3, 4), (4, 1), (4, 3)\}$.

    b) $\{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\}$.

61. Considere a relação R definida no conjunto $A = \{1, 2, 3, 4\}$ e representada pelo digrafo abaixo. Represente o digrafo do fecho transitivo, $\hat{R}$, de R.



Soluções dos exercícios propostos

1. a) $A = \{-1, 1\}$
   b) $B = \emptyset$
   c) $C = \{0, 1, 2, 3, 4\}$
   d) $D = \{-3, 0, 3, 6, 9, 12, 15\}$
   e) $E = \{0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100\}$
   f) $F = \{2\}$

2. a) $A = \{$números pares positivos inferiores a 11$\}$
   b) $B = \{$números primos ímpares inferiores a 20$\}$
   c) $C = \{$múltiplos de 7 positivos e inferiores a 70$\}$

3. a) $\supsetneq$ b) $\subsetneq$ c) $=$

4. $\mathbb{N} \cup \{x \in \mathbb{Z} : x \text{ é par}\} = \{\ldots, -6, -4, -2, 0, 1, 2, 3, \ldots\}$
   $\mathbb{N} \cap \{x \in \mathbb{Z} : x \text{ é par}\} = \{0, 2, 4, 6, 8, 10, \ldots\}$

5. a) (iv) b) (iv) c) (i)

6. a) 250 c) 25 e) 85
   b) 335 d) 360

7. $|A \cup B| = |A| + |B| - |A \cap B|$ e
   $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$.

8. a) 35 b) 65 c) 2 d) 106

9. a) $X \times Y =$ {(André,ananás),(André,banana),(André,cereja),(André,damasco),
   (Bernardo,ananás),(Bernardo,banana),(Bernardo,cereja),(Bernardo,damasco),
   (Cecília,ananás),(Cecília,banana),(Cecília,cereja),(Cecília,damasco)}
   b) $|Y \times Z| = 12$

10. Ou $A = \emptyset$, ou $B = \emptyset$, ou ambos.

11. a) $\mathcal{P}(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$

b) $\mathcal{P}(\emptyset) = \{\emptyset\}$

c) $\mathcal{P}(\{\emptyset, A\}) = \{\emptyset, \{\emptyset\}, \{A\}, \{\emptyset, A\}\}$

12. a) 8 b) 16 c) 2

14. 16

15. $A = \{1, 2, 5, \emptyset\}$

16. $A = \{2, 3, 5, \sqrt{2}\}$, por exemplo.

17. a) designações: (i), (v), (ix), (x)
proposições: (ii) V, (iii) F, (iv) F, (vii) F
expressões ambíguas: (vi), (viii)

b) designações equivalentes: (i) e (v)
proposições equivalentes: (iii), (iv) e (vii)

18. a) $r \wedge \sim q$ c) $p \Rightarrow r$ e) $r \wedge q \Rightarrow p$

b) $q \wedge p \wedge r$ d) $p \wedge \sim q \wedge r$ f) $p \Leftrightarrow q \vee r$

19. a) A Matemática é fácil e interessante.

b) Se eu estudo, então a Matemática é fácil.

c) Se a Matemática é interessante e fácil, então eu estudo.

20. a) $p \vee \sim q$ b) $\sim p \wedge q$ c) $p \wedge q$

21. a) F b) F c) V

22. a) V b) V c) F

23. $V(b) = V(d) = V$ e $V(c) = F$.

24. $V(p) = V(r) = V(s) = F$

25. a)

p	q	$\sim p$	$p \wedge q$	$\sim p \vee q$	$(p \wedge q) \vee (\sim p \vee q)$
V	V	F	V	V	V
V	F	F	F	F	F
F	V	V	F	V	V
F	F	V	F	V	V

b)

p	q	$\sim q$	$p \Rightarrow q$	$p \vee \sim q$	$p \vee q$	$(p \vee \sim q) \Rightarrow (p \vee q)$	$(p \Rightarrow q) \Rightarrow [(p \vee \sim q) \Rightarrow (p \vee q)]$
V	V	F	V	V	V	V	V
V	F	V	F	V	V	V	V
F	V	F	V	F	V	V	V
F	F	V	V	V	F	F	F

c)

p	q	$\sim p$	$p \vee q$	$q \wedge \sim p$	$p \Rightarrow (q \wedge \sim p)$	$(p \vee q) \Leftrightarrow [p \Rightarrow (q \wedge \sim p)]$
V	V	F	V	F	F	F
V	F	F	V	F	F	F
F	V	V	V	V	V	V
F	F	V	F	F	V	F

d)

p	q	r	$\sim p$	$q \wedge r$	$P \equiv p \vee (q \wedge r)$	$Q \equiv \sim p \Rightarrow q$	$R \equiv \sim p \Rightarrow r$	$S \equiv Q \wedge R$	$P \Leftrightarrow S$
V	V	V	F	V	V	V	V	V	V
V	V	F	F	F	V	V	V	V	V
V	F	V	F	F	V	V	V	V	V
V	F	F	F	F	V	V	V	V	V
F	V	V	V	V	V	V	V	V	V
F	V	F	V	F	F	V	F	F	V
F	F	V	V	F	F	F	V	F	V
F	F	F	V	F	F	F	F	F	V

26. a) $\sim p$ b) F c) V

28. a) Não é tautologia nem contradição ($\equiv q$). Fazendo $V(p) = V$ e, respetivamente $V(q) = F$ e $V(q) = V$, temos um contra-exemplo de que não é tautologia, nem é contradição, respetivamente.

    b) É tautologia.

    c) Não é tautologia nem contradição ($\equiv \sim p \vee r$). Fazendo $V(q) = V = V(p)$ e, respetivamente $V(r) = F$ e $V(r) = V$, temos um contra-exemplo de que não é tautologia, nem é contradição, respetivamente.

    d) É contradição.

29. a) $\forall x,\ P(x, \text{Maria})$

    b) $\forall x \exists y,\ P(x, y)$

    c) $\exists y \forall x,\ P(x, y)$

    d) $\sim (\exists x \forall y,\ P(x, y))$

30. a) $\exists x \in \mathbb{Z},\ x = 3x$, verdadeira.

    b) $\forall x \in \mathbb{N},\ x \geq 0$, verdadeira.

    c) $\forall x \in \mathbb{R},\ x < 3 \Rightarrow x < \pi$, verdadeira.

    d) $\forall x \in \mathbb{R} \exists y \in \mathbb{R},\ y = 2x$, verdadeira.

31. a) Há pelo menos um estudante de Engenharia Informática que estuda mais de 10 horas semanais.

    b) Todos os estudantes de Engenharia Informática estudam mais de 10 horas semanais.

    c) Há pelo menos um estudante de Engenharia Informática que não estuda mais de 10 horas semanais.

    d) Nenhum estudante de Engenharia Informática estuda mais de 10 horas semanais.

32. a) $\forall x \in \mathbb{N},\ x \leq x^2$ b) $\forall x \in \mathbb{R},\ x \leq x^2$

33. a) $\forall n \in \mathbb{N},\ n \leq 1 \vee n > 4$
    b) $\exists x \in \mathbb{R},\ (x \leq 2 \vee x \geq 4) \wedge x \leq 3$
    c) $\exists z \in \mathbb{Z},\ (z \geq -2 \wedge z < 3) \wedge |z| \geq 3$
    d) $\forall x, y \in \mathbb{R},\ x > 2 \wedge x + y > 2$

34. a) F, não se verifica para $n = 0$ e $n = 1$.
    b) V, $n = 0$.
    c) V, basta fazer $m = n^2 + 1$, por exemplo.
    d) F, fazendo $m = 0$ não existe n que satisfaça a condição.
    e) V, $n = 1$ é o elemento neutro da multiplicação.
    f) F, basta fazer $m = 3$.

35. a) F b) V c) V d) F

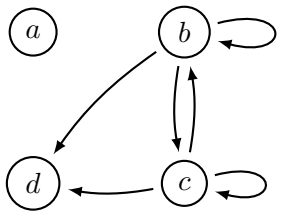
36. a) (i) F (ii) F (iii) V (iv) V

    b) (i) $\exists x \in \mathbb{R}, \forall y \in \mathbb{R},\ xy \neq 1$
    (ii) $\forall z \in \mathbb{R}, \exists x, y \in \mathbb{R},\ x + y \neq z$
    (iii) $\exists x \in \mathbb{R}, \forall y \in \mathbb{R},\ x + y \leq 0$
    (iv) $\forall x \in \mathbb{R}, \exists y \in \mathbb{R},\ xy > 0$

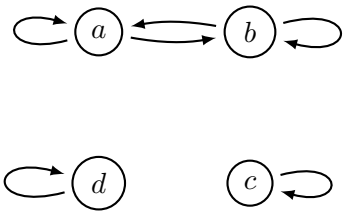
37. a) $R = \{(0,0), (1,1), (2,2)\}$
    b) $R = \{(0,2), (1,1), (2,0)\}$
    c) $R = \{(2,2), (3,1), (3,2)\}$
    d) $R = \{(0,0), (0,1), (0,2), (1,1), (1,2), (2,2)\}$
    e) $R = \{(0,0), (1,0), (1,1), (1,2), (2,0), (2,2), (3,0)\}$

38. $R = \{(1,a), (1,1), (1,3), (2,1), (2,h), (3,a), (3,3), (4,1), (4,h), (4,3)\}$

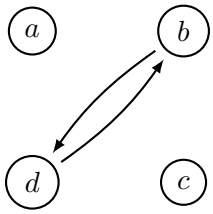
39. a) $\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$



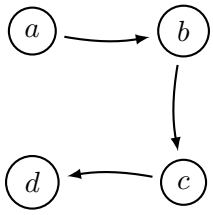
    b) $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$



c) $\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$



d) $\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$



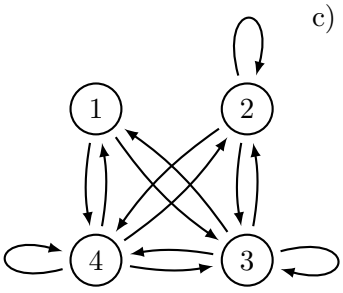
40. a) S b) S e T c) U d) R e S

41. a) S, T e U b) S, T e U c) R, S e T

42. a) simétrica
    b) reflexiva e anti-simétrica
    c) simétrica

43. a) $\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$

b)



c) Não é reflexiva, é simétrica, não é anti-simétrica e não é transitiva.

44. a) simétrica
    b) simétrica
    c) reflexiva, anti-simétrica e transitiva
    d) anti-simétrica
    e) anti-simétrica e transitiva

46. 15

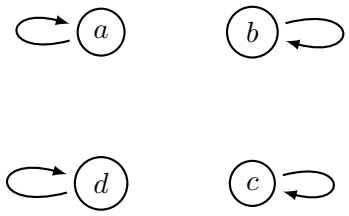
47. a) 1, 4, 64.
    b) 1, 7, 63.
    c) 1, 11, 215.
    d) 2^{n^2-n}, $2^{\binom{n+1}{2}} - 1$, $2^n 3^{\binom{n}{2}} - 1$.

48. a) relação de equivalência e de ordem
    b) não é uma relação de equivalência porque não é reflexiva nem transitiva, nem é uma relação de ordem porque para além de não verificar as duas propriedades anteriores também não é anti-simétrica

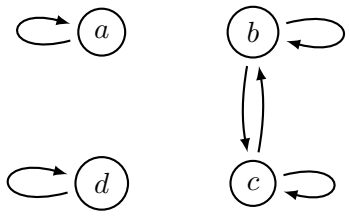
c) relação de equivalência, mas não de ordem porque não é anti-simétrica

d) não é uma relação de equivalência porque não é transitiva nem é uma relação de ordem porque não é anti-simétrica nem transitiva

e) não é uma relação de equivalência porque não é simétrica nem transitiva, nem é uma relação de ordem porque não é anti-simétrica nem transitiva

49. a) 

Classes de equivalência: $\{a\}$, $\{b\}$, $\{c\}$ e $\{d\}$.

c) 

Classes de equivalência: $\{a\}$, $\{b,c\}$ e $\{d\}$.

50. $R = \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3),(4,4)\}$

51. Não. Por exemplo, os elementos 2 e 3 não estão relacionados.

52. a) $\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$

c) $\mathcal{P}(A) = \{\{a,b,f\},\{c\},\{d,e\}\}$

53. a) $\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

c) Ordem parcial.

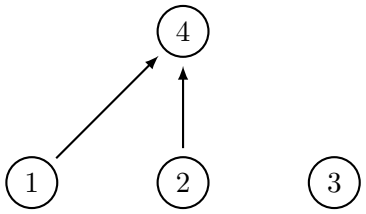
54. $S \circ R = \{(1,0),(1,1),(2,1),(2,2),(3,0),(3,1)\}$

55. a) $\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$

b) $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

56. $S \circ R = \{(a,c)|$ existe b tal que a é progenitor de b e b é irmão de $c\}$;
$R \circ S = \{(a,b)|$ a é tio(a) de $b\}$.

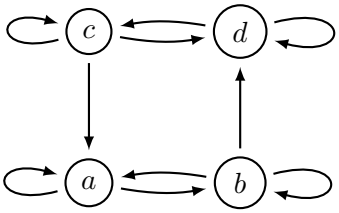
57.



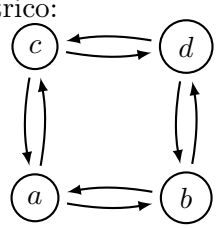
58. a) $R = \{(0,0),(0,1),(1,1),(1,2),(2,0),(2,2),(3,0),(3,3)\}$

b) $R = \{(0,1),(0,2),(0,3),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2),(3,0)\}$

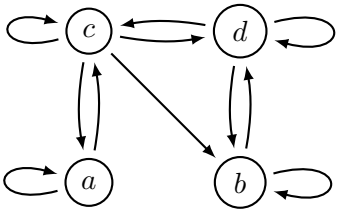
59. a) Fecho reflexivo:



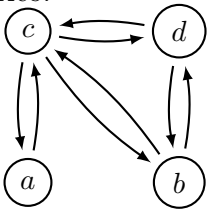
Fecho simétrico:



b) Fecho reflexivo:



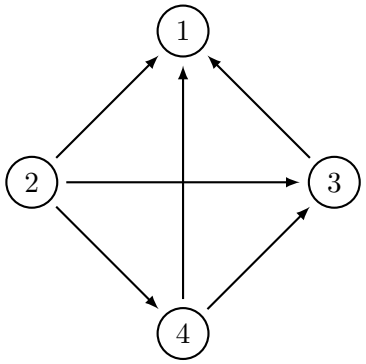
Fecho simétrico:



60. a) $\{(2,1),(2,3),(2,4),(3,1),(3,3),(3,4),(4,1),(4,3),(4,4)\}$.

b) $\{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$.

61.



CAPíTULO 2

Introdução às Estratégias de Demonstração

A prova (da existência de uma infinidade de números primos) é por reductio ad absurdum, e reductio ad absurdum, que Euclides tanto amava, é uma das armas favoritas de um matemático. É uma jogada muito mais refinada do que qualquer jogada de xadrez: um jogador de xadrez pode oferecer o sacrifício de um peão ou até mesmo de outra peça, mas um matemático oferece o jogo.

G. H. Hardy, *A Mathematician's Apology.*

1. Estratégias de demonstração da implicação

A *demonstração* é essencial na Matemática, pois é ela que fundamenta os resultados matemáticos. É, portanto, importante que o leitor possua alguns conhecimentos e competências sobre alguns métodos e técnicas usados numa demonstração.

Entendida como *Ciência do Raciocínio*, a Lógica explicita as leis que nos permitem encadear um raciocínio, ou seja, obter proposições verdadeiras a partir de proposições verdadeiras. Estas leis têm o nome de *Regras de Inferência*.

Um *teorema* é uma proposição que se pode mostrar ser verdadeira, ou seja, que se pode demonstrar. É usualmente escrito sob a forma de implicação $H \Rightarrow T$, onde H é a *hipótese, premissa* ou *antecedente* e T é a *tese, conclusão* ou *consequente*. Significa que, se H é uma proposição verdadeira, então T é também uma proposição verdadeira. A proposição H, que constitui a hipótese, pode ser uma proposição composta, como exemplo, pode ser a conjunção de várias proposições $H_1 \wedge H_2 \wedge \cdots \wedge H_k$.

Um *lema* é um teorema menor, necessário na prova de um outro teorema.

Um *corolário* é uma consequência imediata de um teorema.

Uma *conjetura* é uma afirmação que não se sabe se é verdadeira ou falsa, mas que se acredita ser verdadeira.

Começamos por apresentar as *Regras de Inferência* mais usadas que servirão de utensílio para os três métodos de demonstração de uma implicação apresentados de seguida: a *prova direta*, a *prova por contraposição* e a *prova por redução ao absurdo*.

Notemos que, quando pretendemos provar que uma certa afirmação é verdadeira, temos de fazer uma prova formal, usando uma das técnicas de demonstração existentes. No entanto, para provar que uma certa afirmação é falsa, basta apresentar um *contra-exemplo*, ou seja, uma situação particular onde a afirmação falha.

Exemplo 2.1. **Afirmação:** Todos os números primos são ímpares.

A afirmação é falsa pois 2 é um número primo e é par. Este é então um contra-exemplo da afirmação.

A *prova* de um teorema é uma sequência encadeada de afirmações que formam um argumento de veracidade do teorema. Começa com assunções, que poderão ser hipóteses ou axiomas, e termina no resultado do teorema.

Regras de inferência. As Regras de Inferência são regras lógicas que permitem a dedução de conclusões a partir de permissas, descrevendo formalmente o raciocínio lógico. Foi Aristóteles (384-322 bC) quem primeiro as formalizou, com um exemplo da regra *Modus ponens*:

Todos os homens são mortais.
Sócrates é um homem.
Logo Sócrates é mortal.

A ideia de uma regra de inferência é abstrairmo-nos dos detalhes e concentrarmo-nos na forma do argumento. A cada regra de inferência corresponde uma tautologia e, portanto, a regra é verdadeira independentemente da veracidade das proposições que a compõem.

As regras de inferência mais conhecidas são as apresentadas na tabela seguinte. Usamos a notação

$$\begin{array}{r} p_1 \\ p_2 \\ \vdots \\ \underline{p_k} \\ \therefore q \end{array}$$

para representar $(p_1 \wedge p_2 \cdots \wedge p_k) \Rightarrow q$ e lemos p_1 *e* p_2 *e* ... *e* p_k, *logo* q.

Apresentamos alguns exemplos de uso destas regras:

EXEMPLO 2.2. A *Regra da adição* é a regra de inferência que se baseia na tautologia $p \Rightarrow (p \vee q)$ e é usada no argumento seguinte:

A Maria vai ao cinema.
Logo a Maria vai ao cinema ou vai a um concerto.

EXEMPLO 2.3. A *Regra da simplificação* é a regra de inferência que se baseia na tautologia $(p \wedge q) \Rightarrow p$ e é usada no argumento seguinte:

A Maria vai ao cinema e vai a um concerto.
Logo a Maria vai ao cinema.

EXEMPLO 2.4. O *Silogismo da disjunção* é a regra de inferência que se baseia na tautologia $[(p \vee q) \wedge \sim p] \Rightarrow q$ e é usada no argumento seguinte:

O Joaquim estuda Matemática Discreta ou vê televisão.
O Joaquim não vê televisão.
Logo o Joaquim estuda Matemática Discreta.

EXEMPLO 2.5. *Modus ponens* é, provavelmente, a regra mais famosa. Baseia-se na tautologia $[(p \Rightarrow q) \wedge p] \Rightarrow q$ e é usada no argumento seguinte:

Regras de inferência

Designação da regra	Regra	Tautologia
Regra da adição	$\dfrac{p}{\therefore\ p \vee q}$	$p \Rightarrow (p \vee q)$
Regra da simplificação	$\dfrac{p \wedge q}{\therefore\ p}$	$(p \wedge q) \Rightarrow p$
Regra do absurdo	$\dfrac{p \Rightarrow (q \wedge \sim q)}{\therefore \sim p}$	$[p \Rightarrow (q \wedge \sim q)] \Rightarrow \sim p$
Silogismo da disjunção	$\begin{array}{r} p \vee q \\ \sim p \\ \hline \therefore\ \ q \end{array}$	$[(p \vee q) \wedge \sim p] \Rightarrow q$
Modus ponens	$\begin{array}{r} p \Rightarrow q \\ p \\ \hline \therefore\ \ q \end{array}$	$[(p \Rightarrow q) \wedge p] \Rightarrow q$
Modus tollens	$\begin{array}{r} p \Rightarrow q \\ \sim q \\ \hline \therefore \sim p \end{array}$	$[(p \Rightarrow q) \wedge \sim q] \Rightarrow \sim p$
Transitividade da implicação	$\begin{array}{r} p \Rightarrow q \\ q \Rightarrow r \\ \hline \therefore\ p \Rightarrow r \end{array}$	$[(p \Rightarrow q) \wedge (q \Rightarrow r)] \Rightarrow (p \Rightarrow r)$

Se está sol, eu vou passear até à Foz do Douro.
Está sol.
Logo eu vou passear até à Foz do Douro.

EXEMPLO 2.6. *Modus tollens* é a regra de inferência que se baseia na tautologia $[(p \Rightarrow q) \wedge \sim q] \Rightarrow \sim p$ e é usada no argumento seguinte:

Se está sol, eu vou passear até à Foz do Douro.
Eu não vou passear até à Foz do Douro.
Logo não está sol.

EXEMPLO 2.7. A *Transitividade da implicação* é a regra de inferência que se baseia na tautologia $[(p \Rightarrow q) \wedge (q \Rightarrow r)] \Rightarrow (p \Rightarrow r)$ e é usada no argumento seguinte:

O Joaquim diverte-se quando estuda Matemática Discreta.
O Joaquim é feliz quando se diverte.
Logo o Joaquim é feliz quando estuda Matemática Discreta.

Chamamos de *falácia* a um argumento logicamente inconsistente, sem fundamento, inválido ou incapaz de ser provado.

Há duas falácias muito usuais, que resultam de usarmos, supostamente, uma regra que não é baseda numa tautologia e, portanto, não é uma regra de inferência:

- Afirmar a conclusão e concluir a hipótese de uma implicação.
- Negar a hipótese e concluir a negação da conclusão.

EXEMPLO 2.8. Consideremos a proposição *Se eu estudar muito, então eu terei uma nota excelente a Matemática Discreta.*

É uma falácia o argumento *Se eu tiver uma nota excelente a Matemática Discreta, então eu estudei muito*, pois a equivalência

$$(p \Rightarrow q) \Leftrightarrow (q \Rightarrow p)$$

não é uma tautologia.

É uma falácia o argumento *Se eu não estudar muito, então eu não terei uma nota excelente a Matemática Discreta*, pois a equivalência

$$(p \Rightarrow q) \Leftrightarrow (\sim p \Rightarrow \sim q)$$

não é uma tautologia.

EXEMPLO 2.9. Consideremos o argumento seguinte:

$$\begin{aligned} a = b &\Leftrightarrow a + a = a + b \\ &\Leftrightarrow 2a - 2b = a - b \\ &\Leftrightarrow 2(a - b) = 1(a - b) \\ &\Leftrightarrow 2 = 1 \end{aligned}$$

A falácia ocorre na última equivalência do argumento pois foi feita a divisão por $a - b$, que não é mais do que a divisão por 0 ($a = b$).

Servem estes exemplos para alertar o leitor para os cuidados a ter quando elaboramos um argumento.

Prova direta. Uma demonstração direta de um teorema consiste numa sequência de proposições *válidas* que termina na tese T, onde se considera que uma proposição é *válida* se estiver numa das formas seguintes:

1. É uma das hipóteses $H_1, H_2, \ldots, H_k$.
2. É uma tautologia ou um axioma conhecidos.
3. Obtém-se das proposições anteriores através das Regras de Inferência.

Para demonstrar que um argumento complexo é válido, dividimos o argumento em passos, cada um correspondendo a uma regra de inferência. Vejamos o exemplo que se segue:

EXEMPLO 2.10. Queremos construir um argumento que conclua *Nós estaremos em casa antes do pôr do sol* a partir das hipóteses que se seguem:

Hipótese 1: *Hoje está a chover e está mais frio do que ontem.*

Hipótese 2: *Nós faremos um passeio no Gerês, só se não estiver a chover.*

Hipótese 3: *Se não fizermos um passeio no Gerês, então vamos a uma exposição a Serralves.*

Hipótese 4: *Se formos a uma exposição a Serralves, então estaremos em casa antes do pôr do sol.*

Etiquetamos as proposições:

p**:** *hoje está a chover*

q**:** *hoje está mais frio do que ontem*

r**:** *nós faremos um passeio no Gerês*
s**:** *nós vamos a uma exposição a Serralves*
t**:** *nós estaremos em casa antes do pôr do sol*

e escrevemos as hipóteses na forma simbólica:

H_1**:** $p \wedge q$
H_2**:** $r \Rightarrow \sim p$
H_3**:** $\sim r \Rightarrow s$
H_4**:** $s \Rightarrow t$

Podemos agora partir a nossa prova nos argumentos seguintes:

1. $p \wedge q$ (H_1)
2. p (Regra da simplificação)
3. $r \Rightarrow \sim p$ (H_2)
4. $\sim r$ (Modus tollens, usando (2) e (3))
5. $\sim r \Rightarrow s$ (H_3)
6. s (Modus ponens, usando (4) e (5))
7. $s \Rightarrow t$ (H_4)
8. t (Modus ponens, usando (6) e (7))

e concluir o pretendido.

Apresentamos um outro exemplo que ilustra uma prova direta sem explicitar as regras de inferência (embora se trate de um argumento composto por uma cadeia de *Modus ponens*).

Exemplo 2.11. Usemos a prova direta para demonstrar o teorema seguinte:

$$(\forall n \in \mathbb{N}\backslash\{1\},\ n \text{ é ímpar}) \Rightarrow (\exists k \in \mathbb{N}, n^2 = 8k + 1).$$

Seja $n > 1$ um número ímpar. Então $n = 2p + 1$, para algum $p \in \mathbb{N}$. Temos

$$n^2 = (2p + 1)^2 = 4p^2 + 4p + 1 = 4p(p + 1) + 1.$$

Notemos que $p(p + 1)$ é um número par. Logo, existe $k \in \mathbb{N}$ tal que $p(p + 1) = 2k$. Resulta que

$$n^2 = 4p(p + 1) + 1 = 8k + 1,$$

como queríamos demonstrar.

Demonstração por contraposição. Este método de demonstração baseia-se na tautologia do Cálculo Proposicional, conhecida como Lei da Conversão,

$$(p \Rightarrow q) \Leftrightarrow (\sim q \Rightarrow \sim p).$$

Consiste em demonstrar a implicação $\sim q \Rightarrow \sim p$ para concluir a implicação equivalente $p \Rightarrow q$. Assume-se que a conclusão é falsa e usam-se as regras de inferência, os axiomas e as tautologias, para mostrar que a hipótese também é falsa.

Vejamos alguns exemplos ilustrativos deste tipo de prova:

Exemplo 2.12. Queremos provar a afirmação *Se eu estudar em casa, então vou aos concertos de verão*, usando a prova por contraposição e as hipóteses seguintes:

Hipótese 1: *Se eu estiver com atenção nas aulas ou se estudar em casa, então vou ter aprovação a Matemática Discreta.*

Hipótese 2: *Se eu tiver aprovação a Matemática Discreta, então vou aos concertos de verão.*

Assim, construimos um argumento que conclua *Eu não estudei em casa*, adicionando às hipóteses anteriores a seguinte:

Hipótese 3: *Eu não vou aos concertos de verão.*

Etiquetamos as proposições,

p**:** *eu estive com atenção nas aulas*
q**:** *eu estudei em casa*
r**:** *eu tive aprovação a Matemática Discreta*
s**:** *eu vou aos concertos de verão*

e escrevemos as hipóteses na forma simbólica

H_1**:** $(p \vee q) \Rightarrow r$
H_2**:** $r \Rightarrow s$
H_3**:** $\sim s$

Podemos agora partir a nossa prova nos argumentos seguintes:

1. $\sim s$ (H_3)
2. $r \Rightarrow s$ (H_2)
3. $\sim r$ (Modus tollens, usando (1) e (2))
4. $(p \vee q) \Rightarrow r$ (H_1)
5. $\sim (p \vee q)$ (Modus tollens, usando (3) e (4))
6. $\sim p \wedge \sim q$ (Lei de De Morgan e (5))
7. $\sim q$ (Regra da simplificação e (6))

e concluir que $\sim s \Rightarrow \sim q$, que é equivalente a $q \Rightarrow s$.

EXEMPLO 2.13. Usemos a prova por contraposição para demonstrar o resultado seguinte:

$$\forall n \in \mathbb{N},\ n! > n + 1 \Rightarrow n > 2.$$

Suponhamos que não se tem a conclusão, ou seja, que $n \leq 2$. Como $n \in \mathbb{N}$, então $n = 0$, ou $n = 1$, ou $n = 2$. Temos

$$n = 0 \Rightarrow (0! = 1 \leq 0 + 1 = 1),$$

$$n = 1 \Rightarrow (1! = 1 \leq 1 + 1 = 2)$$

e

$$n = 2 \Rightarrow (2! = 2 \leq 2 + 1 = 3).$$

Resulta que $n! \leq n + 1$, ou seja, a hipótese é falsa. Logo está demonstrado o resultado.

EXEMPLO 2.14. Usemos a prova por contraposição para demonstrar o teorema seguinte:

Se o produto de dois inteiros α e β é par, então pelo menos um deles é par.

Suponhamos que a conclusão é falsa, ou seja, α e β são dois inteiros ímpares. Então existem $m, n \in \mathbb{Z}$ tal que

$$\alpha = 2m + 1 \text{ e } \beta = 2n + 1$$

e, portanto, temos

$$\alpha\beta = (2m + 1)(2n + 1) = 4mn + 2m + 2n + 1 = 2(2mn + m + n) + 1$$

que é, obviamente, um número ímpar. Mostramos então que

Se dois inteiros α e β são ímpares, então o seu produto é ímpar.

o que é equivalente à afirmação que pretendemos mostrar.

Demonstração por redução ao absurdo. Este método de demonstração baseia-se na tautologia do Cálculo Proposicional, conhecida como Relação da Implicação com a Disjunção e a Negação

$$(p \Rightarrow q) \Leftrightarrow (\sim p \vee q)$$

que, composta com a aplicação da Lei de De Morgan

$$(\sim p \vee q) \Leftrightarrow \sim (p \wedge \sim q),$$

resulta na tautologia

$$(p \Rightarrow q) \Leftrightarrow \sim (p \wedge \sim q).$$

Assim, o método consiste em assumirmos que a hipótese é verdadeira e a conclusão é falsa e, usando as regras de inferência, os axiomas e as tautologias, chegarmos a um absurdo. O absurdo resulta de se ter suposto que a conclusão era falsa, o que mostra que é verdadeira.

Vejamos alguns exemplos ilustrativos deste tipo de prova:

Exemplo 2.15. Suponhamos que numa turma teórica de Matemática Discreta estão presentes 50 alunos. Vamos mostrar, por redução ao absurdo, que existem pelo menos 5 alunos com o aniversário no mesmo mês.

Com efeito, suponhamos que tal não é verdadeiro, ou seja, que não existe nenhum grupo de 5 alunos com o aniversário no mesmo mês. Sejam M_i o número de alunos com aniversário no i-ésimo mês, para $i = 1, \ldots, 12$. Temos

$$\begin{cases} M_i \leq 4, \ i = 1, \ldots, 12 \\ \sum_{i=1}^{12} M_i = 50 \end{cases} \Rightarrow 50 = \sum_{i=1}^{12} M_i \leq \sum_{i=1}^{12} 4 = 12 \times 4 = 48$$

o que é uma contradição, pois existem 50 alunos presentes. A contradição resultou de se ter suposto que não existe um grupo de 5 alunos com aniversário no mesmo mês.

Exemplo 2.16. Usemos o método de demonstração por redução ao absurdo para provar o teorema seguinte:

Há um número infinito de números naturais primos.

Suponhamos que a afirmação não é verdadeira e sejam $p_1, p_2, \ldots, p_k$ todos os números naturais primos. Consideremos o número $n = p_1 p_2 \cdots p_k + 1$. Ora, é evidente que este número é maior que qualquer um dos primos $p_1, p_2, \ldots, p_k$ e, portanto, n não é um número primo. Logo n é um número composto. Por outro lado, a divisão deste número por qualquer primo $p_1, p_2, \ldots, p_k$ dá resto 1. No entanto, um número composto é divisível por algum número primo. Resulta daqui um absurdo, que partiu de se ter suposto que o conjunto de números primos é finito.

2. Princípio de Indução Matemática

O *Princípio de indução* é usado frequentemente em Matemática para provar proposições quantificadas universalmente num universo bem-ordenado[1]. Usa-se quando temos uma conjetura sobre o resultado universal e conseguimos provar o resultado de um certo passo usando o passo anterior.

Seja $P(n)$ o predicado *o inteiro não negativo n satisfaz a propriedade P*.

O *Princípio de indução matemática* decorre da regra de inferência seguinte

$$[P(n_0) \wedge \forall_{n \geq n_0}(P(n) \Rightarrow P(n+1))] \Rightarrow \forall_{n \geq n_0} P(n)$$

onde n é uma variável inteira e $\forall_{n \geq n_0}(P(n) \Rightarrow P(n+1))$ denota a conjunção das proposições $P(n) \Rightarrow P(n+1)$ quando n percorre todos os valores inteiros não inferiores a n_0.

Chamamos a $P(n_0)$ o *passo base* e à implicação $P(n) \Rightarrow P(n+1)$ o *passo de indução*, onde $P(n)$ é a *hipótese de indução* e $P(n+1)$ é a *tese de indução*.

O *Princípio de indução matemática* pode então ser descrito na forma seguinte:

Princípio de indução: Temos $P(n)$, $\forall n \geq n_0$, se as condições seguintes forem satisfeitas:

1. $P(n_0)$ é verdadeira;
2. $\forall_{n \geq n_0}$ a implicação $(P(n) \Rightarrow P(n+1))$ é também verdadeira.

De seguida, apresentamos alguns exemplos de aplicação deste método.

EXEMPLO 2.17. *A soma dos primeiros n inteiros positivos é dada pela expressão* $\frac{n(n+1)}{2}$.

A afirmação anterior traduz-se na linguagem matemática por

$$\forall n \in \mathbb{N}\backslash\{0\},\ \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

ou, de um modo expandido,

$$\forall n \in \mathbb{N}\backslash\{0\},\ 1 + 2 + \cdots + n = \frac{n(n+1)}{2}.$$

Provemos, por indução, que ela é verdadeira.

Passo base: $P(1)$: Temos $\sum_{i=1}^{1} i = 1$ e $\frac{1(1+1)}{2} = 1$, logo a condição verifica-se para $n = 1$;

Passo de indução:

Hipótese de indução: $P(n) : \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

Tese de indução: $P(n+1) : \sum_{i=1}^{n+1} i = \frac{(n+1)(n+2)}{2}$

[1] Um conjunto diz-se *bem-ordenado* se todo o seu subconjunto não-vazio tiver elemento mínimo. Em particular, um conjunto bem-ordenado é numerável.

Suponhamos que se tem a hipótese de indução. Mostremos que a tese de indução é válida:
Temos

$$\begin{array}{rll} \sum_{i=1}^{n+1} i & = \sum_{i=1}^{n} i + \sum_{i=n+1}^{n+1} i & \text{propriedade dos somatórios} \\ & = \frac{n(n+1)}{2} + (n+1) & \text{por hipótese de indução} \\ & = (n+1)(\frac{n}{2}+1) & \\ & = (n+1)(\frac{n+2}{2}) & \\ & = \frac{(n+1)(n+2)}{2} & \end{array}$$

ou seja, provamos a validade de $P(n+1)$.

Resulta que se tem $P(n)$, para todo o $n \in \mathbb{N}$.

Exemplo 2.18. Mostramos agora que

$$\forall n \geq 4, n \text{ inteiro, } n! > 2^n.$$

Passo base: $P(4)$: Temos $4! = 4 \times 3 \times 2 \times 1 = 24$ e $2^4 = 16$, logo $4! > 2^4$ e a condição verifica-se para $n = 4$;

Passo de indução:

Hipótese de indução: $P(n) : n! > 2^n$.

Tese de indução: $P(n+1) : (n+1)! > 2^{n+1}$.

Suponhamos que se tem $P(n) : n! > 2^n$. Mostremos que a tese de indução é válida:
Temos

$$\begin{array}{rll} (n+1)! & = (n+1) \times n! & \text{propriedade do fatorial} \\ & > (n+1) \times 2^n & \text{por hipótese de indução} \\ & > 2 \times 2^n & \text{pois } n \geq 4 \\ & = 2^{n+1} & \end{array}$$

e, portanto, $P(n+1)$ é válida.

Resulta que se tem $n! > 2^n$, para todo o $n \geq 4$.

Existe uma variante do princípio de indução, designado por *Princípio de indução completa*. Consiste em provarmos que a propriedade se verifica para um conjunto finito de valores $n_0, n_0 + 1, n_0 + 2, \ldots, n_0 + l$ e, no passo indutivo, usarmos todos estes resultados para provarmos a tese de indução. É então descrito na forma seguinte:

Princípio de indução completa: Temos $P(n)$, $\forall n \geq n_0$, se as condições seguintes forem satisfeitas:

1. $P(n_0), P(n_0+1), P(n_0+2), \ldots, P(n_0+l)$ são verdadeiras;
2. $\forall_{n \geq n_0+l}$ a implicação $[\forall_{k \in [n_0,n]} P(k)] \Rightarrow P(n+1)$ é também verdadeira.

Usemos o Princípio de indução completa para provar os dois resultados seguintes:

Exemplo 2.19. Consideremos a sucessão $(a_n)_{n \in \mathbb{N} \setminus \{0\}}$ definida por recorrência da forma seguinte:

$$\begin{array}{l} a(1) = 3, \\ a(2) = 6, \\ a(n) = a(n-1) + a(n-2), \ n \geq 3. \end{array}$$

Vamos mostrar que se tem

$$a(n) \text{ é divisível por 3, } \forall n \geq 1.$$

Passo base:

$P(1)$**:** $a(1) = 3$ é divisível por 3,

$P(2)$**:** $a(2) = 6$ é divisível por 3,

logo o passo base verifica-se para $n = 1$ e $n = 2$.

Passo de indução:

Hipótese de indução: $a(k)$ é divisível por 3, para $k = 1, 2, \ldots, n$.

Tese de indução: $a(n+1)$ é divisível por 3.

Suponhamos que se tem $P(k)$, para todo o $k = 1, 2, \ldots, n$. Mostremos que a tese de indução é válida:

Temos

$$\begin{aligned} a(n+1) &= a(n) + a(n-1) \\ &= 3m_n + 3m_{n-1} \qquad \text{por hipótese de indução} \\ &= 3(m_n + m_{n-1}) \end{aligned}$$

ou seja, $a(n+1)$ é divisível por 3 e, portanto, $P(n+1)$ é válida.

Resulta que se tem $P(n)$, para todo o $n \geq 1$.

EXEMPLO 2.20. Consideremos a sucessão $(b_n)_{n\in\mathbb{N}_0}$ definida por recorrência da forma seguinte:

$$\begin{aligned} &b(0) = 12, \\ &b(1) = 29, \\ &b(n) = 5b(n-1) - 6b(n-2),\ n \geq 2. \end{aligned}$$

Vamos mostrar que se tem

$$b(n) = 5 \times 3^n + 7 \times 2^n,\ \forall n \in \mathbb{N}.$$

Passo base:

$P(0)$**:** $b(0) = 12$ e $5 \times 3^0 + 7 \times 2^0 = 5 + 7 = 12$,

$P(1)$**:** $b(1) = 29$ e $5 \times 3^1 + 7 \times 2^1 = 15 + 14 = 29$,

logo o passo base verifica-se para $n = 0$ e $n = 1$.

Passo de indução:

Hipótese de indução: $b(k) = 5 \times 3^k + 7 \times 2^k$, para $k = 0, 1, \ldots, n$.

Tese de indução: $b(n+1) = 5 \times 3^{n+1} + 7 \times 2^{n+1}$.

Suponhamos que se tem $P(k)$, para todo o $k = 0, 1, \ldots, n$. Mostremos que a tese de indução é válida. Temos:

$$\begin{aligned} b(n+1) &= 5b(n) - 6b(n-1) \\ &= 5(5 \times 3^n + 7 \times 2^n) - 6(5 \times 3^{n-1} + 7 \times 2^{n-1}) \qquad \text{por hipótese de indução} \\ &= 5^2 \times 3^n + 5 \times 7 \times 2^n - 6 \times 5 \times 3^{n-1} - 6 \times 7 \times 2^{n-1} \\ &= 5 \times 3^{n-1}(15 - 6) + 7 \times 2^{n-1}(10 - 6) \\ &= 5 \times 3^{n-1}3^2 + 7 \times 2^{n-1}2^2 \\ &= 5 \times 3^{n+1} + 7 \times 2^{n+1} \end{aligned}$$

e, portanto, $P(n+1)$ é válida.

Resulta que se tem $P(n)$, para todo o $n \in \mathbb{N}$.

Terminamos convidando o leitor a descobrir o que há de errado na prova de indução que se segue.

Exercício 2.21. Mostramos por indução que

Todas as bolas de bilhar têm a mesma cor.

Com efeito,

Passo base: A condição inicial corresponde a um conjunto singular, ou seja, um conjunto com uma única bola de bilhar. Logo a condição é trivialmente verdadeira.

Passo de indução: Suponhamos que $P(n)$ é verdadeira, onde n é um número natural arbitrário fixo. Queremos provar que $P(n+1)$ é também verdadeira.

Seja $B = \{b_1, b_2, \ldots, b_{n+1}\}$ o conjunto com as $n+1$ bolas de bilhar e sejam $R = \{b_1, b_2, \ldots, b_n\}$ e $S = \{b_2, \ldots, b_{n+1}\}$ dois subconjuntos de B distintos com n bolas de bilhar. Por hipótese de indução, todas as bolas de R e todas as bolas de S têm a mesma cor. Como a interseção destes dois conjuntos não é vazia, ou seja, existem bolas que são comuns aos dois conjuntos, resulta que $R \cup S$ têm todas as bolas da mesma cor.

Resulta, por indução, que *Todas as bolas de bilhar têm a mesma cor!*

Exercícios propostos

1. Em cada um dos argumentos que se seguem, diga quais as regras de inferência que foram usadas.
   a) A Alice estuda Matemática. Logo a Alice estuda Matemática ou Engenharia Informática.
   b) O Joaquim estuda Matemática e Engenharia Informática. Logo o Joaquim estuda Matemática.
   c) Se estiver a chover, então a piscina estará fechada. Está a chover. Logo a piscina está fechada.
   d) Se hoje nevar no Porto, então o ISEP irá fechar. Hoje o ISEP não fechou. Logo hoje não nevou no Porto.
   e) Se eu for à praia, eu apanharei sol durante muitas horas. Se eu apanhar muitas horas de sol, eu ficarei queimado. Logo, se eu for à praia, eu ficarei queimado.

2. Converta cada um dos argumentos seguintes numa prova simbólica e diga quais as regras de inferência usadas em cada passo.
   a) Para eu sair com o guarda-chuva é necessário que chova. Quando chove eu uso sempre chapéu. Hoje eu não uso chapéu. Então eu não saí com o guarda-chuva.
   b) Para eu sair com o guarda-chuva é suficiente que chova. Para eu usar chapéu é necessário que chova. Eu hoje estou a usar chapéu. Então eu trouxe o guarda-chuva.
   c) Não podemos ser simultaneamente felizes e ricos. Portanto, nós não estamos felizes, ou não somos ricos. Neste momento, pareces feliz. Logo, tu não deves ser rico.
   d) Se as taxas de juros baixarem, então o mercado de ações vai subir. Se as taxas de juros não baixarem, então a construção de habitação e os gastos dos consumidores vão diminuir. Agora os gastos dos consumidores não estão a diminuir. Então o índice de construção não está a diminuir ou os gastos dos consumidores não estão a diminuir, ou seja, é falso que a construção de habitação e os gastos dos consumidores estão ambos a diminuir. Isto significa que as taxas de juros estão a cair e que o mercado de ações vai subir.
   e) Se as taxas de juros ou o mercado de títulos cairem, então o mercado de ações vai subir. Se as taxas de juros não cairem, então a construção de habitação vai diminuir. A construção de habitação está a aumentar. Então as taxas de juros estão em queda. Portanto, é verdade que as taxas de juros ou o mercado de títulos está a cair, e assim que o mercado de ações vai subir.

3. Considere cada um dos argumentos seguintes. Escreva-os em forma simbólica e averigúe se são válidos, justificando a sua resposta. No caso de um argumento ser válido, identifique a regra de inferência.
   a) Se este programa está correto, então ele produz o output correto. O programa produz o output correto. Logo o programa está correto.

b) Se eu for ao cinema, não resolverei a folha de exercícios de MDISC. Se eu não fizer a folha de exercícios de MDISC, eu reprovarei nessa unidade curricular. Logo se eu for ao cinema, eu reprovarei a MDISC.

4. Usando as regras de inferência (e indicando-as) e considerando as permissas seguintes:

a) $\begin{cases} p \wedge q \\ p \Rightarrow \sim (q \wedge r) \\ s \Rightarrow r \end{cases}$, prove $\sim s$.

b) $\begin{cases} a \Rightarrow b \\ \sim a \Rightarrow p \\ (d \wedge b) \Rightarrow c \\ \sim c \\ d \end{cases}$, prove p.

5. Construa uma prova, usando as regras de inferência, para mostrar que as hipóteses
   H1. *Faltei às aulas e está frio.*
   H2. *Se falto às aulas, então não é verdade que esteja frio e a nevar.*
   H3. *Se faço snowboard, então está a nevar.*
   implicam que *Eu não fiz snowboard.*

6. Sabendo que:
   Se o Luís tem cabelo comprido, então é um poeta. Se o Luís for um eremita ou tiver cabelo ruivo, então ele tem o cabelo comprido. O Luís não é um poeta.
   é possível concluir que *O Luís é um eremita ou não tem cabelo ruivo.*?

7. Construa uma prova, usando as regras de inferência, para mostrar que as hipóteses
   H1. *Se não chover ou não houver nevoeiro, então a competição de vela vai decorrer e a demonstração de salvamento vai ser efetuada.*
   H2. *Se a competição de vela decorrer, então o troféu vai ser atribuído.*
   H3. *O troféu não foi atribuído.*
   implicam que *Choveu.*

8. Complete a demonstração seguidamente apresentada, identificando em cada passo as hipóteses, as regras de inferência ou as tautologias utilizadas.
   Se $p \Rightarrow (q \wedge r)$, $(q \vee s) \Rightarrow t$ e $p \vee s$, então t.
   1. $p \Rightarrow (q \wedge r)$
   2. $(q \vee s) \Rightarrow t$
   3. $p \vee s$
   4. $(q \wedge r) \Rightarrow q$

5. $p \Rightarrow q$
6. $(p \vee s) \Rightarrow (q \vee s)$
7. $q \vee s$
8. t

9. Mostre, por prova direta, que, se m e n são dois inteiros ímpares, então $m + n$ é par.

10. Mostre, por prova direta, que, para todos os reais x e y, se $(x + y)^2 = x^2 + y^2$, então $x = 0$ ou $y = 0$.

11. Mostre, por prova direta, que, para todo o real x, se $x > 7$, então $x^2 + x > 8x$.

12. Mostre, por contraposição, que, para todo o real x, se $x^2 < 1$, então $x < 1$.

13. Mostre, por contraposição, que, para todos os inteiros positivos m e n, se $mn = 1$, então $m = 1$ e $n = 1$.

14. Mostre, por contraposição, que, se n é produto de dois números inteiros positivos a e b, então $a \leq \sqrt{n}$ ou $b \leq \sqrt{n}$.

15. Mostre, por contraposição, que, se $(x + y)^3 \neq x^3 + y^3$, então $x + y \neq 0$, para quaisquer números reais x e y.

16. Seja A um subconjunto de um conjunto universal Ω. Mostre, por contraposição, que, se A possui a propriedade $A \subseteq B$ para todo $B \subseteq \Omega$, então $A = \emptyset$.

17. Mostre, por redução ao absurdo, que, se m e n são números naturais tais que $mn \geq 1000$, então $m \geq 10$ ou $n \geq 100$.

18. Mostre, por redução ao absurdo, que, para todo o inteiro positivo n, se n^2 é par, então n é par.

19. Mostre, por redução ao absurdo, que, se x e y forem dois números inteiros, então $x^2 - 4y \neq 2$.

20. Mostre que, se n é um número inteiro positivo, então existe um e um só par (a, b) de números inteiros não negativos tais que $n = 2^a(2b + 1)$.

21. Usando o método de indução matemática, mostre que:
    a) Para todo o $n \in \mathbb{N}$, $n^3 + 2n$ é múltiplo de 3.
    b) Para todo o $n \in \mathbb{N}$, $2^{2n} - 1$ é múltiplo de 3.
    c) Para todo o $n \in \mathbb{N}\backslash\{0\}$, $n! \leq n^n$.
    d) Para todo o $n \in \mathbb{N}$, $(1 + x)^n \geq 1 + nx$, com $x \in \mathbb{R}$ tal que $x > -1$.

22. Usando o princípio de indução matemática, mostre que, qualquer que seja o inteiro positivo n se verificam as igualdades:
    a) $1 + 3 + 5 + \cdots + (2n-1) = n^2$.
    b) $1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$.
    c) $1^3 + 2^3 + 3^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$.
    d) $1 \cdot 1! + 2 \cdot 2! + 3 \cdot 3! + \cdots + n \cdot n! = (n+1)! - 1$.

23. Prove, por indução, que:
    a) $\sum_{i=1}^{n} 2i = n^2 + n,\ \forall n \in \mathbb{N}\backslash\{0\}$.
    b) $\sum_{i=1}^{n} \frac{2}{3^i} = 1 - \frac{1}{3^n},\ \forall n \in \mathbb{N}\backslash\{0\}$.
    c) $\sum_{i=1}^{n} i\, 2^{i-1} = (n-1)\, 2^n + 1,\ \forall n \in \mathbb{N}\backslash\{0\}$.

24. Considere a sucessão $(u_n)_{n\geq 1}$ definida pela relação de recorrência
$$u_n = 1 + u_{n-1} + 2\sqrt{1 + u_{n-1}}, \quad n \geq 2,$$
com valor inicial $u_1 = 0$. Usando o Princípio de Indução Matemática, mostre que
$$\forall n \in \mathbb{N}\backslash\{0\},\ u_n = n^2 - 1.$$

25. Usando o princípio de indução matemática mostre que
$$\prod_{k=2}^{n} \left(1 - \frac{1}{k^2}\right) = \frac{n+1}{2n}, \quad \text{para todo o inteiro } n \geq 2.$$

26. Considere a função f definida por $f(x) = \ln(1+x)$ e seja $f^{(n)}$ a sua derivada de ordem n relativamente à variável x, $x > -1$. Usando o princípio de indução matemática mostre que
$$f^{(n)}(x) = (-1)^{n-1} \frac{(n-1)!}{(1+x)^n}, \quad \text{para todo } n \in \mathbb{N}\backslash\{0\}.$$

27. Determine uma fórmula para $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^n}$, com $n \in \mathbb{N}\backslash\{0\}$, e prove o resultado usando indução matemática. (Sugestão: comece por calcular os valores da expressão para os primeiros inteiros positivos n.)

28. Considere a sucessão $(a_n)_{n\in\mathbb{N}\backslash\{0\}}$ definida por recorrência da forma seguinte:
$$\begin{aligned} a_1 &= 1 \\ a_{n+1} &= a_n + 8n. \end{aligned}$$
Encontre uma fórmula para a sucessão e prove o resultado por indução matemática.

29. Considere a sucessão $(a_n)_{n\in\mathbb{N}\backslash\{0\}}$ definida por recorrência da forma seguinte:
$$\begin{aligned} a_1 &= 1 \\ a_2 &= 2 \\ a_n &= a_{n-1} + a_{n-2},\ n \geq 3. \end{aligned}$$
Mostre, usando o método de indução completa, que, para todo o $n \in \mathbb{N}\backslash\{0\}$, $a_n < (\frac{7}{4})^n$.

30. Considere a sucessão $(a_n)_{n\in\mathbb{N}}$ definida por recorrência da forma seguinte:

$$\begin{aligned} a_0 &= 1 \\ a_1 &= 1 \\ a_2 &= 3 \\ a_n &= 3a_{n-1} - 3a_{n-2} + a_{n-3},\ n \geq 3. \end{aligned}$$

Mostre, usando o método de indução completa, que, para todo o $n \in \mathbb{N}$, $a_n = n^2 - n + 1$.

Soluções dos exercícios propostos

1. a) Regra da adição.
   b) Regra da simplificação.
   c) Modus Ponens.
   d) Modus Tollens.
   e) Transitividade da implicação.

2. a) p**:** *eu saio com o guarda-chuva*
   q**:** *está a chover*
   r**:** *eu uso chapéu*
   Por exemplo,
   1. $p \Rightarrow q$ (H_1)
   2. $q \Rightarrow r$ (H_2)
   3. $\sim r$ (H_3)
   4. $\sim q$ (2, 3, Modus Tolens)
   5. $\sim p$ (1, 4, Modus Tolens)

   b) Por exemplo,
   1. $q \Rightarrow p$ (H_1)
   2. $r \Rightarrow q$ (H_2)
   3. r (H_3)
   4. q (2, 3, Modus Ponens)
   5. p (1, 4, Modus Ponens)

   c) Por exemplo,
   p**:** *estar feliz*
   q**:** *ser rico*
   1. $\sim (p \wedge q)$ (H_1)
   2. $\sim p \vee \sim q$ (Lei de De Morgan)
   3. p (H_2)
   4. $\sim q$ (2, 3, Silogismo da Disjunção)

   d) p**:** *as taxas de juro diminuem*
   q**:** *o mercado de ações desce*
   r**:** *a construção de habitação diminui*
   s**:** *os gastos dos consumidores diminuem*
   t**:** *o mercado de títulos desce*
   Por exemplo,
   1. $p \Rightarrow \sim q$ (H_1)
   2. $\sim p \Rightarrow (r \wedge s)$ (H_2)

3. $\sim s$ (H_3)
4. $\sim r \vee \sim s$ (3, Regra da Adição)
5. $\sim (r \wedge s)$ (4, Lei de De Morgan)
6. p (2, 5, Modus Tolens)
7. $\sim q$ (6, 1, Modus Ponens)

e) Por exemplo,

1. $p \vee t \Rightarrow \sim q$ (H_1)
2. $\sim p \Rightarrow r$ (H_2)
3. $\sim r$ (H_3)
4. p (3, 2, Modus Tolens)
5. $p \vee t$ (4, Regra da Adição)
6. $\sim q$ (5, 1, Modus Ponens)

3. a) p**:** *este programa está correto*

q**:** *o programa produz o output correto*

$$\begin{array}{l} p \Rightarrow q \\ \underline{\quad q} \\ \therefore p \end{array}$$

O argumento não é válido.

b) p**:** *eu vou ao cinema*

q**:** *eu resolvo a folha de exercícios de MDISC*

r**:** *eu reprovo a MDISC*

$$\begin{array}{l} p \Rightarrow \sim q \\ \underline{\sim q \Rightarrow r} \\ \therefore p \Rightarrow r \end{array}$$

O argumento é válido. A regra de inferência usada é a transitividade da implicação.

4. a) Por exemplo,

1. $p \wedge q$ (H_1)
2. $p \Rightarrow \sim (q \wedge r)$ (H_2)
3. $s \Rightarrow r$ (H_3)
4. p (H_1 e simplificação)
5. $\sim (q \wedge r)$ (H_2, 4 e Modus Ponens)
6. $\sim q \vee \sim r)$ (5 e De Morgan)
7. q (H_1 e simplificação)
8. $\sim r$ (6, 7 e Silogismo da disjunção)
9. $\sim s$ (8, H_3 e Modus Tollens)

b) Por exemplo,

1. $a \Rightarrow b$ (H_1)
2. $\sim a \Rightarrow p$ (H_2)
3. $(d \wedge b) \Rightarrow c$ (H_3)
4. $\sim c$ (H_4)
5. d (H_5)
6. $\sim (d \wedge b)$ (4, 3 e Modus Tollens)
7. $\sim d \vee \sim b)$ (6 e De Morgan)
8. $\sim b$ (7, 5 e Silogismo da disjunção)
9. $\sim a$ (1, 8 e Modus Tollens)
10. p (2, 9 e Modus Ponens)

5. Fazendo

p**:** *faltar às aulas*
q**:** *estar frio*
r**:** *estar a nevar*
s**:** *fazer snowboard*

temos as hipóteses

H1. $p \wedge q$
H2. $p \Rightarrow \sim (q \wedge r)$
H3. $s \Rightarrow r$

e a demonstração de que se tem $\sim s$ poderá ser:

1. $p \wedge q$ (H_1)
2. $p \Rightarrow \sim (q \wedge r)$ (H_2)
3. $s \Rightarrow r$ (H_3)
4. p (1 e Regra da simplificação)
5. $\sim (q \wedge r)$ (4, 2 e Modus Ponens)
6. $\sim q \vee \sim r$ (5 e Lei de De Morgan)
7. q (1 e Regra da simplificação)
8. $\sim r$ (7, 6 e Silogismo da disjunção)
9. $\sim s$ (8, 3 e Modus Tollens)

6. Fazendo

p**:** *o Luís tem cabelo comprido*
q**:** *o Luís é um poeta*
r**:** *o Luís é um eremita*
s**:** *o Luís tem cabelo ruivo*

temos as hipóteses

H1. $p \Rightarrow q$
H2. $(r \vee s) \Rightarrow p$
H3. $\sim q$
e a demonstração de que se tem $r \vee \sim s$ poderá ser:

1. $p \Rightarrow q$ (H_1)
2. $(r \vee s) \Rightarrow p$ (H_2)
3. $\sim q$ (H_3)
4. $\sim p$ (1, 3, Modus Tollens)
5. $\sim (r \vee s)$ (2, 4, Modus Tollens)
6. $\sim r \wedge \sim s$ (5 e Lei de De Morgan)
7. $\sim s$ (6 e Regra da simplificação)
8. $r \vee \sim s$ (7 e Regra da adição)

7. Fazendo
   p**:** *chove*
   q**:** *há nevoeiro*
   r**:** *a competição de vela vai decorrer*
   s**:** *a demonstração de salvamento vai ser efetuada*
   t**:** *o troféu vai ser atribuído*
   temos as hipóteses
   H1. $(\sim p \vee \sim q) \Rightarrow (r \wedge s)$
   H2. $r \Rightarrow t$
   H3. $\sim t$
   e a demonstração de que se tem p poderá ser:
   1. $(\sim p \vee \sim q) \Rightarrow (r \wedge s)$ (H_1)
   2. $r \Rightarrow t$ (H_2)
   3. $\sim t$ (H_3)
   4. $\sim r$ (3, 2, Modus Tollens)
   5. $\sim r \vee \sim s$ (4 e Regra da adição)
   6. $\sim (r \wedge s)$ (5 e Lei de De Morgan)
   7. $\sim (\sim p \vee \sim q)$ (6, 1, Modus Tollens)
   8. $p \wedge q$ (7 e Lei de De Morgan)
   9. p (8 e Regra da simplificação)

8. Por exemplo,
   1. $p \Rightarrow (q \wedge r)$ (H_1)
   2. $(q \vee s) \Rightarrow t$ (H_2)
   3. $p \vee s$ (H_3)

4. $(q \wedge r) \Rightarrow q$ (simplificação)
5. $p \Rightarrow q$ (1, 4 e transitividade da implicação)
6. $(p \vee s) \Rightarrow (q \vee s)$ (tautologia $(p \Rightarrow q) \Rightarrow [(p \vee s) \Rightarrow (q \vee s)]$, 5 e Modus Ponens)
7. $q \vee s$ (6, 3, Modus Ponens)
8. t (2, 7, Modus Ponens)

27. $a_n = \frac{2^n-1}{2^n},\ n \geq 1.$

28. $a_n = (2n-1)^2,\ n \geq 1.$

CAPíTULO 3

Análise de Algoritmos

Ao verificar que um dado programa está muito lento, uma pessoa prática pede uma máquina mais rápida ao seu chefe. Mas o ganho potencial que uma máquina mais rápida pode proporcionar é tipicamente limitado por um factor de 10*, por razões técnicas ou económicas. Para obter um ganho maior, é preciso encontrar melhores algoritmos. Um bom algoritmo, mesmo correndo numa máquina lenta, acaba sempre por derrotar (para parâmetros grandes do problema) um algoritmo pior correndo numa máquina rápida. Sempre.*

S. S. Skiena, *The Algorithm Design Manual*.

1. O conceito

Chamamos *algoritmo* a uma sequência finita de instruções bem definidas e não ambíguas para executar uma tarefa. Como exemplos, uma receita de culinária é um algoritmo para preparar uma certa refeição, um livro de instruções de um aparelho pode conter um algoritmo para a montagem do mesmo, um programa de computador é um algoritmo que transmite ao computador os passos específicos e a ordem por que devem ser executados para realizar um cálculo ou resolver um problema. É neste último tipo de algoritmos que debruçamos o nosso estudo.

Há vários aspetos que devemos ter em conta num algoritmo. Em primeiro lugar, o *algoritmo deve estar correto*, ou seja, deverá realizar a tarefa pretendida corretamente para todos os parâmetros válidos. Por exemplo, o algoritmo básico da multiplicação de dois números inteiros (extensível a números racionais) que aprendemos nos primeiros anos de escolarização funciona corretamente para quaisquer dois inteiros que pretendamos multiplicar. Um outro exemplo, um algoritmo que efetue a multiplicação de duas matrizes deve realizar corretamente a multiplicação de qualquer par de matrizes $(A, B) \in \mathcal{M}_{m\times n} \times \mathcal{M}_{n\times p}$. Em segundo lugar, devemos analisar os recursos necessários para a execução do algoritmo. Usualmente, as medidas usadas são a quantidade de tempo que ele leva a implementar a tarefa ou o espaço de memória requerido para a executar. Chamamos esta propriedade de *eficiência* ou *complexidade do algoritmo*. No nosso estudo faremos unicamente a análise da complexidade temporal.

2. Provar que um algoritmo está correto

Os algoritmos são ferramentas matemáticas e, como tal, é desejável que sejamos capazes de fornecer argumentos convincentes para mostrarmos que um dado algoritmo está correto. Notemos que, para mostrarmos que um algoritmo não está correto, i.e., que não executa a tarefa pretendida para todos os parâmetros válidos, é suficiente encontrar um contra-exemplo.

No entanto, mostrarmos que ele está correto é idêntico a provarmos um teorema matemático. Para isso, há vários métodos de demonstração que podemos utilizar, alguns dos quais já foram abordados no Capítulo 2. Têm existido tentativas de construção de programas computacionais capazes de verificar se um algoritmo está correto, mas os progressos têm sido limitados. Aliás, matemáticos e teóricos de Ciência da Computação têm vindo a debater que não será possível mecanizar as provas de que os algoritmos estejam corretos (devido à existência de programas bastante complexos). Não desenvolveremos este problema nesta disciplina, pelo que nos restringiremos a esta abordagem superficial.

Provar que um algoritmo está correto consiste em duas partes: em primeiro lugar, que a resposta correta é sempre obtida para cada parâmetro válido; em segundo lugar, que o algoritmo para.

Como exemplo, para calcularmos o máximo divisor comum de dois números inteiros positivos podemos usar o Algoritmo de Euclides, cuja prova matemática de que está correto é bem conhecida.

```
function mdc(a,b)
    while b ≠ 0
       z := b
       b := a mod b
       a := z
    return a
```

TEOREMA 3.1. *O Algoritmo de Euclides está correto.*

DEMONSTRAÇÃO. Sejam a e b, com $a > b$, dois números inteiros positivos para os quais queremos determinar o máximo divisor comum, $\text{mdc}(a, b)$. Consideremos a sequência de inteiros $(r_n)_n$ tais que $r_1 = a$, $r_2 = b$ e r_{n+2} é o resto da divisão inteira de r_n por r_{n+1}. Como o resto da divisão é menor que o divisor, resulta que a sequência $(r_n)_n$ é decrescente e, portanto, chega a 0 num número finito de passos. Logo o algoritmo para.

Seja N o índice do último resto não-nulo da sequência, r_N. Vamos mostrar que $r_N = \text{mdc}(a, b)$. Notemos que se tem a relação $r_n = r_{n+1} \cdot q_{n+2} + r_{n+2}$, para todo o inteiro positivo n. Da igualdade $r_{N-1} = r_N \cdot q_{N+1}$ sabemos que r_N divide r_{N-1} e, da igualdade $r_{N-2} = r_{N-1} \cdot q_N + r_N$, resulta que r_N divide r_{N-2}. Procedendo por recorrência, resulta quer_N divide todos os termos de $(r_n)_n$. Em particular, r_N é divisor comum de a e b. Por outro lado, todo o divisor comum de a e b é também divisor de $a - b \cdot q_2 = r_2$ e, por recorrência, de todos os elementos de $(r_n)_n$ e, em particular, de r_N. Logo $\text{mdc}(a, b) = r_N$. □

Quando implementamos um algoritmo, esperamos obter uma resposta em tempo finito, pelo que é importante ter a garantia que tal algoritmo para. O problema de provar que um algoritmo para para todos os parâmetros válidos é chamado de *problema da paragem*. Alan Turing provou, em 1936, que este problema é insolúvel. Faremos referência a este problema na última secção.

3. Comportamento assintótico de funções: a notação O

A *complexidade de um algoritmo* é a quantidade de trabalho necessária (temporal e espacial) para a sua execução, que depende do algoritmo e do tamanho do *input*.

Ao estudarmos a complexidade algorítmica pretendemos estimar quais os recursos necessários para um algoritmo ser implementado à medida que o tamanho do problema cresce. Além disso, dado um problema, permite-nos escolher, entre os vários algoritmos que resolvem esse problema, qual deles é o mais eficiente e ainda poder desenvolver novos algoritmos mais eficientes para problemas que já têm solução.

Introduzimos, primeiramente, a *notação* O, uma das ferramentas usadas para estimar a complexidade algorítmica.

Sejam f e g funções reais de variável natural x (ou mesmo real). Dizemos que $f(x)$ é $O(g(x))$ se e só se

$$\exists c \in \mathbb{R}^+ \ \exists k \in \mathbb{R}^+ \ \forall x \ (x > k \Rightarrow |f(x)| \leq c|g(x)|). \tag{1}$$

Se $f(x)$ é $O(g(x))$ dizemos ainda que $g(x)$ é o limite assintótico de $f(x)$.

Esta notação é usada para estimar o número de operações efetuadas pelo algoritmo à medida que o tamanho do *input* aumenta, estabelecendo um limite superior desse crescimento. Tem a vantagem de não necessitarmos de preocupar-nos com constantes multiplicativas ($f(x) = ax^2$), com contribuições relativamente pequenas ($f(x) = x^2 + a$) ou valores especiais (fora do padrão) que poderão ocorrer num número finito de casos. Na Figura 1 observamos a interpretação geométrica do conceito dado.

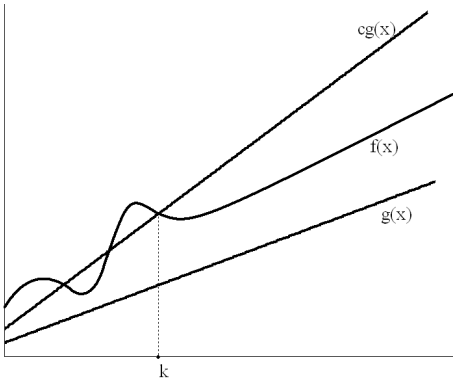


FIGURA 1. Interpretação geométrica da notação O.

EXEMPLO 3.1. Para $x \in \mathbb{R}$, a função $f(x) = x^2 + 7x$ é $O(x^2)$. Notemos que $7x < x^2$ para $x > 7$ e, portanto, $f(x) \leq 2x^2$ para $x > 7$. Logo, fazendo $c = 2$ e $k = 7$ em (1), temos o resultado pretendido.

EXEMPLO 3.2. Se tivermos dois algoritmos para resolver um dado problema, um deles efetuando $10n^2 + 3n + 1$ operações e o outro efetuando n^3 operações, onde n é o tamanho do *input*, a notação O ajuda-nos a entender que o primeiro algoritmo efetuará menos operações que o segundo para $n > 10$. De facto, a primeira expressão é $O(n^2)$ enquanto a segunda é $O(n^3)$ (ver Figura 2).

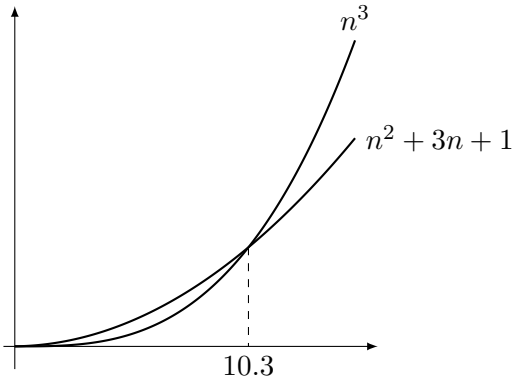


FIGURA 2. Comportamento assintótico de $10n^2 + 3n + 1$ e n^3.

De um modo informal, podemos dizer que o símbolo O seleciona o termo que mais contribui para o valor que uma expressão pode tomar e ignora os fatores constantes desse termo, de que é exemplo o teorema seguinte, onde se mostra que uma função polinomial de grau n é $O(x^n)$. As funções polinomiais da forma x^n são usualmente usadas para estimar o crescimento assintótico de uma função.

TEOREMA 3.2. *Seja* $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, *com* $a_i \in \mathbb{R}$, *para todo o* i. *Então* $f(x)$ *é* $O(x^n)$.

DEMONSTRAÇÃO. Usando a desigualdade triangular, temos

$$\begin{aligned} |f(x)| &\leq |a_n x^n| + |a_{n-1}x^{n-1}| + \cdots + |a_1 x| + |a_0| \\ &= x^n(|a_n| + |a_{n-1}/x| + \cdots + |a_1/x^{n-1}| + |a_0/x^n|) \\ &\leq x^n(|a_n| + |a_{n-1}| + \cdots + |a_1| + |a_0|) \quad \text{para todo o } x > 1. \end{aligned}$$

Logo, fazendo $c = |a_n| + |a_{n-1}| + \cdots + |a_1| + |a_0|$ e $k = 1$ em (1), obtemos o resultado pretendido. □

Notemos que, se $f(x)$ é $O(g(x))$ e $h(x)$ é uma função tal que $|h(x)| \geq |g(x)|$, para valores suficientemente grandes de x, então $f(x)$ é $O(h(x))$. No entanto, $g(x)$ deverá escolher-se tão pequeno quanto possível e é usualmente um elemento do conjunto de funções de referência seguinte:

$$\{1,\ \log x,\ x,\ x\log x,\ x^2,\ x^3, \ldots,\ 2^x,\ 3^x, \ldots, x!\}.$$

O número de passos efetuados por um algoritmo composto por subprocedimentos para resolver um problema, com um *input* de um dado tamanho, é dado pela soma de todos os passos dos subprocedimentos que ele contém. Por outro lado, o número de operações efetuadas por um ciclo com um dado número de operações internas é o produto do número de iterações por esse número de operações. Assim, é importante estimarmos o comportamento assintótico da soma e do produto de funções, que resultam dos teoremas que se seguem.

TEOREMA 3.3. *Sejam* f_1 *e* f_2 *funções tais que* $f_1(x)$ *é* $O(g_1(x))$ *e* $f_2(x)$ *é* $O(g_2(x))$. *Então* $f_1(x) + f_2(x)$ *é* $O(\max\{|g_1(x)|, |g_2(x)|\})$.

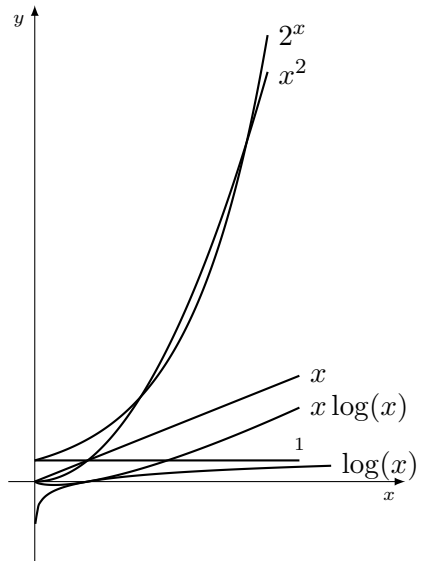


FIGURA 3. Gráfico de funções de referência usadas na notação O.

DEMONSTRAÇÃO. Por hipótese, existem c_1, c_2, k_1 e k_2 tais que $|f_1(x)| \leq c_1|g_1(x)|$, para todo o $x > k_1$, e $|f_2(x)| \leq c_2|g_2(x)|$, para todo o $x > k_2$. Novamente pela desigualdade triangular, temos

$$\begin{aligned} |f_1(x) + f_2(x)| &\leq |f_1(x)| + |f_2(x)| \\ &\leq c_1\,|g_1(x)| + c_2\,|g_2(x)| \quad \text{para todo o } x > \max\{k_1, k_2\} \\ &\leq c_1\,\max\{|g_1(x)|, |g_2(x)|\} + c_2\,\max\{|g_1(x)|, |g_2(x)|\} \\ &\leq (c_1 + c_2)\,\max\{|g_1(x)|, |g_2(x)|\}. \end{aligned}$$

Fazendo $c = c_1 + c_2$ e $k = \max\{k_1, k_2\}$ em (1), resulta que $f_1(x) + f_2(x)$ é $O(\max\{|g_1(x)|, |g_2(x)|\})$. □

TEOREMA 3.4. *Sejam f_1 e f_2 funções tais que $f_1(x)$ é $O(g_1(x))$ e $f_2(x)$ é $O(g_2(x))$. Então $f_1(x)f_2(x)$ é $O(g_1(x)g_2(x))$.*

DEMONSTRAÇÃO. Sejam c_1, c_2, k_1 e k_2 tais que $|f_1(x)| \leq c_1|g_1(x)|$, para todo o $x > k_1$, e $|f_2(x)| \leq c_2|g_2(x)|$, para todo o $x > k_2$. Então, para $x > \max\{k_1, k_2\}$, temos $|f_1(x)f_2(x)| \leq c_1c_2\,|g_1(x)|\,|g_2(x)| = c_1c_2\,|g_1(x)g_2(x)|$. Basta então considerar $c = c_1c_2$ e $k = \max\{k_1, k_2\}$ em (1). □

EXEMPLO 3.3. Consideremos a função $f(x) = (x+1)\log(x^2+1) + 3x^2$. Pelo Teorema 3.2, sabemos que $x + 1$ é $O(x)$. Por outro lado, $x^2 + 1 \leq 2x^2$, para $x \geq 1$. Logo

$$\log(x^2 + 1) \leq \log(2x^2) = \log(2) + 2\log(x) \leq 3\log(x),$$

para $x \geq 2$. Resulta que $\log(x^2+1)$ é $O(\log(x))$. Pelo Teorema 3.4, temos que $(x+1)\log(x^2+1)$ é $O(x\log(x))$. Como $3x^2$ é $O(x^2)$, pelo Teorema 3.3, resulta que $f(x)$ é $O(\max\{x\log(x), x^2\})$. Como $\log(x) \leq x$, para $x > 1$, resulta que $f(x)$ é $O(x^2)$.

4. Complexidade de algoritmos

A complexidade de um algoritmo pode ser medida por dois parâmetros diferentes: o espaço de memória usado ou o tempo necessário para executar o algoritmo. Dados dois ou mais algoritmos que resolvem o mesmo problema, é mais sensato escolher aquele que obtém solução no menor tempo possível e que utiliza a menor quantidade de espaço. Nesta disciplina, estudaremos a complexidade usando unicamente a medida de tempo, designada de *complexidade temporal*.

Ao compararmos dois algoritmos que resolvem o mesmo problema questionamo-nos então qual deles é mais rápido. Naturalmente, a velocidade pode depender da máquina usada, da linguagem em que está escrito ou do número de operações requeridas para realizar o procedimento. Vamos abstrair-nos dos dois primeiros detalhes e analisar o tempo de execução do algoritmo em termos do número de operações requeridas para a sua implementação, quando o *input* tem um dado tamanho.

A complexidade de um algoritmo é então expressa em função do número de *operações elementares (primitivas)* requeridas para a execução do algoritmo num dado tamanho do *input*. Cada operação primitiva tem um custo de execução, custo este que depende também da máquina usada (*hardware*) e da linguagem de programação em que o algoritmo está escrito (*software*). Assim, prever com rigor o tempo de execução de um algoritmo é praticamente impossível. Fazemos então uma análise assintótica da complexidade, conceito introduzido na secção anterior. Para isso, não necessitamos de fazer distinção dos tempos de execução de cada operação primitiva, pelo que consideramos que toda a operação primitiva tem o mesmo tempo de execução, que é uma dada constante.

São *operações primitivas* (ou *elementares*) as operações seguintes:

- Leitura de um elemento de uma lista;
- Atribuição de valores a variáveis ou incrementos;
- Comparação de dois valores;
- Operações aritméticas;
- Chamada de um método;
- Retorno de um método.

Dado um algoritmo, identificamos as operações primitivas (embora não seja necessária a distinção de operações) e calculamos o número de vezes que elas são executadas. Se a este número multiplicarmos o tempo de execução de cada operação, que supomos constante, teremos uma estimativa do tempo necessário para executar o procedimento. Vejamos o exemplo seguinte.

Exemplo 3.4. Consideremos os três algoritmos que se seguem, que calculam a soma dos primeiros n números inteiros positivos. Para cada um, calculamos o número de vezes que cada operação primitiva é executada.

Algoritmo A

```
procedure sumA(n:integer)
   sum:= 0
   for i:= 1 to n
      sum:= sum + i
   return sum
```

Algoritmo B

```
procedure sumB(n:integer)
   sum:= 0
   for i:= 1 to n
      for j:= 1 to i
         sum:= sum + 1
   return sum
```

Algoritmo C

```
procedure sumC(n:integer)
   sum:= n * (n+1)/2
   return sum
```

Apresentamos as operações efetuadas por linha na tabela seguinte, onde usamos as letras seguintes para designar as operações primitivas: atribuição (A), incremento (I), comparação (C), operação aritmética (Op) e retorno (R).

linhas	Algoritmo A	Algoritmo B	Algoritmo C
1^{a}	1A	1A	3Op + 1A
2^{a}	$(n+1)$AouI + $(n+1)$C	$(n+1)$AouI + $(n+1)$C	1R
3^{a}	nOp + nA	$\left(\frac{n(n+1)}{2}+n\right)$ (AouI + C)	-
4^{a}	1R	$\frac{n(n+1)}{2}$ (Op + A)	-
5^{a}	-	1R	-
total	$4n+4$	$\left(2n(n+1)+4n+4\right)$	5
estimativa O	$O(n)$	$O(n^2)$	$O(1)$

Calculemos com cuidado o número de operações primitivas do Algoritmo A. O cálculo das operações do Algoritmo B é deixado ao cuidado do leitor e o do Algoritmo C é facilmente verificado.

Na primeira linha é feita 1 atribuição de valor à variável *sum*. Em seguida, entramos num ciclo for. Nesta linha, é feita uma primeira atribuição à variável i que será incrementada até $n+1$, num total de $n+1$ atribuições ou incrementos. Em cada uma destas atribuições, o valor de i é comparado com o valor n de modo a testar se já foi ultrapassado e o ciclo termina. São então feitas $n+1$ comparações. Na terceira linha, que é percorrida n vezes, pois está dentro do ciclo for, é feita 1 operação de adição e 1 atribuição, num total de $2n$ operações. Finalmente, é feito o retorno. Temos, portanto, um total de $4n+4$ operações primitivas.

Notemos que, dependendo da literatura, esta discriminação das operações poderá ser diferente. Há até autores que não contabilizam as operações realizadas com a variável iterativa para as entradas do cliclo, contando unicamente as operações efetuadas já dentro desse ciclo. Usando a notação O, introduzida na secção anterior, concluiremos que esta diferença de cálculo não afeta a determinação da complexidade do algoritmo.

Aliamos então ao cálculo do número de operações primitivas a notação O e estimamos o tempo de execução de um algoritmo em função do tamanho do *input*. Com esta notação ignoramos as constantes multiplicativas e consideramos apenas o termo que mais contribui para o comportamento assintótico da função, como vimos na secção anterior. É fácil observar que o número de operações efetuadas em cada algoritmo do exemplo anterior é, respetivamente, $O(n)$, $O(n^2)$ e $O(1)$.

Complexidade temporal de alguns algoritmos. Fazemos o estudo da complexidade temporal recorrendo à análise de alguns algoritmos mais conhecidos.

EXEMPLO 3.5. **Procurar o elemento máximo de um conjunto**

Suponhamos que temos uma lista $a_1, a_2, \ldots, a_n$ com n números, distintos ou não, e consideremos o algoritmo que se segue, que procura o elemento máximo dessa lista:

```
procedure max(a[1],a[2],... ,a[n]:integers)
    max:= a[1]
    for i:= 2 to n
       if max < a[i] then max:= a[i]
    return max
```

Inicialmente, o valor máximo temporário é atribuído ao primeiro inteiro da lista e, portanto, é efetuada 1 operação de leitura desse elemento da lista e 1 atribuição. Após isso, o algoritmo efetua n atribuições ou incrementos à variável iterativa i e n comparações, até que ela toma o valor $n+1$ e sai do ciclo for. Em cada uma destas atribuições (excetuando a última), há a entrada no ciclo e é feita a leitura do i-ésimo elemento da lista. Este elemento é comparado com o valor máximo temporário. São então efetuadas $n-1$ leituras e $n-1$ comparações. Sempre que o valor máximo temporário é substituído por outro é efetuada uma operação de substituição (atribuição), sendo, no máximo, $n-1$. Resulta que o número de operações elementares do algoritmo é, no máximo, $5n$, considerando ainda a operação de retorno. Usando a notação O, concluimos que o algoritmo apresentado para determinar o máximo de uma lista de n elementos tem complexidade temporal $O(n)$.

Repare-se que, no exemplo anterior, não é possível determinar se a substituição do máximo temporário por outro valor é efetuada sempre que percorremos uma iteração do ciclo for, pois depende do *input*. Assim, é feita a *análise do pior caso*, ou seja, calculamos o número máximo de operações que garantem a obtenção da solução pelo algoritmo, para cada *input* de um tamanho específico. Chamamos esta análise de *complexidade do pior caso*, i.e., calculamos qual é o maior tempo gasto pelo algoritmo para produzir uma solução para um *input* de tamanho n.

Outro tipo importante de análise de complexidade de algoritmos é a *complexidade do caso médio*, i.e., o tempo médio gasto pelo algoritmo para produzir a solução para um *input* de tamanho n. Embora a complexidade do caso médio seja, na maior parte das vezes, a que melhor define o comportamento do algoritmo, esta é mais difícil de analisar pois é necessário entrar em consideração com a distribuição de probabilidade dos *inputs*.

Nesta disciplina restringimos o estudo da complexidade algorítmica à análise da complexidade do pior caso, que usa a notação O introduzida na secção anterior.

Os algoritmos de procura são bastante úteis. Temos, como exemplos, um programa que procure uma dada palavra num dicionário, um programa que procure se um cidadão com um dado número de cartão do cidadão está recenseado numa certa freguesia, etc. Vejamos dois exemplos de algoritmos de procura.

EXEMPLO 3.6. **Algoritmo de Procura Linear (Linear Search)**

Dado um elemento x, pretendemos verificar se x se encontra numa lista de n elementos, distintos ou não.

```
procedure linear_search(x:integer,a[1],a[2],...,a[n]:integers)
                        i:= 1
                        while (i ≤ n and x ≠ a[i])
                           i:= i+1
                        if i ≤ n then location:= i
                        else location:= 0
                        return location
```

Analisemos a complexidade deste algoritmo. No primeiro passo é feita 1 atribuição inicial.

No caso de a_i ser o elemento procurado, são feitas 2 comparações e 1 leitura em cada passo do ciclo while, num total de $2i$ comparações e i leituras. Neste caso, entramos no ciclo $i-1$ vezes, pelo que são ainda feitos $i-1$ incrementos à variável i. Finalmente, são efetuadas 1 comparação e 1 atribuição, quando determinamos a localização do elemento usando o comando if, e 1 retorno. Há então um total de $4i+3$ operações.

No caso do elemento não estar na lista, serão efetuadas $2n+1$ comparações, n leituras e n atribuições no ciclo while. Seguidamente, são efetuadas 1 comparação e 1 atribuição para concluir a localização e 1 retorno, num total de $4n+5$ operações (contabilizando também a atribuição inicial).

Logo o algoritmo efetua, no máximo, $4n+5$ operações elementares. Resulta que este algoritmo de procura linear tem complexidade temporal $O(n)$.

EXEMPLO 3.7. **Algoritmo de Procura Binária (Binary Search)**

Podemos usar este tipo de algoritmo de procura sempre que os elementos estão listados por ordem crescente (ou, no caso dual, por ordem decrescente). Começamos por comparar o elemento que está no meio da lista (se o "meio da lista" não for inteiro, escolhemos o lugar imediatamente abaixo). A lista é dividida em duas sublistas e é escolhida aquela que poderá ter o elemento de procura. Procedemos sucessivamente do mesmo modo até obtermos uma sublista com 1 elemento. Comparamos este elemento com o pretendido e obtemos uma resposta.

```
procedure binary_search(x:integer,a[1],a[2],...,a[n]:increasing integers)
                        i:= 1   (corresponde ao início da sublista)
                        j:= n   (corresponde ao final da sublista)
                        while i < j
                           m:= floor ((i+j)/2)
                           if x > a[m] then i:= m+1
                           else j:= m
                        if x = a[i] then location:= i
                        else location:= 0
                        return location
```

EXERCÍCIO 3.8. Consideremos a lista de números primos menores que 100,

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.$$

Apresente os passos efetuados pelo Algoritmo de Procura Binária quando pretendemos verificar que:

(i) 27 não é número primo;
(ii) 61 é número primo.

Vamos então calcular um majorante do número de operações efetuadas pelo algoritmo, de modo a determinarmos a complexidade do pior caso. Para simplificar o cálculo, assumimos que o tamanho da lista é um potência de 2, ou seja, $n = 2^k$. A primeira e a segunda linha só são executadas uma vez. Vejamos quantas vezes é executada a terceira linha. Inicialmente $i = 1$ e $j = 2^k$. Após a primeira iteração, teremos $i = 1$ e $j = 2^{k-1}$, ou $i = 2^{k-1}+1$ e $j = 2^k$. Em qualquer dos casos, sobram 2^{k-1} elementos para fazer a procura. Procedendo sucessivamente, após a segunda iteração teremos 2^{k-2} elementos até à última iteração onde teremos unicamente 1 elemento. Logo a linha 3 é executada $k+1$ vezes (a última vez para sair do ciclo), as linhas 4 e 5 são executadas k vezes e a linha 6 no máximo k vezes. Finalmente, as linhas 7 e 8 serão executas uma vez ou no máximo uma vez, respetivamente. Como em cada linha só são executadas operações elementares, obtemos a complexidade total de $O(1)+O(1)+O(k)+O(k)+O(k)+O(k)+O(1)+O(1) = O(k)$, pelo Teorema 3.3. Como $k = \log_2 n$, resulta que a complexidade do algoritmo é $O(\log_2 n)$, onde n é o comprimento da lista.

Os algoritmos de ordenação de elementos são outro tipo de algoritmos bastante usados. Por exemplo, dada a lista de todos os alunos inscritos em Matemática Discreta, podemos ordená-los por número de aluno ou por ordem alfabética. Têm sido dispendidos muitos esforços no aperfeiçoamento dos algoritmos de ordenação, de modo a tornar este problema o mais eficiente quanto possível, visto ser útil em muitos problemas importantes.

EXEMPLO 3.9. **Algoritmo de Ordenação por Flutuação (Bubble Sort)**

Trata-se de um dos algoritmos de ordenação mais simples, mas nem por isso mais eficiente. Ordena os elementos por ordem crescente simplesmente comparando valores adjacentes dois a dois e colocando-os por ordem crescente, no caso de não estarem. Se imaginarmos os elementos da lista colocados numa coluna, poderemos imaginar que os elementos mais leves flutuam até ao topo, como uma bolha de ar num tanque. E temos a justificação para o nome do algoritmo!

```
procedure bubble_sort(a[1],a[2],...,a[n]:real)
    for i:= 1 to n-1
       for j:= 1 to n-i
          if a[j] > a[j+1] then swap a[j] and a[j+1]
```

EXERCÍCIO 3.10. Apresente os passos efetuados pelo Algoritmo de Ordenação por Flutuação para ordenar a lista dos primeiros 5 números perfeitos[1] 496, 28, 8128, 6, 33550336.

[1]Um número perfeito é um número inteiro positivo que é igual à soma de todos os seus divisores próprios.

Procedemos então à análise da complexidade do pior caso. A primeira linha do algoritmo é efetuada $n-1$ vezes. Para cada iteração i do primeiro ciclo for, serão efetuadas $n-i$ iterações do segundo ciclo for e, em cada uma destas iterações, serão feitas $n-i$ comparações e, no máximo, $n-i$ trocas. Assim, o número total de comparações efetuadas (que é também o número total de iterações do segundo ciclo e o número máximo de trocas) é dado por

$$\sum_{i=1}^{n-1}(n-i) = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i = n(n-1) - \frac{1+(n-1)}{2}(n-1) = \frac{n(n-1)}{2}.$$

Notemos que a primeira e a segunda linha do algoritmo foram ainda efetuadas, respetivamente, 1 e n vezes, de cada vez que se concluiu que o ciclo terminou. Resulta que a complexidade do pior caso é dada por $O(n) + O(n^2) + O(n^2) + O(n^2) = O(n^2)$.

Terminamos esta subsecção com uma pequena nota. Por vezes, ao calcularmos o número de operações, além de se ignorarem as comparações efetuadas pela variável iterativa para concluir que se finalizou o ciclo, também se ignoram as leituras do *input* ou ainda as operações de atribuição de valor à variável, contando-se somente as comparações dos elementos e as operações aritméticas. Isto não afeta a análise de complexidade, nem deve criar confusão ao leitor. Neste texto optou-se por apresentar todas as contagens das operações efetuadas pelos algoritmos, exceto nas duas últimas análises, onde se contaram unicamente o número de vezes que uma linha é executa, determinando-se diretamente a complexidade da linha e fazendo o cálculo final da complexidade usando os teoremas do comportamento assintótico de funções apresentados na secção anterior.

Classes de complexidade. Terminamos esta secção apresentando uma tabela com as *classes de complexidade* mais comuns e, por isso, que apresentam uma terminologia própria. Apresentamos ainda uma tabela com exemplos de tempos de execução de algoritmos de uma dada complexidade em função do tamanho do *input*, n, do problema que resolvem. Finalmente, referimos alguns exemplos de algoritmos de algumas destas classes de complexidade.

Notação O	Terminologia
$O(1)$	Complexidade constante
$O(\log n)$	Complexidade logarítmica
$O(n)$	Complexidade linear
$O(n \log n)$	Complexidade $n \log n$
$O(n^b)$	Complexidade polinomial
$O(b^n)$ $(b > 1)$	Complexidade exponencial
$O(n!)$	Complexidade fatorial

FIGURA 4. Classes de complexidade dos algoritmos.

EXEMPLO 3.11. O Algoritmo C do Exemplo 3.4 tem complexidade constante, pois são efetuadas 5 operações, independentemente do tamanho do *input*.

tamanho *input*	$\log_2 n$	n	$n \log_2 n$	n^2	2^n
10	3s	10s	33s	100s	10^3s
10^2	7s	10^2s	664s	10^4s$\sim$ 3h	10^{30}s$\sim$ 10^{20}sec
10^3	10s	10^3s	9965s	10^6s$\sim$ 12dias	10^{300}s$\sim$ 10^{291}sec
10^4	13s	10^4s$\sim$ 3h	10^5s$\sim$ 1, 5dias	10^8s$\sim$ 3anos	10^{3000}s$\sim$???sec
10^5	17s	10^5s$\sim$ 1dia	10^6s$\sim$ 19dias	10^{10}s$\sim$ 3séculos	10^{3000}s$\sim$???sec

FIGURA 5. Tempos de execução de algoritmos de uma dada classe de complexidade.

EXEMPLO 3.12. Consideremos um algoritmo que encontra o maior dos valores das 100 primeiras entradas de uma lista com n elementos. Notemos que, qualquer que seja o tamanho $n \geq 100$ da lista, se aplicarmos o algoritmo do Exemplo 3.5 aos primeiros 100 elementos da lista, então serão sempre efetuadas 99 comparações na linha 3, independentemente do tamanho da lista, que é n. Logo o algoritmo terá complexidade constante, $O(1)$.

EXEMPLO 3.13. Os algoritmos que procuram o elemento máximo de um conjunto e de procura linear numa lista de números, apresentados, respetivamente, nos Exemplos 3.5 e 3.6, têm complexidade linear.

EXEMPLO 3.14. O algoritmo de procura binária do Exemplo 3.7 tem complexidade logarítmica, $O(\log n)$.

EXEMPLO 3.15. O algoritmo de ordenação por flutuação do Exemplo 3.9 tem complexidade quadrática (polinomial de grau 2), $O(n^2)$.

EXEMPLO 3.16. O algoritmo que determina se uma proposição composta por n proposições simples é uma tautologia analisando todas as combinações de valores lógicos que se podem atribuir às proposições simples (como quando construímos uma tabela de verdade) é um algoritmo com complexidade exponencial. De facto, notemos que há 2^n combinações diferentes de atribuição de valores lógicos às proposições simples. Assim, o algoritmo poderá ter de fazer 2^n operações (no caso de ser tautologia é mesmo o número de operações que efetua), em que cada operação é a verificação do valor lógico da proposição composta para cada atribuição das variáveis. Resulta que a complexidade do algoritmo é exponencial, $O(2^n)$.

EXEMPLO 3.17. Finalmente, como exemplo de um algoritmo de complexidade fatorial, fazemos referência ao Problema do Caixeiro Viajante, um dos problemas mais conhecidos da Teoria de Grafos: *Um caixeiro viajante quer visitar todas as n cidades exatamente uma vez e voltar à cidade de partida. Pretende fazê-lo percorrendo a distância menor possível.* Como resolvemos este problema? A maneira menos eficiente seria construirmos um algoritmo que calculasse todos os caminhos possíveis e a distância total deles. Mas isto dá $(n-1)!$ possibilidades. Assim, tal algoritmo terá complexidade fatorial, $O(n!)$.

5. Algoritmos iterativos *versus* algoritmos recursivos

Uma definição recursiva expressa o valor de um termo de uma sequência em função dos valores de termos inferiores dessa sequência.

Diz-se que um algoritmo é *recursivo* quando ele resolve um problema reduzindo-o a uma instância do mesmo problema com *input* inferior.

O uso de algoritmos recursivos é frequente pois simplifica a escrita e leitura de algoritmos que resolvem um certo problema, mas, como teremos oportunidade de verificar nesta secção, o esforço computacional exigido na sua execução poderá ser muito maior.

No que se segue, faremos um breve estudo comparativo da complexidade temporal entre algoritmos recursivos e os algoritmos iterativos que lhes são equivalentes. Para isso, recorreremos aos resultados apresentados no Apêndice B, onde é feita uma breve abordagem matemática a algumas relações de recorrência. Para facilidade de leitura, apresenta-se em seguida uma tabela que resume tais resultados.

Classe	Relação de recorrência	Solução geral
Homogéneas		
grau 1	$a_n = ca_{n-1}$	$a_n = \alpha c^n$
grau 2, raízes distintas	$a_n = c_1 a_{n-1} + c_2 a_{n-2}$	$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$
grau 2, raízes iguais	$a_n = c_1 a_{n-1} + c_2 a_{n-2}$	$a_n = \alpha_1 r^n + \alpha_2 n r^n$
Não-homogéneas		
grau 1	$a_n = ca_{n-1} + k$, $c \neq 1$	$a_n = \alpha c^n + \frac{k}{1-c}$
grau 1	$a_n = a_{n-1} + k$	$a_n = \alpha + nk$
grau 2, raízes distintas	$a_n = c_1 a_{n-1} + c_2 a_{n-2} + k$, $c_1 + c_2 \neq 1$	$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n - \frac{k}{c_1 + c_2 - 1}$

FIGURA 6. Soluções de algumas relações de recorrência lineares de coeficientes constantes.

Quando fazemos a análise da complexidade temporal de um algoritmo recursivo, a contagem do número de operações efetuadas para um *input* de tamanho n é dada naturalmente por uma relação de recorrência. Assim, será necessário determinar a solução fechada desta relação de recorrência para obtermos uma estimativa O da complexidade.

O primeiro exemplo a estudar refere-se ao cálculo do fatorial de um inteiro não negativo.

EXEMPLO 3.18. **Algoritmo Iterativo *versus* Algoritmo Recursivo para o Fatorial**

Para o cálculo do fatorial de um número inteiro não negativo, podemos usar o algoritmo iterativo ou o algoritmo recursivo apresentados abaixo. O segundo recorre à relação de recorrência, onde o cálculo de $n!$ se reduz ao produto de n pelo fatorial do inteiro anterior, $(n-1)!$. No primeiro, o procedimento inicia no valor 1 e vai multiplicando sucessivamente todos os inteiros até n. Analisemos a complexidade temporal de cada algoritmo.

```
procedure ItFatorial(n: non-negative integer)
    fatorial:= 1
    for i:= 1 to n
       fatorial:= fatorial * i
```

```
    return fatorial
```

Dado um *input* n, o algoritmo efetua uma atribuição inicial, seguindo-se $n+1$ atribuições e comparações da variável iterativa do ciclo **for**. De cada vez que entra no ciclo, realiza uma multiplicação seguida de uma atribuição. Em termos de multiplicações, o algoritmo efetua n multiplicações. Resulta que a sua complexidade temporal é $O(n)$.

```
procedure RecFatorial(n: non-negative integer)
    fatorial:= 1
    if n ≠0 then
       fatorial:= n * RecFatorial(n-1)
    return fatorial
```

Para qualquer *input* n é feita uma atribuição inicial e uma comparação com o inteiro0. No caso de $n \neq 0$ é ainda realizada uma multiplicação seguida da chamada de **RecFatorial**$(n-1)$ (e respetiva atribuição à variável *fatorial*).

Seja a_n o número de multiplicações realizadas por **RecFatorial**(n). Temos

$$a_0 = 0 \text{ e } a_n = 1 + a_{n-1}, \text{ para } n > 0.$$

Ou seja, definimos o número de multiplicações que o algoritmo efetua em função do *inputn* por uma relação de recorrência. Trata-se de uma relação de recorrência linear não-homogénea do 1º grau com coeficientes constantes. Assim, pela Proposição B.5, a solução da relação é

$$a_n = \alpha + nk,$$

com $k = 1$ e $a_0 = 0$. Logo $\alpha = 0$ e $a_n = n$, ou seja, são efetuadas n multiplicações para o cálculo de **RecFatorial**(n). Resulta que a complexidade temporal do algoritmo é $O(n)$.

Concluimos então que os algoritmos têm a mesma complexidade temporal. Recorde-se que tal não significa que demoram o mesmo tempo de computação, mas sim que, à medida que o *input* cresce, o tempo de computação dos algoritmos é o mesmo a menos de uma constante multiplicativa.

No exemplo que se segue, apresentam-se dois algortimos para o cálculo dos números de Fibonacci. A Sequência de Fibonacci tem como dois primeiros termos o valor 1 (ou 0 e 1, dependendo da convenção feita) e cada termo seguinte da sequência é obtido pela soma dos dois termos imediatamente anteriores. Tal denominação deveu-se a Fibonacci ter usado tal sequência para descrever o crescimento de uma população de coelhos.

EXEMPLO 3.19. **Algoritmo Iterativo *versus* Algoritmo Recursivo para os Números de Fibonacci**

Os Números de Fibonacci são os que constituem a sequência

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ 144,\ 233,\ 377,\ 610,\ 987,\ \ldots$$

Os dois algoritmos que se seguem calculam o n-ésimo termo desta sequência de um modo iterativo e recursivo, respetivamente. Analisemos cada um dos procedimentos e concluamos acerca da

sua complexidade temporal. Para isso, faremos unicamente a contagem do número de adições efetuadas, omitindo as atribuições e comparações da variável iterativa.

```
procedure ItFibonacci(n: positive integer)
    old_fib:= 0
    fibonacci:= 1
    for i:= 1 to (n-1)
       z:= old_fib + fibonacci
       old_fib:= fibonacci
       fibonacci:= z
    return fibonacci
```

Neste procedimento iterativo é realizada uma adição sempre e somente quando se entra no ciclo **for**, num total de $n-1$ adições. Notemos que as inicializações das variáveis e o retorno são $O(1)$, sendo as restantes operações primitivas $O(n)$, tal como as adições. Logo o algoritmo tem complexidade temporal linear, $O(n)$.

```
procedure RecFibonacci(n: positive integer)
    if n ≤2 then
       fibonacci:= 1
    else
       fibonacci:= RecFibonacci(n-1) + RecFibonacci(n-2)
    return fibonacci
```

O algoritmo recursivo calcula o n-ésimo termo da sequência chamando novamente o procedimento para calcular os termos de ordem $n-1$ e $n-2$. Seja a_n o número de adições efetuadas no cálculo deste n-ésimo termo. Este número não é mais do que a soma das adições efetuadas pelos procedimentos **RecFibonacci**$(n-1)$ e **RecFibonacci**$(n-2)$ mais uma adição extra, ou seja,

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

com $a_1 = 0$ e $a_2 = 0$, visto que, para $n \leq 2$ o algoritmo não efetua nenhuma adição. Ora, estamos perante uma relação de recorrência linear não-homogénea de coeficientes constantes de grau 2. Como a soma dos coeficientes de a_{n-1} e a_{n-2} é 2 ($\neq 1$), podemos usar o Teorema B.6.

As soluções da equação característica $x^2 - x - 1 = 0$ são $r_1 = \frac{1+\sqrt{5}}{2}$ e $r_2 = \frac{1-\sqrt{5}}{2}$, pelo que a relação de recorrência tem solução

$$a_n = \alpha_1 \Big(\frac{1+\sqrt{5}}{2}\Big)^n + \alpha_2 \Big(\frac{1-\sqrt{5}}{2}\Big)^n - 1,$$

onde a última parcela é dada por $-\frac{k}{c_1+c_2-1} = -\frac{1}{1+1-1} = -1$. Fazendo $n = 1$ e $n = 2$, respetivamente, obtemos

$$\begin{cases} 0 = \alpha_1 r_1 + \alpha_2 r_2 - 1 \\ 0 = \alpha_1 r_1^2 + \alpha_2 r_2^2 - 1 \end{cases}.$$

Multiplicando a primeira linha por r_1 e subtraindo à segunda linha obtemos

$$\begin{cases} 0 = \alpha_1 r_1 + \alpha_2 r_2 - 1 \\ 0 = \alpha_2(r_2^2 - r_1 r_2) + r_1 - 1 \end{cases} \Leftrightarrow \begin{cases} 0 = \alpha_1 r_1 + \alpha_2 r_2 - 1 \\ \alpha_2 = \frac{1-r_1}{r_2^2 - r_1 r_2} \end{cases} \Leftrightarrow \cdots \Leftrightarrow \begin{cases} \alpha_1 = \frac{1}{\sqrt{5}} \\ \alpha_2 = -\frac{1}{\sqrt{5}} \end{cases} .$$

Resulta que o número total de adições efetuadas pelo algoritmo é

$$a_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right] - 1$$

e, portanto, tem uma complexidade exponencial.

No exemplo anterior, pudemos observar que um algoritmo recursivo pode requerer um número de cálculos bastante superior ao do algoritmo iterativo. Mesmo assim, é frequente optar-se pelo modo recursivo, visto ter uma escrita mais elegante e uma leitura mais fácil.

6. Complexidade de problemas

Quando pretendemos resolver o problema da procura de um dado elemento numa lista há vários algoritmos que podemos implementar, tendo como exemplos os dos Exemplos 3.6 e 3.7, onde o segundo tem complexidade inferior ao primeiro. Uma questão que se coloca é se existe um algoritmo que resolva o mesmo problema com complexidade ainda menor.

Define-se a *complexidade temporal de um problema* como a complexidade temporal do algoritmo que resolve esse problema mais eficientemente, ou seja, com menor complexidade. Assim, sabendo a complexidade de um problema podemos concluir que: existe um algoritmo com essa complexidade que resolve o problema; não existe um algoritmo com complexidade menor que o resolva.

No entanto, determinar a complexidade de um problema pode ser uma tarefa muito difícil. Portanto, na maior parte dos casos, determina-se unicamente um majorante para essa complexidade (e, por vezes, um minorante). Notemos que para determinar um majorante para a complexidade do problema basta apresentar um algoritmo com essa complexidade que resolva o problema. Concluímos então, para o exemplo referido, que o problema de encontrar um dado elemento numa lista tem complexidade, no máximo, logarítmica.

Problemas tratáveis e problemas intratáveis. Todos os exemplos de problemas que apresentamos na Secção 4 são finitos, ou seja, dão uma resposta num número finito de passos. Acreditando na capacidade e velocidade dos computadores seria de esperar que todos estes algoritmos nos dessem uma resposta em tempo satisfatório.

Exemplo 3.20. Consideremos o Problema do Caixeiro Viajante e o algoritmo explicado no Exemplo 3.17. Ora, se o problema consistir de 5 cidades, bastará determinar as $4! = 24$ rotas e respetivas distâncias. No entanto, se o problema tiver 70 cidades, teremos de determinar todos os 69! caminhos. Como $69! = 1,71122 \times 10^{98}$, se assumirmos que um computador examina 10^{10} caminhos por segundo, então ele demoraria $1,71122 \times 10^{88}$ segundos a resolver o problema, o que são aproximadamente $5,42626 \times 10^{78}$ séculos para examinar todas as rotas de modo a escolhermos a que tem menos custo!

Exemplo 3.21. De modo análogo, se implementássemos o algoritmo do Exemplo 3.16 para uma proposição composta por 100 proposições simples, teríamos de analisar as 2^{100} combinações de atribuição de valores lógicos às proposições simples. Novamente, se assumirmos que um computador gasta 10^{-10} segundos para examinar o valor lógico da proposição composta para cada combinação, então demoraria cerca de 4×10^{12} anos para obtermos a resposta de que tal proposição é uma tautologia.

Notemos que as funções polinomiais crescem muito mais devagar que as funções exponenciais e fatoriais. Assim, consideramos que um algoritmo cuja complexidade temporal é $O(n^k)$, para um inteiro k, é *eficiente*, e um algoritmo cuja complexidade temporal cresce mais rapidamente que n^k, para todo o inteiro k, é *ineficiente*.

Temos, naturalmente, o conceito seguinte. Um problema é considerado *tratável* (ou *computacionalmente fácil*) se puder ser resolvido por um algoritmo eficiente e é considerado *intratável* (ou *computacionalmente difícil*) se não existir um algoritmo eficiente que o resolva. Mais uma vez, para mostrarmos que um problema é computacionalmente difícil teremos de mostar que não existe nenhum algoritmo eficiente que o resolva, o que poderá ser uma tarefa bastante árdua.

Problemas solúveis e problemas insolúveis: o problema da paragem. Dizemos que um problema é *solúvel* se existir um algoritmo que o resolva. Surge de imediato a seguinte questão: *Existem problemas insolúveis?* Alan Turing foi o primeiro a provar a existência de tais problemas ao mostrar, em 1936, que o *problema da paragem* é insolúvel.

Um *problema de decisão* é um problema computacional cuja resposta é *sim* ou *não* (ou, 1 ou 0). O objetivo é, para um dado problema e um algoritmo que o resolva, se dado um *input*, o algoritmo aceita esse *input*, dando a resposta *sim* ao problema, ou rejeita o *input*, dando a resposta *não* ao problema.

Exemplo 3.22. **O Problema da Paragem (The Halting Problem)**

O problema consiste em decidir se existe um algoritmo que decida, em tempo finito, se, dado um algoritmo A e um *input* P, o algoritmo A para quando corre com o *input* P.

Alan Turing, usando um argumento de redução ao absurdo, mostrou que tal algoritmo não existe e, portanto, o problema da paragem é insolúvel. Não faz parte do programa desta disciplina o estudo dessa demonstração. Mesmo assim, apresentamos uma demonstração simples do resultado, de modo a satisfazer um leitor mais curioso.

Demonstração. Suponhamos que existe um algoritmo que decide, dado um programa arbitrário P e um *input* I, se o programa P para quando corre com o *input* I, ou seja, retorna *True*, se P para com I, e *False* se entrar num ciclo infinito. Seja $H(P, I)$ tal algoritmo.

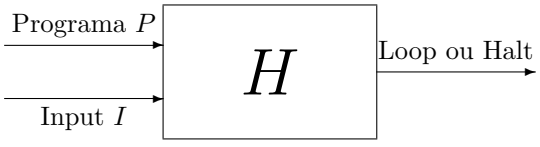


Figura 7. The Halting Problem (Alan Turing, 1936).

Consideremos o programa K que tem como *input* um programa X, lê o *output* de H com *input* (X, X) e faz o seguinte:

```
procedure K(X:program)
   if H(X,X) then
      loop forever
   else
      halt
```

Vejamos o que acontece quando o algoritmo K corre com o *input* K:

Case 1: Se H der como resposta que K para com o *input* K, então K entra em *loop*, o que é uma contradição.

Case 2: Se H der como resposta que K não para com o *input* K, então K para, o que é, mais uma vez, uma contradição.

Em ambos os casos, H dá a resposta errada sobre a paragem do programa K. Logo o programa H não funciona em todos os casos. Conclui-se que é impossível construir um tal algoritmo! □

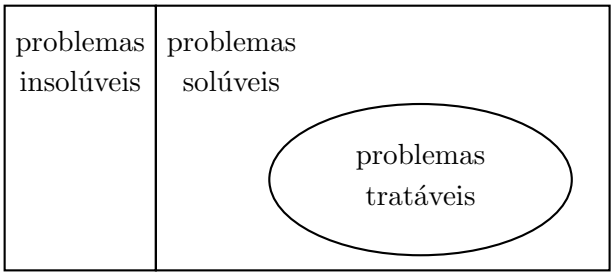


FIGURA 8. Diagrama do tipo de problemas.

Classes de complexidade. O estudo da complexidade vai muito além da abordagem que aqui foi feita. Terminamos fazendo referência a três classes de complexidade de problemas: **P**, **NP** e **NP-completos**.

A *classe* **P** é a classe dos problemas tratáveis, ou seja, que podem ser resolvidos por um algoritmo com complexidade polinomial.

EXEMPLO 3.23. Todos os algoritmos que resolvem o problema do Caixeiro Viajante são, pelo menos, exponenciais. No entanto, não se conhece uma prova de que seja impossível obter um algoritmo polinomial que resolva este problema. Logo não se sabe se este problema pertence ou não à classe **P**.

A *classe* **NP**[2] é constituída pelos problemas que se acredita não serem resolvidos por nenhum algoritmo com complexidade polinomial, mas uma possível solução do problema pode ser testada em tempo polinomial.

[2]O termo **NP** vem de Não-determinístico Polinomial. Esta classe consiste daqueles problemas que podem ser resolvidos polinomialmente por uma máquina não-determinística!

Exemplo 3.24. A fatorização de inteiros é um problema **NP**: verificar se um número é produto de dois primos dados é um problema extremamente simples, mas, dado um número inteiro, descobrir a sua fatorização em primos poderá ser bastante intrincado. É na dificuldade deste problema que se baseiam vários sistemas criptográficos (segurança de computadores, sistemas bancários, etc.).

A *classe dos problemas* **NP-completos** é uma subclasse de **NP** constituída pelos problemas com a propriedade seguinte: se um problema é **NP-completo**, então qualquer problema **NP** pode ser reduzido a ele em tempo polinomial.

Exemplo 3.25. O Problema de Satisfazibilidade Booleana (SAT) foi o primeiro problema identificado como pertencente à classe de complexidade NP-completo. Consiste em determinar se uma dada fórmula, usando somente os conectivos $\sim$, $\wedge$ e $\vee$, tem alguma atribuição de valores lógicos das variáveis que a torne verdadeira. Notemos que facilmente se verifica se uma atribuição de valores lógicos das variáveis torna uma fórmula uma proposição verdadeira, mas ainda não se conhece um algoritmo polinomial que, dada uma qualquer fórmula nas condições indicadas, encontre uma atribuição de valores que a torne verdadeira.

Note-se que, ao definirmos a classe **NP**, usou-se a expressão *acreditar não serem resolvidos por nenhum algoritmo com complexidade polinomial*. De facto, determinar se um problema cuja solução pode ser testada em tempo polinomial é também resolúvel em tempo polinomial é um problema da Matemática e da Ciência da Computação atualmente em aberto, o chamado *Problema* **P** *versus* **NP**. Considerado um dos grandes problemas por resolver, o Clay Mathematics Institute oferece o prémio de 1 milhão de dólares a quem mostrar formalmente que **P=NP** ou que **P≠NP**.

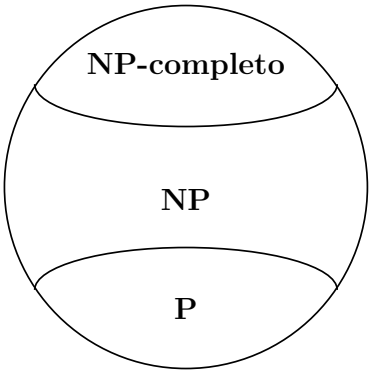


Figura 9. Diagrama das classes de complexidade **P**, **NP** e **NP-completo**.

Exercícios propostos

1. Usando a definição de O, verifique se cada uma das funções que se seguem é $O(x)$:
   a) $f(x) = 8$.
   b) $f(x) = 2x + 9$.
   c) $f(x) = x^2$.
   d) $f(x) = 3 \log x$.
   e) $f(x) = x^2 + 2x + 3$.

2. Usando a definição de O, verifique se cada uma das funções que se seguem é $O(x^2)$:
   a) $f(x) = 17x + 11$.
   b) $f(x) = x^2 + 100$.
   c) $f(x) = x \log x$.
   d) $f(x) = \frac{x^4}{2}$.
   e) $f(x) = 2^x$.

3. Usando os teoremas estudados, mostre que:
   a) $x^4 + 9x^3 + 4x + 7$ é $O(x^4)$.
   b) $2^x + 17$ é $O(3^x)$.
   c) $\frac{x^2+1}{x+1}$ é $O(x)$.
   d) $\frac{x^3+2x}{2x+1}$ é $O(x^2)$.

4. Determine o menor inteiro n tal que $f(x)$ é $O(x^n)$ para cada uma das funções seguintes:
   a) $f(x) = 2x^3 + x^2 \log x$.
   b) $f(x) = \frac{x^4+x^2+1}{x^3+1}$.
   c) $f(x) = \frac{x^3+2 \log x}{x^3+1}$.
   d) $f(x) = 3x^3 + (\log x)^4$.

5. Mostre, usando a definição de O, que:
   a) x^3 é $O(x^4)$ mas x^4 não é $O(x^3)$.
   b) $3x^4 + 1$ é $O(\frac{x^4}{2})$ e $\frac{x^4}{2}$ é $O(3x^4 + 1)$.
   c) $x \log x$ é $O(x^2)$ mas x^2 não é $O(x \log x)$.
   d) 2^n é $O(3^n)$ mas 3^n não é $O(2^n)$.
   e) $\log_2 n$ é $O(\log_3 n)$ e $\log_3 n$ é $O(\log_2 n)$.

6. Determine os menores inteiros a e b de modo a que f seja $O(n^a (\log n)^b)$:
   a) $f(n) = (n^2 + 8)(n + 1)$.
   b) $f(n) = (n \log n + n^2)(n^3 + 2)$.
   c) $f(n) = (n! + 2^n)(n^3 + \log(n^2 + 1))$.

d) $f(n) = n\log(n^2+1) + n^2\log n$.

e) $f(n) = (n\log n + 1)^2 + (\log n + 1)(n^2+1)$.

7. Escreva um algoritmo que encontre o menor número natural numa lista de n números naturais. Quantas comparações são usadas?

8. Indique uma estimativa O para o número de comparações usadas pelo algoritmo que determina o número de 1 numa string de bits, examinando cada bit da string para determinar se é um bit 1.

9. O Algoritmo de Procura Binária (Binary Search) é usado para procurar um elemento numa lista ordenada por ordem crescente (ou decrescente):

```
procedure binary_search(x:integer,a[1],a[2],...,a[n]:increasing integers)
                        i:= 1
                        j:= n
                        while i < j
                           m:= floor ((i+j)/2)
                           if x > a[m] then i:= m+1
                           else j:= m
                        if x = a[i] then location:= i
                        else location:= 0
                        return location
```

a) Consideremos a lista de números primos menores que 100,

$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.$

Apresente os passos efetuados pelo algoritmo quando pretendemos verificar que:

(i) 27 não é número primo;

(ii) 61 é número primo.

b) Supondo que o tamanho da lista é $n = 2^k$, calcule quantas vezes é executada cada linha do algoritmo.

c) Mostre que a complexidade deste algoritmo é $O(\log_2 n)$.

10. Suponha que se sabe que um elemento está entre os primeiros quatro elementos de uma lista de 32 elementos. Que algoritmo localiza este elemento mais rapidamente: pesquisa linear ou pesquisa binária?

11. O Algoritmo de Ordenação por Flutuação (Bubble Sort) ordena por ordem crescente os elementos de uma lista comparando dois valores adjacentes e colocando-os por ordem crescente (Se supusermos que os elementos da lista estão em coluna, podemos imaginar que os elementos mais leves flutuam até ao topo, como uma bolha de ar num tanque).

```
procedure bubble_sort(a[1],a[2],...,a[n]:real)
    for i:= 1 to n-1
       for j:= 1 to n-i
          if a[j] > a[j+1] then swap a[j] and a[j+1]
```

a) Apresente os passos efetuados pelo algoritmo para ordenar a lista dos primeiros 5 números perfeitos[3] 496, 28, 8128, 6, 33550336.

b) Calcule quantas vezes é executada cada linha do algoritmo.

c) Prove que a complexidade do pior caso é $O(n^2)$.

12. Escreva um algoritmo que coloque os primeiros quatro termos de uma lista de comprimento arbitrário por ordem crescente. Calcule a complexidade temporal considerando unicamente o número de comparações efetuadas.

13. Considere o algoritmo seguinte:

```
procedure p(n:integer)
   total := 0
   for i := 1 to n
      for j := 1 to n
         total := total + i * j
   return total
```

a) Determine uma fórmula para o número de multiplicações efetuadas pelo algoritmo em termos do valor do *input* n.

b) Para cada valor n do *input*, qual o cálculo que o algoritmo efetua?

14. O algoritmo que se segue pode ser usado para avaliar o polinómio $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, de grau n, no ponto $x = c$.

```
procedure poly1(c:real,a[0 to n]:real)
   power := 1
   y := a[0]
   for i := 1 to n
      power := power * c
      y := y + a[i] * power
```

a) Avalie $5x^2 + x + 3$ em $x = 2$ percorrendo cada passo do algoritmo.

b) Escreva uma fórmula para o número de operações de multiplicação necessárias para avaliar um polinómio de grau n.

c) Obtenha uma fórmula para o número de operações de adição necessárias para avaliar um polinómio de grau n (excluindo as adições para incrementar i).

[3]Um número perfeito é um número inteiro positivo que é igual à soma de todos os seus divisores próprios.

d) Apresente uma estimativa O para o tempo de execução deste algoritmo em função do grau do polinómio.

15. O algoritmo seguinte representa um modo alternativo ao do exercício anterior para avaliar o polinómio $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, de grau n, no ponto $x = c$, conhecido por Método de Horner.

```
procedure poly2(c:real,a[0 to n]:real)
   y := a[n]
   for i := 1 to n
      y := y * c + a[n - i]
```

a) Avalie $5x^2 + x + 3$ em $x = 2$ percorrendo cada passo do algoritmo.

b) Escreva uma fórmula para o número de operações de multiplicação necessárias para avaliar um polinómio de grau n.

c) Obtenha uma fórmula para o número de operações de adição necessárias para avaliar um polinómio de grau n (excluindo as adições para incrementar i).

d) Apresente uma estimativa O para o tempo de execução deste algoritmos em função do grau do polinómio.

e) Qual dos dois algoritmos tem um tempo de execução inferior?

16. Considere o algoritmo que se segue, que calcula o produto de duas matrizes $a[\,]$ e $b[\,]$, guardando o resultado na matriz $c[\,]$.

```
procedure matrixmult(a[1 to n,1 to n],b[1 to n,1 to n]:real)
   for i := 1 to n
      for j := 1 to n
         c[i,j] := 0
         for k := 1 to n
            c[i,j] := c[i,j] + a[i,k] * b[k,j]
```

a) Calcule quantas vezes são executadas as linhas 3 e 5 do algoritmo.

b) Obtenha uma estimativa O para o tempo de execução do algoritmo.

17. Quanto tempo é necessário para que um algoritmo execute 2^{50} operações com bits, se cada operação com bits demora:

a) 10^{-6} segundos.

b) 10^{-9} segundos.

c) 10^{-12} segundos.

18. Considere os dois algoritmos recursivos que se seguem.

```
procedure linrec(n:integer)
   if n = 0 then
```

```
        result := 1
    else
        result := linrec(n-1) * linrec(n-1) + 5

procedure linrec2(n:integer)
    if n = 0 then
        result := 1
    else
        temp := linrec2(n-1)
        result := temp * temp + 5
```

a) Escreva uma relação de recorrência para o número de multiplicações efetuadas pelo algoritmo **linrec**. Resolva-a.

b) Escreva uma relação de recorrência para o número de multiplicações efetuadas pelo algoritmo **linrec2**. Resolva-a.

c) Compare os resultados anteriores.

19. Considere o algoritmo recursivo que se segue. Escreva, e resolva, uma relação de recorrência para

a) o número de adições efetuadas pelo algoritmo.

b) o número de multiplicações efetuadas pelo algoritmo.

c) o número de comparações efetuadas pelo algoritmo.

d) Obtenha uma estimativa O para o tempo de execução do algoritmo em função do inteiro n.

```
procedure linrec3(n: non-negative integer, a[0 to 1]: integer)
    if n = 0 then
        result := 3 + a[n]
    if n = 1 then
        result := 5 + a[n]
    else
        result := 3 * linrec3(n-1) * linrec3(n-2)
```

20. Considere a sequência

$$a_0 = 1,\ a_1 = 2 \text{ e } a_n = a_{n-1} \cdot a_{n-2},\ \text{para } n > 1.$$

a) Escreva um algoritmo recursivo que determine o n-ésimo termo da sequência, para $n \geq 0$.

b) Escreva um algoritmo iterativo que determine o n-ésimo termo da sequência, para $n \geq 0$.

c) Qual dos algoritmos é mais eficiente? Justifique.

Soluções dos exercícios propostos

1. a) Por exemplo, $c = 1$ e $k = 8$.
   b) Por exemplo, $c = 3$ e $k = 9$.
   c) Não é $O(x)$.
   d) Por exemplo, $c = 3$ e $k = 1$.
   e) Não é $O(x)$.

2. a) Por exemplo, $c = 2$ e $k = 17$.
   b) Por exemplo, $c = 2$ e $k = 32$.
   c) Por exemplo, $c = 1$ e $k = 1$.
   d) Não é $O(x^2)$.
   e) Não é $O(x^2)$.

4. a) $n = 3$
   b) $n = 1$
   c) $n = 0$
   d) $n = 3$

6. a) $a = 3$ e $b = 0$.
   b) $a = 5$ e $b = 0$.
   c) Não existem, pois $f(n)$ é $O(n!)$.
   d) $a = 2$ e $b = 1$.
   e) $a = 2$ e $b = 2$.

7. $2n - 1$, no caso de ser o Algoritmo de Procura Linear.

8. $O(n)$

10. Pesquisa Linear

12. $O(1)$

13. a) n^2
    b) $\frac{n^2(n+1)^2}{4}$

14. b) $2n$
    c) n
    d) $O(n)$

15. b) n
    c) n
    d) $O(n)$
    e) Algoritmo de 15.

16. a) Linha 3: n^2 vezes; linha 5: n^3 vezes.
    b) $O(n^3)$

17. a) Aproximadamente 36 anos.
    b) Aproximadamente 13 dias.
    c) Aproximadamente 19 minutos.

18. a) $a_n = 1 + 2a_{n-1}$, para $n > 0$, $a_0 = 0$; solução: $a_n = 2^n - 1$.
    b) $a_n = 1 + a_{n-1}$, para $n > 0$, $a_0 = 0$; solução: $a_n = n$.
    c) Complexidade temporal: $O(2^n)$ *versus* $O(n)$.

19. a) $a_n = a_{n-1} + a_{n-2}$, para $n > 1$, $a_0 = 1$, $a_1 = 1$;
    solução: $a_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$.

    b) $b_n = 2 + b_{n-1} + b_{n-2}$, para $n > 1$, $b_0 = 0$, $b_1 = 0$;
    solução: $a_n = \left(1 - \frac{1}{\sqrt{5}}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n + \left(1 + \frac{1}{\sqrt{5}}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n - 2$.

    c) $c_n = 2 + c_{n-1} + c_{n-2}$, para $n > 1$, $c_0 = 1$, $c_1 = 2$;
    solução: $a_n = \left[(1 - 3\sqrt{5})\left(\frac{1-\sqrt{5}}{2}\right)^n + (5 + 3\sqrt{5})\left(\frac{1+\sqrt{5}}{2}\right)^n\right] - 2$.

    d) $O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$.

20. a)
```
procedure rec_a(n: non-negative integer)
   if n = 0 then
      result := 1
   else if n = 1 then
      result := 2
   else
      result := rec_a(n-1) * rec_a(n-2)
```

    b)
```
procedure it_a(n: non-negative integer)
   if n = 0 then
      result := 1
   else
      prev := 1
      result := 2
      for i := 1 to n-1
         temp := result * prev
         prev := result
         result := temp
```

    c) Complexidade exponencial (a relação de recorrência é a mesma do Algoritmo Recursivo para os Números de Fibonacci) *versus* complexidade linear.

CAPíTULO 4

Grafos

Um aluno meu [F. Guthrie] pediu-me, hoje, para dar-lhe uma razão para um facto que eu não sabia que era um facto - e ainda não sei. Ele diz que, se uma figura for dividida de qualquer maneira e as divisões forem coloridas de modo que divisões com parte da linha de fronteira comuns estejam coloridas de cor diferente, então quatro cores poderão ser necessárias, mas não mais. O que se segue é o caso em que quatro cores são necessárias. Ponho em dúvida se existe um caso em que cinco ou mais sejam necessárias... Se me responder com alguns casos muito simples que façam de mim um animal estúpido, eu farei o que a Esfinge[1] *fez...*

Carta de De Morgan a Hamilton, 23 de outubro de 1852.

1. Conceitos básicos

Os grafos são estruturas discretas que consistem num conjunto de pontos e num conjunto de linhas que unem alguns pares de pontos. São ferramentas matemáticas de extrema importância pois facilitam a resolução de problemas em várias áreas da Engenharia, da Indústria, da Economia, da Ecologia, da Genética, do Planeamento, etc.

Este conceito já foi introduzido na Secção 1.4, onde pudemos observar que a representação de uma relação binária por um grafo torna mais evidente as propriedades da relação. É pela facilidade e pela transparência na representação por grafos, bem como por resultados surpreendentes atingidos nesta teoria, que se modelam inúmeros problemas em diferentes áreas do saber e da indústria. Relembramos então este conceito e introduzimos mais algumas definições e notação básicas da Teoria de Grafos.

Grafo não orientado e grafo orientado. Um *grafo não orientado* é um par $G = (V, E)$, onde V é um conjunto não vazio, designado por conjunto dos *vértices*, e E é um subconjunto de V^2, onde não é considerada nenhuma ordem nos pares de V^2, designado por conjunto das *arestas*.

Um *grafo orientado* (ou *digrafo*) é um par $\vec{G} = (V, E)$, em que E é um subconjunto de V^2, onde os pares são ordenados, designado por conjunto das *arestas orientadas*. Se $\vec{e} = (u, v)$ é uma aresta de $\vec{G}$, dizemos que u é a *cauda* e v é a *cabeça* da aresta $\vec{e}$.

Dependendo do problema que pretendamos resolver, faremos uso de um grafo orientado ou de um grafo não orientado. Por esta razão, abordamos neste texto os dois conceitos, bem como as propriedades que lhes são comuns e as que não são. Por existirem conceitos e propriedades semelhantes nos dois tipos de grafos, optamos por tratá-los em paralelo.

[1] *O Enigma da Esfinge*, em http://pt.wikipedia.org/wiki/Esfinge, a 31 de janeiro de 2025.

Um grafo não orientado e um digrafo podem ser representados por uma figura que não é única, visto a disposição dos vértices e o traçado das arestas ser livre. No entanto, toda a representação gráfica determina um único grafo (ver Figura 1).

EXEMPLO 4.1. O grafo não orientado G e o grafo orientado $\vec{G}$, cujos conjuntos dos vértices e das arestas são, respetivamente,

$$\begin{aligned} V(G) = \;& V(\vec{G}) = \{1,2,3,4,5\} \\ E(G) = \;& \{(1,1),(1,4),e_1=(2,1),e_2=(2,1),(3,1),(3,2),(3,3),(4,3),(4,5),(5,3)\} \\ E(\vec{G}) = \;& \{(1,1),(1,4),\vec{e}_1=(2,1),\vec{e}_2=(2,1),(3,1),(3,2),(3,3),(4,3),(4,5),(5,3)\} \end{aligned}$$

podem ser representados como na Figura 1, não sendo uma representação única.

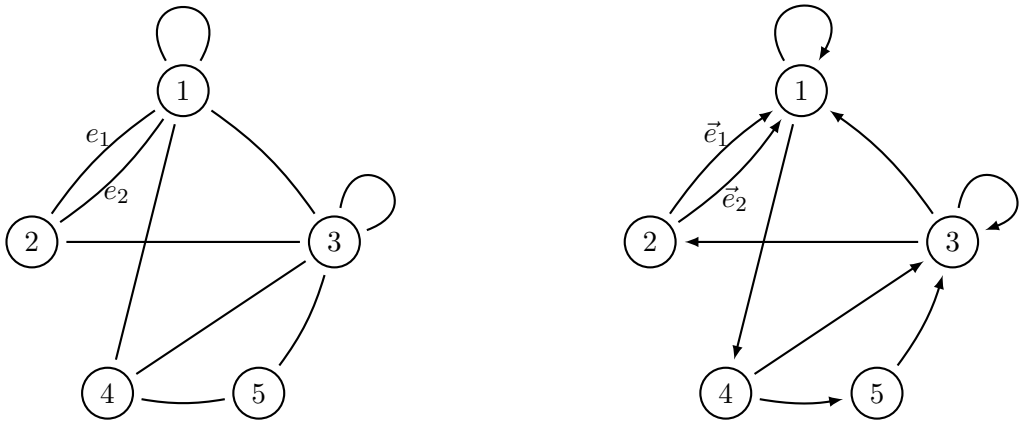


FIGURA 1. Representação gráfica de um grafo não orientado e de um digrafo.

Dizemos que uma aresta é *incidente* nos seus vértices extremos e que tais vértices são *adjacentes*. O conjunto de todos os vértices adjacentes a um dado vértice v designa-se de por *vizinhança* de v e denota-se por $\mathcal{N}_G(v)$. Na Figura 1, os vértices 1 e 3 são adjacentes e $\mathcal{N}_G(1) = \{1,2,3,4\}$.

Uma aresta com extremos no mesmo vértice denomina-se de *lacete*. Na Figura 1, as arestas $(1,1)$ e $(3,3)$ são lacetes.

Duas ou mais arestas unindo o mesmo par de vértices designam-se de *arestas múltiplas*. Notemos que, num digrafo, duas arestas múltiplas têm a mesma orientação de um vértice u para um vértice v. Na Figura 1, as arestas que unem o vértice 2 ao vértice 1 são arestas múltiplas. (De modo a distinguirmos duas arestas múltiplas, etiquetamos as arestas, fazendo, por exemplo, $e_1 = (2,1)$ e $e_2 = (2,1)$).

Um grafo sem arestas múltiplas nem lacetes denominamos de *grafo simples*.

A *ordem* de um grafo G é o número de vértices que o constitui, ou seja, $|V(G)|$. A *dimensão* de um grafo G é o número de arestas que ele possui, ou seja, $|E(G)|$. Os grafos da Figura 1 têm ordem 5 e dimensão 10.

Grau de um vértice. Seja G um grafo não orientado e $v \in V(G)$. Designamos por *grau* do vértice v, e denotamos por $grau_G(v)$ (ou $grau(v)$, se não houver confusão a que grafo nos referimos), o número de arestas incidentes no vértice v, onde os lacetes, caso existam, contam duas vezes para o grau do vértice.

Seja $\vec{G}$ um grafo orientado e $v \in V(\vec{G})$. Designamos por *grau de entrada* do vérticev, e denotamos por $grau^e(v)$, o número de arestas que entram no vértice v. Dualmente, designamos por *grau de saída* do vértice v, e denotamos por $grau^s(v)$, o número de arestas que saem do vértice v.

Exemplo 4.2. No primeiro grafo do Exemplo 4.1, tem-se:

$$grau(1) = 6, \quad grau(2) = 3, \qquad grau(3) = 6,$$
$$grau(4) = 3 \quad \text{e} \quad grau(5) = 2,$$

e, no segundo grafo do Exemplo 4.1, tem-se:

$$grau^e(1) = 4, \quad grau^s(1) = 2, \qquad grau^e(2) = 1, \quad grau^s(2) = 2,$$
$$grau^e(3) = 3, \quad grau^s(3) = 3, \qquad grau^e(4) = 1, \quad grau^s(4) = 2,$$
$$grau^e(5) = 1 \quad \text{e} \quad grau^s(5) = 1.$$

É fácil notar que, qualquer que seja o grafo, orientado ou não, cada aresta contribui duas vezes para a soma dos graus dos vértices, visto que é incidente nos dois vértices extremos, mesmo quando se trata de um lacete. Isto significa que a soma dos graus de todos os vértices é o dobro do número de arestas do grafo, como enunciamos no lema seguinte.

Lema 4.1 (Lema do Aperto de Mão). *Seja $G = (V, E)$ um grafo não orientado. Então*

$$2|E| = \sum_{v \in V} grau(v).$$

Exemplo 4.3. Um grafo com 7 vértices, cada um dos quais de grau 4 tem 14 arestas. Com efeito, a soma dos graus dos vértices é $7 \times 4 = 28$, pelo que $2|E| = 28$. Logo $|E| = 14$.

Lema 4.2. *Seja $\vec{G} = (V, E)$ um grafo orientado. Então*

$$|E| = \sum_{v \in V} grau^e(v) = \sum_{v \in V} grau^s(v).$$

Exemplo 4.4. No grafo orientado do Exemplo 4.1 temos:

$\sum_{v \in V} grau^e(v) = 4 + 1 + 3 + 1 + 1 = 10$ e $\sum_{v \in V} grau^s(v) = 2 + 2 + 3 + 2 + 1 = 10$

que é, efetivamente, o número de arestas do grafo.

Lema 4.3. *Um grafo não orientado tem um número par de vértices de grau ímpar.*

Demonstração. Sejam V_p e V_i o conjunto dos vértices do grafo $G = (V, E)$ de grau, respetivamente, par e ímpar. Então tem-se

$$2|E| = \sum_{v \in V} grau(v) = \sum_{v \in V_p} grau(v) + \sum_{v \in V_i} grau(v).$$

Como $grau(v)$ é par, para todo o $v \in V_p$, então a primeira parcela do último membro das igualdades é par. Como a soma das duas parcelas é par, resulta que a segunda parcela tem

também de ser par. Ora, os termos deste somatório são todos ímpares, logo o número de parcelas do somatório tem de ser par, ou seja, o número de vértices de grau ímpar é par. □

Matriz de adjacências. Definimos a *matriz de adjacências* de um grafo não orientado $G(V, E)$, e denotamos por $M_G(a_{ij})$ (ou, simplesmente, M_G), como a matriz de dimensão $|V|\times|V|$ onde a_{ij} é igual ao número de arestas entre os vértices v_i e v_j. No caso de se tratar de um digrafo $\vec{G}(V, E)$, $M_{\vec{G}}(a_{ij})$ é tal que a_{ij} é igual ao número de arestas com cauda em v_i e cabeça em v_j.

EXEMPLO 4.5. As matrizes de adjacências dos grafos do Exemplo 4.1 são, respetivamente,

$$\begin{pmatrix} 1 & 2 & 1 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Notemos que, num grafo não orientado, a matriz de adjacências é simétrica. Observamos ainda que:

- Num grafo não orientado, a soma das entradas de cada linha, multiplicando por 2 a entrada da diagonal principal, é igual ao grau do vértice correspondente a essa linha. O mesmo resultado se tem em relação às colunas da matriz.
- Num digrafo, a soma das entradas de cada linha é igual ao grau de saída do vértice correspondente e a soma das entradas de cada coluna é igual ao grau de entrada do vértice correspondente.

Isomorfismo de grafos. Dois grafos $G = (V(G), E(G))$ e $H = (V(H), E(H))$ dizem-se *isomorfos* se existir uma bijeção entre o conjunto dos vértices de G e o conjunto dos vértices de H e uma bijeção entre o conjunto das arestas de G e o conjunto das arestas de H que preservam a relação de adjacência e de incidência. De outra forma, existem

$$\varphi : V(G) \to V(H) \quad \text{e} \quad \psi : E(G) \to E(H)$$

tais que

$$e = (u, v) \in E(G) \text{ se e só se } \psi(e) = (\varphi(u), \varphi(v)) \in E(H).$$

EXEMPLO 4.6. Os grafos orientados seguintes são isomorfos:
Basta observar que a bijeção entre os vértices seguinte preserva a relação de adjacência:

$$\begin{array}{rccc} \varphi : & V(G) & \to & V(H) \\ & 1 & \mapsto & a \\ & 2 & \mapsto & b \\ & 3 & \mapsto & c \\ & 4 & \mapsto & d \\ & 5 & \mapsto & e \end{array}$$

e construir a bijeção $\psi : E(G) \to E(H)$ de modo a satisfazer a relação de incidência.

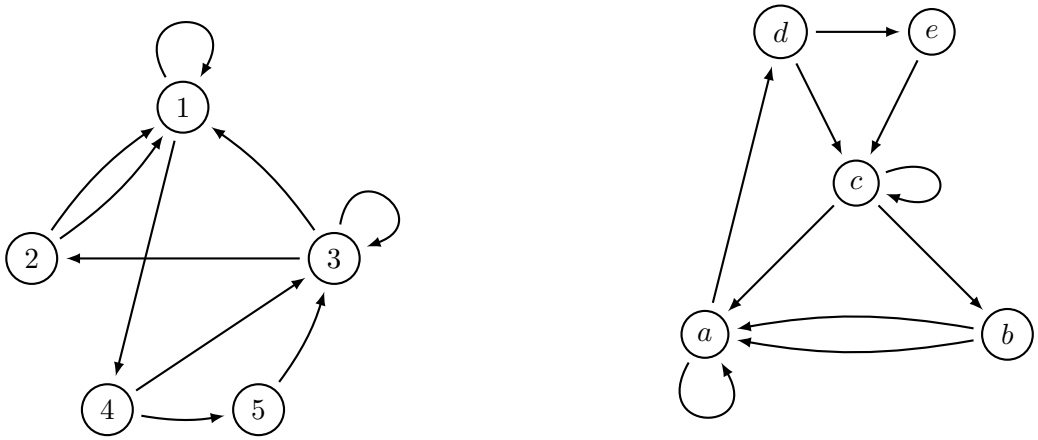


FIGURA 2. Grafos isomorfos.

Subgrafos. Um *subgrafo* de um grafo $G = (V(G), E(G))$ é um grafo $H = (V(H), E(H))$ onde $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Se H for um subgrafo de G diferente de G, então H denomina-se de *subgrafo próprio*.

EXEMPLO 4.7. Os grafos seguintes são subgrafos próprios dos grafos do exemplo anterior:

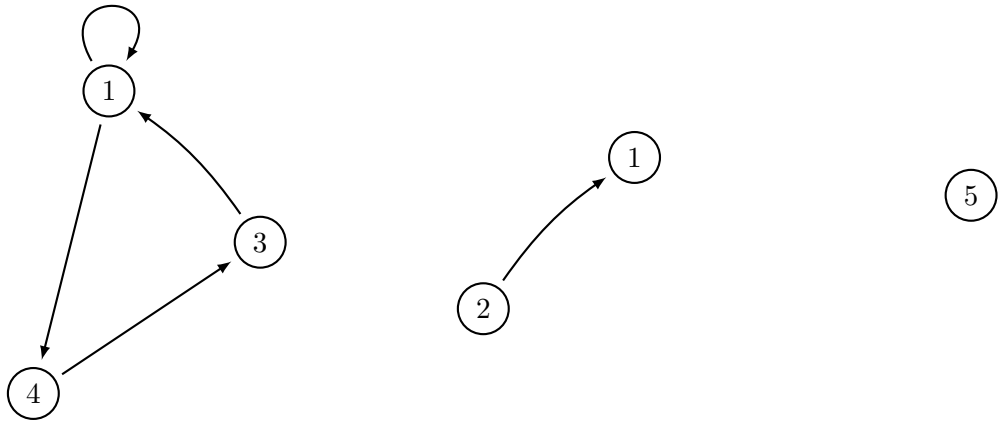


FIGURA 3. Subgrafos.

Passeios. Muitos problemas resolúveis com Teoria de Grafos são modelados por grafos que envolvem percursos que percorrem as arestas do grafo. É por isso necessário introduzir os conceitos que se seguem.

Dado um grafo $G = (V(G), E(G))$, designamos por *passeio* em G toda a sequência não vazia

$$P = (v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$$

com $v_0, v_1, \ldots v_k \in V(G)$, $e_1, e_2, \ldots e_k \in E(G)$ e os vértices v_{i-1} e v_i são os extremos da aresta e_i, para todo $i = 1, \ldots, k$. Designamos o vértice v_0 de *vértice inicial*, o vértice v_k de *vértice final* e

os vértices $v_1, \ldots, v_{k-1}$ de *vértices intermédios* do passeio P. Dizemos ainda que P é um passeio entre os vértices v_0 e v_k.

O passeio P designa-se por *trajeto* se todas as arestas forem distintas. Se, adicionalmente, todos os vértices forem distintos, o passeio P designa-se por *caminho simples* (ou, unicamente, *caminho*).

Se o grafo G for um grafo sem arestas múltiplas, então, a existir, existe uma única aresta de u para v, quaisquer que sejam os vértices u e v, pelo que podemos apresentar o passeio P simplesmente pela sequência dos seus vértices, ou seja,

$$P = (v_0, v_1, v_2, \ldots, v_k).$$

EXEMPLO 4.8. As sequências seguintes são exemplos de, respetivamente, um passeio, um trajeto e um caminho simples. Os números representam a ordem pela qual as arestas são percorridas.

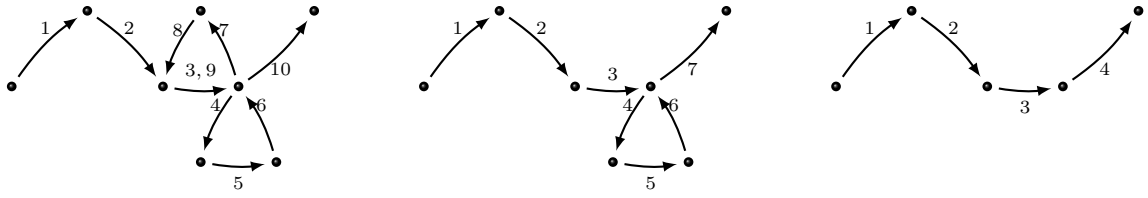


FIGURA 4. Passeios.

No caso do vértice inicial ser igual ao vértice final, então designamos os conceitos anteriores, respetivamente, por *passeio fechado*, *trajeto fechado* e *circuito* (ou *ciclo*), em que, neste último conceito, não existe repetição dos vértices exceto o inicial e o final.

EXEMPLO 4.9. As sequências seguintes são exemplos de, respetivamente, um passeio fechado, um trajeto fechado e um ciclo. Os números representam a ordem pela qual as arestas são percorridas.

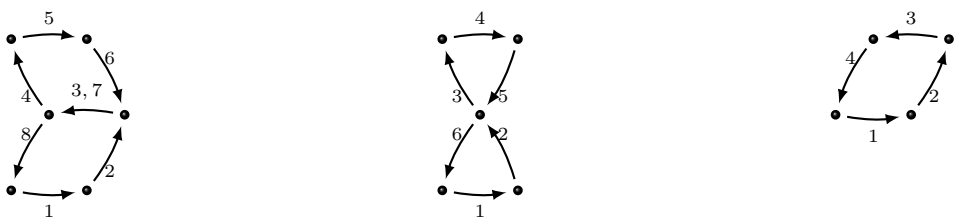


FIGURA 5. Passeios fechados.

Dado um passeio P de um grafo G designa-se por *comprimento* de P, e denota-se por $comp(P)$, o número de arestas (com eventual repetição) que o constitui.

Assim, dado um grafo G, a função seguinte define a *distância* entre dois quaisquer vértices u e v de G, onde $\mathcal{P}_{u,v}$ é o conjunto de todos os passeios entre u e v:

$$dist_G(u,v) = \begin{cases} \min_{P \in \mathcal{P}_{u,v}} comp(P) & \text{se } \mathcal{P}_{u,v} \neq \emptyset, \\ \infty & \text{se } \mathcal{P}_{u,v} = \emptyset. \end{cases}$$

Conexidade. Finalmente, terminamos esta secção com o conceito de *conexidade* de um grafo.

Um grafo não orientado diz-se *conexo* se, para cada par de vértices, existe um passeio que os une. Caso contrário, o grafo diz-se *desconexo*.

Um grafo orientado é *conexo* se o seu *grafo suporte*, isto é, o grafo que se obtém ignorando a orientação das arestas, for conexo.

Um grafo orientado diz-se *fortemente conexo* se, para cada par de vértices, existe um passeio orientado que os une.

Exemplo 4.10. Na figura que se segue o primeiro grafo é fortemente conexo. Basta observar que o passeio fechado $P = (1, 4, 5, 3, 2, 1)$ passa por todos os vértices do grafo, definindo um passeio entre quaisquer dois vértices.

O segundo grafo não é fortemente conexo pois não é possível atingir o vértice 2 partindo de qualquer outro vértice. No entanto, este grafo é conexo, pois o seu grafo suporte é conexo.

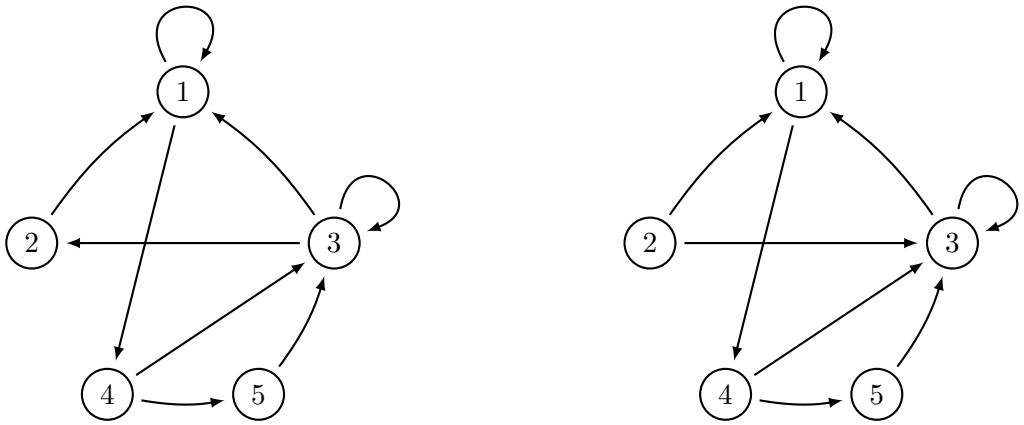


Figura 6. Digrafo fortemento conexo e digrafo que não é fortemento conexo mas é conexo.

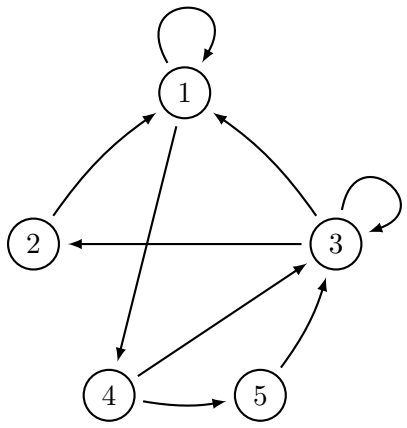
2. Passeios

Contar o número de passeios entre dois vértices. O número de passeios entre dois vértices de um grafo é facilmente determinado usando a sua matriz de adjacências, como é enunciado no teorema seguinte.

TEOREMA 4.4. *Seja G um grafo com matriz de adjacências M, com respeito à ordem dos vértices $v_1, v_2, \ldots, v_n$. O número de diferentes passeios de comprimentos c do vértice v_i para o vértice v_j é igual à entrada $a_{(i,j)}$ da matriz M^c.*

Notemos que o teorema anterior é válido para grafos não orientados, digrafos, grafos com arestas múltiplas ou lacetes.

EXEMPLO 4.11. Dado o grafo seguinte e a sua matriz de adjacências,



$$M = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

o número de passeios de comprimento 4 entre dois quaisquer vértices é dado pela matriz M^4,

$$M^4 = \begin{pmatrix} 6 & 3 & 5 & 2 & 1 \\ 2 & 1 & 3 & 1 & 1 \\ 8 & 2 & 6 & 5 & 3 \\ 8 & 2 & 3 & 4 & 1 \\ 5 & 1 & 2 & 3 & 1 \end{pmatrix}.$$

Assim, existem 5 passeios de comprimentos 4 do vértice 5 para o vértice 1.

Definimos o *fecho transitivo* de uma matriz de adjacências M, de ordem n, de um grafo G como a matriz F definida por

$$F = M \oplus M^2 \oplus \cdots \oplus M^n,$$

onde a adição e a multiplicação das matrizes são as Booleanas (ver Lema 1.3).

COROLÁRIO 4.5. *Um grafo não orientado (orientado) é conexo (fortemente conexo) se o fecho transitivo da sua matriz de adjacências não tiver entradas nulas.*

EXEMPLO 4.12. O grafo do exemplo anterior é fortemente conexo pois

$$F = M \oplus M^2 \oplus M^3 \oplus M^4 \oplus M^5 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

não tem entradas nulas.

Trajetos Eulerianos. (Texto retirado de [9]) A origem da Teoria de Grafos é, em geral, associada ao *Problema das pontes de Königsberg*, cidade da Prússia que agora se designa por Kaliningrado. Parte desta cidade localizava-se em duas ilhas do rio Pregel as quais estavam ligadas às margens e uma à outra através de sete pontes, conforme a Figura 7 documenta.

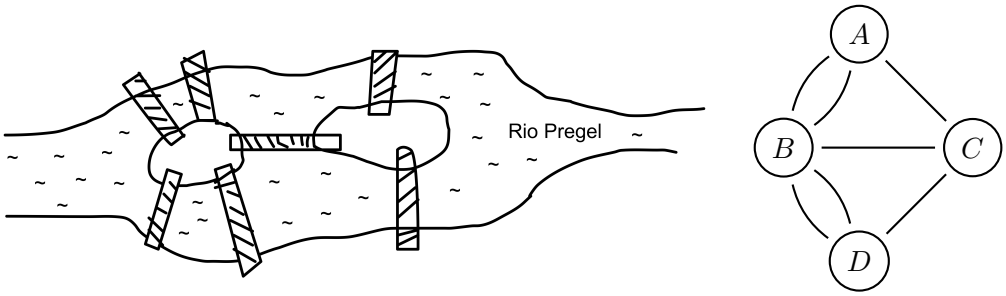


FIGURA 7. Pontes de Königsberg em 1736 e o respetivo grafo.

Consta que os habitantes de Königsberg gostavam de dar passeios de modo a atravessar todas as pontes e que alguns andavam particularmente aborrecidos pelo facto de não encontrarem um trajeto, com partida e chegada ao mesmo lugar, que lhes permitisse atravessar apenas uma vez cada uma das pontes. O matemático suíço Leonhard Euler (1707-1783), ao tomar conhecimento deste problema, resolveu-o, indicando a impossibilidade da existência de um tal percurso, modelando-o pelo gráfico representado na Figura 7.

Denominamos de *trajeto de Euler* (*trajeto de Euler fechado*) um trajeto (trajeto fechado) que contém todas as arestas de um grafo, ou seja, um passeio (passeio fechado) que passa por todas as arestas do grafo exatamente uma vez.

Um grafo diz-se *Euleriano* (ou *grafo de Euler*) se admite um trajeto de Euler fechado e diz-se *semi-Euleriano* se admite um trajeto de Euler.

Obviamente, um grafo Euleriano é também semi-Euleriano.

O problema das pontes de Königsberg reduz-se agora ao problema de saber se o grafo da Figura 7 é ou não Euleriano.

TEOREMA 4.6. *Um grafo não orientado e conexo é Euleriano se e só se todos os seus vértices têm grau par.*

TEOREMA 4.7. *Um grafo não orientado e conexo é semi-Euleriano (e não Euleriano) se e só se possuir exatamente dois vértices de grau ímpar.*

EXEMPLO 4.13. O grafo que modela o problema das pontes de Königsberg não é nem Euleriano, nem semi-Euleriano, visto possuir quatro vértices de grau ímpar. Resulta que o problema de encontrar um passeio que passe uma única vez por todas as pontes é impossível.

EXEMPLO 4.14. Os grafos da Figura 8 são, respetivamente, Euleriano e semi-Euleriano.

Em 1883, Fleury apresenta um algoritmo eficiente (complexidade polinomial) que determina os trajetos Eulerianos.

FIGURA 8. Grafo Euleriano e grafo semi-Euleriano.

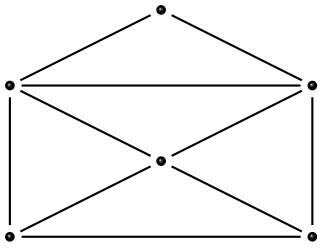
Algoritmo de Fleury

Input: grafo G.

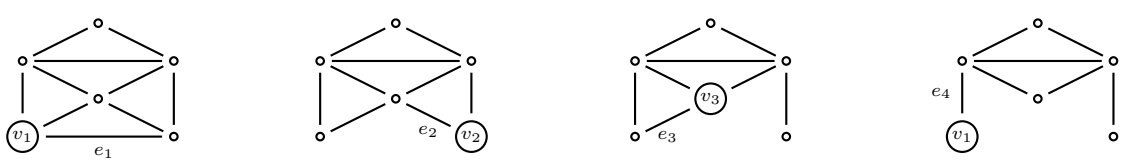
Output: trajeto de Euler ϵ, caso exista.

1. Determinam-se os graus de todos os vértices.
2. Se o grafo tiver um número de vértices de grau ímpar diferente de 2 e 0, então o algoritmo para e dá como resposta *"O grafo não é Euleriano nem semi-Euleriano"*.
3. Se houver 2 vértices de grau ímpar, escolhe-se um deles. Se não, escolhe-se qualquer vértice. Seja v o vértice escolhido e $\epsilon = v$.
4. Se não houver nenhuma aresta incidente em v, então o algoritmo para e devolve o trajeto de Euler ϵ. Caso contrário, se houver apenas uma aresta, escolhe-se essa aresta e removem-se essa aresta e o vértice do grafo; se houver mais do que uma aresta incidente no vértice, escolhe-se uma aresta que, ao ser removida, não desconecte o grafo; remove-se essa aresta. Seja $e = (v, w)$ a aresta escolhida.
5. Adiciona-se ew como sufixo de ϵ.
6. Substitui-se v por w e regressa-se a (4).

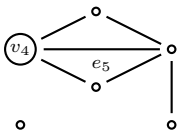
EXEMPLO 4.15. Determinemos um trajeto semi-Euleriano para o grafo seguinte, usando o Algoritmo de Fleury:



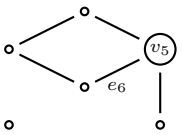
Os passos efetuados pelo algoritmo (não determinístico) podem ser observados na figura seguinte. Optou-se por escrever o trajeto unicamente usando a sequência dos vértices percorridos.



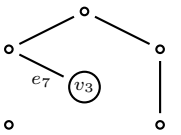
$TE = v_1$ $\quad$ $TE = v_1v_2$ $\quad$ $TE = v_1v_2v_3$ $\quad$ $TE = v_1v_2v_3v_1$



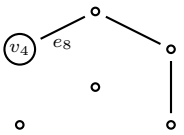
$TE = v_1v_2v_3v_1v_4$



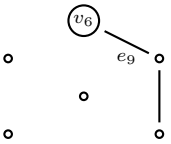
$TE = v_1v_2v_3v_1v_4v_5$



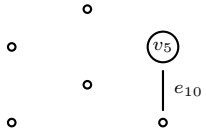
$TE = v_1v_2v_3v_1v_4v_5v_3$



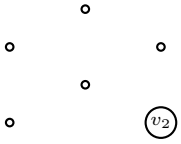
$TE = v_1v_2v_3v_1v_4v_5v_3v_4$



$TE = v_1v_2v_3v_1v_4v_5v_3v_4v_6$



$TE = v_1v_2v_3v_1v_4v_5v_3v_4v_6v_5$



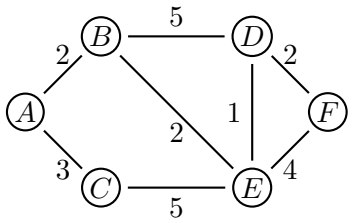
$TE = v_1v_2v_3v_1v_4v_5v_3v_4v_6v_5v_2$

3. Grafos ponderados

Muitos problemas podem ser modelados por grafos com custos (ou pesos) nas suas arestas. Como exemplos, em problemas sobre distâncias, os vértices representam as cidades e as arestas estradas etiquetadas com os valores das distâncias entre as cidades que elas ligam; em problemas que envolvem tempos de voo, às arestas são atribuídos o tempo de voo entre os dois aeroportos; em problemas que envolvem custos, como o custo do transporte de energia, as etiquetas nas arestas representam o custo do transporte de energia entre duas centrais; etc.

Dizemos que um grafo é *ponderado* (ou *pesado*) se a cada uma das suas arestas está associado um número, denominado por *peso* da aresta. Denotamos um grafo ponderado por um terno $G = (V, E, W)$, onde $W = (w_{ij})$ denota a matriz de custos. Nesta matriz, a entrada w_{ij} corresponde ao custo da aresta (i, j), se tal aresta existe, ou $w_{ij} = \infty$, se $(i, j) \notin E(G)$. Adicionalmente, assume-se que $w_{ii} = 0$, para cada i.

O *custo* (ou *peso*) de um passeio é a soma dos pesos das arestas desse passeio.



$$M = \begin{pmatrix} 0 & 2 & 3 & \infty & \infty & \infty \\ 2 & 0 & \infty & 5 & 2 & \infty \\ 3 & \infty & 0 & \infty & 5 & \infty \\ \infty & 5 & \infty & 0 & 1 & 2 \\ \infty & 2 & 5 & 1 & 0 & 4 \\ \infty & \infty & \infty & 2 & 4 & 0 \end{pmatrix}$$

FIGURA 9. Grafo ponderado e matriz de custos do grafo.

Um dos problemas usuais que se pretende resolver é o de encontrar o caminho de menor custo entre dois vértices. Há vários algoritmos que resolvem este problema. Neste texto, apresentamos o Algoritmo de Dijkstra (1959), que possui complexidade quadrática. Este algoritmo aceita como *input* todo o grafo não orientado, conexo com pesos positivos nas arestas, mas pode ser adaptado para grafos orientados e com custos arbitrários.

A ideia básica deste algoritmo é ir determinando sucessivamente passeios de maior comprimento e com menor distância (custo) que partem de a e chegam a um vértice do grafo, até alcançarmos o vértice final z. No passo corrente marcam-se temporariamente os vértices que nesse passo se consideram mais próximos de a e muda-se a marca temporária de um vértice v para marca permanente quando se obtém o caminho mais curto entre a e v.

Usamos a notação que se segue, onde a é o vértice inicial e v é um vértice do grafo:

- $Marca(v)$: comprimento do caminho mais curto entre a e v;
- $Antecessor(v)$: antecessor do vértice v no caminho mais curto entre a e v de entre os já determinados;
- $Temporarios$: conjunto dos vértices com marca temporária;
- v^*: vértice com menor marca temporária corrente, a qual vai passar a marca permanente.

Algoritmo de Dijkstra

Input: grafo $G = (V, E, W)$ não orientado, conexo, com pesos positivos nas arestas; vértice inicial a e vértice final z.

Output: caminho simples de menor custo entre os dois vértices a e z, e custo desse caminho.

1. Marcam-se o vértice a com marca permanente 0 (pois $dist(a, a) = 0$) e todos os restantes vértices com a marca temporária ∞:
$$Marca(a) = 0 \text{ e } Marca(v) = \infty, \text{ para todo o } v \neq a.$$
2. Define-se o conjunto dos vértices com marca temporária, $Temporarios = V(G)\backslash\{a\}$, e o vértice de marca mínima $v^* = a$.
3. Para cada vértice $v \in Temporarios$, determina-se uma nova marca segundo a lei seguinte:
$$Marca(v) = \begin{cases} Marca(v^*) + w_{v^*v} & \text{se } Marca(v) > Marca(v^*) + w_{v^*v}, \\ Marca(v) & \text{se } Marca(v) \leq Marca(v^*) + w_{v^*v}, \end{cases}$$
(ou seja, se $Marca(v) > Marca(v^*) + w_{v^*v}$, então o caminho mais curto entre a e v passa por v^*, pelo que se atualiza $Marca(v)$.)
4. Se houver atualização da marca de v em (3), então $Antecessor(v) = v^*$.
5. Determina-se o novo vértice v^* de marca mínima de entre os elementos de $Temporarios$. Caso haja mais do que um vértice nestas condições, escolhe-se um deles. Retira-se este vértice de $Temporarios$, ou seja, a sua marca passa a marca permanente.
6. Se $v^* \neq z$, então volta-se ao Passo (3).
7. O algoritmo devolve o custo de um caminho de menor custo entre a e z, $Marca(z)$, e esse caminho, $(a, \ldots, Antecessor(Antecessor(z)), Antecessor(z), z)$.

EXEMPLO 4.16. Usando o Algoritmo de Dijkstra, vamos determinar o caminho de menor custo entre os vértices A e F do grafo da Figura 9. Na tabela seguinte, apresentamos os valores obtidos ao longo da aplicação do algoritmo: para cada vértice v, calculamos os sucessivos pares $(Marca(v), Antecessor(v))$ e etiquetamos com $\_^*$ o vértice que nesse passo passa a ser o vértice de marca mínima e passamos a sua marca a permanente.

A	B	C	D	E	F
$(0,-)^*$	$(\infty,-)$	$(\infty,-)$	$(\infty,-)$	$(\infty,-)$	$(\infty,-)$
	$(2,A)^*$	$(3,A)$	$(\infty,-)$	$(\infty,-)$	$(\infty,-)$
		$(3,A)^*$	$(7,B)$	$(4,B)$	$(\infty,-)$
			$(7,B)$	$(4,B)^*$	$(\infty,-)$
			$(5,E)^*$		$(8,E)$
					$(7,D)^*$

Resulta que um caminho de menor custo entre o vértice A e o vértice F tem custo 7 e é

$$(A,B,E,D,F).$$

Exemplo 4.17. Usando o Algoritmo de Dijkstra, determinemos o caminho de menor custo entre o segundo e o sexto vértice do grafo definido pela matriz de adjacências seguinte:

$$M = \begin{pmatrix} 0 & 9 & 4 & 15 & \infty & 2 \\ 9 & 0 & 2 & 4 & \infty & \infty \\ 4 & 2 & 0 & 3 & 6 & 7 \\ 15 & 4 & 3 & 0 & 20 & \infty \\ \infty & \infty & 6 & 20 & 0 & 5 \\ 2 & \infty & 7 & \infty & 5 & 0 \end{pmatrix}.$$

Usamos novamente a tabela introduzida no exemplo anterior, para registarmos os cálculos ao longo da implementação do algoritmo:

1	2	3	4	5	6
$(\infty,-)$	$(0,-)^*$	$(\infty,-)$	$(\infty,-)$	$(\infty,-)$	$(\infty,-)$
$(9,2)$		$(2,2)^*$	$(4,2)$	$(\infty,-)$	$(\infty,-)$
$(6,3)$			$(4,2)^*$	$(8,3)$	$(9,3)$
$(6,3)^*$				$(8,3)$	$(9,3)$
				$(8,3)$	$(8,1)^*$

Resulta que um caminho de menor custo entre o vértice 2 e o vértice 6 tem custo 8 e é

$$(2,3,1,6).$$

Notemos que, na última iteração do algoritmo, há dois vértices com menor marca temporária. Neste caso, podemos escolher qualquer um dos vértices e continuar o procedimento do algoritmo.

Teorema 4.8. *O Algoritmo de Dijkstra, que encontra o caminho de menor custo entre dois vértices num grafo não orientado, conexo, com pesos positivos nas arestas e com n vértices, tem complexidade* $O(n^2)$.

Demonstração. Basta observar que o algoritmo faz, no máximo, $n-1$ iterações do ciclo constituído pelos Passos $(3)-(6)$. Em cada iteração, o algoritmo atualiza as marcas dos vértices do conjunto dos temporários, ou seja, $O(n)$ atualizações. Como todas as operações envolvidas são operações primitivas, resulta que o algoritmo efetua $O(n^2)$ operações primitivas. □

4. Árvores geradoras de custo mínimo

Nesta secção estudamos um tipo de grafos chamados árvores. São particularmente úteis em Ciências da Computação e Informática pois modelam, entre outros, problemas de decisão. São também usadas como modelos em diversas áreas, tais como Química, Biologia e Geologia (ver Figura 10[2]).

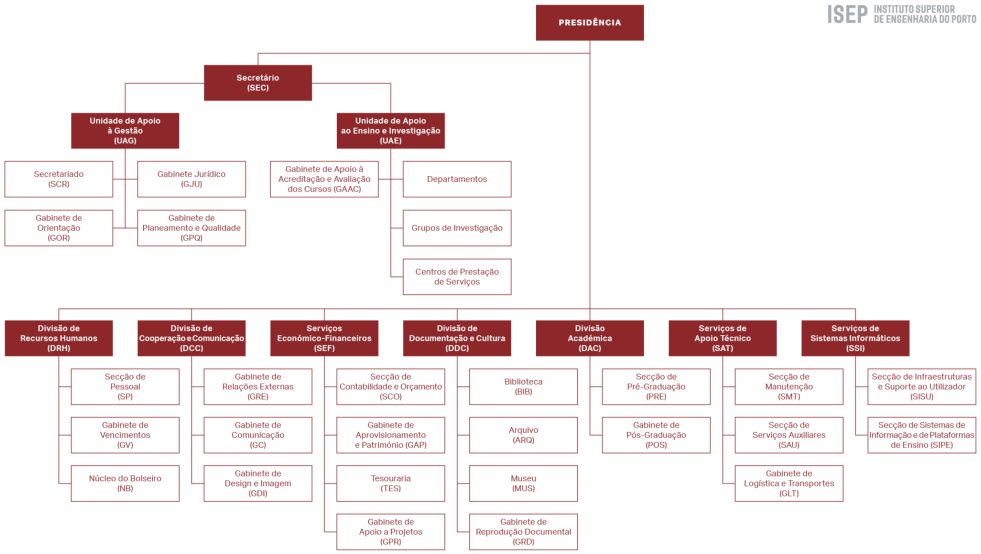


FIGURA 10. O organigrama dos serviços do ISEP representado por uma árvore.

Um grafo G diz-se uma *árvore* se for conexo e não contiver ciclos. Em particular, G não tem lacetes nem arestas múltiplas, pelo que é um grafo simples.

EXEMPLO 4.18. O grafo da figura seguinte é uma árvore.

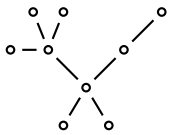


FIGURA 11. Árvore.

TEOREMA 4.9. *Seja $G = (V, E)$ uma árvore com n vértices. As condições seguintes são verificadas:*

1. *G tem $n - 1$ arestas.*
2. *Quaisquer dois vértices de G estão ligados por um único caminho simples.*
3. *G não tem ciclos, mas acrescentando uma aresta obtém-se um ciclo.*

Dado um grafo conexo G, denominamos de *árvore geradora* de G todo o subgrafo de G que é uma árvore e contém todos os vértices de G.

[2]Imagem autorizada e cedida pelo ISEP - Instituto Superior de Engenharia do Porto.

EXEMPLO 4.19. Na figura seguinte o segundo e terceiro grafos são árvores geradoras do primeiro grafo.

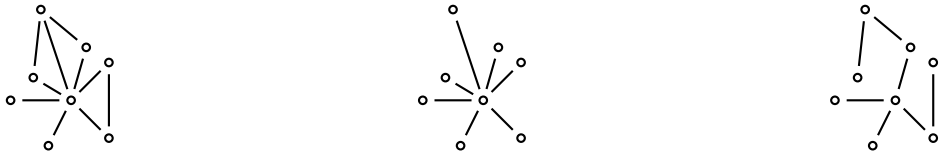


FIGURA 12. Árvores geradoras.

As árvores geradoras são subgrafos de um grafo que podem ser entendidas como esqueletos desse grafo, no sentido em que ligam todos os vértices do grafo e são minimais para esta condição (ou seja, não possuem um subgrafo próprio que satisfaça a propriedade). Vários problemas modelados por grafos recorrem às arvores geradoras do grafo.

TEOREMA 4.10. *Todo o grafo conexo admite uma árvore geradora.*

O problema que abordamos neste texto é o de encontrar a árvore geradora de um grafo ponderado cuja soma dos pesos das suas arestas seja mínima.

Definimos *árvore geradora minimal* de um grafo ponderado como a árvore geradora cuja soma dos pesos das arestas é miníma.

Apresentamos o Algoritmo de Kruskal (1956) que encontra a árvore geradora minimal de um grafo ponderado. Este algoritmo é um algoritmo guloso, no sentido em que, em cada iteração, faz a melhor escolha. Nem sempre um algoritmo guloso produz a solução óptima do problema. No caso do Algoritmo de Kruskal tal solução é produzida.

A ideia base do algoritmo é ir escolhendo sempre a aresta de menor custo que ainda não foi adicionada à arvore geradora em construção e que não forma um ciclo com nenhum conjunto de arestas dessa árvore. O algoritmo para após termos escolhido $n-1$ arestas, onde n é o número de vértices do grafo.

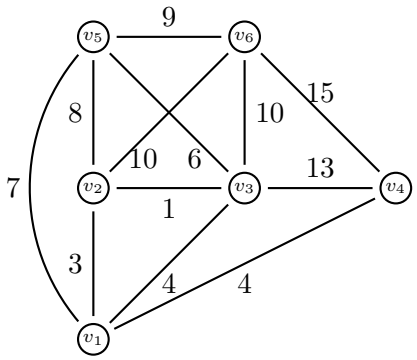
Algoritmo de Kruskal

Input: grafo $G = (V, E, W)$ ponderado não orientado, conexo, com n vértices, $v_1, \ldots, v_n$;
Output: árvore geradora de custo mínimo.

1. Definem-se A como a árvore geradora que será construída, onde, inicialmente, é vazio, e $S_i = \{v_i\}$, $i = 1, \ldots, n$. Ordenam-se as arestas por ordem crescente dos seus pesos. Caso haja arestas com o mesmo peso, qualquer ordenação entre elas é válida. Seja n_A o número de arestas de A, onde inicialmente $n_A = 0$.
2. Repete-se o procedimento seguinte enquanto $n_A < n - 1$:
   3. Escolhe-se a aresta seguinte, (v_i, v_j), da sequência ordenada das arestas (na primeira iteração, escolhe-se a primeira aresta).
   4. Sejam S_p e S_q tais que $v_i \in S_p$ e $v_j \in S_q$. Se $S_p \neq S_q$, então adiciona-se a aresta (v_i, v_j) a A, $S_p = S_p \cup S_q$, elimina-se S_q e $n_A = n_A + 1$.

5. O algoritmo devolve A.

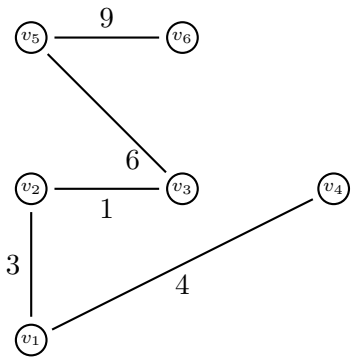
EXEMPLO 4.20. Usando o Algoritmo de Kruskal, determinamos uma árvore geradora de custo mínimo do grafo da figura seguinte.



Apresentamos os passos do algoritmo na tabela seguinte:

iteração	(v_i, v_j)	S_p	S_q	A	S_1	S_2	S_3	S_4	S_5	S_6	n_A
					v_1	v_2	v_3	v_4	v_5	v_6	0
1	(v_2, v_3)	S_2	S_3	(v_2, v_3)	v_1	v_2, v_3		v_4	v_5	v_6	1
2	(v_1, v_2)	S_1	S_2	(v_1, v_2)	v_1, v_2, v_3			v_4	v_5	v_6	2
3	(v_1, v_3)	S_1	S_1		v_1, v_2, v_3			v_4	v_5	v_6	2
4	(v_1, v_4)	S_1	S_4	(v_1, v_4)	v_1, v_2, v_3, v_4				v_5	v_6	3
5	(v_3, v_5)	S_1	S_5	(v_3, v_5)	v_1, v_2, v_3, v_4, v_5					v_6	4
6	(v_1, v_5)	S_1	S_1		v_1, v_2, v_3, v_4, v_5					v_6	4
7	(v_2, v_5)	S_1	S_1		v_1, v_2, v_3, v_4, v_5					v_6	4
8	(v_5, v_6)	S_1	S_6	(v_5, v_6)	$v_1, v_2, v_3, v_4, v_5, v_6$						5
9	(v_2, v_6)										
10	(v_3, v_6)										
11	(v_3, v_4)										
12	(v_4, v_6)										

Resulta que a árvore geradora minimal tem custo 23 e é



TEOREMA 4.11. *O Algoritmo de Kruskal, que encontra a árvore geradora de custo mínimo, tem complexidade $O(e \log e)$, onde e é o número de arestas do grafo.*

5. Coloração de grafos planares

Um grafo diz-se *planar* se puder ser representado no plano sem que as arestas se cruzem. Tal representação diz-se a *representação planar* do grafo.

Exemplo 4.21. O grafo representado na figura seguinte é planar. A segunda representação é uma representação planar desse grafo.

Figura 13. Grafo planar.

Um dos problemas mais antigos da Teoria de Grafos é o da *coloração de mapas*. Pretende-se saber qual o menor número de cores necessárias para colorir um mapa de modo a que regiões que partilhem uma fronteira tenham diferentes colorações. Considera-se que regiões que só se tocam num ponto não são adjacentes.

Um mapa pode ser representado por um grafo da forma seguinte: cada região do mapa é representada por um vértice e dois vértices que representem regiões adjacentes são ligados por uma aresta.

Designa-se de *coloração* de um grafo simples a uma correspondência de uma cor a cada vértice do grafo, de modo a que vértices adjacentes não tenham a mesma correspondência de cor.

O *número cromático* de um grafo é o menor número de cores necessárias para colorir o grafo.

Durante mais de cem anos conjeturou-se que bastariam quatro cores para colorir qualquer grafo plano, resultado que foi finalmente provado em 1976.

Teorema 4.12. *O número cromático de um grafo planar é, no máximo, quatro.*

Apresentamos o Algoritmo de Welsh-Powell para coloração de grafos. É um algoritmo guloso, no sentido em que ordena os vértices por ordem decrescente de grau e vai atribuindo a primeira cor possivel a cada vértice, de uma sequência também ordenada de cores. Notemos que este algoritmo não devolve sempre a coloração mínima de um grafo.

Algoritmo de Welsh-Powell

Input: grafo G.

Output: coloração de G.

1. Ordenam-se os vértices por ordem decrescente dos seus graus. Caso haja vértices com o mesmo grau, escolhe-se qualquer ordenação entre eles.

2. Faz-se corresponder a primeira cor ao primeiro vértice e, seguidamente, faz-se corresponder essa cor ao primeiro vértice da sequência que não é adjacente a nenhum vértice ao qual já foi atribuída essa cor.
3. Repete-se o passo anterior com uma nova cor onde o conjunto dos vértices a colorir contém os que ainda não estão coloridos.
4. Repete-se o item anterior enquanto houver vértices por colorir.

EXEMPLO 4.22. Usemos o Algoritmo de Welsh-Powell para colorir o mapa com os distritos de Portugal[3]. Este mapa pode ser representado pelo grafo da figura.

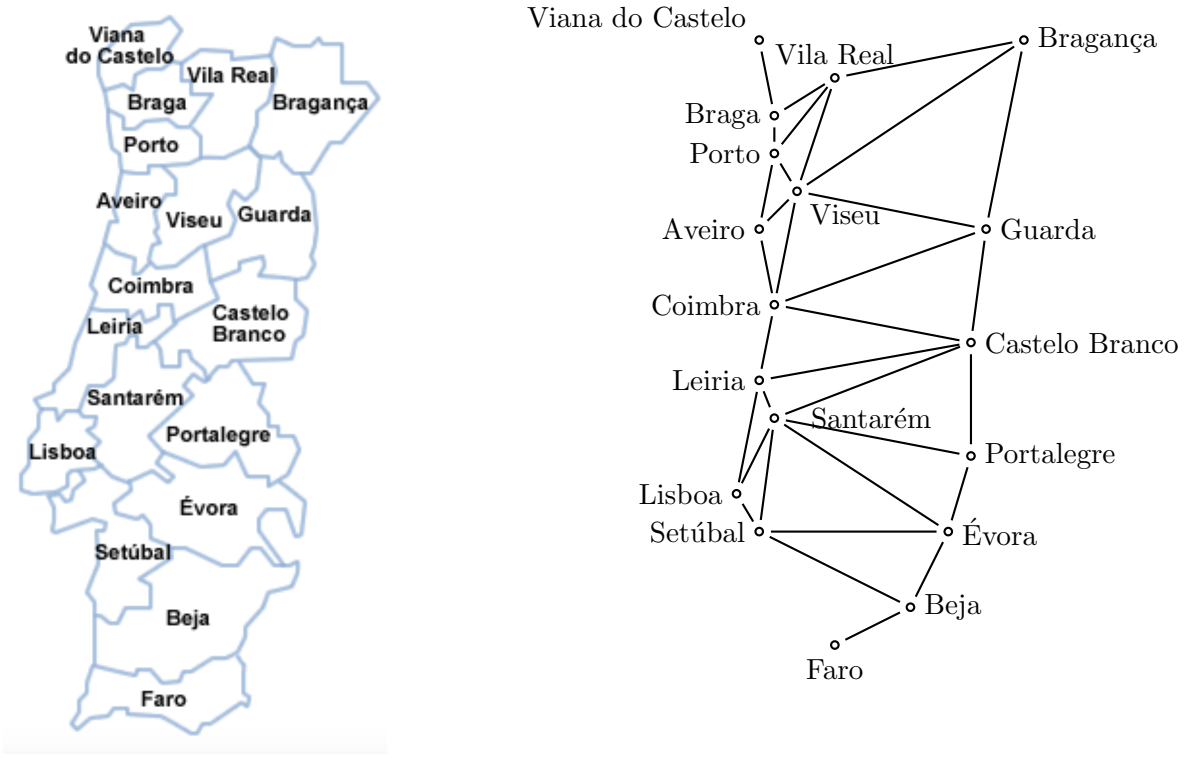


FIGURA 14. Mapa dos distritos de Portugal e representação por grafo.

Apresentamos a correspondência de cores aos vértices do grafo na tabela seguinte:

[3]Imagem disponível na página web das Conservatórias do Registo Comercial, https://conservatorias.com/conservatorias-do-registo-comercial, em 31 de janeiro de 2025.

Vértice	Grau	Cor atribuída	Vértice	Grau	Cor atribuída
Viseu	6	1	Évora	4	3
Santarém	6	1	Braga	3	1
Coimbra	5	2	Bragança	3	3
Castelo Branco	5	3	Aveiro	3	4
Vila Real	4	2	Lisboa	3	3
Porto	4	3	Portalegre	3	2
Guarda	4	4	Beja	3	1
Leiria	4	4	Viana do Castelo	1	2
Setúbal	4	2	Faro	1	2

De onde resulta a coloração seguinte:

FIGURA 15. Coloração do mapa dos distritos de Portugal.

Exercícios propostos

1. Considere os grafos não orientados $G_i = (V(G_i), E(G_i))$, $i = 1, 2, 3$, seguintes:

G_1 : $V(G_1) = \{1, 2, 3, 4, 5\}$
$E(G_1) = \{(1, 2), (1, 4), (1, 5), (2, 3), (3, 4), (4, 4)\}$

G_2 : $V(G_2) = \{1, 2, 3, 4, 5, 6\}$
$E(G_2) = \{(1, 2), (1, 4), (1, 4), (2, 3), (2, 5), (3, 5)\}$

G_3 : $V(G_3) = \{A, B, C, D, E, F\}$
$E(G_3) = \{(A, B), (A, C), (A, D), (B, E), (B, F), (C, E), (C, F), (D, E), (D, F)\}$

Para cada um dos grafos anteriores:

a) Represente-o graficamente.

b) Determine a sua matriz de adjacências.

c) Usando a matriz de adjacências determinada na alínea anterior, calcule o grau de cada um dos seus vértices.

d) Calcule o número de passeios de comprimento 3 entre o segundo e o quarto vértice.

e) Indique, justificando, quais dos grafos são grafos simples.

f) Indique, justificando, quais dos grafos são conexos.

2. Considere os grafos orientados $G_i = (V(G_i), E(G_i))$, $i = 1, 2$, seguintes:

G_1 :
$V(G_1) = \{1, 2, 3, 4\}$
$E(G_1) = \{(2, 1), (2, 3), (2, 4), (3, 1), (4, 1), (4, 3)\}$

G_2 :
$V(G_2) = \{A, B, C, D, E\}$
$E(G_2) = \{(A, B), (B, C), (B, E), (C, B), (C, C), (C, D), (D, E), (E, E), (E, D), (E, A)\}$

Para cada um dos grafos anteriores:

a) Represente-o graficamente.

b) Determine a sua matriz de adjacências.

c) Usando a matriz de adjacências determinada na alínea anterior, calcule os graus de entrada e de saída de cada um dos seus vértices.

d) Calcule o número de passeios de comprimento 3 do segundo para o primeiro vértice.

e) Comente a afirmação: "O grafo G_1 é conexo, mas não é fortemente conexo".

f) Mostre que o grafo G_2 é fortemente conexo.

3. Existe algum grafo com 11 vértices em que cada vértice tem grau 5? Em caso afirmativo, construa o grafo.

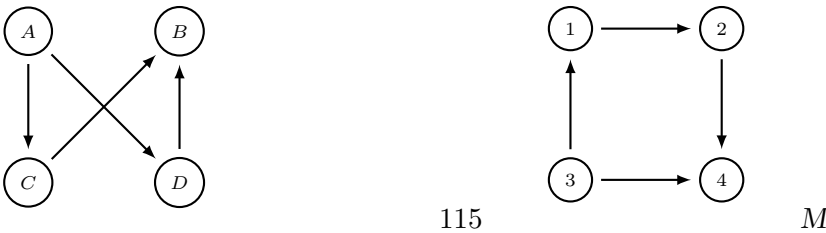
4. Represente graficamente os grafos dados pelas matrizes de adjacências seguintes:
   a) Grafo não orientado:
$$\begin{pmatrix} 1 & 2 & 0 & 1 \\ 2 & 0 & 3 & 0 \\ 0 & 3 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$
   b) Grafo orientado:
$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$
   c) Pode a matriz da alínea *b*) ser a matriz de adjacências de um grafo não orientado? Justifique.
   d) Verifique se o grafo da alínea *b*) é fortemente conexo e justifique a sua resposta.

5. Verifique se os pares de grafos que se seguem são isomorfos:

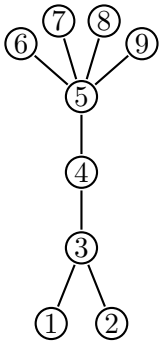
a)


b)

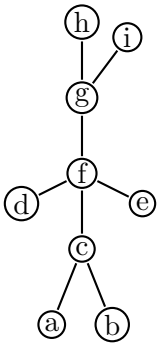

c)


d)


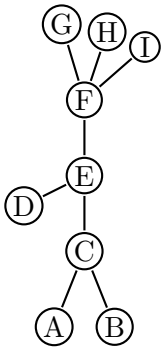
6. Quais dos grafos G_1, G_2, G_3 e G_4 são isomorfos? Justifique a sua resposta: para os pares de grafos isomorfos, construa um isomorfismo entre eles; para os que não forem isomorfos, indique uma propriedade satisfeita por um dos grafos que não seja satisfeita pelo outro e que quebre a relação de isomorfismo.



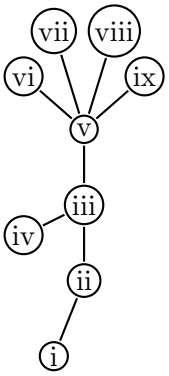
G_1



G_2

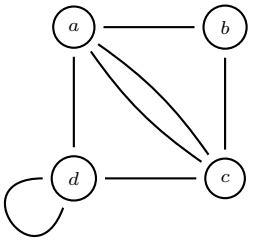


G_3



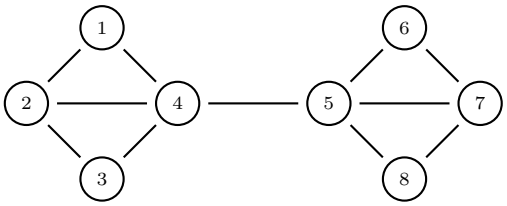
G_4

7. No grafo representado na figura seguinte, procure:



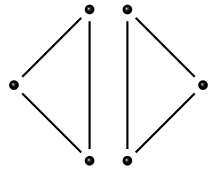
   a) Um passeio de comprimento 7 do vértice d para o vértice b.
   b) Todos os circuitos de comprimento 4.
   c) Um caminho simples de maior comprimento. Qual é o comprimento desse caminho?

8. Considerando o grafo a seguir representado, determine todos os caminhos simples do vértice 1 para o vértice 8.

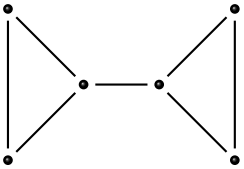


9. Verifique quais dos grafos seguintes admitem uma orientação nas arestas de tal forma que o digrafo obtido seja fortemente conexo. Justifique.

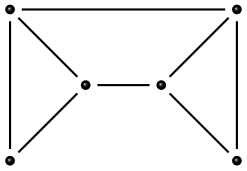
a)



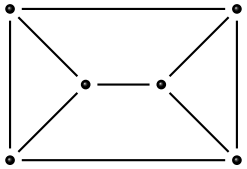
b)



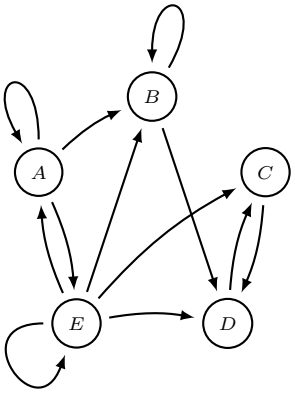
c)



d)



10. Considere o digrafo seguinte:

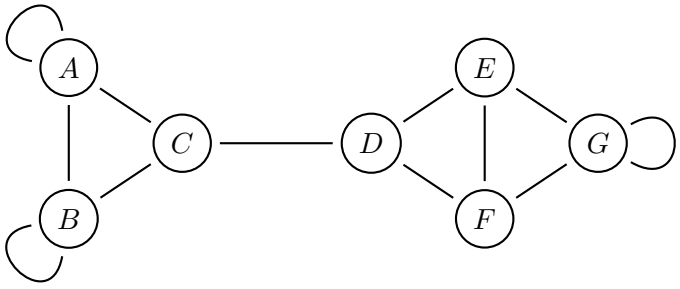


a) Determine a matriz de adjacências do grafo.

b) Calcule, matricialmente, quantos passeios de comprimento igual a 4 existem do vértice A para o vértice C.

c) Tente identificar todos os passeios da alínea anterior.

11. Considere o grafo não-orientado G representado na figura seguinte.



a) Apresente a matriz de adjacências de G.

b) Usando a matriz de adjacências apresentada em a), calcule o número de diferentes passeios de comprimento 3 que existem do vértice D para o vértice G. Determine todos esses passeios.

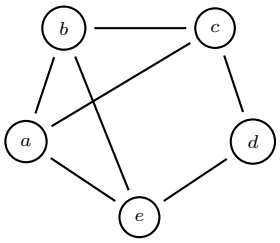
c) Faça a correspondência biunívoca [4] entre os itens da coluna da esquerda e os da coluna da direita:

• Passeio	• (E, F, G, G, E)
• Trajeto	• (E, D, F, G, E)
• Trajeto fechado	• (C, A, A, C, D)
• Caminho simples	• (F, D, E, F, G, E)
• Ciclo	• (A, C, D)

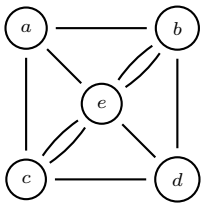
d) Verifique se o grafo admite uma orientação nas arestas de tal forma que o digrafo obtido seja fortemente conexo. Justifique convenientemente a sua resposta.

12. Averigue se os grafos seguintes são Eulerianos ou semi-Eulerianos. Em caso afirmativo, determine um trajeto de Euler fechado (ou um trajeto de Euler) usando o Algoritmo de Fleury.

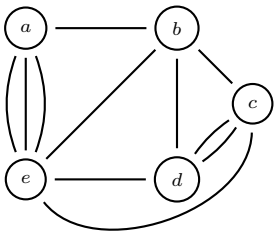
a)



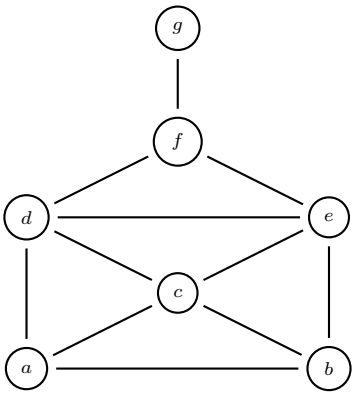
b)



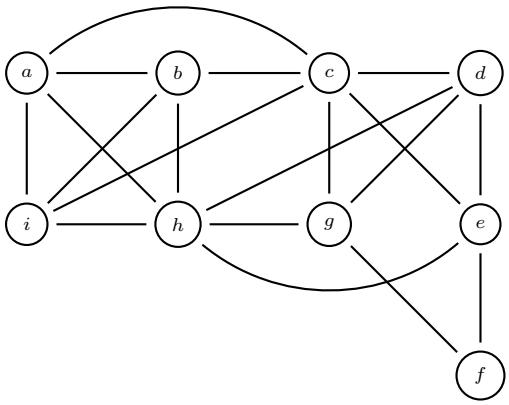
c)



d)

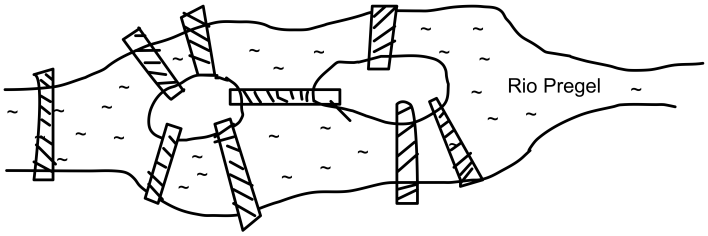


e)

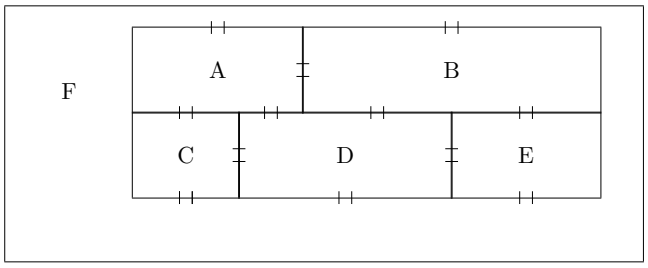


[4] Uma correspondência diz-se *biunívoca* se a cada elemento do primeiro conjunto corresponde um só elemento do segundo conjunto, e vice-versa.

13. Em Kaliningrado (o nome atual de Königsberg) há, atualmente, mais duas pontes para além das 7 pontes do século XVIII, como é apresentado na figura seguinte. É, atualmente, possível atravessar todas as 9 pontes exatamente uma vez e regressar ao ponto de partida?

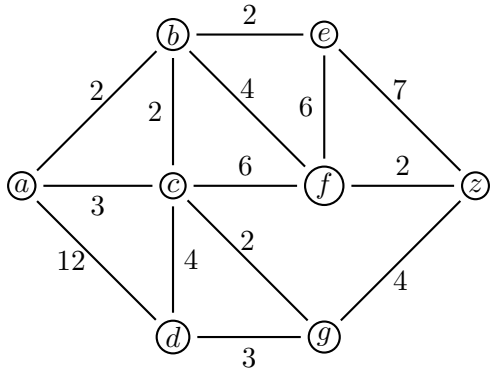


14. A figura seguinte representa a planta de uma casa com jardim. Verifique se é possível atravessar todas as portas sem passar duas ou mais vezes por uma mesma porta.

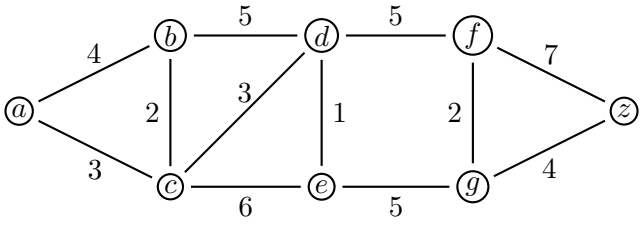


15. Determine um caminho de menor custo, e o seu custo, entre os vértices a e z de cada um dos grafos seguintes:

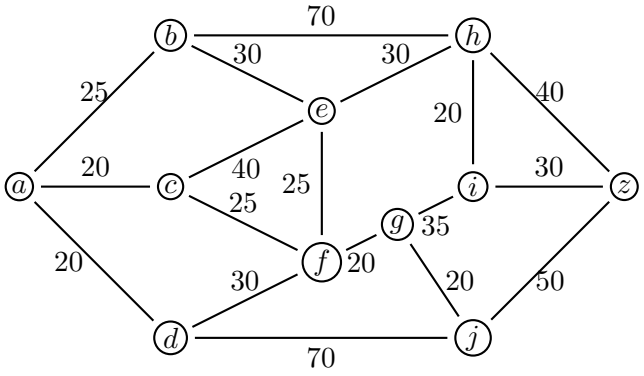
a)



b)



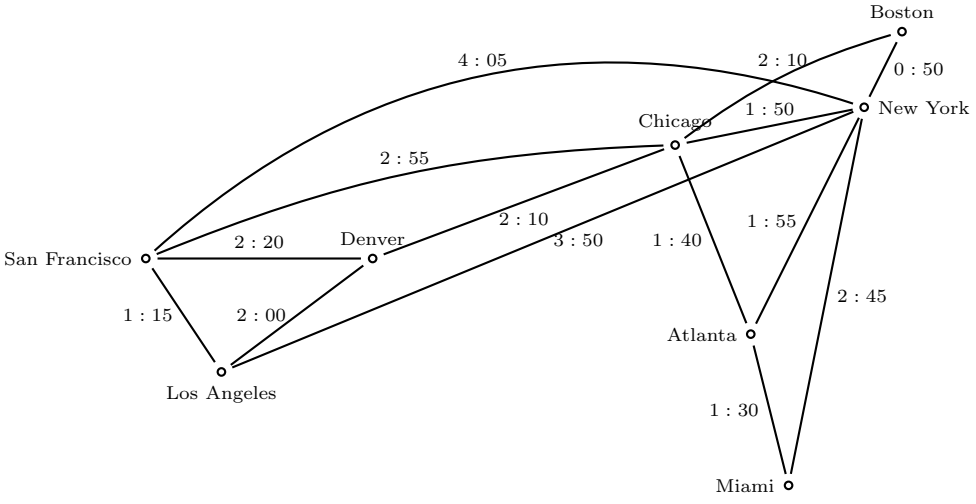
c)



16. Determine um caminho de menor custo, e o seu custo, entre os vértices 5 e 8 do grafo definido pela seguinte matriz de custos nas arestas:

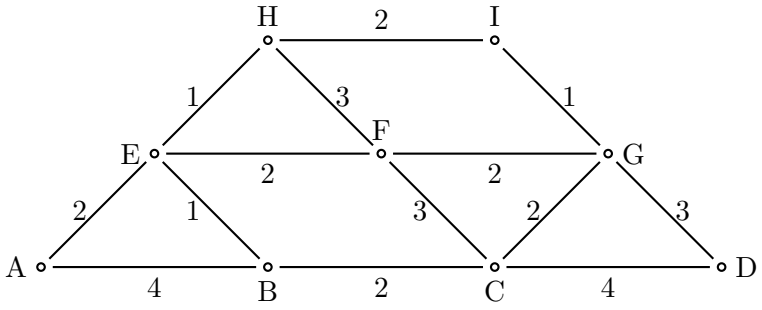
$$M = \begin{pmatrix} 0 & 12 & \infty & \infty & 12 & \infty & \infty & \infty \\ 12 & 0 & 13 & \infty & 12 & 14 & 15 & \infty \\ \infty & 13 & 0 & 13 & \infty & \infty & 11 & 15 \\ \infty & \infty & 13 & 0 & \infty & \infty & \infty & 11 \\ 12 & 12 & \infty & \infty & 0 & 15 & \infty & \infty \\ \infty & 14 & \infty & \infty & 15 & 0 & 13 & \infty \\ \infty & 15 & 11 & \infty & \infty & 13 & 0 & 12 \\ \infty & \infty & 15 & 11 & \infty & \infty & 12 & 0 \end{pmatrix}.$$

17. O grafo seguinte modela um sistema de linhas aéreas nos Estados Unidos da América, onde a etiqueta de cada aresta representa o tempo de voo entre dois aeroportos.
    a) Qual o percurso mais rápido, em termos de tempo de voo, entre os aeroportos de San Francisco e Miami? Qual o tempo total de voo desse percurso?
    b) Qual o percurso mais rápido e tempo de voo entre os mesmos aeroportos, mas de modo a fazermos escala em New York?

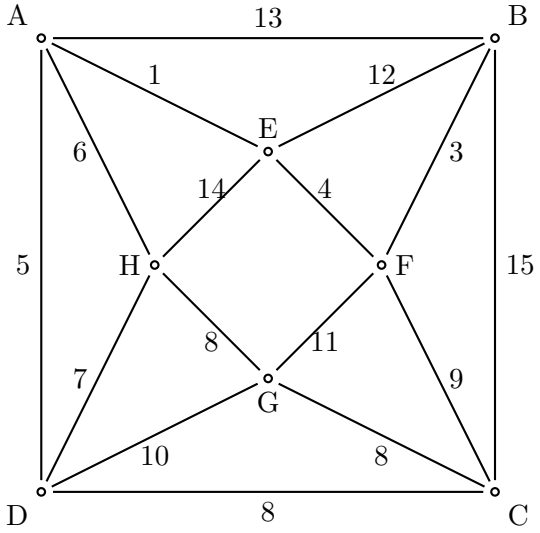


18. Determine uma árvore geradora de custo mínimo de cada um dos grafos seguintes:

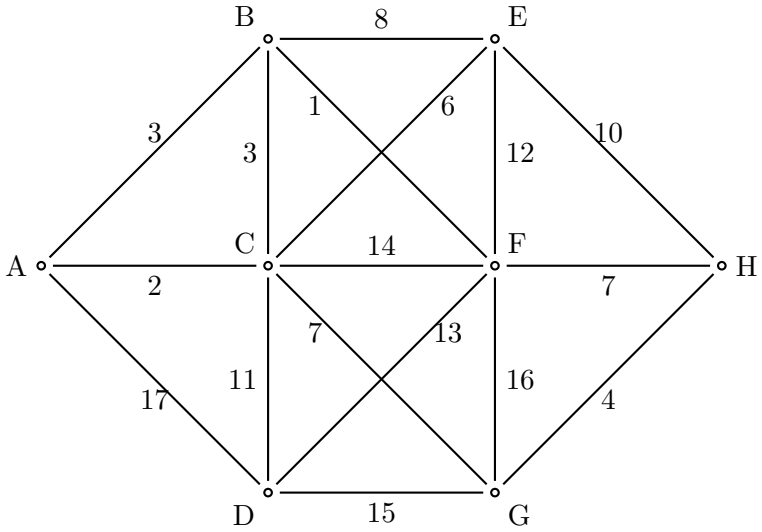
a)



b)

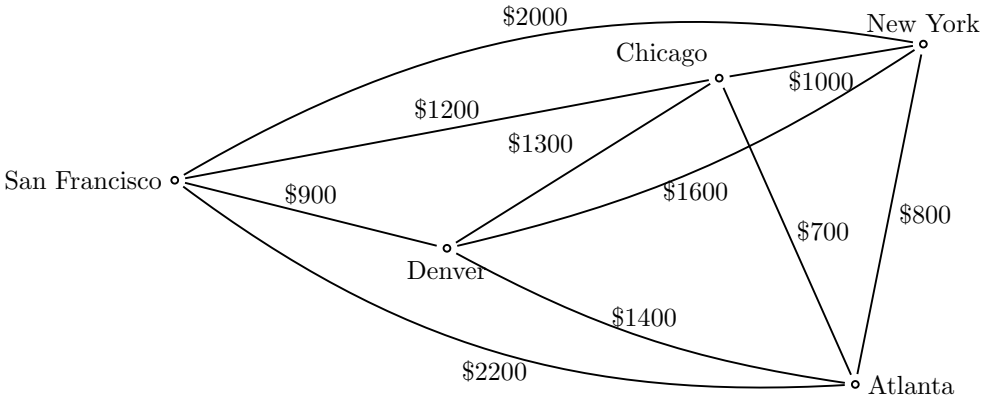


c)

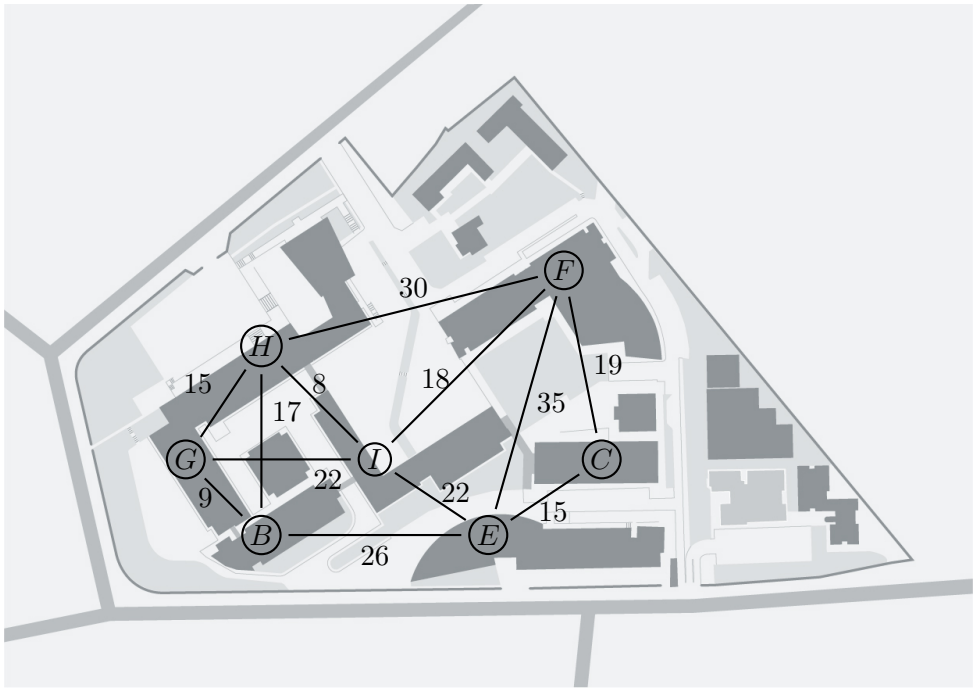


19. O grafo seguinte modela uma rede de computadores nos Estados Unidos da América, onde a etiqueta de cada aresta representa o custo mensal de arrendamento da linha

telefónica. Quais as ligações que se devem manter de modo a assegurar a ligação entre dois quaisquer computadores e tal que o custo total de arrendamento de linhas seja mínimo?



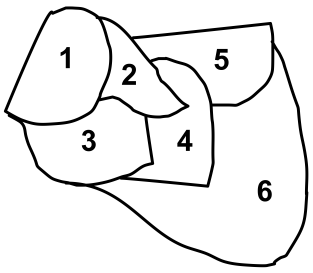
20. O ISEP pretende instalar novas canalizações de gás natural para fornecer os sete edifícios assinalados na figura[5], onde estão indicadas as distâncias, em metros, entre os edifícios. Usando o Algoritmo de Kruskal, determine a solução que utiliza menor canalização e assegura a ligação entre quaisquer dois edifícios.



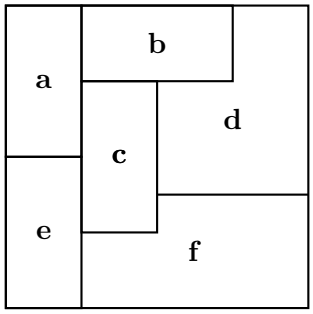
[5]Imagem autorizada e cedida pelo ISEP - Instituto Superior de Engenharia do Porto.

21. Use o Algoritmo de Welsh-Powell para colorir os grafos seguintes:

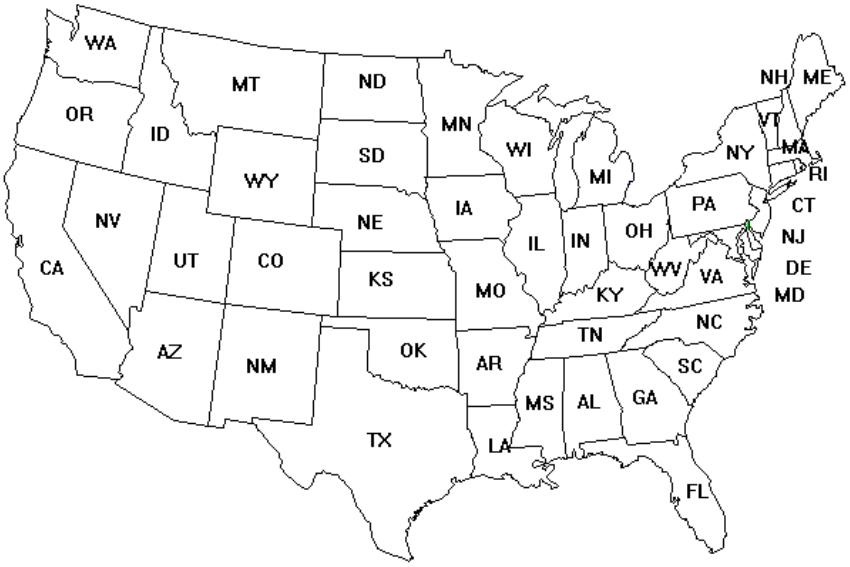
a)



b)



c)

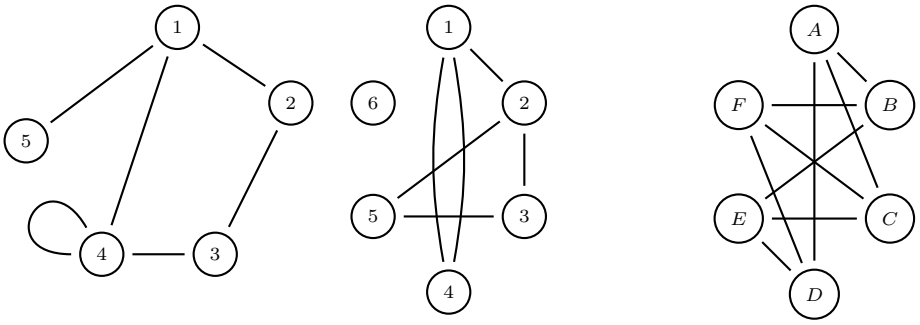


22. Pretende-se fazer o agendamento dos exames das seguintes unidades curriculares de Engenharia Informática: AMATA, ALGAN, MATCP, MDISC, ESOFT, PPROG, LPROG, APROG. Sabendo que não há alunos que estão inscritos simultaneamente nos pares de exames seguintes: AMATA e APROG, ALGAN e APROG, MDISC e ESOFT, MDISC e PPROG, AMATA e ALGAN, AMATA e MATCP, MATCP e MDISC, sugira um agendamento que use o menor número de dias e de modo a que cada aluno não tenha de realizar mais do que um exame em cada dia.

Soluções dos exercícios propostos

1.

a)



b)

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

c)

$$\begin{array}{lll} G_1: & grau(1)=3 & grau(2)=2 \quad grau(3)=2 \\ & grau(4)=4 & grau(5)=1 \\ G_2: & grau(1)=3 & grau(2)=3 \quad grau(3)=2 \\ & grau(4)=2 & grau(5)=2 \quad grau(6)=0 \\ G_3: & grau(A)=3 & grau(B)=3 \quad grau(C)=3 \\ & grau(D)=3 & grau(E)=3 \quad grau(F)=3 \end{array}$$

d) G_1: 2 passeios; G_2: 0 passeios; G_3: 0 passeios.

e) G_3.

f) G_1 e G_3.

2.

a)

b)

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

c)

G_1 : $grau^e(1) = 3$ $grau^e(2) = 0$ $grau^e(3) = 2$ $grau^e(4) = 1$
$grau^s(1) = 0$ $grau^s(2) = 3$ $grau^s(3) = 1$ $grau^s(4) = 2$

G_2 : $grau(A)^e = 1$ $grau(B)^e = 2$ $grau(C)^e = 2$ $grau^e(D) = 2$ $grau^e(E) = 3$
$grau^s(A) = 1$ $grau^s(B) = 2$ $grau^s(C) = 3$ $grau^s(D) = 1$ $grau^s(E) = 3$

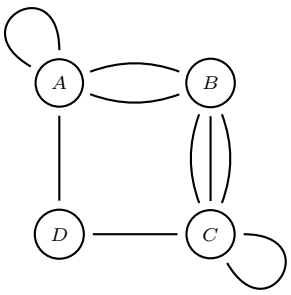
d) G_1: 1 caminho; G_2: 1 caminho.

e) Verdadeira.

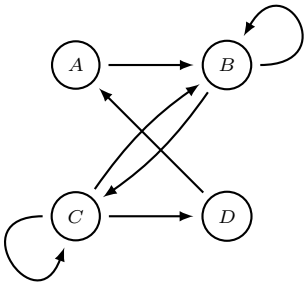
f) (A, B, C, D, E, A) é um passeio fechado que passa por todos os vértices.

3. Não, a soma dos graus dos vértices tem de ser par.

4.



a)



b)

c) Não, pois não é simétrica.

d) Sim, (A, B, C, D, A) é um passeio fechado que passa por todos os vértices.

5. a) Sim.
b) Não.
c) Sim.
d) Não.

6. G_1 e G_4 não podem ser isomorfos a G_2 ou G_3: têm um vértice de grau 5 e os segundos não.

G_1 não pode ser isomorfo a G_4: no primeiro, o vértice de grau 5 é adjacente a um vértice de grau 2; no segundo, ele é adjacente a um de grau 3.

G_2 não pode ser isomorfo a G_3: no primeiro, o vértice de grau 4 é adjacente a dois vértices de grau 1; no segundo, ele é adjacente a três de grau 1.

7. a) (d, d, a, b, c, d, a, b)

   b) (a, b, c, d, a) e todos os que resultam deste por escolha de outro vértice de partida ou mudança de sentido.

   c) (b, c, a, d); 3.

8. $(1, 4, 5, 8)$, $(1, 4, 5, 7, 8)$, $(1, 4, 5, 6, 7, 8)$, $(1, 2, 4, 5, 8)$, $(1, 2, 4, 5, 7, 8)$, $(1, 2, 4, 5, 6, 7, 8)$, $(1, 2, 3, 4, 5, 8)$, $(1, 2, 3, 4, 5, 7, 8)$ e $(1, 2, 3, 4, 5, 6, 7, 8)$.

9. a) Não, o grafo não é conexo. b) Não. c) Sim. d) Sim.

10. a)
$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

    b) 10

    c) (A, A, B, D, C), $(A, B, B, D, C,)$, (A, E, B, D, C), (A, E, C, D, C), (A, A, E, D, C), (A, E, E, D, C), (A, A, A, E, C), (A, E, A, E, C), (A, A, E, E, C) e (A, E, E, E, C).

11. a)
$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

    b) 4: (D, E, G, G), (D, E, F, G), (D, F, G, G) e (D, F, E, G).

    c) (Passeio,3º), (Trajeto,4º), (Trajeto fechado,1º), (Caminho simples,5º) e (Ciclo,2º).

    d) Não, a aresta (C, D) é uma aresta de corte.

12. a) Não é Euleriano nem semi-Euleriano.

    b) É semi-Euleriano: $(a, e, c, e, b, e, d, b, a, c, d)$.

    c) É Euleriano: $(a, b, c, d, c, e, d, b, e, a, e, a)$.

    d) Não é Euleriano nem semi-Euleriano.

    e) É Euleriano: $(a, i, h, gd, e, f, g, c, e, h, d, c, a, b, i, c, b, h, a)$.

13. Não, existem duas regiões com um número ímpar de pontes.

14. Não, o grafo não é Euleriano nem semi-Euleriano.

15. a) (a, b, f, z); 8.

    b) (a, c, d, e, g, z); 16.

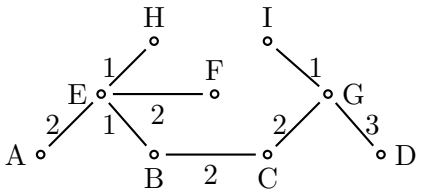
    c) (a, b, e, h, z); 125.

16. $(5, 2, 7, 8)$; 39.

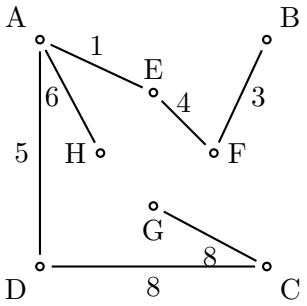
17. a) (San Francisco,Chicago,Atlanta,Miami); 6 : 05.

    b) (San Francisco,New York,Miami); 6 : 50.

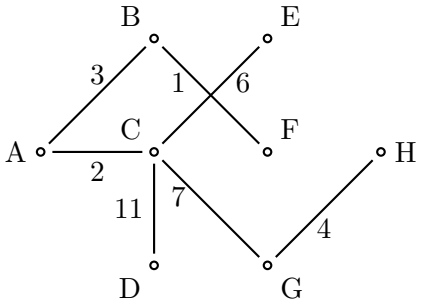
18. a) Por exemplo,



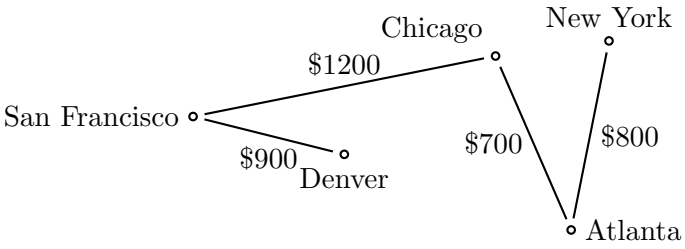
    b)



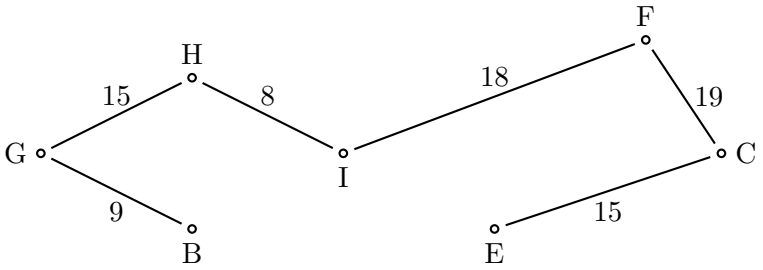
    c)



19.



20.



22. Dia 1: LPROG; Dia 2: PPROG e MDISC; Dia 3: ESOFT; Dia 4: MATCP e AMATA; Dia 5: ALGAN e APROG.

APÊNDICE A

Breve revisão do Cálculo Combinatório

Qual é a probabilidade de, na tua turma de Matemática Discreta, existirem dois alunos com o mesmo dia de aniversário?

Quantas chaves existem no euromilhões?

Quantas passwords de duas letras, maiúsculas ou minúsculas, seguidas de quatro dígitos todos diferentes podes formar?

O Cálculo Combinatório é a área da Matemática Discreta que se dedica à contagem do número de elementos de um conjunto. À partida, pode parecer uma tarefa simples, mas, se quisermos contar, por exemplo, o número de chaves do euromilhões enumerando-as uma a uma, este processo de contagem será bastante moroso. A contagem de elementos é necessária para resolver vários e diferentes tipos de problemas na Matemática e na Informática. Como exemplo, na determinação da complexidade de algoritmos, abordada no Capítulo 3.

Neste capítulo, fazemos referência aos métodos de contagem fundamentais: princípios da adição, de inclusão-exclusão e da multiplicação, permutações, arranjos e combinações. Optou-se por introduzir cada método recorrendo a exemplos.

1. Princípios básicos de contagem

Princípio da adição. Eis a ementa do restaurante *O Grande Repasto*:

Sopas
Caldo Verde
Juliana de Legumes

Entradas
Alheira de caça
Salada de polvo
Moelas

Pratos de Peixe
Bacalhau à Lagareiro
Dourada grelhada
Espetada de polvo
Lulas grelhadas

Pratos de Carne
Posta à Mirandesa
Tripas à moda do Porto
Vitela assada na brasa

Sobremesa
Foundant de chocolate
Romeu e Julieta
Gelados variados

Se quisermos escolher um prato principal, de peixe ou de carne, quantas possibilidades temos de escolha?

Consideremos os conjuntos

$$\text{Peixe} = \{B, D, E, L\} \text{ e Carne} = \{P, T, V\}.$$

Pretendemos escolher um elemento de um deles. Como não há elementos comuns, basta adicionar o número de elementos de cada um dos conjuntos para determinar o número total de possibilidades, $4 + 3$. Este é o primeiro lema de enumeração:

LEMA A.1. *Se a escolha num conjunto A puder ser feita de m maneiras distintas e a escolha de um conjunto B puder ser feita de n maneiras distintas e os conjuntos não tiverem elementos comuns, então o número possível de escolhas no conjunto A ou no conjunto B é $m + n$.*

Representamos por $|A|$ o cardinal ou número de elementos do conjunto A. Assim, se A e B forem dois conjuntos disjuntos (sem elementos comuns), escrevemos o lema anterior da forma seguinte:

$$|A \cup B| = |A| + |B|.$$

Este lema pode ser estendido a um número finito de conjuntos finitos, dois a dois disjuntos, obtendo-se o primeiro princípio básico de enumeração:

PROPOSIÇÃO A.2 (Princípio da Adição). *Sejam $A_1, A_2, \ldots, A_n$ conjuntos finitos, dois a dois disjuntos (ou seja, tais que $A_i \cap A_j = \emptyset$, para $i \neq j$). Então*

$$|\bigcup_{i=1}^{n} A_i| = \sum_{i=1}^{n} |A_i|.$$

EXEMPLO A.1. Consideremos o algoritmo seguinte e determinemos o valor de retorno k.

```
procedure valor( )
    k := 0
    for i_1 := 1 to n_1
        k := k + 1
    for i_2 := 1 to n_2
        k := k + 1
    ⋮
    for i_m := 1 to n_m
        k := k + 1
    return k
```

O valor de k inicia em 0. Seguidamente, o algoritmo percorre m ciclos for distintos. Em cada ciclo j, o valor de k é incrementado de 1 unidade n_j vezes. Logo o valor final de k é $n_1 + n_2 + \cdots + n_m$.

Princípio de inclusão-exclusão. E no caso dos conjuntos não serem disjuntos?
Consideremos agora os conjuntos

$$\text{Primos} = \{2, 3, 5, 7, 11, 13\} \text{ e Pares} = \{2, 4, 6, 8, 10, 12\}.$$

Quantas maneiras há de se escolher um inteiro entre os elementos dos dois conjuntos? Como os conjuntos não são disjuntos, ao fazermos a soma do número de elementos dos dois conjuntos, $6 + 6$, estaremos a contar duas vezes o inteiro 2. Assim, teremos que subtrair esta dupla contagem, como é referido no princípo que se segue.

LEMA A.3. *Se a escolha num conjunto A puder ser feita de m maneiras distintas e a escolha de um conjunto B puder ser feita de n maneiras distintas e os conjuntos tiverem l elementos comuns, então o número possível de escolhas no conjunto A ou no conjunto B é $m + n - l$.*

De outro modo, o número possível de escolhas no conjunto A ou no conjunto B é:

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Estendemos este lema a um número finito de conjuntos finitos da forma seguinte:

PROPOSIÇÃO A.4 (Princípio de Inclusão-Exclusão). *Sejam $A_1, A_2, \ldots, A_n$ conjuntos finitos arbitrários. Então*

$$|\bigcup_{1\leq i<j\leq n} A_i| = \sum_{1\leq i\leq n} |A_i| - \sum_{1\leq i<j\leq n} |A_i \cap A_j| + \sum_{1\leq i<j<k\leq n} |A_i \cap A_j \cap A_k| - \cdots + (-1)^{n-1}|A_1 \cap A_2 \cap \cdots \cap A_n|.$$

Princípio da multiplicação. O restaurante *O Grande Repasto* faz um menu a um preço especial, constituído por uma sopa ou entrada, um prato de peixe ou de carne e uma sobremesa. De quantas maneiras diferentes podemos escolher o menu?

Consideremos os conjuntos

$$\text{Sopa} = \{C, J\},\ \text{Entrada} = \{A, S, M\}$$

$$\text{Peixe} = \{B, D, E, L\},\ \text{Carne} = \{P, T, V\} \text{ e Sobremesa} = \{F, R, G\}.$$

Podemos elaborar o menu com um elemento de cada um dos conjuntos seguintes: Sopa $\cup$ Entrada $= \{C, J, A, S, M\}$, Peixe $\cup$ Carne $= \{B, D, E, L, P, T, V\}$, e Sobremesa $= \{F, R, G\}$.

Suponhamos que escolhemos Caldo Verde (C) para iniciar a refeição. Temos, em seguida, 7 hipóteses de escolher o prato principal. A cada uma dessas hipóteses podemos associar uma das 3 sobremesas. Assim, existem 7×3 refeições diferentes que começam com C. No total, há $5 \times 7 \times 3 = 105$ formas diferentes de elaborar o menu.

Usando notação matemática, o conjunto Menu tem a seguinte representação:

$$\text{Menu} = \{(x, y, z) : x \in \text{Sopa} \cup \text{Entrada}, y \in \text{Peixe} \cup \text{Carne}, z \in \text{Sobremesa}\}$$

e o seu cardinal é dado por:

$$|\text{Menu}| = |\text{Sopa} \cup \text{Entrada}| \times |\text{Peixe} \cup \text{Carne}| \times |\text{Sobremesa}|.$$

Estamos perante o Princípio da Multiplicação.

PROPOSIÇÃO A.5 (Princípio da Multiplicação). *Sejam $A_1, A_2, \ldots, A_n$ conjuntos não-vazios finitos. Então o conjunto dos n-uplos $(a_1, a_2, \ldots, a_n)$, onde $a_i \in A_i$, $i = 1, 2, \ldots, n$, e que se denota por $A_1 \times A_2 \times \cdots \times A_n$ é tal que*

$$|A_1 \times A_2 \times \cdots \times A_n| = |A_1| \times |A_2| \times \cdots \times |A_n|.$$

EXEMPLO A.2. Qual é o valor de k que o algoritmo retorna?

```
procedure valor( )
    k := 0
    for i_1 := 1 to n_1
       for i_2 := 1 to n_2
          ⋮
          for i_m := 1 to n_m
             k := k + 1
    return k
```

O valor de k inicia em 0. Seguidamente, o algoritmo percorre m ciclos for imbricados. O ciclo 1 é percorrido n_1 vezes. De cada vez que é percorrido, o ciclo 2 é percorrido n_2 vezes, ou seja, num total de $n_1 \cdot n_2$ vezes. Seguindo o mesmo raciocínio, o ciclo j é percorrido $n_1 \cdot n_2 \cdots n_j$ vezes. Logo o valor final de k é $n_1 \cdot n_2 \cdots n_m$.

EXEMPLO A.3. Quantas *passwords* de 6 a 8 carateres, onde cada carater é uma letra minúscula ou um dígito, e que tenha pelo menos um dígito, se podem formar?

Sejam P_i, $i = 6, 7, 8$, o número de *passwords* de comprimento i e $P = P_6 + P_7 + P_8$. Para determinar o valor de cada P_i, calculamos o número de *passwords* de comprimento i onde cada carater é uma letra minúscula ou um dígito e subtraímos o número de *passwords* que não possuem dígitos. Pelo Princípio da Multiplicação, temos:

$$\begin{aligned} P_6 &= 36^6 - 26^6 = 1\,867\,866\,560, \\ P_7 &= 36^7 - 26^7 = 70\,332\,353\,920, \\ P_8 &= 36^8 - 26^8 = 2\,612\,282\,842\,880 \end{aligned}$$

e, portanto, $P = 2\,684\,483\,063\,360$.

Pode acontecer que o os elementos de um conjunto A_i, $i > 1$, dependam das escolhas feitas nos conjuntos anteriores. Para isso, generalizamos o Princípio da Multiplicação, como se segue:

PROPOSIÇÃO A.6 (Princípio da Multiplicação Generalizada). *Se uma tarefa envolve n etapas onde a primeira tem r_1 maneiras distintas de ser realizada, a segunda tem r_2 formas distintas de ser realizada, ..., e a n-ésima tem r_n formas distintas de ser realizada, então há $r_1 \cdot r_2 \cdots r_n$ formas de realizar a tarefa.*

EXEMPLO A.4. Quantos números existem com 4 algarismos todos distintos e pertencentes ao conjunto $\{1, 2, \ldots, 9\}$?

Este é um exemplo em que temos de usar o Princípio da Multiplição Generalizada. Queremos preencher os 4 dígitos do número, _ _ _ _. Para o primeiro dígito, podemos escolher um elemento de $A_1 = \{1, 2, \ldots, 9\}$. No entanto, não podemos explicitar os conjuntos A_2, A_3 e A_4, para cada um dos dígitos seguintes. Não obstante, sabemos o cardinal que cada um destes conjuntos tem, que é, respetivamente, 8, 7 e 6. Obtemos então o resultado

$$9 \times 8 \times 7 \times 6 = 3\,024$$

para o número pretendido.

Exemplo A.5. Quantas *strings* de *bits* de comprimento 8 que começam com 1 ou terminam com 00 existem?

Ora, o número de *strings* de comprimento 8 que começam em 1 é $2^7 = 128$, pelo Princípio da Multiplicação, visto que o primeiro dígito tem de ser 1 e os restantes podem ser escolhidos num conjunto de 2 elementos. Pelo mesmo racíocínio, o número de *strings* de comprimento 8 que terminam em 00 é $2^6 = 64$. No entanto, as *strings* de comprimento 8 que começam em 1 e terminam em 00 estão a ser contadas 2 vezes e o número delas é $2^5 = 32$. Assim, pelo Princípio de Inclusão-Exclusão, obtemos $128 + 64 - 32 = 160$ para o número de *strings* pedido.

Exemplo A.6. Um estudante da LEI pretende fazer uma planificação para um período de 7 dias no qual estudará uma UC por dia. As UC que pretende estudar são: MDISC, MATCP, ESOFT e PPROG. Quantas planificações diferentes existem que garantam que pelo menos um dia é dedicado a cada UC?

Facilmente se conclui que o número de diferentes planificações é 4^7. Teremos de excluir aquelas que não contemplam pelo menos uma UC em pelo menos um dia.

Sejam A_1, A_2, A_3 e A_4 os conjuntos de planificações que não contemplam o estudo de MDISC, MATCP, ESOFT e PPROG, respetivamente. Então

$$A_1 \cup A_2 \cup A_3 \cup A_4$$

é o conjunto de planificações em que pelo menos uma UC não é contemplada. Pelo Princípio de Inclusão-Exclusão, temos

$$|A_1 \cup A_2 \cup A_3 \cup A_4| = \sum_{1 \leq i \leq 4} |A_i| - \sum_{1 \leq i < j \leq 4} |A_i \cap A_j| + \sum_{1 \leq i < j < k \leq 4} |A_i \cap A_j \cap A_k| - |A_1 \cap A_2 \cap A_3 \cap A_4|.$$

Note-se que os somatórios da expressão têm, respetivamente, 4, 6 e 4 parcelas. O valor de cada parcela é obtido pelo Princípio da Multiplicação:

$$\begin{aligned} &|A_i| = 3^7, \text{ para todo o } i, \\ &|A_i \cap A_j| = 2^7, \text{ para todos os } i, j, \\ &|A_i \cap A_j \cap A_k| = 1^7 = 1, \text{ para todos os } i, j, k, \\ &|A_1 \cap A_2 \cap A_3 \cap A_4| = 0 \end{aligned}$$

e, portanto,

$$|A_1 \cup A_2 \cup A_3 \cup A_4| = 4 \times 3^7 + 6 \times 2^7 + 4 \times 1 + 0.$$

Resulta que o número de planificações diferentes que garantem que pelo menos um dia é dedicado a cada UC é

$$4^7 - (4 \times 3^7 - 6 \times 2^7 + 4 \times 1 - 0) = 8\,400.$$

2. Permutações, arranjos e combinações

Arranjos e permutações. Os alunos de uma turma de Matemática Discreta, que são 25, vão fazer uma corrida de 10 km. Existem prémios para os primeiros três classificados. De quantas maneiras diferentes pode ser constituído o pódio?
Existem três posições diferentes para preencher, _ _ _. Para a primeira temos 25 alunos possíveis.

Depois de se escolher o primeiro classificado, restam 24 possibilidades para o segundo classificado, restando 23 para o terceiro lugar. Há, portanto, $25 \times 24 \times 23 = 13\,800$ maneiras diferentes de preencher o pódio.

Usando notação matemática, sendo Alunos o conjunto dos 25 alunos da turma, o conjunto dos diferentes pódios escreve-se da forma:

$$\text{Podios} = \{(x, y, z) : x, y, z \in \text{Alunos} \wedge x, y, z \text{ são distintos entre si}\}.$$

Cada elemento do conjunto Podios denomina-se de arranjo dos 25 elementos de Alunos, organizados 3 a 3.

DEFINIÇÃO A.1 (Arranjo). Dado um conjunto finito, chama-se *arranjo* a qualquer sequência constituída por elementos todos distintos desse conjunto (ou seja, onde não figuram elementos repetidos).

O número de arranjos que se podem formar com p elementos de um conjunto com n elementos denota-se por

$$A_p^n$$

e lê-se *"número de arranjos de n elementos p a p"*.

Na notação fatorial, o número de arranjos de n elementos p a p é dado por

$$A_p^n = \frac{n!}{(n-p)!}.$$

E se quiséssemos agora apresentar a ordenação da chegada destes alunos à meta? De quantas maneiras diferentes estes alunos podem ser ordenados?
Basta seguir o raciocínio anterior, considerando agora as 25 posições. Obteríamos $25 \times 24 \times \cdots \times 2 \times 1 = 15\,511\,210\,043\,330\,985\,984\,000\,000$ ordenações possíveis!

DEFINIÇÃO A.2 (Permutação). Uma *permutação* é um arranjo de todos os elementos de um dado conjunto finito. A ordem porque aparecem os elementos é importante.

O número de permutações que se podem formar com os elementos de um conjunto com n elementos é

$$n! = n \times (n-1) \times \cdots \times 1.$$

EXEMPLO A.7. Queremos alinhar estes 25 alunos de modo a que os 3 alunos do pódio fiquem em posições consecutivas. De quantas formas diferentes o podemos fazer?

Comecemos por substituir estes 3 alunos do pódio por um único aluno fictício e trabalhemos então com 23 alunos (os 22 que não pertencem ao pódio e o fictício). Ora, existem 23! alinhamentos diferentes. Agora, como podemos alinhar os 3 alunos que constituem o aluno fictício? Podemos fazê-lo de 3! formas diferentes. Pelo Princípio da Multiplicação, resulta que temos $3! \cdot 23!$ alinhamentos diferentes.

Combinações. Serão escolhidos 5 alunos de uma turma de Matemática Discreta para fazerem uma apresentação do que aprenderam ao longo da unidade curricular aos restantes colegas. De quantas maneiras diferentes podemos escolher os alunos?
Neste caso, não é importante a ordem por que os elementos são escolhidos, mas unicamente

o subconjunto de alunos que é escolhido. Chama-se a esta escolha uma combinação dos 25 elementos, 5 a 5.

Definição A.3 (Combinação). Dado um conjunto finito, chama-se *combinação* a uma coleção de elementos desse conjunto para o qual não interessa a ordem. Note-se que uma combinação não é mais do que um subconjunto de um conjunto inicial.

O número de combinações que se podem formar com p elementos de um conjunto com n elementos denota-se por

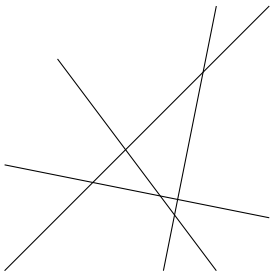
$$C_p^n \text{ ou } \binom{n}{p}$$

e lê-se *"número de combinações de n elementos p a p"*.

Na notação fatorial é dado por

$$C_p^n = \frac{n!}{p! \cdot (n-p)!}.$$

Exemplo A.8. Se desenharmos n retas no plano, de modo a que não haja um par de retas paralelas e sem que três retas se intersetem num único ponto, quantos pontos são definidos pelas interseções dessas retas?



Como cada ponto é determinado pela interseção de duas retas e a interseção de duas retas define um único ponto, o número de pontos definidos pelas interseções é igual ao número de subconjuntos de duas retas do conjunto das retas desenhadas, ou seja,

$$\binom{n}{2} = \frac{n(n-1)}{2}.$$

Arranjos com repetição. Em todas as aulas de Matemática Discreta, a professora pede a um dos seus 25 alunos para resolver um exercício no quadro. Ocorreram 22 aulas da disciplina. De quantas formas diferentes a professora pode escolher os alunos?
É fácil perceber que, em cada aula, a professora teve 25 escolhas diferentes, escolhas essas que foram independentes das restantes aulas. Pelo Princípio da Multiplicação, houve 25^{22} possibilidades.

Definição A.4. Um *arranjo com repetição* é uma sequência de elementos de um conjunto, podendo existir repetição de elementos.

O número de arranjos de n elementos p a p é

$$n^p.$$

Combinações com repetição. Pretendemos colocar p bolas iguais em n caixas distintas. De quantas formas diferentes o podemos fazer?
Representemos cada bola com o símbolo 0 e separemos as diferentes caixas, ordenadas, pelo símbolo 1, existindo $n-1$ separadores. Por exemplo, para 10 bolas e 5 caixas podemos ter a distribuição seguinte:

$$\underbrace{}_{caixa1}\ 1\ \underbrace{000}_{caixa2}\ 1\ \underbrace{000}_{caixa3}\ 1\ \underbrace{00}_{caixa4}\ 1\ \underbrace{00}_{caixa5}$$

Assim, facilmente se concluiu que o número de distribuições diferentes das p bolas pelas n caixas é igual ao número de *strings* binárias de comprimento $p+n-1$ e com p zeros. Ora, isto é equivalente a escolher, de entre $p+n-1$ posições, p posições para colocar os zeros, ou seja, $\binom{n+p-1}{p}$.

Definição A.5. Uma *combinação com repetição* é uma coleção de elementos de um conjunto, podendo existir repetição de elementos.

O número de combinações com repetição de n elementos p a p é

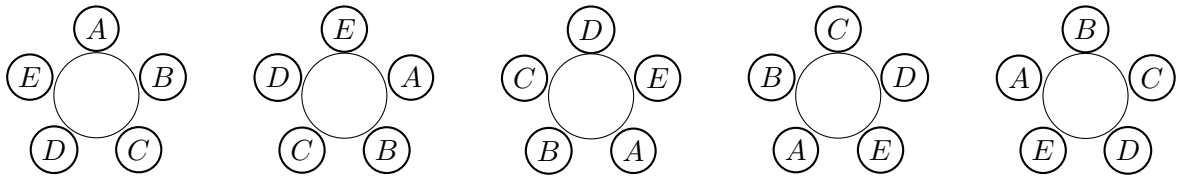
$$\binom{n+p-1}{p}.$$

Exemplo A.9. Qual o número de possibilidades de colocar 20 bolas em 5 caixas, garantindo que cada caixa tem pelo menos 2 bolas?

Para satisfazer a exigência de existirem 2 bolas em cada caixa, podemos, à partida, usar 10 das 20 bolas iniciais e distribuir 2 por caixa. O problema resume-se agora a distribuir as 10 sobrantes de qualquer forma pelas 5 caixas, existindo $\binom{14}{10} = 1\,001$ possibilidades.

Arranjos circulares. Quando abordamos os arranjos, demos como exemplo a ordenação de n indivíduos em fila. E se quiséssemos sentar estes n indivíduos numa mesa redonda, quantas organizações diferentes existem?

Seja, por exemplo, $n=5$. Então os arranjos $ABCDE$, $BCDEA$, $CDEAB$, $DEABC$ e $EABCD$ conduzem à mesma organização da mesa, como podemos observar na figura seguinte:



Portanto, a cada arranjo circular corresponde um grupo de $n=5$ arranjos.

E se a mesa só tivesse 3 lugares, de quantas formas diferentes podemos sentar 3 das 5 pessoas?

O raciocínio é idêntico: há A_3^5 formas de alinhar 3 das 5 pessoas. Mas cada grupo de 3 que diferem de uma rotação dos elementos dá origem a uma única organização da mesa. Logo há $A_3^5/3$ organizações de mesa diferentes.

Definição A.6. Um *arranjo circular* é uma organização de elementos em círculo, para a qual interessa a ordem.

O número de arranjos circulares de p elementos de um conjunto de n elementos é

$$\frac{A_p^n}{p}.$$

Exercícios propostos

1. No bar do ISEP há 8 tipos diferentes de bebidas sem gás e 4 tipos diferentes de bebidas com gás. O Joaquim pede uma bebida com gás e outra sem gás. De quantas formas diferentes ele pode fazer a escolha?

2. Quantos pares distintos de números inteiros $p, q \in \{1, 2, 3, \ldots, n\}$ existem com $p \leq q$?

3. Quantas são as chaves possíveis do totobola? (As apostas são para 13 jogos.)

4. No ISEP, os docentes são identificados por uma sigla com três letras (alfabeto de 26 letras). Quantas siglas é possível formar?

5. A empresa JOGOS tem um anúncio luminoso com as letras do seu nome. São usadas cores para as letras podendo aparecer cores repetidas (por exemplo, (amarelo, verde, azul, amarelo, vermelho)). A empresa gostaria de usar um esquema de cores diferente para cada um dos seus 365 dias do ano. Qual o número mínimo de cores que deverá usar para garantir tal exigência.

6. No Campeonato Nacional de Futebol participam 18 equipas, devendo cada uma delas jogar com cada uma das restantes duas vezes (como visitante e como visitado).
   a) Qual é o número máximo de partidas que se podem jogar em cada jornada?
   b) Qual é o número total de jornadas, admitindo que em cada jornada se joga o número máximo de partidas?

7. Na sala de aula de Matemática Discreta existem 5 lugares na primeira fila. A Maria, a Leonor, o Manuel, o Francisco e o Rodrigo querem sentar-se nessa primeira fila. De quantas maneiras diferentes é possível sentá-los?

8. Quantas matrículas de carro da forma $XX - YY - XX$, onde X é um algarismo entre 0 e 9, e Y é uma das 23 letras do alfabeto, é possível fazer em Portugal?

9. De quantas maneiras diferentes é possível pintar bandeiras com duas cores diferentes, escolhidas as cores entre 9 cores possíveis?

10. Quantos números inteiros formados por algarismos todos diferentes são maiores do que 2000 e menores que 10000?

11. A professora de Matemática Discreta vai dar boleia a 3 dos seus 19 alunos. De quantas formas diferentes ela pode escolher esse grupo de alunos?

12. Quantas são as chaves possíveis do euromilhões? (5 números de 1 a 50, 2 estrelas de 1 a 11)

13. Um aluno tem 80 livros. De quantas formas pode encher uma estante na qual cabem apenas 10 livros,

a) sem termos em consideração a ordenação dos livros?

b) tendo em consideração a ordenação dos livros?

14. Num saco há 6 rebuçados de 6 sabores diferentes.
    a) De quantas maneiras diferentes se podem escolher dois sabores diferentes?
    b) Quantos sabores são necessários para durante um ano inteiro se conseguir comer sempre 2 combinações de sabores diferentes?

15. Quantos triângulos se podem construir usando os vértices de um polígono regular com n lados?

16. O Francisco acordou antes de amanhecer e estava sem luz em casa. Ele guarda as meias numa gaveta sem juntá-las aos pares. Sabendo que ele tem 6 meias azuis, 8 meias brancas, 4 meias verdes e 8 meias pretas, qual o número mínimo de meias que ele deverá tirar de modo a garantir um par de meias da mesma cor.

17. Num grupo existem n homens e n mulheres. De quantas maneiras diferentes podemos ordená-los de modo que homens e mulheres alternem?

18. Numa festa todas as pessoas cumprimentaram-se com um aperto de mão. Sabendo que foram dados 78 apertos de mão, quantas pessoas estavam na festa?

19. Sabendo que $a = 3^5 \cdot 5^2 \cdot 7 \cdot 13^3 \cdot 17^2$, $b = 10a$ e $c = 3^4 \cdot 7^4 \cdot 13 \cdot 19$, calcule:
    a) O número de divisores de a e o número de divisores de b.
    b) O número de divisores comuns de a e c.

20. Quantas palavras (com ou sem significado) constituídas por 5 letras têm exatamente 2 vogais? (Considere um alfabeto com 25 letras e 5 vogais).

21. Num jogo de póquer, qual a probabilidade de sair um *full house* (três cartas do mesmo valor facial e duas do mesmo valor facial)? (Considere um baralho com 52 cartas.)

22. Qual a probabilidade de numa reunião de 25 pessoas existirem pelo menos duas que façam anos no mesmo dia? (Considere um ano com 365 dias.)

23. Quantas palavras se podem formar, reordenando todas as letras da palavra

$$PIPAPAPIPAPAPIGRAFO\ ?$$

24. No *jogo de pares* dispõem-se as cartas do primeiro baralho numa linha e por cima de cada carta coloca-se uma carta de um segundo baralho, ambos os baralhos com 52 cartas. Atribui-se 1 ponto sempre que duas cartas coincidam.
    a) Qual a probabilidade de se obter a pontuação máxima?
    b) Qual a probabilidade de se obter 0 pontos?
    c) Qual a probabilidade de se obter uma pontuação não inferior a 2 pontos?

Soluções dos exercícios propostos

1. $8 \times 4 = 32$
2. $\frac{n(n+1)}{2}$
3. $3^{13} = 1\,594\,323$
4. $26^3 = 17\,576$
5. 4
6. a) 9
   b) 34
7. $5! = 120$
8. $10^4 \times 23^2 = 5\,290\,000$
9. $A_2^9 = 72$
10. 4032
11. $C_3^{19} = 969$
12. $C_5^{50} \times C_2^{11} = 116\,531\,800$
13. a) $\frac{80!}{70!\,10!}$ b) $\frac{80!}{70!}$
14. a) $C_2^6 = 15$
    b) 28
15. $\frac{n!}{(n-3)!\,3!}$
16. 5
17. $2(n!)^2$
18. 13
19. a) 432 divisores de a; 1152 divisores de b.
    b) 20 divisores comuns de a e c.
20. 2000000
21. $0,144\%$
22. $56,9\%$
23. $\frac{19!}{7!\,5!\,3!}$
24. a) $\frac{1}{52!}$
    b) $\frac{1}{2!} - \frac{1}{3!} + \cdots + \frac{1}{52!}$
    c) $1 - \frac{2}{2!} + \frac{2}{3!} + \cdots + \frac{2}{51!} - \frac{1}{52!}$

APÊNDICE B

Relações de recorrência

Um professor americano escreveu um artigo em Inglês e pretendeu publicá-lo numa revista francesa. Como ele não sabe Francês, pediu ao seu colega francês, Professor Bestougeff, para traduzir o documento. Quando a tradução foi concluída, o professor americano entendeu que deveria incluir uma nota de agradecimento no jornal para reconhecer a contribuição do seu colega. Ele escreveu a nota "O autor agradece ao Professor Bestougeff pela tradução do seu artigo para Francês"em Inglês e pediu ao Professor Bestougeff para traduzi-la para Francês, o que o professor fez de bom grado. Em seguida, ocorreu ao professor americano que também deveria agradecer a tradução desta nota. Então ele escreveu outra nota, "O autor agradece ao Professor Bestougeff a tradução da nota anterior para Francês", pediu ao Professor Bestougeff para traduzi-la para Francês e copiou a tradução francesa duas vezes como duas notas de rodapé adicionais. O seu problema foi assim completamente resolvido!

em C. L. Liu, *Elements of Discrete Mathematics*.

Uma *relação de recorrência* (ou *recursiva*) para uma sequência $(a_n)_{n\in\mathbb{N}}$ é uma fórmula que expressa a_n em termos de um ou mais termos de ordem inferior da sequência, i.e., a_{n-1}, a_{n-2}, ..., a_{n-k}, com $n \geq k \geq n_0$, para $n_0 \in \mathbb{N}_0$.

De um modo informal, uma resolução recursiva de um problema consiste em expressar a solução de um problema de certa ordem à custa da solução de um ou mais problemas idênticos mas de ordem inferior.

Resolver uma relação de recorrência consiste em determinar uma fórmula não recursiva para a_n.

Exemplo B.1. Foram investidos 1 000 euros num depósito a prazo com um juro composto anual de 5%. Seja P_n o montante dessa conta ao fim de n anos. Ora, este montante não é mais do que o montante ao fim de $n-1$ anos, P_{n-1}, adicionado do juro de 5% obtido no final desse ano, ou seja,

$$P_n = P_{n-1} + 0,05P_{n-1} = 1,05P_{n-1},$$

com a condição inicial $P_0 = 1\,000$. Temos então P_n definido recursivamente à custa do termo anterior.

Facilmente se obtém uma fórmula não recursiva para P_n, usando um processo iterativo:

$$P_n = 1,05P_{n-1} = (1,05)^2P_{n-2} = (1,05)^3P_{n-3} = \cdots = (1,05)^nP_0 = (1,05)^n1\,000.$$

Se pretendermos determinar o montante dessa conta ao final de 10 anos, bastará substituir n por 10 na fórmula fechada:

$$P_{10} = (1,05)^{10}1\,000 \approx 1\,629 \text{ euros}.$$

No exemplo anterior, determinou-se facilmente uma fórmula fechada solução da relação de recorrência. No entanto, a dedução de tais fórmulas nem sempre é uma tarefa fácil.

Em seguida, apresentamos duas classes de relações de recorrência que ocorrem na modelação de problemas e que possuem métodos sistemáticos de resolução.

1. Relações de recorrência lineares homogéneas do 1º e 2º grau

Uma *relação de recorrência linear homogénea de grau k com coeficientes constantes* é uma relação de recorrência da forma

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k},$$

onde $c_1, c_2, \ldots, c_k \in \mathbb{R}$ e $c_k \neq 0$.

A relação diz-se *linear* pois o membro da direita é uma soma de múltiplos (*constantes*) de termos de ordem inferior da sequência. Diz-se *homogénea* por não existirem termos que não são múltiplos de termos da forma a_j. Por fim, é de *grau k* pois é expressa à custa dos k termos anteriores da sequência.

Na Secção 2.2, pudemos observar que uma sequência definida por uma relação de recorrência de grau k é unicamente determinada por essa relação e por k condições iniciais, $a_0 = C_0, a_1 = C_1, \ldots, a_{k-1} = C_{k-1}$.

Do Exemplo B.1, é fácil concluir a proposição que se segue, que fornece a solução de uma relação de recorrência linear homogénea do 1º grau.

PROPOSIÇÃO B.1. *Seja $a_n = ca_{n-1}$ uma relação de recorrência para a sequência $(a_n)_{n\in\mathbb{N}_0}$, com $c \in \mathbb{R}$. Então $a_n = \alpha c^n$ é uma solução da relação, onde α é determinado pela condição inicial a_0.*

DEMONSTRAÇÃO. O resultado demonstra-se facilmente pelo método de indução. Para o passo indutivo, assumimos que a solução se verifica para $n-1$. Mostremos que se verifica também para n. De facto, tem-se, pela relação de recorrência,

$$a_n = ca_{n-1} = c\alpha c^{n-1} = \alpha c^n$$

usando a hipótese de indução.

Finalmente, para determinar o valor de α, faz-se $n = 0$

$$a_0 = \alpha c^0 = \alpha$$

e obtém-se a solução $a_n = a_0 c^n$ da relação de recorrência. □

Vejamos agora como obter a solução de uma relação de recorrência linear homogénea do 2º grau. Define-se a *equação característica* corresponde à relação de recorrência

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

como sendo a equação da forma

$$x^2 - c_1 x - c_2 = 0.$$

A solução da relação de recorrência depende do tipo de soluções da equação característica: raízes distintas (Teorema B.2) ou raízes iguais (Teorema B.3). Notemos que as raízes de uma equação

característica correspondente a uma relação de recorrência do 2º grau são não nulas (pois $c_2 \neq 0$) e não são necessariamente reais (pelo Teorema Fundamental da Álgebra, um polinómio de grau n de coeficientes complexos tem exatamente n raízes complexas não necessariamente distintas).

TEOREMA B.2. *Sejam $c_1, c_2 \in \mathbb{R}$ tais que a equação $x^2 - c_1 x - c_2 = 0$ tem duas raízes distintas, r_1 e r_2. Então a relação de recorrência $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ tem como solução $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$, onde α_1 e α_2 são determinados pelas condições iniciais a_0 e a_1.*

DEMONSTRAÇÃO. Procedemos por indução completa, onde os casos $n = 0$ e $n = 1$ serão validados no final da demonstração, pelo cálculo de α_1 e α_2. Assumimos que o resultado se verifica para $n-1$ e $n-2$. Mostramos que se verifica para n. Com efeito,

$$\begin{aligned} a_n &= c_1 a_{n-1} + c_2 a_{n-2} \\ &= c_1(\alpha_1 r_1^{n-1} + \alpha_2 r_2^{n-1}) + c_2(\alpha_1 r_1^{n-2} + \alpha_2 r_2^{n-2}) \\ &= \alpha_1 r_1^{n-2}(c_1 r_1 + c_2) + \alpha_2 r_2^{n-2}(c_1 r_2 + c_2). \end{aligned}$$

Por serem raízes da equação $x^2 - c_1 x - c_2 = 0$, tem-se $c_1 r_1 + c_2 = r_1^2$ e $c_1 r_2 + c_2 = r_2^2$. Logo

$$a_n = \alpha_1 r_1^{n-2} r_1^2 + \alpha_2 r_2^{n-2} r_2^2 = \alpha_1 r_1^n + \alpha_2 r_2^n.$$

Fazendo $n = 0$ e $n = 1$ na solução geral obtida, temos

$$\begin{cases} \alpha_1 + \alpha_2 = a_0 \\ \alpha_1 r_1 + \alpha_2 r_2 = a_1 \end{cases} \Leftrightarrow \begin{cases} \alpha_2 = a_0 - \alpha_1 \\ \alpha_1 r_1 + (a_0 - \alpha_1) r_2 = a_1 \end{cases} \Leftrightarrow \begin{cases} \alpha_2 = a_0 - \alpha_1 \\ \alpha_1 (r_1 - r_2) = a_1 - a_0 r_2 \end{cases} \Leftrightarrow \begin{cases} \alpha_2 = a_0 - \alpha_1 \\ \alpha_1 = \frac{a_1 - a_0 r_2}{r_1 - r_2} \quad \text{pois } r_1 \neq r_2 \end{cases} \Leftrightarrow \begin{cases} \alpha_2 = a_0 - \frac{a_1 - a_0 r_2}{r_1 - r_2} = \frac{a_0 r_1 - a_1}{r_1 - r_2} \\ \alpha_1 = \frac{a_1 - a_0 r_2}{r_1 - r_2} \end{cases}$$

Assim, para os valores de α_1 e α_2 determinados, a expressão $\alpha_1 r_1^n + \alpha_2 r_2^n$ é solução para a relação de recorrência. □

EXEMPLO B.2. Determinemos a solução da relação de recorrência

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

com $a_0 = 2$ e $a_1 = 7$.

A equação $x^2 - x - 2 = 0$ tem duas raízes distintas, $r_1 = 2$ e $r_2 = -1$. Assim, pelo Teorema B.2, temos a solução da relação de recorrência dada por

$$a_n = \alpha_1 2^n + \alpha_2 (-1)^n,$$

onde α_1 e α_2 são determinados pelas condições iniciais

$$\begin{cases} 2 = \alpha_1 2^0 + \alpha_2(-1)^0 \\ 7 = \alpha_1 2^1 + \alpha_2(-1)^1 \end{cases} \Leftrightarrow \begin{cases} 2 = \alpha_1 + \alpha_2 \\ 7 = 2\alpha_1 - \alpha_2 \end{cases} \Leftrightarrow \begin{cases} \alpha_1 = 2 - \alpha_2 \\ 7 = 4 - 2\alpha_2 - \alpha_2 \end{cases} \Leftrightarrow \begin{cases} \alpha_1 = 3 \\ \alpha_2 = -1 \end{cases}.$$

Logo $a_n = 3 \cdot 2^n - (-1)^n$ é solução da relação de recorrência.

TEOREMA B.3. *Sejam $c_1, c_2 \in \mathbb{R}$ com $c_2 \neq 0$ e tais que a equação $x^2 - c_1 x - c_2 = 0$ tem uma única raiz não nula, r. Então a relação de recorrência $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ tem como solução $a_n = \alpha_1 r^n + \alpha_2 n r^n$, onde α_1 e α_2 são determinados pelas condições iniciais a_0 e a_1.*

DEMONSTRAÇÃO. Procedemos por indução completa e assumimos que o resultado se verifica para $n-1$ e $n-2$. Mostramos que se verifica para n. Com efeito,

$$\begin{aligned} a_n &= c_1 a_{n-1} + c_2 a_{n-2} \\ &= c_1[\alpha_1 r^{n-1} + \alpha_2(n-1)r^{n-1}] + c_2[\alpha_1 r^{n-2} + \alpha_2(n-2)r^{n-2}] \\ &= \alpha_1 r^{n-2}(c_1 r + c_2) + \alpha_2 r^{n-2}[c_1(n-1)r + c_2(n-2)]. \end{aligned}$$

Por r ser raiz da equação $x^2 - c_1 x - c_2 = 0$, tem-se $c_1 r + c_2 = r^2$. Mais ainda, por r ser a única raiz da equação $x^2 - c_1 x - c_2 = 0$, tem-se

$$x^2 - c_1 x - c_2 = (x - r)^2 = x^2 - 2rx + r^2$$

e, portanto, $c_1 = 2r$ e $c_2 = -r^2$. Logo

$$\begin{aligned} a_n &= \alpha_1 r^{n-2} r^2 + \alpha_2 r^{n-2}[2r(n-1)r - r^2(n-2)] \\ &= \alpha_1 r^n + \alpha_2 r^{n-2} r^2 [2(n-1) - (n-2)] \\ &= \alpha_1 r^n + \alpha_2 n r^n. \end{aligned}$$

Fazendo $n = 0$ e $n = 1$ na solução geral obtida, temos

$$\begin{cases} \alpha_1 = a_0 \\ \alpha_1 r + \alpha_2 r = a_1 \end{cases} \Leftrightarrow \begin{cases} \alpha_1 = a_0 \\ \alpha_2 = \frac{a_1 - a_0 r}{r} \end{cases}.$$

Assim, para os valores de α_1 e α_2 determinados, a expressão $\alpha_1 r^n + \alpha_2 n r^n$ é solução para a relação de recorrência. □

EXEMPLO B.3. Determinemos a solução da relação de recorrência

$$a_n = 6a_{n-1} - 9a_{n-2}$$

com $a_0 = 1$ e $a_1 = 6$.

A equação característica $x^2 - 6x + 9 = 0$ tem uma raiz dupla $r = 3$. Pelo Teorema B.3, temos a solução da relação de recorrência da forma

$$a_n = \alpha_1 3^n + \alpha_2 n 3^n$$

onde α_1 e α_2 determinam-se à custa das condições iniciais

$$\begin{cases} \alpha_1 = 1 \\ 3\alpha_1 + 3\alpha_2 = 6 \end{cases} \Leftrightarrow \begin{cases} \alpha_1 = 1 \\ \alpha_2 = 1 \end{cases}.$$

Resulta que $a_n = 3^n + n3^n = (n+1)3^n$ é solução da relação de recorrência.

2. Algumas relações de recorrência lineares não-homogéneas do 1º e 2º grau

Uma *relação de recorrência linear não-homogénea de grau k com coeficientes constantes* é uma relação de recorrência da forma

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + F(n),$$

onde $c_1, c_2, \ldots, c_k \in \mathbb{R}$ e $c_k \neq 0$.

Abordaremos unicamente o caso em que $F(n)$ é constante. As proposições seguintes fornecem a solução de uma relação de recorrência do 1º grau, $a_n = ca_{n-1} + k$, respetivamente, quando $c \neq 1$ e $c = 1$.

Proposição B.4. *Seja $a_n = ca_{n-1} + k$ uma relação de recorrência, com $c \neq 1$. Então $a_n = \alpha c^n + \frac{k}{1-c}$ é uma solução da relação, onde α é determinado pela condição inicial a_0.*

Demonstração. Provamos, mais uma vez, usando o método de indução. Assumindo que a fórmula se verifica para $n-1$, temos

$$\begin{aligned} a_n &= ca_{n-1} + k \\ &= c(\alpha c^{n-1} + \tfrac{k}{1-c}) + k \\ &= \alpha c^n + k(\tfrac{c}{1-c} + 1) \\ &= \alpha c^n + \tfrac{k}{1-c}. \end{aligned}$$

O valor de α determina-se a partir da condição inicial a_0. Fazendo $n = 0$, temos

$$a_0 = \alpha + \frac{k}{1-c} \Leftrightarrow \alpha = a_0 - \frac{k}{1-c}.$$

□

Proposição B.5. *A solução da relação de recorrência $a_n = a_{n-1} + k$ é $a_n = \alpha + nk$, onde α é determinado pela condição inicial a_0.*

Demonstração. Estamos perante uma progressão aritmética. Suponhamos que o resultado se verifica para $n-1$. Então

$$a_n = a_{n-1} + k = \alpha + (n-1)k + k = \alpha + nk.$$

O valor de α obtém-se da condição inicial, $a_0 = \alpha$. □

No próximo teorema, obtemos uma solução para uma certa classe de relações de recorrência lineares não-homogéneas do 2º grau.

TEOREMA B.6. *Sejam $c_1, c_2 \in \mathbb{R}$ com $c_1 + c_2 \neq 1$ e tais que a equação característica $x^2 - c_1 x - c_2 = 0$ tem duas raízes distintas, r_1 e r_2 (não necessariamente reais). Então a relação de recorrência $a_n = c_1 a_{n-1} + c_2 a_{n-2} + k$ tem como solução $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n - \frac{k}{c_1+c_2-1}$, onde α_1 e α_2 são determinados pelas condições iniciais a_0 e a_1.*

DEMONSTRAÇÃO. Procedendo por indução completa, suponhamos que é verdade para $n-1$ e $n-2$. Então

$$\begin{aligned} a_n &= c_1 a_{n-1} + c_2 a_{n-2} + k \\ &= c_1(\alpha_1 r_1^{n-1} + \alpha_2 r_2^{n-1} - \tfrac{k}{c_1+c_2-1}) + c_2(\alpha_1 r_1^{n-2} + \alpha_2 r_2^{n-2} - \tfrac{k}{c_1+c_2-1}) + k \\ &= \alpha_1 r_1^{n-2}(c_1 r_1 + c_2) + \alpha_2 r_2^{n-2}(c_1 r_2 + c_2) + k - \tfrac{k(c_1+c_2)}{c_1+c_2-1}) \\ &= \alpha_1 r_1^n + \alpha_2 r_2^n - \tfrac{k}{c_1+c_2-1} \end{aligned}$$

pois r_1 e r_2 são soluções da equação característica.

Falta unicamente determinar os valores de α_1 e α_2 usando as condições iniciais a_0 e a_1. Para simplificar as expressões, façamos $b = -\frac{k}{c_1+c_2-1}$. Substituindo n por 0 e por 1, temos

$$\begin{cases} a_0 = \alpha_1 + \alpha_2 + b \\ a_1 = \alpha_1 r_1 + \alpha_2 r_2 + b \end{cases} \Leftrightarrow \begin{cases} \alpha_1 = a_0 - \alpha_2 - b \\ a_1 = (a_0 - \alpha_2 - b) r_1 + \alpha_2 r_2 + b \end{cases}$$

$$\Leftrightarrow \begin{cases} \alpha_1 = a_0 - \alpha_2 - b \\ \alpha_2(r_1 - r_2) = a_0 r_1 - b r_1 - a_1 + b \end{cases}$$

$$\Leftrightarrow \begin{cases} \alpha_1 = a_0 - \frac{a_0 r_1 - b r_1 - a_1 + b}{r_1 - r_2} - b \\ \alpha_2 = \frac{a_0 r_1 - b r_1 - a_1 + b}{r_1 - r_2} \end{cases}$$

$$\Leftrightarrow \begin{cases} \alpha_1 = \frac{a_1 - a_0 r_2 + b(r_2 - 1)}{r_1 - r_2} \\ \alpha_2 = \frac{a_0 r_1 - a_1 + b(1 - r_1)}{r_1 - r_2} \end{cases}.$$

□

EXEMPLO B.4. Determinemos a solução da relação de recorrência

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

com $a_0 = 1$ e $a_1 = 1$.

A equação característica $x^2 - x - 2 = 0$ tem duas raízes distintas, $r_1 = 2$ e $r_2 = -1$. Como $c_1 + c_2 \neq 1$, podemos usar o Teorema B.6 para encontrar a solução. Assim,

$$a_n = \alpha_1 2^n + \alpha_2 (-1)^n - \frac{1}{2}$$

é solução da relação, onde α_1 e α_2 satisfazem

$$\begin{cases} 1 = \alpha_1 + \alpha_2 - \frac{1}{2} \\ 1 = 2\alpha_1 - \alpha_2 - \frac{1}{2} \end{cases} \Leftrightarrow \begin{cases} 2 = 3\alpha_1 - 1 \\ 1 = 2\alpha_1 - \alpha_2 - \frac{1}{2} \end{cases} \Leftrightarrow \begin{cases} \alpha_1 = 1 \\ \alpha_2 = \frac{1}{2} \end{cases}.$$

Resulta que

$$a_n = 2^n + \frac{1}{2}(-1)^n - \frac{1}{2}$$

é solução da relação de recorrência.

Exercícios propostos

1. Suponhamos que o número de bactérias numa colónia triplica em cada hora.
   a) Escreva uma relação de recorrência para o número de bactérias numa colónia ao final de n horas.
   b) São usadas 100 bactérias para iniciar uma nova colónia. Quantas bactérias existirão nessa colónia após 10 horas?

2. Mostre que as sequências a_n seguintes são solução da relação de recorrência $a_n = a_{n-1} + 2a_{n-2} + 2n - 9$.
   a) $a_n = -n + 2$
   b) $a_n = 5(-1)^n - n + 2$
   c) $a_n = 3(-1)^n + 2^n - n + 2$
   d) $a_n = 7 \cdot 2^n - n + 2$

3. Resolva cada uma das relações de recorrência seguintes, para as condições iniciais dadas.
   a) $a_n = a_{n-1} + 6a_{n-2}$, para $n \geq 2$, $a_0 = 3$ e $a_1 = 6$.
   b) $a_n = 7a_{n-1} - 10a_{n-2}$, para $n \geq 2$, $a_0 = 2$ e $a_1 = 1$.
   c) $a_n = 4a_{n-1} - 4a_{n-2}$, para $n \geq 2$, $a_0 = 6$ e $a_1 = 8$.
   d) $a_n = 7a_{n-1} - 10a_{n-2} + 2$, para $n \geq 2$, $a_0 = 4$ e $a_1 = 1$.
   e) $a_{n+2} = -4a_{n+1} + 5a_n$, para $n \geq 0$, $a_0 = 2$ e $a_1 = 8$.

Soluções dos exercícios propostos

1. a) $a_n = 3a_{n-1}$
   b) $a_{10} = 100 \cdot 3^{10} = 5\,904\,900$

3. a) $a_n = \frac{3}{5}(-2)^n + \frac{12}{5}3^n$
   b) $a_n = 3 \cdot 2^n - 5^n$
   c) $a_n = 6 \cdot 2^n - 2n \cdot 2^n$
   d) $a_n = 5 \cdot 2^n - 2 \cdot 5^n + 1$
   e) $a_n = 3 - (-5)^n$

Bibliografia

[1] Agnarsson, G., & Greenlaw, G. (2006). *Graph Theory: Modeling, Applications, and Algorithms*. Pearson Education.

[2] Alves de Sá, A., & et al (2012). *Introdução ao Cálculo*. Escolar Editora.

[3] Belcastro, S.-M. (2018). *Discrete Mathematics with Ducks*. Chapman and Hall/CRC.

[4] Biggs, N. L. (1993). *Discrete Mathematics*. Oxford University Press.

[5] Constantinides, G. (2006). *Lecture Notes in Discrete Mathematics and Computational Complexity*. Imperial College. http://cas.ee.ic.ac.uk/people/gac1/Complexity/.

[6] Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2022). *Introduction to Algorithms*. MIT Press.

[7] Liu, C. L. (1985). *Elements of Discrete Mathematics*. McGraw-Hill.

[8] Loura, L. & Martins, M. (2003). *Cálculo Combinatório*. Departamento de Estatística e Investigação Operacional, Faculdade de Ciências da Universidade de Lisboa.

[9] Moreira Cardoso, D., Szymanski, J., & Rostami, M. (2009). *Matemática Discreta*. Escolar Editora.

[10] Rosen, K. H. (2019). *Discrete Mathematics and Its Applications*. McGraw-Hill.

[11] Ross, K. A., & Wright, C. R. B. (1999). *Discrete Mathematics*. Prentice-Hall.

[12] Simões Pereira, J. M. S. (2013). *Introdução à Matemática Combinatória*. Interciência.

[13] Sullivan, B. W. (2013). *Everything You Always Wanted To Know About Mathematics*. Institution Carnegie Mellon University Pittsburgh.