Probabilidade com Algoritmos

e vice-versa

Uma introdução à probabilidade discreta e aos algoritmos probabilísticos

— Yair Donadelli —

última modificação 21/8/2021

[*Rosencrantz and Guildenstern are riding horses down a path - they pause*]
R: Umm, uh...
[*Guildenstern rides away, and Rosencrantz follows. Rosencrantz spots a gold coin on the ground*]
R: Whoa - whoa, whoa.
[*Gets off horse and starts flipping the coin*] **R**: Hmmm. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads. Heads.
[*Guildenstern grabs the coin, checks both sides, then tosses it back to Rosencrantz*]
R: Heads.
[*Guildenstern pulls a coin out of his own pocket and flips it*]
R: Bet? Heads I win?
[*Guildenstern looks at coin and tosses it to Rosencrantz*]
R: Again? Heads.

[. . .]

R: Heads
G: A weaker man might be moved to re-examine his faith, if in nothing else at least in the law of probability.
R: Heads
G: Consider. One, probability is a factor which operates within natural forces. Two, probability is not operating as a factor. Three, we are now held within um...sub or supernatural forces. Discuss!
R: What?
[. . .]
R: Heads, getting a bit of a bore, isn't it?

[. . .]

R: 78 in a row. A new record, I imagine.
G: Is that what you imagine? A new record?
R: Well...
G: No questions? Not a flicker of doubt?
R: I could be wrong.
G: No fear?
R: Fear?
G: Fear!
R: Seventy nine.

[. . .]

G: I don't suppose either of us was more than a couple of gold pieces up or down. I hope that doesn't sound surprising because its very unsurprisingness is something I am trying to keep hold of. The equanimity of your average tosser of coins depends upon a law, or rather a tendency, or let us say a probability, or at any rate a mathematically calculable chance, which ensures that he will not upset himself by losing too much nor upset his opponent by winning too often. This made for a kind of harmony and a kind of confidence. It related the fortuitous and the ordained into a reassuring union which we recognized as nature. The sun came up about as often as it went down, in the long run, and a coin showed heads about as often as it showed tails.

Tom Stoppard, *Rosenkrantz and Guildenstern are dead* (1996).

Sumário

INTRODUÇÃO

Algoritmo é um conceito antigo, o algoritmo de Euclides tem mais de dois mil anos, porém uma formalização satisfatória do que é um algoritmo só se deu por volta de 1936 com os trabalhos de alguns matemáticos, notadamente o célebre Alan Turing. Mais recente é o uso de aleatoriedade em algoritmos e foram introduzidos na Computação na década de 70 com o objetivo de projetar algoritmos eficientes e que tomam decisões com base no resultado do cara ou coroa. No século 18, Buffon usou um método aleatorizado para determinar uma aproximação para π. Em um artigo de 1917, Henry Pocklington apresentou um algoritmo para encontrar $\sqrt{a} \bmod p$ que dependia de escolher números ao acaso. Em 1953 foi publicado um método para se obter uma sequência de amostras aleatórias de acordo com uma distribuição de probabilidade para a qual a amostragem direta é difícil, o conhecido algoritmo Metropolis–Hastings. O estudo moderno e sistematizado de modelos probabilísticos de computação começou por volta de 1976 com Gary Miller, Michael Rabin, Robert Solovay e Volker Strassen que escreveram algoritmos aleatorizados muito eficientes para testar a primalidade de um número. Também por volta de 1976, John Gill estendeu a definição de Máquina de Turing para incluir bits aleatórios e definiu classes de complexidade computacional para essas máquinas.

Por volta dos anos de 1990 a Computação vivenciou um rápido crescimento no uso de algoritmos aleatorizados, ou probabilísticos, em problemas de várias áreas distintas. Duas características marcantes nesses algoritmos e que favoreceram muito esse crescimento foram a simplicidade e eficiência. Em muitos casos é o algoritmo mais simples disponível, ou o mais rápido, ou ambos. Existem, ainda, alguns problemas interessantes para os quais existe um algoritmo probabilístico e eficiente mas não conseguimos encontrar um algoritmo determinístico eficiente. Atualmente são conhecidas soluções aleatorizadas para problemas que não admitem solução determinística, como compartilhamento de recursos em sistemas distribuídos. A aleatorização é figura central em alguma áreas como a criptografia de chave pública.

Um punhado de princípios gerais está no centro do projeto de quase todos os algoritmos aleatorizados, apesar do grande variedade de áreas nas quais eles encontram aplicação. Uma delas, para dar um exemplo, é conhecida na literatura como *fingerprinting*. De modo abstrato, temos um conjunto universo U muito grande e queremos decidir se os objetos x e y desse universo, cujas representações têm tamanho da ordem de $\log(|U|)$, são o mesmo objeto. A estratégia probabilística para solução é criar uma função aleatorizada $f\colon U \to V$ com $|V|$ muito menor que $|U|$ e declarar $x = y$ se $f(x) = f(y)$.

Deve estar claro que $f(x) = f(y)$ é verificada mais rapidamente que $x = y$ e que essa igualdade não significa que $x = y$, mas precisamos que $\mathbb{P}[\, f(x) = f(y) \mid x \neq y \,]$ seja limitado por uma constante pequena.

Uma característica marcante dos algoritmos aleatorizados é que duas execuções do mesmo algoritmo com a mesma entrada podem não produzir a mesma resposta ou não executarem o mesmo número de instruções. Para um algoritmo aleatório ser interessante ou não está correto o tempo todo, ou nem sempre é executado no tempo esperado.

Algoritmos Monte Carlo: Algoritmos aleatorizados com tempo de execução que depende somente da entrada e não dos sorteios e que podem responder errado, são conhecidos na literatura como algoritmos *Monte Carlo*. O teste de primalidade (aleatorizado) de Miller–Rabin é um algoritmo que recebe um inteiro positivo e responde *primo* ou *composto*. A resposta *composto* sempre é certa. A resposta *primo*, por outro lado, pode estar errada, mas a probabilidade desse erro é controlada. É possível, ainda, tornar a probabilidade da resposta errada arbitrariamente pequena às custas do aumento no tempo de execução, a probabilidade de erro é tão pequena e o algoritmo tão eficiente que mesmo após a descoberta de um algoritmo determinístico de tempo polinomial para o problema 30 anos depois[1], esse teste desenvolvido na década de 1970 continua sendo amplamente utilizado. A complexidade de pior caso para decidir primalidade de n (que tem $\log n$ algarismos, aproximadamente) em 10 repetições é $O((\log n)^3)$ e a probabilidade de erro é menor que $9{,}5 \times 10^{-7}$.

Para um base de comparação, sem a intenção de sermos rigorosos, *hardware* apresentam falhas no processamento, transmissão e armazenamento de informação. Um tipo de erro é o da inversão de bit, quando um bit 0 vira 1, ou 1 vira 0, por motivo de falha. Há vários estudos que medem a taxa de erro devido as várias causas possíveis como, por exemplo, os raios cósmicos que têm energia suficiente para alterar os estados dos componentes de circuitos integrados causando erros transitórios e aleatórios. Estudos da IBM na década de 1990 sugerem que os computadores normalmente experimentam um erro induzido por raios cósmicos por 256 megabytes de RAM por mês (O'Gorman et al., 1996). Isso significa uma probabilidade de erro de 1.4×10^{-15} por byte por segundo. Estudos mais recentes nos centros de dados da Google, publicados em 2009, mostram uma taxa de erro dentre 1 erro em 1,4 anos e 1 erro em 950 anos por megabit de DRAM, o que numa máquina de de 4GiB de DRAM significa 3 erros por mês com a taxa mais baixa ou 2000 erros por mês com a taxa mais alta (Schroeder, Pinheiro e Weber, 2009). Em 2011, um estudo da Microsoft em grande escala em computadores que rodam o Windows da Microsoft conclui que, por exemplo, máquinas com pelo menos 5 dias de tempo de CPU tem probabilidade 1/2.700 de falha na DRAM (Nightingale, Douceur e Orgovan, 2011).

[1] Agrawal, Kayal e Saxena publicaram em 2004 um algoritmo determinístico com complexidade de pior caso $O(\log^{12} n)$ Agrawal, Kayal e Saxena, 2004.

Algoritmos Las Vegas: Algoritmos aleatorizados que têm probabilidade pequena de executarem por muito tempo, ou seja, a complexidade de tempo depende dos sorteios feitos durante a execução, são conhecidos na literatura como algoritmos Las Vegas. A complexidade é uma variável aleatória e, idealmente, o algoritmo é eficiente em média com alta probabilidade. Em geral, nos algoritmos Las Vegas, a resposta sempre está correta. O *Quicksort* aleatorizado é um algoritmo que com probabilidade maior que $1 - 1/n^{-6}$ ordena n elementos em tempo $O(n \log n)$, mas sempre responde corretamente.

Soluções eficientes: Um dos atrativos dos algoritmos aleatorizados reside em que vários problemas computacionais admitem algoritmo aleatorizado simples e eficiente enquanto que algoritmos determinísticos de tempo subexponencial não são conhecidos. Por exemplo, o problema

Problema do teste de identidade entre polinômios.	
Instância	: Um circuito aritmético que calcula um polinômio p com n variáveis, grau $\leqslant d$ com coeficientes num corpo $\mathbb{F}$.
Resposta	: Decidir se $p \equiv 0$.

em que $p \equiv 0$ significa que p é idênticamente nulo. Esse é o caso quando precisamos decidir se

$$\sum_{\sigma} \operatorname{sinal}\left(\prod_{1 \leqslant i < j \leqslant n} (\sigma(j) - \sigma(i))\right) \prod_{i=1}^{n} x_i^{\sigma(i)-1} \equiv \prod_{1 \leqslant i < j \leqslant n} (x_j - x_i) \,? \tag{1}$$

com a soma sobre toda permutação σ de $\{1, 2, \ldots, n\}$. A identidade da equação (1) é verdadeira, o lado esquerdo é o determinante da matriz de Vandermonde. Esse é um problema para o qual não se conhece algoritmo determinístico de tempo subexponencial no tamanho da entrada mas tem uma solução probabilística cuja probabilidade de erro é no máximo $(1/3)^k$ se repetirmos k vezes o algoritmo; a repetição 15 vezes resulta em chance de erro de $0,00000007$.

Pak provou que um problema computacional da teoria dos grupos pode ser resolvido por um algoritmo probabilístico de tempo polinomial, mas que qualquer algoritmo determinístico para esse problema é no mínimo quadrático.

Desaleatorização: Diminuir a quantidade de bits aleatórios utilizados sem penalizar muito os outros recursos de um algoritmo, como por exemplo, o número de operações elementares realizadas, é o que é chamado de ***desaleatorização***. Do ponto de vista teórico, o algoritmo para identidade polinomial pode ser desaleatorizado e o modo mais trivial de fazer isso é rodar o algoritmo para todas as possíveis escolhas aleatórias. Obviamente isso resulta num algoritmo exponencial. Não se sabe se desaleatorização de algoritmos eficientes pode ser feita de modo eficiente (veja Kabanets e Impagliazzo, 2004). Em outras palavras, não se sabe responder

onde P é a classe do problemas computacionais que podem ser decididos por um algoritmo determinístico de tempo polinomial e BPP é a classe dos problemas computacionais que podem ser decididos por algoritmo aleatorizado de tempo polinomial com probabilidade de erro menor que 1/3.

Um exemplo de desaleatorização eficiente é o algoritmo AKS para teste de primalidade de Agrawal, Kayal e Saxena, 2004, ele é resultado de uma desaleatorização do algoritmo de Agrawal e Biswas, 2003, ou seja, o AKS é a desaleatorização de um teste probabilístico de identidade polinomial, entretanto, essa desaleatorização só funciona no caso específico.

Uma técnica geral de desaleatorização de identidades polinomiais teria grande impacto na Teoria da Computação.

Para um algoritmo aleatório, como mensurar a sua correção e como mensurar seu tempo de execução? O objetivo deste texto é dar ferramentas para responder essas questões e exibir algumas das principais contribuições da Probabilidade para a Computação.

> *"Numbers that fool the Fermat test are called Carmichael numbers, and little is known about them other than that they are extremely rare. There are 255 Carmichael numbers below 100,000,000. The smallest few are 561, 1105, 1729, 2465, 2821, and 6601. In testing primality of very large numbers chosen at random, the chance of stumbling upon a value that fools the Fermat test is less than the chance that cosmic radiation will cause the computer to make an error in carrying out a "correct"algorithm. Considering an algorithm to be inadequate for the first reason but not for the second illustrates the difference between mathematics and engineering."* Abelson & Sussman

CONVEÇÕES NOTACIONAIS

1. Dos conjuntos numéricos, $\mathbb{N} = \{1, 2, \ldots\}$ denota o conjunto dos números naturais; $\mathbb{Z}$ denota o conjunto dos números inteiros e $\mathbb{Z}^+ = \{0, 1, 2, \ldots\}$; $\mathbb{R}$ denota o conjunto dos números reais.

2. Usamos ":=" com o significado de "é definido igual a".

3. A notação $A \subset B$ para "A é subconjunto de B" não exclui a possibilidade de valer o caso $A = B$; quando queremos indicar a inclusão própria escrevemos $A \subsetneq B$.

   O conjunto de todos os subconjuntos de B com cardinalidade k é denotado $\binom{B}{k}$ O conjunto de todos os subconjuntos de B, ou seja, o conjunto as partes de B, é denotado por 2^B.

4. Usamos *enumerável* para designar tanto conjunto finito quanto conjunto infinito enumerável (i.e., que pode ser posto em correspondência bijetiva com $\mathbb{N}$).

5. se A e B são conjuntos então B^A é o conjunto de todas as funções $A \to B$.

6. log é a função logaritmo natural, ou seja, logaritmo na base $e = 2{,}71\ldots$. Em base a, usamos $\log_a$.

7. Usamos $\sum_{i \geqslant 1}$ tanto quanto $\sum_{i=1}^{\infty}$ para séries.

1 | ESPAÇOS DE PROBABILIDADE

Intuitivamente, uma medida de probabilidade é uma forma quantitativa de expressar a chance com que um conjunto de resultados de um experimento *aleatório*, um processo que nos fornece resultados os quais não podem ser previamente determinados, ocorra. A Probabilidade é a disciplina dedicada a modelagem desses fenômenos com condições de incerteza. Um modelo probabilístico é um modelo matemático de um experimento aleatório. A modelagem probabilística tem sido importante em praticamente todas as áreas do conhecimento e o desenvolvimento da Teoria da Probabilidade tem sido estimulada pela ampla variedade de suas aplicações. Neste capítulo introduzimos o tratamento axiomático moderno da probabilidade introduzido pelo matemático russo Andrei Nikolaevich Kolmogorov (1903-1987) por volta de 1930 e que, ao contrário da interpretações que estabelecem uma forma explícita de calcular probabilidades, estuda as propriedades que uma probabilidade deve satisfazer.

1.1 ESPAÇOS DE PROBABILIDADE DISCRETOS

Monty Hall é o nome do apresentador de um concurso televisivo exibido na década de 1970 nos Estados Unidos chamado *Let's Make a Deal*, e é o nome de um problema agora clássico em probabilidade. O jogo consistia em o apresentador Monty Hall apresentar três portas a um espectador que concorre a um prêmio escondido pela porta escolhida através de um processo de escolhas que será descrito a seguir. O protocolo da brincadeira é: Monty Hall escolhe, ao acaso com igual probabilidade, uma das portas para esconder um carro; nas outras duas esconde um bode cada. Na primeira etapa o concorrente escolhe uma porta ao acaso (que ainda não é aberta); em seguida Monty Hall abre uma das outras duas portas que o concorrente não escolheu, sabendo que ela esconde um bode. Se são duas possibilidades, ele escolhe uma ao acaso. Com duas portas fechadas apenas, e sabendo que o carro está atrás de uma delas, o apresentador oferece ao concorrente a oportunidade de trocar de porta. O concorrente tem que decidir se permanece com a porta que escolheu no início do jogo ou se muda para a outra porta que ainda está fechada; feita a escolha, o apresentador abre a porta escolhida e o concorrente leva o prêmio escondido pela porta.

Assumindo que o objetivo do jogador é ganhar o carro, o problema é determinar uma estratégia de decisão que maximiza a chance de ganhar o carro. A resposta para esse problema será dada mais a frente no texto, no momento convidamos o leitor a refletir um pouco sobre o problema antes de passar adiante na leitura, para, ao menos, identificar os experimentos aleatórios escondidos na descrição feita no parágrafo acima.

Um modelo probabilístico para um experimento aleatório é caracterizado por um *espaço amostral* — conjunto dos resultados possíveis — um *espaço de eventos* — família[1] dos subconjuntos de resultados que admitem uma probabilidade — e uma *(medida de) probabilidade* — uma função que associa um valor numérico a cada evento.

ESPAÇO AMOSTRAL. O espaço amostral de um experimento aleatório, quase sempre denotado por Ω, é um conjunto não vazio em que cada elemento representa um resultado possível do experimento e cada resultado tem um representante que pertence ao conjunto. Um elemento de Ω é chamado de **ponto amostral** e a escolha de algum ponto amostral representa uma realização do experimento.

Exemplo 1.1. São experimentos com respectivos espaços amostrais

1. um dado é lançado e observamos a face para cima, $\Omega = \{1, 2, 3, 4, 5, 6\}$;

2. uma moeda é lançada e observamos sua face para cima, $\Omega = \{\text{Ca}, \text{Co}\}$;

[1] Família é usado como sinônimo de conjunto.

3. uma moeda é lançada até sair coroa, $\Omega = \{(\text{Co}), (\text{Ca},\text{Co}), \ldots, (\text{Ca},\text{Ca},\ldots,\text{Ca},\text{Co}), \ldots, (\text{Ca},\text{Ca},\ldots)\}$, cada ponto amostral é representado por uma sequência de Ca que, eventualmente, termina com Co.

4. uma moeda honesta é lançada sucessivamente e pergunta-se o que ocorre primeiro, uma sequência de três caras ou três coroas consecutivas. Um espaço amostral é considerar todas as sequências $(a_i \in \{\text{Ca},\text{Co}\}\colon i \geqslant 1)$ de resultados possíveis, isto é, $\Omega = \{\text{Ca},\text{Co}\}^{\mathbb{N}}$;

5. observamos tempo de vida de uma lâmpada em minutos, $\Omega = \{t \in \mathbb{R}\colon t \geqslant 0\}$;

6. um dardo é lançado num alvo circular de raio 1 e observamos o ponto atingido, um espaço amostral é obtido usando um sistema de coordenadas cartesianas com a origem no centro do alvo de modo que $\Omega = \{(x,y) \in \mathbb{R}^2\colon x^2 + y^2 \leqslant 1\}$. ◊

O espaço amostral de um experimento aleatório reflete a observação do resultado de um experimento e não é único; no item 3 do exemplo acima, podemos escrever o espaço amostral $\{1,2,3,\ldots,\infty\}$ para representar os resultados do experimento. No item 6 do exemplo acima, podemos escrever o espaço amostral com pontos dados em coordenadas polares $\{(r,\theta) \in \mathbb{R}^2\colon 0 \leqslant r \leqslant 1 \text{ e } -\pi < \theta \leqslant \pi\}$.

Exercício 1.2. Identifique os experimentos aleatórios e descreva um espaço amostral para o problema de Monty Hall.

Espaço de eventos. Intuitivamente um evento aleatório de um experimento aleatório é um acontecimento observável ao final da realização do experimento. Quando da realização do experimento deve ser sempre possível dizer se tal fenômeno ocorreu ou não ocorreu. Por exemplo, se um dado é lançado então o resultado "é um número par" e o resultado "é um número maior que 3" são eventos do experimento de lançar um dado. Um evento E é representado no modelo probabilístico por um subconjunto E_Ω do espaço amostral Ω o qual fica definido pela coleção de resultados *possíveis* do experimento que satisfazem a descrição do evento. No exemplo do dado, se $\Omega = \{1,2,3,4,5,6\}$ então "é um número par" e "é um número maior que 3" são modelados por $\{2,4,6\}$ e $\{4,5,6\}$, respectivamente. Usualmente, por abuso de notação, usamos E para denotar E_Ω embora o modelo para um evento depende do espaço amostral construído.

Assim, um modelo de um evento aleatório é subconjunto do espaço amostral Ω também chamado de **evento aleatório**. Na realização de um experimento o evento $A \subset \Omega$ *ocorre* se o resultado observado é representado por um elemento de A, caso contrário o evento A *não ocorre*. Em especial, $\varnothing$ é o evento *impossível*; Ω é o evento *certo*; $\{\omega\}$ é um evento *elementar* para cada elemento $\omega \in \Omega$; o *complemento* do evento A é o evento *não*-A dado por

$$\overline{A} = \Omega \setminus A := \{\omega \in \Omega\colon \omega \notin A\}.$$

Em um lançamento de dados $\Omega = \{1,2,3,4,5,6\}$ e são exemplos de eventos

- $A = \{2,4,6\}$, ou seja, A representa o evento "número par";
- $\overline{A} = \{1,3,5\}$, ou seja, $\overline{A}$ representa o evento "não é número par";
- $B = \{4,5,6\}$, ou seja, B representa o evento "número maior que 3";
- $C = \{4\}$, ou seja, C representa o evento "número 4";
- $A \cap \overline{A} = \varnothing$, ou seja, $A \cap \overline{A}$ representa o evento "número par *e* número ímpar", que é o evento impossível;
- $A \cup \overline{A} = \Omega$, ou seja, $A \cup \overline{A}$ representa o evento "número par *ou* número ímpar", que é o evento certo;
- $B \cap C = \{4\}$, ou seja, $B \cap C$ representa o evento "número maior que 3 *e* número 4" que é o mesmo evento que "número 4";
- $B \cap A = \{4,6\}$, ou seja, $B \cap A$ representa o evento "número maior que 3 *e* número par";
- o evento "múltiplo de 2 *ou* múltiplo de 3 mas não múltiplo de ambos" é representado pela diferença simétrica $\{2,4,6\} \triangle \{3,6\} = (\{2,4,6\} \cup \{3,6\}) \setminus (\{2,4,6\} \cap \{3,6\}) = \{2,3,4\}$.

Dizemos que A e B são eventos **disjuntos** ou **eventos mutuamente exclusivos** quando não têm elementos em comum, isto é, $A \cap B = \varnothing$. Os eventos $A_1, A_2, \ldots, A_n$ são ditos **mutuamente exclusivos** se são disjuntos tomados dois-a-dois, isto é, $A_i \cap A_j = \varnothing$ sempre que $i \neq j$. Embora eventos sejam conjuntos e a Teoria dos Conjuntos tem uma linguagem tradicional e bem aceita a Probabilidade tem um linguagem particular para os eventos e a descrevemos na tabela 1.1 abaixo.

Denotemos por $\mathcal{A}$ um conjunto de eventos aleatórios que podem ocorrer num experimento aleatório. Para ser consistente com a intuição $\mathcal{A}$ deve ter $\varnothing$ e Ω entre seus elementos, ser fechado para as operações usuais de conjunto e, também, pedimos que satisfaça o seguinte: se $A_i \in \mathcal{A}$ para todo $i \geqslant 1$, então $\bigcup_{i \geqslant 1} A_i \in \mathcal{A}$ e uma justificativa para isso é dada adiante.

Um **espaço de eventos** é um conjunto $\mathcal{A}$ de eventos aleatórios de um experimento aleatório. Quais são as famílias de subconjuntos de Ω que podem ser tomadas como espaço de eventos é um assunto que não trataremos. Uma escolha óbvia é o conjunto 2^Ω das partes de Ω, mas acontece que em muitos casos é preciso restringir essa família a um subconjunto próprio de 2^Ω para que questões probabilísticas façam sentido. Por ora, chamamos atenção ao fato de ser possível haver subconjuntos de um espaço amostral Ω que não são eventos aleatórios, como é o caso dado no exemplo 1.10 adiante, na página 16. Esse fenômeno só é importante quando Ω é grande, infinito e não enumerável.

Notação	*Eventos*	*Conjunto*
Ω	espaço amostral, evento certo	universo
$\varnothing$	evento impossível	vazio
$\{\omega\}$	evento elementar	conjunto unitário
A	evento	subconjunto
A	ocorre A	$\omega \in A$
$\overline{A}$	não ocorre A	$\omega \notin A$ (complemento)
$A \cap B$	ocorre A e B	$\omega \in A \cap B$ (intersecção)
$A \cup B$	ocorre A ou B	$\omega \in A \cup B$ (união)
$A \setminus B$	ocorre A e não ocorre B	$\omega \in A$ e $\omega \notin B$ (diferença)
$A \triangle B$	ocorre A ou B, não ambos	$\omega \in A \cup B$ e $\omega \notin A \cap B$ (diferença simétrica)
$A \subset B$	se ocorre A, então ocorre B	$\omega \in A \Rightarrow \omega \in B$ (inclusão)

Tabela 1.1: termos da Probabilidade.

Exercício 1.3. Descreva, segundo a solução dada no exercício 1.2, o evento de interesse no problema de Monty Hall, isto é, o subconjunto que modela o evento "o espectador concorrente ganha o carro".

Medida de probabilidade. Uma medida de probabilidade sobre um espaço de eventos $\mathcal{A}$ de um espaço amostral Ω é uma função, genericamente denotada por $\mathbb{P}$, que atribui a cada evento aleatório $A \in \mathcal{A}$ um número real $\mathbb{P}(A)$ satisfazendo os seguintes axiomas

A1 – *não negatividade*: $\mathbb{P}(A) \geqslant 0$;

A2 – *normalização*: $\mathbb{P}(\Omega) = 1$;

A3 – *aditividade enumerável*: $\mathbb{P}\left(\bigcup_{i\geqslant 1} A_i\right) = \sum_{i\geqslant 1} \mathbb{P}(A_i)$ sempre que $\{A_i : i \geqslant 1\}$ é um conjunto de eventos mutuamente exclusivos.[2]

As primeiras consequências importantes desses axiomas são enunciadas na proposição a seguir.

Proposição 1.4 *São consequências desses axiomas:*

1. *A probabilidade do evento impossível é* $\mathbb{P}(\varnothing) = 0$.
2. *Aditividade finita: se* $A_1, A_2, \ldots, A_n$ *são eventos mutuamente exclusivos então*

$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \mathbb{P}(A_i).$$

[2] O lado esquerdo da igualdade não depende de uma enumeração particular dos conjuntos A_i e, nesse caso, o mesmo vale para o lado direito, veja (s.1) do apêndice.

3. *A probabilidade do complemento é* $\mathbb{P}(\overline{A}) = 1 - \mathbb{P}(A)$, *para todo evento* A.

4. *Monotonicidade: se* $A \subset B$ *então* $\mathbb{P}(A) \leqslant \mathbb{P}(B)$.

5. *Regra da adição:* $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)$ *para quaisquer eventos* A *e* B.

Demonstração. Fazendo $A_1 = \Omega$ e $A_i = \varnothing$ para todo $i \geqslant 2$ temos, pela aditividade enumerável, que

$$\mathbb{P}(\Omega) = \mathbb{P}(\Omega \cup \varnothing \cup \varnothing \cup \cdots \cup \varnothing \cup \cdots) = \mathbb{P}(\Omega) + \sum_{i \geqslant 2} \mathbb{P}(\varnothing)$$

portanto, pela não-negatividade, resta que $\mathbb{P}(\varnothing) = 0$. Agora, definindo $A_i = \varnothing$ para todo $i > n$

$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) = \mathbb{P}\left(\bigcup_{i \geqslant 1} A_i\right) = \sum_{i \geqslant 1} \mathbb{P}(A_i) = \sum_{i=1}^{n} \mathbb{P}(A_i) + \sum_{i > n} \mathbb{P}(\varnothing) = \sum_{i=1}^{n} \mathbb{P}(A_i)$$

que é o resultado afirmado.

A probabilidade da complemento segue do item anterior e da normalização. Os detalhes ficam a cargo do leitor.

Para monotonicidade, consideremos A e B eventos tais que $A \subset B$. Usamos que B pode ser escrito como a união disjunta $A \cup (\overline{A} \cap B)$, donde $\mathbb{P}(B) = \mathbb{P}(A) + \mathbb{P}(\overline{A} \cap B)$ e como $\mathbb{P}(\overline{A} \cap B) \geqslant 0$ temos $\mathbb{P}(B) \geqslant \mathbb{P}(A)$. Notemos que, como consequência imediata, temos para todo evento A $\mathbb{P}(A) \leqslant 1$.

Finalmente, a união $A \cup B$ pode ser escrita como duas uniões disjuntas $(A \setminus B) \cup (B \setminus A) \cup (A \cap B)$ donde concluímos que

$$\mathbb{P}(A \cup B) = \mathbb{P}(A \setminus B) + \mathbb{P}(B \setminus A) + \mathbb{P}(A \cap B). \tag{1.1}$$

Agora, A pode ser escrito como a união disjunta $(A \setminus B) \cup (A \cap B)$ e, analogamente, $B = (B \setminus A) \cup (A \cap B)$, portanto $\mathbb{P}(A) = \mathbb{P}(A \setminus B) + \mathbb{P}(A \cap B)$ assim como $\mathbb{P}(B) = \mathbb{P}(B \setminus A) + \mathbb{P}(A \cap B)$. Isolando $\mathbb{P}(A \setminus B)$ e $\mathbb{P}(B \setminus A)$ nessas duas igualdades e substituindo na equação (1.1) prova a regra da adição. □

O seguinte limitante é bastante útil e podes ser facilmente provado usando indução e a regra da adição.

Corolário 1.5 (Desigualdade de Boole) *Se* $A_1, A_2, \ldots, A_n$ *são eventos então*

$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) \leqslant \sum_{i=1}^{n} \mathbb{P}(A_i).$$

Exemplo 1.6 (*lançamento de uma moeda equilibrada*). O modelo probabilístico para o lançamento de uma moeda equilibrada é $\Omega = \{\text{Ca}, \text{Co}\}$ e $\mathbb{P}(\varnothing) = 0$, $\mathbb{P}(\{\text{Ca}\}) = \mathbb{P}(\{\text{Co}\}) = 1/2$, $\mathbb{P}(\Omega) = 1$. ◊

Exemplo 1.7 (*lançamento de um dado equilibrado*). No caso do lançamento de um dado equilibrado atribuímos a probabilidade $1/6$ a cada uma das faces, o que é interpretado como todas as faces serem

equiprováveis. A partir disso qualquer subconjunto $A \subset \Omega$ de faces do dado é um evento que tem probabilidade de ocorrência dada por

$$\mathbb{P}(A) = \frac{|A|}{6}$$

de modo que os axiomas de probabilidade ficam satisfeitos. ◊

Nesses dois exemplos vimos o modo clássico de interpretar probabilidade no caso finito, os eventos elementares são equiprováveis e nos referimos a eles como uma "escolha aleatória", ou uma "ocorrência ao acaso". Nos espaços amostrais infinitos essenão é o caso, pode não haver uma interpretação viável ou podem haver mais de um modo natural de definir probabilidade para o significado intuitivo clássico.

Exemplo 1.8. Quando escolhemos um inteiro positivo com a probabilidade de escolher i dada por $(1/2)^i$ e estendemos a probabilidade a qualquer subconjunto A de inteiros positivos pondo

$$\mathbb{P}(A) := \sum_{a\in A} \mathbb{P}(\{a\}) \tag{1.2}$$

temos um modelo probabilístico. De fato, temos (veja (s.6a) do apêndice)

$$\mathbb{P}(\Omega) = \sum_{i\geqslant 1} \left(\frac{1}{2}\right)^i = 1$$

e a convergência absoluta dessa série implica que toda subsérie dela é convergente (veja (s.4) do apêndice), assim temos que a probabilidade dada na equação (1.2) está bem definida, isto é, $\mathbb{P}(A)$ como definido acima é um número real não negativo menor ou igual a 1.

Também segue da convergência absoluta que um rearranjo da série resulta noutra série que converge para o mesmo resultado donde obtemos a aditividade enumerável da medida de probabilidade,

$$\mathbb{P}\left(\bigcup_{i\geq 1} A_i\right) = \sum_{a\in\bigcup_i A_i} \mathbb{P}(\{a\}) = \sum_{i\geq 1}\sum_{a\in A_i} \mathbb{P}(\{a\}) = \sum_{i\geq 1} \mathbb{P}(A_i) \tag{1.3}$$

para qualquer conjunto $\{A_i : i \geqslant 1\}$ de eventos mutuamente exclusivos.

Nesse exemplo, a probabilidade de escolher um número par é

$$\sum_{a \text{ par}} \mathbb{P}(\{a\}) = \sum_{k\geqslant 1} \left(\frac{1}{2}\right)^{2k} = \sum_{k\geqslant 1} \left(\frac{1}{4}\right)^k = \frac{1}{3}$$

portanto, calculando a probabilidade do complemento, a probabilidade de escolher um número ímpar é 2/3. A probabilidade de escolha de um múltiplo de 3 é 1/7 e a probabilidade da escolha de um múltiplo de 6 é 1/31 (verifique). Usando a regra da adição e o fato de que ser múltiplo de 6 equivale a ser múltiplo de 2 *e* múltiplo de 3, temos que a probabilidade de escolha de um múltiplo de 2 *ou* um múltiplo de 3 é a probabilidade de escolha de um múltiplo de 2 mais a probabilidade de escolha de um múltiplo de 3 menos a probabilidade de escolha de um múltiplo de 6, ou seja, a probabilidade de escolha de um múltiplo de 2 *ou* um múltiplo de 3 é $1/3 + 1/7 - 1/31 = 289/651 \approx 0{,}444$. ◊

Exemplo 1.9. No intervalo $\Omega = [0,1]$ da reta real podemos definir uma medida de probabilidade $\mathbb{P}$ de modo que os intervalos (a,b), $(a,b]$, $[a,b)$, $[a,b]$ tenham probabilidade $|b-a|$, entretanto não há tal medida de modo que $\mathbb{P}(A)$ esteja definida para todo $A \subset \Omega$, ou seja, nem todo subconjunto do espaço amostral é evento (veja, e.g., Rosenthal, 2006, proposição 1.2.6). O conjunto dos eventos aleatórios é subconjunto próprio do conjunto das partes do intervalo. ◊

Exemplo 1.10 (*probabilidade geométrica*). Consideremos o experimento 6 do exemplo 1.1. É possível definir uma medida de probabilidade para $A \subset \Omega$ como a área de A proporcionalmente a de Ω, i.e.,

$$\mathbb{P}(A) = \frac{\text{Área}(A)}{\pi}$$

Assim, a probabilidade de um lançamento aleatório acertar o círculo de mesmo centro do alvo e raio 1/2 é 1/4. Ademais, há subconjuntos de Ω que não têm uma probabilidade associada pois não é possível definir área para todo subconjunto do plano (veja, e.g., Gelbaum e Olmsted, 1964, capítulo 11). ◊

O próximo exemplo é conhecido como o paradoxo de Bertrand mas que, a rigor, não é um paradoxo, é passível de mais de uma interpretação para "ao acaso", ou "aleatório".

Exemplo 1.11 (*paradoxo de Bertrand*). Qual é a probabilidade de que uma corda AB escolhida ao acaso numa circunferência de raio 1 tenha comprimento maior que $\sqrt{3}$? Numa circunferência de raio 1, um triângulo equilátero inscrito tem lado $\sqrt{3}$ (figura 1.1). Numa primeira interpretação a

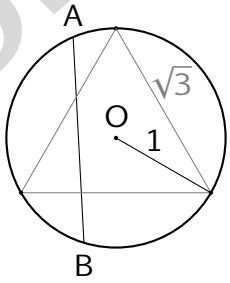
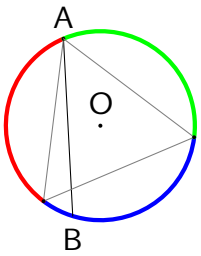


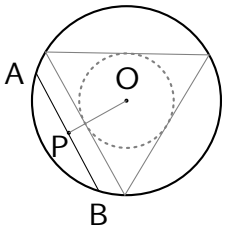
Figura 1.1: paradoxo de Bertrand.

escolha da corda se dá ao tomarmos A e B escolhidos ao acaso dentre os pontos da circunferência. Consideremos o triângulo rotacionado de modo que um de seus vértices coincida com o ponto A. A corda tem comprimento maior que o lado do triângulo se B está no arco da circunferência entre os dois outros vértices do triângulo, o que ocorre com probabilidade 1/3 pois os vértices dividem a circunferência em três arcos de mesmo comprimento (figura 1.2(a)).

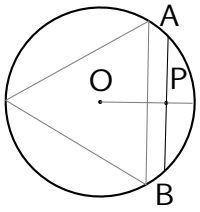
Na segunda interpretação, a corda é obtida por uma escolha de P no interior da circunferência e AB é a corda cujo ponto médio é P (figura 1.2(b)). A corda é maior que o lado do triângulo se P está no interior da circunferência de centro O e raio 1/2, o que ocorre com probabilidade 1/4.



(a) a corda é dada por uma escolha aleatória de A e de B na circunferência. A probabilidade procurada é 1/3.



(b) a corda AB é definida por P, seu ponto médio. A probabilidade procurada é 1/4.



(c) a corda é dada pela escolha de um raio e pela escolha de um ponto P nesse raio. A probabilidade procurada é 1/2.

Figura 1.2: As três interpretações do paradoxo de Bertrand.

Na terceira é última interpretação para uma corda aleatória nós fixamos um raio. A corda é obtida escolhendo um ponto P no raio e tomando a corda que passa por P e perpendicular ao raio (figura 1.2(c)). A corda é maior do que um lado do triângulo, se o ponto escolhido está mais próximo do centro do círculo, que o ponto onde o lado do triângulo intersecta o raio, logo se $|OP| \in (0, 1/2)$ o que ocorre com probabilidade 1/2. ◊

Há uma diferença fundamental entre os modelos probabilísticos dos exemplos 1.7 e 1.8 e os modelos dos exemplos 1.9 e 1.10. Nos dois primeiros é possível atribuir probabilidade a todo subconjunto do espaço amostral, o que não é possível nos outros dois. A explicação desse fenômeno é muito técnica para ser dada aqui, mas está relacionada a cardinalidade do espaço amostral. Um espaço amostral enumerável (finito ou infinito) é chamado de **espaço amostral discreto**. Um espaço que têm a mesma cardinalidade dos reais, que também é o caso do item 4 do exemplo 1.1, é chamado de **espaço amostral contínuo**. Para o espaço de eventos de um espaço amostral contínuo qualquer vale que *não há medida de probabilidade que possa ser definida para todo subconjunto* desses espaços. Esse é um resultado difícil para demonstrarmos aqui, as ferramentas necessárias vão além do escopo deste texto, e que só ocorre em espaços amostrais não enumeráveis.

Em resumo, o espaço de eventos $\mathcal{A}$ é uma necessidade técnica e sua compreensão vai muito além do que precisamos neste texto que é dedicado ao caso discreto.

Exercício 1.12. Determine uma medida de probabilidade para os eventos do problema de Monty Hall.

1.1.1 ESPAÇO DE PROBABILIDADE

Probabilidade pode ser estudada do ponto de vista abstrato sem se referir a experimentos aleatórios e sem que os números associados aos eventos tenham qualquer interpretação. Formalmente,

exigimos que qualquer medida de probabilidade $\mathbb{P}$ esteja definida sobre uma família $\mathcal{A}$ de subconjuntos de Ω que deve satisfazer: (i) $\Omega \in \mathcal{A}$; (ii) se $A \in \mathcal{A}$ então $\overline{A} \in \mathcal{A}$; (iii) se $A_i \in \mathcal{A}$ para todo $i \geqslant 1$, então $\bigcup_{i\geqslant 1} A_i \in \mathcal{A}$. Uma família de subconjuntos como acima é dita σ-álgebra de subconjuntos de Ω. Um **espaço de probabilidade**, assim como um **modelo probabilístico**, é uma terna $(\Omega, \mathcal{A}, \mathbb{P})$ tal que Ω é um conjunto não vazio, chamado **espaço amostral**; $\mathcal{A}$ é uma σ-álgebra de subconjuntos de Ω ditos **eventos**; e $\mathbb{P}\colon \mathcal{A} \to [0,1]$ é uma **medida de probabilidade**.

Deixamos para a reflexão do leitor o fato de que todo modelo probabilístico de um experimento aleatório corresponde a um espaço de probabilidades e todo espaço de probabilidades corresponde ao modelo probabilístico de um experimento ideal e usaremos essas terminologias sem distinção.

1.1.2 MODELO PROBABILÍSTICO DISCRETO

Um **modelo probabilístico discreto**, ou **espaço de probabilidade discreto**, é um espaço $(\Omega, 2^\Omega, \mathbb{P})$ em que Ω é enumerável (finito ou infinito). No caso de espaço amostral discreto, todo experimento tem seu modelo probabilístico especificado quando estabelecemos

(D1) um espaço amostral enumerável Ω que pode ser finito ou infinito;

(D2) uma função de probabilidade $p\colon \Omega \to [0,1]$ tal que $\sum_{\omega\in\Omega} p(\omega) = 1$.

De fato, dado (Ω, p) como acima podemos definir uma função sobre 2^Ω tomando

$$\mathbb{P}(A) := \sum_{\omega\in A} p(\omega)$$

que é um número real positivo para qualquer $A \subset \Omega$, como já observamos no exemplo 1.8 (veja (s.4) do apêndice).

Convencionamos a notação $\mathbb{P}(\omega) := \mathbb{P}(\{\omega\})$ para os eventos elementares.

Claramente, $\mathbb{P}(A) \geqslant 0$ e $\mathbb{P}(\Omega) = \sum_i \mathbb{P}(\omega_i) = 1$. Ainda, se A_i para $i \geqslant 1$ são eventos mutuamente exclusivos então $\mathbb{P}\big(\bigcup_{i\geqslant 1} A_i\big) = \sum_{i\geqslant 1} \mathbb{P}(A_i)$, como na equação (1.3), segue da convergência absoluta e da exclusão mútua.

Para registro, enunciamos o seguinte resultado sem prova.

TEOREMA *Se $\Omega \neq \varnothing$ é enumerável e $p\colon \Omega \to [0,1]$ é tal que $\sum_{\omega\in\Omega} p(\omega) = 1$ então $(\Omega, 2^\Omega, \mathbb{P})$ com $\mathbb{P}(A) = \sum_{\omega\in A} p(\omega)$ para todo $A \in 2^\Omega$ é um espaço de probabilidade. Reciprocamente, se $(\Omega, 2^\Omega, \mathbb{P})$ é um espaço de probabilidade sobre Ω enumerável então (Ω, p) com $p(\omega) := \mathbb{P}(\{\omega\})$ satisfaz as condições (D1) e (D2) de um modelo probabilístico discreto.*

Exemplo 1.13. No item 3 do exemplo 1.1 uma moeda equilibrada é lançada e observamos o resultado até sair coroa. Esse experimento é modelado pelo espaço amostral $\Omega = \{(\text{Co}), (\text{Ca},\text{Co}), \ldots, (\text{Ca},\text{Ca},\ldots)\}$

munido da função de probabilidade $p((c_1, c_2, \ldots, c_i)) = 2^{-i}$ onde $c_j = \text{Co}$ se $j = i$ e $c_j = \text{Ca}$ caso contrário, e $p((\text{Ca}, \text{Ca}, \ldots)) = 0$. Da argumentação feita no exemplo 1.8, página 15, deduzimos igualmente que Ω e p definem um modelo probabilístico discreto para o experimento. $\Diamond$

Exemplo 1.14 (*um modelo probabilístico para Monty Hall*). No caso do problema de Monty Hall, consideremos o experimento que consiste das seguintes três etapas

1. o apresentador esconde o carro atrás de uma das portas escolhida com probabilidade 1/3;
2. com probabilidade 1/3, uma porta é escolhida pelo jogador;
3. o apresentador revela, dentre as duas que o jogador não escolheu, aquela que não esconde o carro. Se houver duas possibilidades então o apresentador escolhe uma delas com probabilidade 1/2.

O espaço amostral é definido pelas ternas (e_1, e_2, e_3) em que e_i é a porta escolhida no passo i descrito acima e se a portas estão numeradas por 1,2 e 3 então definimos um modelo probabilístico discreto com $Omega$ dado por

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

e probabilidades de acordo com o diagrama de árvore mostrado na figura 1.3 abaixo; um caminho seguido pelo jogador numa rodada do jogo corresponde a um caminho na árvore, a partir da raiz (o ponto mais alto) até uma folha (um dos pontos mais baixos). A primeira ramificação corresponde a escolha de porta para esconder o carro, as segundas ramificações correspondem a escolha do jogador e as terceiras ramificações correspondem a escolha de porta para abrir feita pelo apresentador. Os eventos que interessam, a saber "o jogador vence trocando de porta" e "o jogador vence

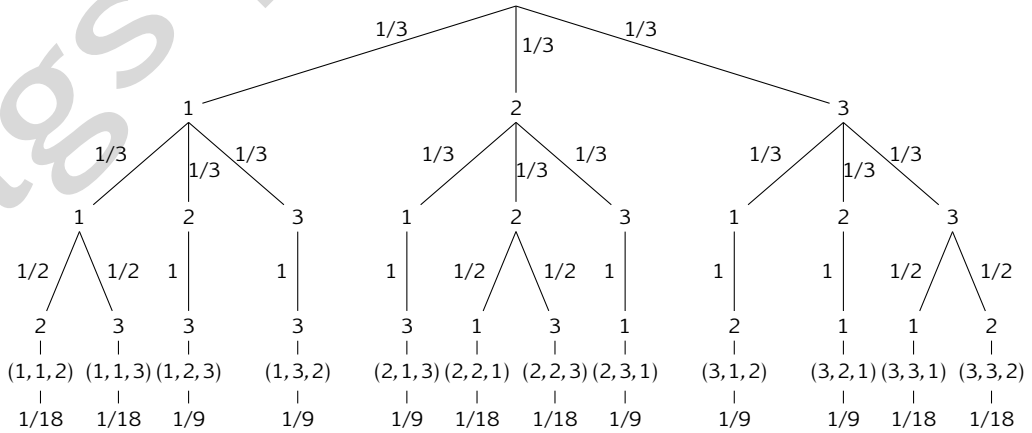


Figura 1.3: diagrama de árvore de um modelo para Monty Hall.

não trocando de porta", são complementares e denotados por A e $\overline{A}$ respectivamente, de modo que $A = \{(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)\}$ e $\overline{A}$ é dada pelas ternas restantes de Ω. O jogador ganha o carro trocando de porta com probabilidade

$$\mathbb{P}(A) = \mathbb{P}(\{(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)\}) = \frac{2}{3}$$

portanto, ganha sem trocar de porta com probabilidade $1 - 2/3 = 1/3$, que corresponde à probabilidade de ter escolhido a porta certa já na primeira oportunidade de escolha. Portanto, a melhor estratégia é trocar de porta quando é oferecida essa oportunidade. ◊

1.1.3 CONTINUIDADE DE UMA MEDIDA DE PROBABILIDADE

Consideremos novamente o lançamento de uma moeda equilibrada até sair coroa, citado no exemplo 1.13, modelado por $\Omega = \{(\text{Co}),(\text{Ca},\text{Co}),(\text{Ca},\text{Ca},\text{Co}),\ldots,(\text{Ca},\text{Ca},\ldots)\}$ munido da função de probabilidade $p((c_1,c_2,\ldots,c_i)) = 2^{-i}$ e $p((\text{Ca},\text{Ca},\ldots)) = 0$. Como cada resultado de um lançamento é igualmente provável e não depende dos resultados dos outros lançamentos deve ser intuitivamente válido[3] que devemos assumir que $(\text{Ca},\text{Ca},\ldots,\text{Ca},\text{Co})$ tenha probabilidade de ocorrer igual a

$$\left(\frac{1}{2}\right)^{\text{número de lançamentos}} \tag{1.4}$$

e como cada ponto amostral em $\Omega \setminus \{(\text{Ca},\text{Ca},\ldots)\}$ está associado a um único inteiro maior que zero $\mathbb{P}(\Omega \setminus \{(\text{Ca},\text{Ca},\ldots)\}) = \sum_{n\geqslant 1} 2^{-n} = 1$ o que nos obriga a tomar como 0 a probabilidade para o evento "nunca sair coroa". Nessa seção veremos que essa obrigação condiz com a proposta intuitiva para finitos lançamentos tomada em na equação (1.4) acima.

Consideremos o evento A_n definido por "não sai coroa até o n-ésimo lançamento" que ocorre com probabilidade 2^{-n}. Falando inda de modo intuitivo, queremos que o evento "nunca sair coroa", representado por $\lim_{n\to\infty} A_n$, tenha probabilidade $\lim_{n\to\infty} \mathbb{P}(A_n) = \lim_{n\to\infty} 2^{-n} = 0$. Essa "passagem ao limite", $\mathbb{P}(\lim_{n\to\infty} A_n) = \lim_{n\to\infty} \mathbb{P}(A_n)$, é garantida pela aditividade enumerável.

Uma sequência qualquer $(A_n : n \geqslant 1)$, de eventos em um espaço de probabilidade $(\Omega, \mathcal{A}, \mathbb{P})$ é dita **monótona** se vale um dos casos

crescente: $A_1 \subset A_2 \subset \cdots \subset A_n \subset A_{n+1} \subset \cdots$ e definimos

$$\lim_{n\to\infty} A_n := \bigcup_{n\geqslant 1} A_n.$$

decrescente: $A_1 \supset A_2 \supset \cdots \supset A_n \supset A_{n+1} \supset \cdots$ e definimos

$$\lim_{n\to\infty} A_n := \bigcap_{n\geqslant 1} A_n.$$

[3] Essa noção intuitiva é formalizada na seção 1.4. Por ora, notemos que em n lançamentos a probabilidade de ocorrer um resultado específico é $(1/2)^n$ quando todos os resultados são igualmente prováveis.

Se $(A_n : n \geqslant 1)$ é uma sequência crescente, então o limite pode ser escrito como uma união de eventos disjuntos $A_1 \cup (A_2 \setminus A_1) \cup (A_3 \setminus A_2) \cup \cdots$ de modo que se tomamos $A_0 := \varnothing$ então

$$\mathbb{P}\Big(\lim_{n\to\infty} A_n\Big) = \lim_{n\to\infty} \sum_{i=1}^{n} \mathbb{P}(A_i \setminus A_{i-1}) = \lim_{n\to\infty} \sum_{i=1}^{n} (\mathbb{P}(A_i) - \mathbb{P}(A_{i-1})) = \lim_{n\to\infty} \mathbb{P}(A_n).$$

No caso em que $(A_n : n \geqslant 1)$ é decrescente tomamos os complementos e temos que $(\overline{A_n} : n \geqslant 1)$ é crescente, portanto,

$$\mathbb{P}\Big(\lim_{n\to\infty} \overline{A_n}\Big) = \mathbb{P}\Big(\bigcup_{n\geqslant 1} \overline{A_n}\Big) = \mathbb{P}\Big(\overline{\bigcap_{n\geqslant 1} A_n}\Big) = \mathbb{P}\Big(\overline{\lim_{n\to\infty} A_n}\Big) = 1 - \mathbb{P}\Big(\lim_{n\to\infty} A_n\Big)$$

por outro lado

$$\mathbb{P}\Big(\lim_{n\to\infty} \overline{A_n}\Big) = \lim_{n\to\infty} \mathbb{P}\Big(\overline{A_n}\Big) = \lim_{n\to\infty} \Big(1 - \mathbb{P}\Big(\overline{A_n}\Big)\Big) = 1 - \lim_{n\to\infty} \mathbb{P}(A_n)$$

portanto

$$\lim_{n\to\infty} \mathbb{P}(A_n) = \mathbb{P}\Big(\lim_{n\to\infty} A_n\Big).$$

Em resumo, se $(A_n : n \geqslant 1)$, é uma sequência monótona de eventos então

$$\mathbb{P}\Big(\lim_{n\to\infty} A_n\Big) = \lim_{n\to\infty} \mathbb{P}(A_n). \tag{1.5}$$

De volta ao exemplo do início da seção, consideremos o evento A_n definido por "não sai coroa até o n-ésimo lançamento". Então a sequência $(A_n : n \geqslant 1)$ é monótona pois $A_n \supset A_{n-1}$ para todo $n > 1$, portanto,

$$\mathbb{P}((\mathrm{Ca}, \mathrm{Ca}, \ldots)) = \mathbb{P}\Big(\lim_{n\to\infty} A_n\Big) = \lim_{n\to\infty} \mathbb{P}(A_n) = 0.$$

Exemplo 1.15. Consideremos o lançamento de uma moeda equilibrada infinitas vezes, o que pode ser modelado pelo espaço amostral contínuo $\Omega = \{\mathrm{Ca}, \mathrm{Co}\}^{\mathbb{N}}$. Intuitivamente, parece ser claro que esperaríamos que a probabilidade de nunca sair cara deveria ser zero: se lançarmos n vezes, a probabilidade de nunca sair cara é 2^{-n}, então no limite a probabilidade é 0. A propriedade dada na equação (1.5) permite a passagem ao limite: A_n é o evento "nos primeiros n lançamentos ocorre pelo menos uma cara"; para $n \geqslant 1$ temos uma sequência monótona de eventos. O limite é o evento "em algum momento, ocorre cara" cuja probabilidade é

$$\lim_{n\to\infty} \mathbb{P}(A_n) = \lim_{n\to\infty} 1 - 2^{-n} = 1.$$

Portanto, de fato, a probabilidade de nunca sair cara é zero. ◊

Vimos que a propriedade dada na equação (1.5) segue dos axiomas A1 (não negatividade), A2 (normalização) e A3 (aditividade enumerável) para uma medida de probabilidade. Se tomarmos por axiomas de probabilidade os axiomas A1, A2, a aditividade finita (isto é, o item 2 da proposição 1.4) e

a propriedade dada na equação (1.5) para sequências monótonas de eventos, então vale a aditividade enumerável.

De fato, assumindo os axiomas A1 e A2, a equação (1.5) e o item 2 da proposição 1.4, se $(A_n : n \geqslant 1)$ é qualquer sequência de eventos mutuamente exclusivos então

$$B_n := \bigcup_{i \geqslant n} A_i$$

é uma sequência monótona decrescente e $\lim_{n\to\infty} B_n = \varnothing$. Usando a aditividade finita de $\mathbb{P}$

$$\mathbb{P}\left(\bigcup_{i\geqslant 1} A_i\right) = \mathbb{P}\left(\bigcup_{i=1}^{n-1} A_i\right) + \mathbb{P}\left(\bigcup_{i\geqslant n} A_i\right) = \sum_{i=1}^{n-1} \mathbb{P}(A_i) + \mathbb{P}(B_n)$$

e se tomamos o limite quando n tende ao infinito

$$\mathbb{P}\left(\bigcup_{i\geqslant 1} A_i\right) = \sum_{i\geqslant 1} \mathbb{P}(A_i).$$

Diferente do que fizemos no exemplo 1.8, página 15, a demonstração para espaços contínuos de que uma função candidata a medida de probabilidade é enumeravelmente aditiva é difícil. Usualmente, o que é feito é verificar que uma função é finitamente aditiva e contínua para toda sequência $(B_n : n \geqslant 1)$ tal que $\lim_{n\to\infty} B_n = \varnothing$, isso é suficiente para garantir que é enumeravelmente aditiva e, em geral, uma tarefa mais fácil de realizar.

1.2 CONVENÇÕES DE NOTAÇÃO

Consideremos E um evento aleatório e E_Ω o subconjunto que o modela em $(\Omega, \mathcal{E}, \mathbb{P})$. Denotamos por $\mathbb{P}[E]$ a probabilidade do evento E, isto é, $\mathbb{P}[E] := \mathbb{P}(E_\Omega)$. Por exemplo, suponha que uma moeda equilibrada é lançada até sair coroa, então a probabilidade do evento "o número de lançamentos é par" com essa convenção fica $\mathbb{P}[\text{o número de lançamentos é par}]$ que é o mesmo que $\mathbb{P}(\{(c_1, \ldots, c_i) : i \text{ é par}\})$. Caso haja a necessidade de evidenciar o espaço amostral escreveremos

$$\mathop{\mathbb{P}}_{\Omega}[E] \quad \text{ou} \quad \mathop{\mathbb{P}}_{\omega\in\Omega}[\omega \text{ satisfaz } E] \quad \text{ou} \quad \mathop{\mathbb{P}}_{\omega\in\Omega}[\omega \in E] \tag{1.6}$$

com o mesmo sentido, o de $\mathbb{P}(E_\Omega)$.

Caso Ω seja finito e a menos que seja dada explicitamente outra medida, então a notação na equação (1.6) significa que estamos assumindo a medida de **probabilidade uniforme**: $\mathbb{P}(\omega) = 1/|\Omega|$ para todo $\omega \in \Omega$. Por exemplo, seja $p(x)$ um polinômio não nulo com coeficientes inteiros e Ω um conjunto finito de números inteiros. A probabilidade de que o sorteio de um elemento de Ω resulte numa raiz do polinômio é descrita por

$$\mathop{\mathbb{P}}_{x\in\Omega}[p(x) = 0]$$

que é a probabilidade do evento $R = \{\omega \in \Omega : p(\omega) = 0\}$ e que, caso não seja dito nada a respeito da medida, é dada por $\mathbb{P}(R) = |R|/|\Omega|$.

Nos algoritmos assumiremos a possibilidade de se fazer escolhas aleatórias, ou seja, assumiremos que os algoritmos dispõem de uma fonte de bits aleatórios e escrevemos a instrução

$$a \leftarrow_{\mathcal{U}} \{0,1\}$$

para denotar o fato de que a é uma variável do algoritmo e que após a execução da atribuição $\leftarrow_{\mathcal{U}}$ o valor da variável a é um elemento qualquer de $\{0,1\}$ com probabilidade $1/2$. De um modo geral, se Ω é um conjunto finito, então escrevemos a instrução

$$a \leftarrow_{\mathcal{U}} \Omega$$

chamada de atribuição por uma **escolha aleatória uniforme** em Ω, o que significa que a assume qualquer um dos elementos de Ω com igual probabilidade, a saber $1/|\Omega|$.

1.2.1 SIGILO PERFEITO

Vejamos como aplicação dos conceitos elementares de probabilidade uma das contribuições do grande matemático americano Claude Shannon (1916 – 2001) que é considerado fundador da Teoria da Informação.

Um *sistema de codificação* é definido por uma quina $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ de conjuntos, onde $\mathcal{P}$ é o conjunto dos textos comuns (ou legíveis); $\mathcal{C}$ é o conjunto dos textos codificados (ou ilegíveis); $\mathcal{K}$ é o espaço das chaves que são usadas para codificar e decodificar um texto; $\mathcal{E}$ é o conjunto das funções de codificação $E_k : \mathcal{P} \to \mathcal{C}$ para $k \in \mathcal{K}$; $\mathcal{D}$ é o conjunto das funções de decodificação $D_k : \mathcal{C} \to \mathcal{P}$ para $k \in \mathcal{K}$. Essas funções são tais que para cada $e \in \mathcal{K}$ existe $d \in \mathcal{K}$ para as quais vale $D_d(E_e(p)) = p$.

Exemplo 1.16 (*cifra de César*). Essa técnica identifica o alfabeto $\{a, b, \ldots, z\}$ com o conjunto $\{0, \ldots, 25\}$ dos restos da divisão inteira por 26 e $\mathcal{K} = \mathcal{P} = \mathcal{C} := \{0, \ldots, 25\}^\ell$ em que ℓ é o comprimento da mensagem. Para uma chave $e \in \mathcal{K}$ a mensagem $\mathbf{x} = x_1 x_2 \ldots x_\ell$ é codificada como $E_e(\mathbf{x}) = y_1 y_2 \ldots y_\ell$ com $y_i = (x_i + e) \bmod 26$, para todo i, e é decodificada como $D_e(\mathbf{x}) = y_1 y_2 \ldots y_\ell$ com $y_i = (x_i - e) \bmod 26$ para todo i. Por exemplo, para a chave $e = 3$ o texto "essaauladasono" é codificado como "hvvddzodgdvrqr".

A cifra de César deve seu nome ao imperador romano Júlio César que a usou com a chave fixa $e = 3$. Tal codificação é facilmente decifrada não oferecendo segurança na comunicação e sua efetividade na época de César deveu-se principalmente ao fato de que a maioria das pessoas eram analfabetas.

No caso $\ell = 1$ conseguimos um cifra segura se escolhemos uma chave aleatoriamente. Tomemos $(\mathcal{K},\mathbb{P})$ com $\mathbb{P}$ a medida uniforme. Dadas duas mensagens legíveis $m_1, m_2 \in \mathcal{P}$ quaisquer e uma

mensagem codificada $y \in \mathcal{C}$ qualquer, temos

$$\mathbb{P}(\{k \in \mathcal{K}\colon \mathsf{E}_k(m_1) = y\}) = \frac{1}{26} = \mathbb{P}(\{k \in \mathcal{K}\colon \mathsf{E}_k(m_2) = y\})$$

ou seja, o conhecimento do texto codificado não dá nenhuma informação a respeito do texto legível. No caso $\ell = 2$ a situação é outra. Se $ab, az \in \mathcal{P}$ e $bc \in \mathcal{C}$ então $\mathbb{P}(\{k \in \mathcal{K}\colon \mathsf{E}_k(ab) = bc\}) = 1/26$ pois podemos tomar $k = 1$ e essa é a única chave que codifica ab em bc, por outro lado não existe chave que codifica az em bc de modo que $\mathbb{P}(\{k \in \mathcal{K}\colon \mathsf{E}_k(az) = bc\}) = 0$. Agora, o conhecimento do texto codificado dá alguma informação a respeito do texto legível. ◊

Um sistema de codificação tem **sigilo perfeito** se para quaisquer $m_1,\ m_2 \in \mathcal{P}$ de mesmo comprimento e para todo $\mathsf{C} \in \mathcal{C}$ vale

$$\mathop{\mathbb{P}}_{k\in\mathcal{K}}[\mathsf{E}_k(m_1) = \mathsf{C}] = \mathop{\mathbb{P}}_{k\in\mathcal{K}}[\mathsf{E}_k(m_2) = \mathsf{C}].$$

Pelas convenções de notação, o espaço amostral é o conjunto das chaves, a descrição $[\mathsf{E}_k(m_1) = \mathsf{C}]$ corresponde ao evento $\{k \in \mathcal{K}\colon \mathsf{E}_k(m_1) = \mathsf{C}\}$ formado por todas as chaves que codificam m_1 como C e, também, está implícito que probabilidade de uma chave qualquer é $1/|\mathcal{K}|$.

Em outras palavras, o sigilo perfeito requer que, dado um texto cifrado, qualquer texto legível tem a mesma probabilidade de ser o texto legível subjacente ao texto cifrado.

Exemplo 1.17 (*one-time pad*). O seguinte sistema de codificação, conhecido por *one-time pad*, foi descrito pela primeira vez em 1882 por Frank Miller e reinventado, também patenteado, por Gilbert Sandford Vernam em 1919 e, posteriormente, aperfeiçoado por Joseph Mauborgne, que reconheceu que se a chave fosse aleatória e usada uma única vez o sistema seria muito seguro.

Tomamos $\mathcal{P} = \mathcal{K} = \mathcal{C} := \{0,1\}^n$ e para uma chave k escolhida previamente definimos

$$\mathsf{E}_k(x) := x \oplus k \qquad \text{e} \qquad \mathsf{D}_k(y) := y \oplus k. \tag{1.7}$$

em que $x \oplus y$ é a soma módulo 2 (ou o *ou exclusivo*) coordenada-a-coordenada das sequências binárias x e y. O leitor pode verificar que em (1.7) vale $\mathsf{D}_k(\mathsf{E}_k(x)) = x$. ◊

Claude Shannon provou que o *one-time pad* é uma codificação "inviolável" no sentido de que o sistema tem sigilo perfeito. O *one-time pad* não é o único sistema que possui sigilo perfeito, mas foi o primeiro a ser descoberto.

A codificação do texto legível $m \in \mathcal{P}$ usando a chave $k \in \mathcal{K}$ é o texto cifrado $\mathsf{C} = m \oplus k$, logo $m \oplus \mathsf{C} = m \oplus (m \oplus k) = (m \oplus m) \oplus k = k$, portanto, dados m e C existe uma única chave $k \in \mathcal{K}$ tal que $\mathsf{E}_k(m) = \mathsf{C}$, de modo que

$$\mathop{\mathbb{P}}_{k\in\mathcal{K}}[m_1 \oplus k = \mathsf{C}] = \frac{|\{k \in \mathcal{K}\colon k \oplus m_1 = \mathsf{C}\}|}{|\mathcal{K}|} = \frac{1}{|\mathcal{K}|} = \mathop{\mathbb{P}}_{k\in\mathcal{K}}[m_2 \oplus k = \mathsf{C}]$$

para todos os textos legíveis $m_1,\ m_2 \in \mathcal{P}$ e todo texto cifrado $\mathsf{C} \in \mathcal{C}$. Isso demonstra o seguinte resultado.

Teorema 1.18 *O* one-time pad *tem sigilo perfeito.* □

1.2.2 TESTE DE IDENTIDADE POLINOMIAL

Nosso primeiro exemplo de um algoritmo aleatorizado é um teste de identidade entre polinômios: dados dois polinômios p e q, decidir de modo eficiente se eles são idênticos.

Há muitas questões a serem esclarecidas nessa formulação do problema: o que significam "eficiente", "dado um polinômio" e "idênticos"? Isso será tratado mais tarde, na seção 2.2.3, por ora basta saber que "dado um polinômio p" significa que p é um polinômio com coeficientes inteiros dado por uma caixa preta da qual a função polinomial $p(x)$ pode ser avaliada em qualquer número x. Além disso, vamos considerar o problema equivalente de decidir se um polinômio dado f é identicamente nulo. Uma algoritmo que resolve esse problema também resolve o problema original, basta tomarmos $f(x) = p(x) - q(x)$.

Um algoritmo para esse problema funciona do seguinte modo: dado $f(x)$ de grau no máximo d, escolhemos aleatoriamente $a \in \{1, 2, \ldots, 4d\}$ e avaliamos $f(a)$; se $f(a) = 0$, então respondemos *sim*, caso contrário respondemos *não*. A resposta *sim* significa que $f(x)$ é o polinômio nulo e a resposta *não* significa que $f(x)$ não é o polinômio nulo. Esse algoritmo pode responder errado dependendo de f e da escolha aleatória a e devemos tentar garantir que a probabilidade de ocorrer o erro seja pequena.

Se o polinômio f é nulo então a resposta está sempre certa. Suponhamos que f não é nulo. Nesse caso, se a escolha aleatória a for uma raiz do polinômio então $f(a) = 0$ e a resposta a resposta *sim* dada pelo algoritmo está errada e se a escolha aleatória a não for uma raiz do polinômio então $f(a) \neq 0$ e a resposta a resposta *sim* dada pelo algoritmo está correta. Em resumo, uma resposta *não* dada pelo algoritmo está correta e uma resposta *sim* pode estar errada. O algoritmo descrito abaixo resume essa estratégia.

Instância : d inteiro positivo e f polinômio de grau no máximo d.
Resposta : *não* se f não é nulo, senão *sim* com probabilidade de erro no máximo 1/4.

1 $a \leftarrow_U \{1, 2, \ldots, 4d\}$;
2 **se** $f(a) = 0$ **então responda** *sim*.
3 **senão responda** *não*.

Algoritmo 2: teste de identidade entre polinômios.

Para determinar um limitante para a probabilidade do algoritmo responder errado, seja f um polinômio não nulo e de grau no máximo d e consideremos o evento E formado pelas raízes de f que pertencem ao espaço amostral $\Omega = \{1, 2, \ldots, 4d\}$. Então $|\mathsf{E}| \leqslant \text{grau}(f) \leqslant d$ pois f tem no máximo grau(f) raízes distintas pelo teorema fundamental da álgebra. O algoritmo erra se a escolha aleatória resulta num elemento de E, portanto,

$$\mathbb{P}[\text{erro}] \leqslant \frac{1}{4}.$$

Proposição 1.19 *Sejam d um inteiro positivo, f um polinômio não nulo de grau no máximo d e $\Omega \subset \mathbb{Z}$*

finito. Então a probabilidade com que uma escolha aleatória uniforme em Ω seja raiz de f é no máximo $d/|\Omega|$, portanto, o algoritmo 2 erra com probabilidade no máximo $1/4$. □

Veremos mais adiante que é possível fazer essa probabilidade arbitrariamente pequena ao custo de um pouco mais computação. Intuitivamente, imagine tal algoritmo sendo executado em dois computadores diferentes concomitantemente. Basta um deles responder *não* para que a resposta definitiva seja *não*. Agora se f é não nula, com que probabilidade todos respondem *sim*? Uma resposta em $E \times E \subset \Omega \times \Omega$ (uma coordenada para cada computador) ocorre com probabilidade no máximo $(1/4)^2$. Se são dez computadores a probabilidade de erro é no máximo $(1/4)^{10} < 10^{-6}$ (em metros é menos que o diâmetro do fio da teia de aranha).

O problema da identidade polinomial tem importância central em Complexidade Computacional. Em resumo, podemos dizer que o algoritmo aleatorizado dado acima resolve esse problema e é muito eficiente (executa poucas instruções), enquanto que um algoritmo eficiente que resolve esse problema sem usar aleatoriedade não é conhecido e não se sabe se pode existir. A existência de tal algoritmo determinístico e eficiente teria implicações profundas na Teoria da Computação.

1.3 PROBABILIDADE CONDICIONAL

Lançamos dois dados equilibrados, um deles é vermelho e tem doze faces numeradas de 1 a 12 e o outro preto com vinte faces numeradas de 1 a 20.

Suponhamos que temos a informação de que a soma dos resultados é 15, e isso é tudo que sabemos. Qual é a probabilidade do dado vermelho ter resultado 6?

Definimos um modelo discreto para o experimento tomando o espaço amostral Ω composto pelos $12 \cdot 20 = 240$ pontos amostrais, dados pelos pares ordenados de resultados de cada dado, com a medida uniforme de probabilidade. Sejam Q_Ω o subconjunto dos 12 eventos elementares de Ω que representa o evento "a soma é 15" e S_Ω o evento "o valor do dado vermelho é 6".

Se é certo que ocorre Q_Ω então vamos renormalizar a probabilidade de cada ponto amostral $\omega \in Q_\Omega$ para $\mathbb{P}_Q(\omega) = (1/|\Omega|)/(|Q_\Omega|/|\Omega|) = 1/12$ de modo que Q_Ω tenha probabilidade 1. Ademais $S_Q = S_\Omega \cap Q_\Omega = \{(6,9)\}$ tem probabilidade $\mathbb{P}_Q(S_Q) = 1/12$. Essa é a probabilidade de ocorrer um 6 vermelho sob a condição de que a soma dos dados é 15.

Em um espaço de probabilidade discreto definido por Ω e $\mathbb{P}$, a **probabilidade condicional** do

evento A dado que ocorre o evento E, em que $\mathbb{P}(E) \neq 0$, é definida por

$$\mathbb{P}(A \mid E) := \frac{\mathbb{P}(A \cap E)}{\mathbb{P}(E)} \tag{1.8}$$

e $\mathbb{P}(A \mid E)$ é lido como a **probabilidade de** A **dado** E.

Por exemplo, se Ω é finito com medida de probabilidade uniforme e $E \neq \varnothing$ então

$$\mathbb{P}(A \mid E) = \frac{\mathbb{P}(A \cap E)}{\mathbb{P}(E)} = \frac{|A \cap E|}{|E|}$$

que é, essencialmente, a medida uniforme em E. No exemplo acima, do par de dados, usando a definição de probabilidade condicional com S e Q eventos do modelo probabilístico discreto $(\Omega, \mathbb{P})$ para o lançamento dos dados

$$\mathbb{P}(S \mid Q) = \frac{\mathbb{P}(S \cap Q)}{\mathbb{P}(Q)} = \frac{|\{(6,9)\}|}{|\{(i,j)\colon i+j=15\}|} = \frac{1}{12}.$$

Exercício 1.20. Considere um espaço de probabilidade $(\Omega, \mathcal{A}, \mathbb{P})$ e $E \in \mathcal{A}$ um evento com probabilidade positiva. Verifique que $\mathbb{P}_E(A) := \mathbb{P}(A \mid E)$ é uma medida de probabilidade para os eventos em $\mathcal{A}$ (i.e, satisfaz os axiomas de probabilidade da página 13). Verifique, também, que $(E, \{A \cap E\colon A \in \mathcal{A}\}, \mathbb{P}_E)$ é um espaço de probabilidade.

Exemplo 1.21. Uma urna tem 20 bolas azuis e 10 bolas brancas. Das bolas azuis, 5 têm a letra X e 15 têm a letra Y gravada nelas; das bolas brancas, 1 têm a letra X e 9 tem a letra Y. Uma bola é escolhida ao acaso. Qual é a probabilidade dessa bola ser azul e com a letra X? Se A representa o evento "bola azul" e X o evento "letra X" então $\mathbb{P}(X \mid A) = 5/20 = 1/4$, que é a proporção de bolas azuis com a letra X. Usando a equação (1.8) podemos deduzir que a probabilidade de sortear uma bola azul e com a letra X é $\mathbb{P}(A \cap X) = \mathbb{P}(X \mid A) \cdot \mathbb{P}(A) = (1/4) \cdot (20/30) = 1/6$. ◊

O teorema da multiplicação e experimentos compostos. A igualdade $\mathbb{P}(A \cap E) = \mathbb{P}(A \mid E) \cdot \mathbb{P}(E)$ usada no exemplo 1.21 é consequência direta da definição de probabilidade condicional e é conhecida como **teorema da multiplicação** ou regra da multiplicação. Um caso geral dela é dado no exercício 1.24 abaixo. Nos próximos exemplos ilustramos como o teorema da multiplicação pode ser usado para definir um modelo probabilístico para um experimento composto por dois ou mais experimentos aleatórios, como foi o caso do modelo probabilístico para o problema de Monty Hall, exemplo 1.14.

Consideremos três urnas, digamos A, B e C, cada uma com a mesma probabilidade de ser escolhida, 1/3. Em cada uma das urnas há seis bolas, cada uma com a mesma probabilidade de ser escolhida, 1/6. Na urna A temos três bolas pretas e três bolas vermelhas; na urna B temos duas bolas pretas e quatro vermelhas; na urna C todas as bolas são pretas. Uma urna é escolhida aleatoriamente

e, em seguida, uma bola é escolhida aleatoriamente e observamos a cor dessa bola. Vamos definir um modelo probabilístico discreto $(\Omega, \mathbb{P})$ para esse *experimento composto* de modo que $\mathbb{P}$ respeita as probabilidades em cada experimento num sentido que ficará claro abaixo.

Temos dois experimentos aleatórios, o primeiro consiste de sortear uma urna e o segundo e sortear uma bola da urna que foi escolhida. Para o primeiro experimento temos o modelo discreto $(\Omega_1, \mathbb{P}_1)$ dado pelo espaço amostral $\Omega_1 = \{A, B, C\}$ e a medida de probabilidade uniforme $\mathbb{P}_1$. Para o segundo experimento tomamos $(\Omega_2, \mathbb{P}_2)$ com o espaço amostral $\Omega_2 = \{V, P\}$ em que usamos os eventos atômicos (pontos amostrais) $V \in \Omega_2$ para representar "bola vermelha" e $P \in \Omega_2$ para representar "bola preta" e cujas probabilidades dependem da urna e são dadas na tabela 1.2 abaixo, mas que por abuso de notação escrevemos $\mathbb{P}_2$ em todos os casos.

urna	$\mathbb{P}_2(V)$	$\mathbb{P}_2(P)$
A	1/2	1/2
B	2/3	1/3
C	0	1

Tabela 1.2: probabilidade dos eventos "bola vermelha" e "bola preta" em cada urna.

Um espaço amostral para o experimento composto pelos dois sorteios é $\Omega := \Omega_1 \times \Omega_2 = \{A, B, C\} \times \{V, P\}$. A probabilidade $\mathbb{P}$ é de tal modo que $\mathbb{P}(E \times \Omega_2) = \mathbb{P}_1(E)$ e $\mathbb{P}(\Omega_1 \times F) = \mathbb{P}_2(F)$.

Agora, por exemplo, o evento "urna A" é representado no primeiro experimento e no experimento composto por, respectivamente

$$U_{\Omega_1} = \{A\} \quad \text{e} \quad U_\Omega = \{A\} \times \Omega_2 = \{(A, V), (A, P)\}$$

de modo que para a medida $\mathbb{P}_\Omega$ em Ω temos

$$\mathbb{P}(U_\Omega) = \mathbb{P}(\{A\} \times \Omega_2) = \mathbb{P}_1(U_{\Omega_1}) = \frac{1}{3}.$$

O evento "bola preta" é modelado em Ω por $E_\Omega = \Omega_1 \times \{P\} = \{(A, P), (B, P), (C, P)\}$ de modo que

$$\mathbb{P}(E_\Omega) = \mathbb{P}(\Omega_1 \times \{P\}) = \mathbb{P}((A, P)) + \mathbb{P}((B, P)) + \mathbb{P}((C, P))$$

entretanto, diferente do caso anterior, o resultado do segundo experimento depende do resultado do primeiro; o que conhecemos são as probabilidades condicionais de "cor" dado "urna". O ponto amostral (A, P) de Ω é dado por

$$(\Omega_1 \times \{P\}) \cap (\{A\} \times \Omega_2) = E_\Omega \cap U_\Omega \tag{1.9}$$

e tem probabilidade dada pelo teorema da multiplicação da seguinte forma

$$\mathbb{P}(E_\Omega \cap U_\Omega) = \mathbb{P}(E_\Omega \mid U_\Omega)\mathbb{P}(U_\Omega) = \mathbb{P}_2(E_{\Omega_2})\mathbb{P}_1(U_{\Omega_1}) = \frac{1}{2} \cdot \frac{1}{3} \tag{1.10}$$

pois dado que ocorre "urna A", a probabilidade de "bola preta" é $\mathbb{P}(\,E_\Omega \mid U_\Omega\,) = \mathbb{P}_2(\,P_{\Omega_2}\,) = 1/2$.

Podemos determinar de maneira análoga a probabilidade de todo ponto amostral de Ω, cada um é dado por um interseção e pode ser escrito como na equação (1.9) e a probabilidade é calculada pelo teorema da multiplicação como na equação (1.10).

Quando o espaço amostral é pequeno, como nesse exemplo, pode ser conveniente descrevermos o modelo probabilístico através de um diagrama de árvore como o da figura 1.4 abaixo e como já fizemos para Monty Hall (figura 1.3, pág. 19). O diagrama de árvore da figura 1.4 representa cada

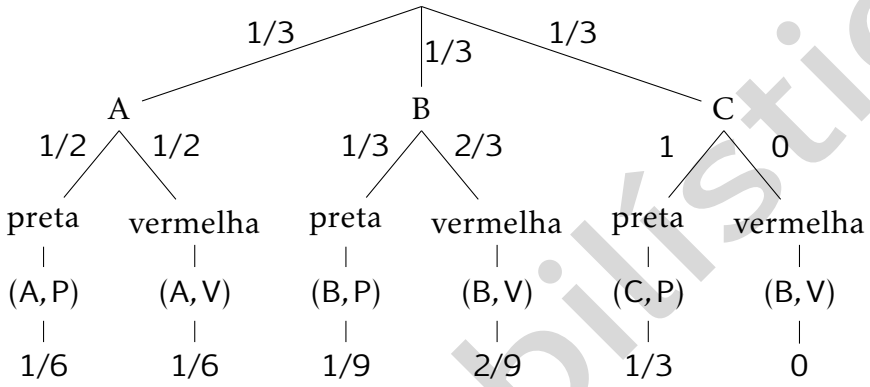


Figura 1.4: diagrama de árvore.

etapa do experimento em um nível da árvore, com as respectivas probabilidades nas ramificações correspondentes aos resultados de cada etapa. A partir do segundo nível essas probabilidades são condicionadas ao que ocorreu na etapas anteriores. Uma maneira prática de calcular a probabilidade a um ponto amostral é tomar o produto das probabilidades no caminho até ele nessa árvore, por exemplo, $\mathbb{P}((A,P)) = 1/3 \cdot 1/2$.

Como o modelo é discreto, estendemos a probabilidade para qualquer subconjunto de $\Omega_1 \times \Omega_2$ somando a probabilidade de seus elementos. Por exemplo, o evento dado por "a bola sorteada é preta" ocorre com probabilidade

$$\mathbb{P}((A,P)) + \mathbb{P}((B,P)) + \mathbb{P}((C,P)) = \frac{11}{18}.$$

Notemos que essa probabilidade não depende do número de bolas pretas na urna C, portanto, embora a medida de probabilidade em cada experimento seja uniforme a probabilidade de "a bola sorteada é preta" não é a quantidade de bolas pretas dividido pelo número total de bolas, que é um erro cometido frequentemente nesse caso. ◊

Exemplo 1.22. Numa cômoda há três gavetas e em cada gaveta um par de meias. Na primeira gaveta há um par de meias brancas, na segunda um par de meias pretas e na terceira gaveta um par com um pé de cada cor. Uma gaveta é escolhida uniformemente e, sem olhar para o interior da gaveta,

um pé de meia é escolhido uniformemente e em seguida a gaveta é fechada. O pé de meia retirado é branco. Qual a probabilidade de o outro pé que ficou sozinho na gaveta ser preto?

Vamos denotar por B o evento "retirou uma meia branca" e por T o evento "ficou uma meia preta", ambos eventos do experimento composto por dois experimentos realizados consecutivamente. Queremos determinar $\mathbb{P}(\,T \mid B\,)$.

Em metade dos resultados possíveis a meia escolhida é branca, portanto $\mathbb{P}(B) = 1/2$. A probabilidade de ocorrer ambos os eventos é a probabilidade de um ponto amostral específico que corresponde a abrir a terceira gaveta e retirar uma meia branca, o que ocorre com probabilidade $\mathbb{P}(T \cap B) = \mathbb{P}(\,B \mid T\,)\mathbb{P}(T)$ pela regra da multiplicação. A probabilidade condicional $\mathbb{P}(\,B \mid T\,)$ corresponde a sortear no segundo experimento uma meia branca na terceira gaveta, o que ocorre com probabilidade 1/2 e $\mathbb{P}(T)$ corresponde a probabilidade de sortear a terceira gaveta no primeiro experimento, o que ocorre com probabilidade 1/3. Logo, $\mathbb{P}(T \cap B) = 1/2 \cdot 1/3 = 1/6$. Portanto, segue da definição que $\mathbb{P}(\,T \mid B\,) = 1/3$. ◊

Exercício 1.23. Verifique a seguinte igualdade para o teorema da multiplicação com três eventos

$$\mathbb{P}(A \cap B \cap C) = \mathbb{P}(A) \cdot \mathbb{P}(\,B \mid A\,) \cdot \mathbb{P}(\,C \mid A \cap B\,) \tag{1.11}$$

e identifique seu uso no exemplo 1.14, o modelo probabilístico para o problema de Monty Hall.

Exercício 1.24 (teorema da multiplicação). Sejam $A_1, A_2, \ldots, A_n$ eventos de um modelo probabilístico. Prove que

$$\mathbb{P}\left(\bigcap_{i=1}^{n} A_i\right) = \mathbb{P}(A_1) \prod_{i=2}^{n} \mathbb{P}\left(\,A_i \;\middle|\; \bigcap_{j=1}^{i-1} A_j\right)$$

sempre que as probabilidades condicionais estão definidas (veja que é suficiente pedir que $\mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_{n-1}) > 0$).

Exercício 1.25. Sejam A, B e C eventos de um mesmo espaço amostral. Verifique que vale a seguinte igualdade

$$\mathbb{P}(\,C \cap A \mid B\,) = \mathbb{P}(\,C \mid A \cap B\,)\mathbb{P}(\,A \mid B\,)$$

sempre que as condicionais estão definidas.

1.3.1 OS TEOREMAS DA PROBABILIDADE TOTAL E DE BAYES

Se E e A são eventos, com $0 < \mathbb{P}(E) < 1$, então o evento A ocorre se, e somente se, ocorre (A e E) ou (A e $\overline{E}$) e esses eventos entre parênteses são disjuntos; mais que isso $\{A \cap E, A \cap \overline{E}\}$ é uma partição de

A, portanto,

$$\begin{aligned}\mathbb{P}(A) &= \mathbb{P}\Big((A\cap E)\cup(A\cap \overline{E})\Big)\\ &= \mathbb{P}(A\cap E)+\mathbb{P}(A\cap \overline{E})\\ &= \mathbb{P}(A\mid E)\mathbb{P}(E)+\mathbb{P}(A\mid \overline{E})\mathbb{P}(\overline{E}).\end{aligned} \tag{1.12}$$

No problema de Monty Hall se o convidado fica com a porta que escolheu inicialmente, então a probabilidade de ganhar um carro é 1/3, que é a probabilidade dele ter escolhido a porta certa logo de início. Agora, vamos supor que o convidado troca de porta. Nesse caso, denotamos por A o evento "ganha o carro" e por E o evento "a porta escolhida na primeira etapa esconde o carro". Claramente, $\mathbb{P}(A\mid E)=0$ e $\mathbb{P}(E)=1/3$. Se a primeira escolha não era a correta então o convidado ganha o carro, ou seja, $\mathbb{P}(A\mid \overline{E})=1$. Com isso temos por (1.12)

$$\mathbb{P}(A)=\mathbb{P}(A\mid E)\mathbb{P}(E)+\mathbb{P}(A\mid \overline{E})\mathbb{P}(\overline{E})=0\frac{1}{3}+1\frac{2}{3}=\frac{2}{3}$$

portanto, é melhor trocar de porta.

O caso geral dessa igualdade é conhecido como o Teorema da Probabilidade Total. Segue da dedução acima e usando indução em n que se $\{E_1,E_2,\ldots,E_n\}$ é um conjunto de eventos que particionam o espaço amostral Ω com $\mathbb{P}(E_i)>0$ para todo $i\geqslant 1$, então vale

$$\mathbb{P}(A)=\sum_{i=1}^{n}\mathbb{P}(A\cap E_i)=\sum_{i=1}^{n}\mathbb{P}(A\mid E_i)\mathbb{P}(E_i) \tag{1.13}$$

para qualquer evento A. Deixamos a prova por conta do leitor, abaixo provamos um pouco mais, considerando partições enumeráveis.

Teorema 1.26 (Teorema da Probabilidade Total) *Seja $\{E_i : i\in\mathbb{N}\}$ um conjunto de eventos que particionam o espaço amostral Ω com $\mathbb{P}(E_i)>0$ para todo i. Então*

$$\mathbb{P}(A)=\sum_{i\geqslant 1}\mathbb{P}(A\cap E_i)=\sum_{i\geqslant 1}\mathbb{P}(A\mid E_i)\mathbb{P}(E_i) \tag{1.14}$$

para qualquer evento A.

Demonstração. Os conjuntos $A\cap E_i$, para $i\in\mathbb{N}$, são disjuntos dois a dois e a da união deles resulta A. A primeira igualdade em (1.14) segue da aditividade enumerável. A segunda igualdade segue de $\mathbb{P}(A\cap E_i)=\mathbb{P}(A\mid E_i)\mathbb{P}(E_i)$ para todo i. □

Exemplo 1.27 (Ross, 2010). As seguradoras de automóveis classificam motoristas em *mais propensos a acidentes* e *menos propensos a acidentes*. Com isso estimam que os mais propensos são 30% da população e que esses se envolvem em acidente no período de um ano com probabilidade 0,4, enquanto

que os menos propensos a acidentes se envolvem em acidente no período de um ano com probabilidade 0,2. Denotemos por A^+ o evento definido pelos motoristas mais propensos a acidentes. Então a probabilidade de um novo segurado se envolver em acidente em um ano é

$$\mathbb{P}(A_1) = \mathbb{P}(A_1 \mid A^+)\mathbb{P}(A^+) + \mathbb{P}(A_1 \mid \overline{A^+})\mathbb{P}(\overline{A^+}) = 0{,}4 \cdot 0{,}3 + 0{,}2 \cdot 0{,}7 = 0{,}26$$

e se um novo segurado se envolve em acidente nesse prazo, a probabilidade dele ser propenso a acidentes é

$$\mathbb{P}(A^+ \mid A_1) = \frac{\mathbb{P}(A_1 \cap A^+)}{\mathbb{P}(A_1)} = \frac{\mathbb{P}(A_1 \mid A^+)\mathbb{P}(A^+)}{\mathbb{P}(A_1)} = \frac{0{,}4 \cdot 0{,}3}{0{,}26} = \frac{6}{13}.$$

Portanto, a probabilidade de ser propenso a acidente dado que se envolve em acidente em um ano é 0,46, aproximadamente. Dado que o motorista não se envolve em acidente em um ano, a probabilidade de ser propenso a acidente é $\mathbb{P}(A^+ \mid \overline{A}_1) \approx 0{,}24$. ◊

Urna de Pólya. Uma urna contém duas bolas, uma branca e uma preta. Em cada instante $t \in \{1, 2, \dots\}$ sorteamos uma bola da urna. A bola sorteada é devolvida para a urna junto com uma outra bola da mesma cor dessa sorteada. Assim, o t-ésimo sorteio ($t \geqslant 1$) ocorre com $t+1$ bolas na urna. Neste exemplo nós vamos calcular a a probabilidade com que uma bola preta é sorteada em cada instante.

Seja P_t ($t \geqslant 1$) o evento "a t-ésima bola sorteada é preta"; se não é sorteada uma bola preta então é sorteada uma bola branca, cujo evento é denotado por B_t. Certamente,

$$\mathbb{P}(P_1) = \frac{1}{2}.$$

Pelo teorema de probabilidade total (equação (1.12)) $\mathbb{P}(P_2) = \mathbb{P}(P_2 \mid P_1)\mathbb{P}(P_1) + \mathbb{P}(P_2 \mid B_1)\mathbb{P}(B_1)$ e se ocorre P_1, então para o segundo sorteio há 2 bolas pretas dentre 3 bolas, portanto, $\mathbb{P}(P_2 \mid P_1) = 2/3$ e, analogamente, $\mathbb{P}(P_2 \mid B_1) = 1/3$, de modo que

$$\mathbb{P}(P_2) = \frac{2}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{2}.$$

Para computar $\mathbb{P}(P_3)$ precisamos de um pouco mais de esforço. Pelo teorema da probabilidade total, equação (1.13), temos

$$\mathbb{P}(P_3) = \mathbb{P}((P_1 \cap P_2) \cap P_3) + \mathbb{P}((P_1 \cap B_2) \cap P_3) + \mathbb{P}((B_1 \cap P_2) \cap P_3) + \mathbb{P}((B_1 \cap B_2) \cap P_3)$$

e cada termo dessa soma pode ser computado pelo teorema da multiplicação (especificamente, equação (1.11) na página 30), por exemplo

$$\mathbb{P}(P_1 \cap P_2 \cap P_3) = \mathbb{P}(P_1)\mathbb{P}(P_2 \mid P_1)\mathbb{P}(P_3 \mid P_1 \cap P_2) = \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{3}{4}$$

de modo que temos

$$\begin{aligned} \mathbb{P}(P_3) &= \mathbb{P}((P_1 \cap P_2) \cap P_3) + \mathbb{P}((P_1 \cap B_2) \cap P_3) + \mathbb{P}((B_1 \cap P_2) \cap P_3) + \mathbb{P}((B_1 \cap B_2) \cap P_3) \\ &= \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{3}{4} + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{2}{4} + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{2}{4} + \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{4} = \frac{1}{2}. \end{aligned} \tag{1.15}$$

Até aqui, $\mathbb{P}(P_1) = \mathbb{P}(P_2) = \mathbb{P}(P_3) = 1/2$.

Notemos que $\mathbb{P}(B_1 \cap B_2 \cap P_3) = \mathbb{P}(P_1 \cap P_2 \cap B_3)$ e que $\mathbb{P}(B_1 \cap P_2 \cap P_3) = \mathbb{P}(P_1 \cap B_2 \cap B_3)$ de modo que, substituindo em na equação (1.15) acima

$$\mathbb{P}(P_3) = \mathbb{P}(P_1 \cap (P_2 \cap P_3)) + \mathbb{P}(P_1 \cap (B_2 \cap P_3)) + \mathbb{P}(P_1 \cap (P_2 \cap B_3)) + \mathbb{P}(P_1 \cap (B_2 \cap B_3))$$

que por (1.13) é $\mathbb{P}(P_1)$ e, então, $\mathbb{P}(P_1) = \mathbb{P}(P_3)$. Tal simetria vale para qualquer t de modo que $\mathbb{P}(P_t) = \mathbb{P}(P_1)$ para todo $t \geqslant 1$. Vamos demonstrar esse fato.

Consideremos uma sequência de eventos $X_1, X_2, \ldots, X_{t-1}, P_t$ em que para cada i, $1 \leqslant i < t$, temos $X_t \in \{P_t, B_t\}$. As 2^{t-1} possíveis sequências de eventos $X_1, \ldots, X_{t-1}$ particionam o espaço amostral de modo que, pelo teorema da probabilidade total e o caso geral do teorema da multiplicação (exercício 1.24 na página 30), a probabilidade de P_t é

$$\sum_{(X_1,\ldots,X_{t-1})} \mathbb{P}(X_1 \cap X_2 \cap \cdots \cap X_{t-1} \cap P_t) = \sum_{(X_1,\ldots,X_{t-1})} \mathbb{P}(X_1) \cdot \left(\prod_{i=2}^{t-1} \mathbb{P}\left(X_i \,\middle|\, \bigcap_{j=1}^{i-1} X_j \right) \right) \cdot \mathbb{P}\left(P_t \,\middle|\, \bigcap_{j=1}^{t-1} X_j \right) \tag{1.16}$$

em que a soma é sobre todas as 2^{t-1} sequências $X_1, \ldots, X_{t-1}$ de eventos. Os somandos no lado direito da equação (1.16) são

$$\mathbb{P}(X_1)\mathbb{P}(X_2 \mid X_1)\mathbb{P}(X_3 \mid X_1 \cap X_2) \cdots \mathbb{P}(P_t \mid X_1 \cap \cdots \cap X_{t-1}) = \prod_{i=1}^{t} \frac{n_i}{i+1} \tag{1.17}$$

onde n_i é a quantidade de bolas da cor X_i sorteada no i-ésimo sorteio e os denominadores são $i+1$ porque em cada sorteio o número total de bolas aumenta de 1.

Fixado um instante t e supondo que até esse instante tenham sido sorteadas m bolas brancas e, portanto, $t-m$ bolas pretas, sejam $1 \leqslant t_1 < t_2 < \cdots < t_m \leqslant t$ os instantes em que ocorrem sorteio de bola branca. Quando foi realizado o primeiro sorteio de uma bola branca, havia uma bola branca de modo que $n_{t_1} = 1$, no segundo sorteio $n_{t_2} = 2$, e assim por diante, até o último sorteio de bola branca no instante t_m quando $n_{t_m} = m$. No momentos em que não foram sorteados bolas brancas foram sorteados bolas pretas, sejam $1 \leqslant s_1 < s_2 < \cdots < s_{t-m} \leqslant t$ tais instantes em que ocorrem sorteio de bolas pretas. De modo análogo temos que $n_{s_1} = 1$, $n_{s_2} = 2$, $\ldots$, $n_{s_{t-m}} = t-m$.

Agora, notemos que nos numeradores no lado direito da equação (1.17) ocorrem os números $1, 2, \ldots, m$ e $1, 2, \ldots, t-m$ de modo que o fator determinante no cálculo é a probabilidade de ocorrer m sorteios de bolas brancas e $t-m$ sorteios de bolas pretas, a ordem não importa. Dessa observação concluímos que os somandos no lado direito da equação (1.16) são

$$\frac{1}{2} \cdot \frac{2}{3} \cdots \frac{m}{m+1} \cdot \frac{1}{m+2} \cdot \frac{2}{m+3} \cdots \frac{t-m}{t+1} = \frac{m!(t-m)!}{(t+1)!} = \frac{1}{(t+1)\binom{t}{m}}. \tag{1.18}$$

Há $\binom{t-1}{m}$ sequências $X_1, X_2, \ldots, X_{t-1}$ de eventos com m posições correspondentes ao sorteio de bola

branca e cada uma tem probabilidade dada pela equação (1.18), portanto

$$\mathbb{P}(\mathrm{P}_t) = \sum_{m=0}^{t-1} \binom{t-1}{m} \frac{1}{(t+1)\binom{t}{m}} = \sum_{m=0}^{t-1} \frac{1}{(t+1)} \frac{t-m}{t} = \frac{1}{t(t+1)} \sum_{m=1}^{t} m = \frac{1}{2}$$

ou seja, $\mathbb{P}(\mathrm{P}_t) = 1/2$ para todo $t \geqslant 1$. ◊

Um fato interessante que deduzimos do exemplo acima é que usando equação (1.18) podemos concluir que a probabilidade de haver m bolas brancas após t-ésimo sorteio é

$$\mathbb{P}[\text{há } m \text{ bolas brancas após } t\text{-ésimo sorteio}] = \binom{t}{m} \frac{1}{(t+1)\binom{t}{m}} = \frac{1}{t+1}$$

que não depende de m.

Teorema de Bayes. Seja $\mathrm{E}_1, \ldots, \mathrm{E}_n$ uma partição do espaço amostral. Se soubermos que o evento A ocorre, então qual é a probabilidade (condicionada) com que E_j tenha ocorrido? Para todo j

$$\mathbb{P}(\mathrm{E}_j \mid \mathrm{A}) = \frac{\mathbb{P}(\mathrm{A} \mid \mathrm{E}_j)\mathbb{P}(\mathrm{E}_j)}{\mathbb{P}(\mathrm{A})}$$

usando o teorema da probabilidade total obtemos

$$\mathbb{P}(\mathrm{E}_j \mid \mathrm{A}) = \frac{\mathbb{P}(\mathrm{A} \mid \mathrm{E}_j)\mathbb{P}(\mathrm{E}_j)}{\sum_{i=1}^{n} \mathbb{P}(\mathrm{A} \mid \mathrm{E}_i)\mathbb{P}(\mathrm{E}_i)}$$

para todo evento com probabilidade positiva A. Esse resultado é conhecido como teorema de Bayes.

Teorema 1.28 (Teorema de Bayes) *Se* $\{\mathrm{E}_i : i \in \mathbb{N}\}$ *é uma partição do espaço amostral com* $\mathbb{P}(\mathrm{E}_i) > 0$ *para todo* i *e* $\mathbb{P}(\mathrm{A}) > 0$, *então*

$$\mathbb{P}(\mathrm{E}_j \mid \mathrm{A}) = \frac{\mathbb{P}(\mathrm{A} \mid \mathrm{E}_j)\mathbb{P}(\mathrm{E}_j)}{\sum_{i \geqslant 1} \mathbb{P}(\mathrm{A} \mid \mathrm{E}_i)\mathbb{P}(\mathrm{E}_i)}.$$

Suponha que *probabilite* é um vírus que afeta 10% da população de estudantes universitários. Um professor de Probabilidade aplica um teste que detecta *probabilite* mas eventualmente se engana: 3% de falsos positivos e 1% de falsos negativos. Se for detectado *probabilite* em um indivíduo escolhido ao acaso, qual é a probabilidade que ele tenha o vírus? Queremos determinar $\mathbb{P}(\mathrm{B} \mid \mathrm{A})$ onde A é o evento "foi detectado *probabilite*" e B o evento "tem *probabilite*". Usando o teorema de Bayes com $\mathbb{P}(\mathrm{A} \mid \mathrm{B}) = 0{,}99$, pois $\mathbb{P}(\overline{\mathrm{A}} \mid \mathrm{B}) = 0{,}01$ é a chance de ocorrer um falso negativo, e $\mathbb{P}(\mathrm{A} \mid \overline{\mathrm{B}}) = 0{,}03$ é a chance de ocorrer um falso positivo

$$\mathbb{P}(\mathrm{B} \mid \mathrm{A}) = \frac{\mathbb{P}(\mathrm{A} \mid \mathrm{B})\mathbb{P}(\mathrm{B})}{\mathbb{P}(\mathrm{A} \mid \mathrm{B})\mathbb{P}(\mathrm{B}) + \mathbb{P}(\mathrm{A} \mid \overline{\mathrm{B}})\mathbb{P}(\overline{\mathrm{B}})} = \frac{0{,}99 \cdot 0{,}1}{0{,}99 \cdot 0{,}1 + 0{,}03 \cdot 0{,}9} \approx 0{,}78.$$

Agora, supondo que *probabilite* seja uma contaminação muito rara, que afeta só 1,05% da população universitária, e que o teste seja um pouco mais acurado, só há 1% de chance de falsos positivos e falsos negativos, então

$$\mathbb{P}(\mathrm{B} \mid \mathrm{A}) = \frac{0{,}99 \cdot 0{,}0105}{0{,}99 \cdot 0{,}0105 + 0{,}01 \cdot 0{,}9895} \approx 0{,}51$$

logo o teste é essencialmente tão efetivo quanto decidir lançando uma moeda. Se o vírus for ainda mais raro, digamos que apena 0,5% da população universitária tenha o vírus. Assim,

$$\mathbb{P}(\,\mathsf{B} \mid \mathsf{A}\,) = \frac{0{,}99 \cdot 0{,}005}{0{,}99 \cdot 0{,}005 + 0{,}01 \cdot 0{,}995} \approx 0{,}34$$

ou seja, se o teste do professor detectou *probabilite* em um estudante é duas vezes mais provável que o indivíduo não tenha *probabilite*. Esse resultado aparentemente paradoxal ocorre porque o número de indivíduos não contaminados pelo *probabilite* é muito grande em relação ao número de contaminados, de modo que a quantidade de falsos positivos supera a quantidade positivos verdadeiros. Se 100.000 indivíduos forem testados, esperamos que aproximadamente 99.500 não tenham *probabilite* mas que ocorram $0{,}01 \cdot 99.500 \approx 1.000$ falsos positivos; também, esperamos que 500 indivíduos tenham *probabilite* e deles $0{,}99 \cdot 500 \approx 500$ verdadeiros positivos, portanto, 1/3 dos resultados positivos são genuínos.

Filtro bayesiano para mensagens eletrônicas indesejáveis *(Spam)*. O uso de técnicas baseadas no teorema de Bayes para classificar mensagens eletrônicas (*emails*) surgiu em 1996 num trabalho de Jason Rennie chamado *Ifile* e ganhou impulso com o ensaio de Graham (2002).

Previamente identificamos algumas características das mensagens que estão classificadas em dois conjuntos, as que são *spam* e as que não são *spam*, da seguinte forma. A frequência com que ocorre uma palavra no conjunto das mensagens que são *spam* define uma probabilidade da palavra condicionada à mensagem ser um *spam* e as palavras características de *spam* são as mais relevantes de acordo com essas probabilidades. Por exemplo, nas minhas mensagens muitos dos *spam*s têm a palavra *watch* enquanto que muitos dos não *spam*s têm a palavra "reunião"; a maioria das mensagens têm a palavra "a", tantos *spam*s quanto não *spam*s, logo "a" não deve ser uma característica classificatória; separamos as palavras tais que $\mathbb{P}[\,\text{palavra} \mid \mathit{spam}\,]$ seja bem maior que 1/2 e $\mathbb{P}[\,\text{palavra} \mid \text{não } \mathit{spam}\,]$ seja bem menor que 1/2. Ao final temos algumas características classificatórias, digamos n características, que podem estar ou não estar presentes nas mensagens futuras e que vão ajudar a classificá-las.

Dadas as n características, cada mensagem fica associada uma sequência binária de $\Omega := \{0,1\}^n$ em que cada coordenada da sequência correspondente indica se a mensagem tem ou não tem uma determinada característica. A primeira coordenada, especificamente, é 1 se a mensagem é *spam* e 0 caso, contrário. Assim, $(1,0,0,1,1)$ corresponde a uma mensagem que é *spam* não tem as características 2 e 3, mas tem as características 4 e 5. Denotemos por S o evento "*spam*" e por C_i o evento "tem a característica i". Na classificação prévia contamos a quantidade k_i de mensagens *spam* que têm a característica i dentre as K mensagens classificadas. Também, determinamos a quantidade ℓ_i de mensagens não *spam* que têm a característica i dentre as L mensagens classificadas. Com essa

informação determinamos

$$\mathbb{P}(\,C_i \mid S\,) = \frac{k_i}{K} \quad \text{e} \quad \mathbb{P}(\,C_i \mid \overline{S}\,) = \frac{\ell_i}{L}$$

para cada característica $i > 1$. Pelo teorema de Bayes, a probabilidade de uma mensagem que apresenta a característica i ser *spam* é

$$p_i := \mathbb{P}(\,S \mid C_i\,) = \frac{\mathbb{P}(\,C_i \mid S\,)\mathbb{P}(S)}{\mathbb{P}(\,C_i \mid S\,)P(S) + \mathbb{P}(\,C_i \mid \overline{S}\,)\mathbb{P}(\overline{S})}.$$

Se assumimos que, a priori, temos a mesma chance de receber um *spam* quanto um não *spam* então $\mathbb{P}(S) = \mathbb{P}(\overline{S}) = 1/2$ e assumindo $K = L$, para simplificar, a equação acima se resume a

$$p_i = \frac{(k_i/K)\mathbb{P}(S)}{(k_i/K)\mathbb{P}(S) + (\ell_i/L)\mathbb{P}(\overline{S})} = \frac{k_i}{k_i + \ell_i}.$$

Recebida uma mensagem como classificá-la? Determinamos quais das n características estão presentes na mensagem, digamos que para algum subconjunto de índices I a mensagem tem C_i para todo $i \in I$ e com essa informação calculamos $\mathbb{P}(\,S \,\big|\, \bigcap_{i\in I} C_i\,)$. Se essa probabilidade for maior que um limiar $\varepsilon \in (0,1)$ estabelecido, então a mensagem recebida é classificada como *spam*, senão é classificada como não *spam*. No que segue vamos provar que a probabilidade dessa mensagem ser *spam* é dada por

$$\mathbb{P}\left(\, S \,\middle|\, \bigcap_{i\in I} C_i \right) = \frac{\prod_{i\in I} p_i}{\prod_{i\in I} p_i + \prod_{i\in I}(1-p_i)}. \tag{1.19}$$

Para isso vamos assumir que valem

$$\mathbb{P}\left(\, \bigcap_{i\in I} C_i \,\middle|\, S \right) = \prod_{i\in I} \mathbb{P}(\,C_i \mid S\,) \tag{1.20}$$

$$\mathbb{P}\left(\, \bigcap_{i\in I} C_i \,\middle|\, \overline{S} \right) = \prod_{i\in I} \mathbb{P}(\,C_i \mid \overline{S}\,), \tag{1.21}$$

para todo $I \subset \{2, 3, \ldots, n\}$, o que pode não ser uma hipótese muito realista. Usando a definição de probabilidade condicional

$$\mathbb{P}\left(\, S \,\middle|\, \bigcap_{i\in I} C_i \right) = \frac{\mathbb{P}(\,\bigcap_{i\in I} C_i \mid S\,)\mathbb{P}(S)}{\mathbb{P}(\bigcap_{i\in I} C_i\,)}$$

e da lei da probabilidade total

$$\frac{\mathbb{P}(\,\bigcap_{i\in I} C_i \,\big|\, S\,)\mathbb{P}(S)}{\mathbb{P}(\bigcap_{i\in I} C_i\,)} = \frac{\mathbb{P}(\,\bigcap_{i\in I} C_i \,\big|\, S\,)\mathbb{P}(S)}{\mathbb{P}(\,\bigcap_{i\in I} C_i \,\big|\, S\,)\mathbb{P}(S) + \mathbb{P}(\,\bigcap_{i\in I} C_i \mid \overline{S}\,)\mathbb{P}(\overline{S})}$$

e pelas equações (1.20) e (1.21)

$$\frac{\mathbb{P}(\,\bigcap_{i\in I} C_i \,\big|\, S\,)\mathbb{P}(S)}{\mathbb{P}(\,\bigcap_{i\in I} C_i \,\big|\, S\,)\mathbb{P}(S) + \mathbb{P}(\,\bigcap_{i\in I} C_i \mid \overline{S}\,)\mathbb{P}(\overline{S})} = \frac{\prod_{i\in I}\mathbb{P}(\,C_i \mid S\,)\mathbb{P}(S)}{\prod_{i\in I}\mathbb{P}(\,C_i \mid S\,)\mathbb{P}(S) + \prod_{i\in I}\mathbb{P}(\,C_i \mid \overline{S}\,)\mathbb{P}(\overline{S})}$$

e, usando que $\mathbb{P}(C_i \mid S) = \mathbb{P}(S \mid C_i)\mathbb{P}(C_i)/\mathbb{P}(S)$ e a igualdade análoga para $\overline{S}$

$$\frac{\prod_{i\in I}\mathbb{P}(C_i \mid S)\mathbb{P}(S)}{\prod_{i\in I}\mathbb{P}(C_i \mid S)\mathbb{P}(S)+\prod_{i\in I}\mathbb{P}(C_i \mid \overline{S})\mathbb{P}(\overline{S})} = \frac{\prod_{i\in I}\mathbb{P}(S \mid C_i)}{\prod_{i\in I}\mathbb{P}(S \mid C_i)+\prod_{i\in I}\mathbb{P}(\overline{S} \mid C_i)}$$

donde seque a equação (1.19).

As hipóteses assumidas nas equações (1.20) e (1.21) significam, grosso modo, que o conhecimento de algumas das características não dá nenhuma pista sobre a presença ou não das outras características; estamos assumindo independência dos eventos e o significado preciso disso é o assunto da próxima seção.

1.4 INDEPENDÊNCIA DE EVENTOS

Se um dado é lançado duas vezes, então temos

$$\mathbb{P}[\text{ a soma é } 7 \mid \text{o primeiro resultado é } 4\,] = \frac{1}{6} = \mathbb{P}[\text{a soma é } 7\,]$$

entretanto

$$\mathbb{P}[\text{ a soma é } 12 \mid \text{o primeiro resultado é } 4\,] = 0 \neq \mathbb{P}[\text{a soma é } 12\,].$$

O condicionamento de ocorrência de um evento A à ocorrência de B pode afetar ou não a probabilidade de ocorrência de A. Definimos que o evento A é independente do evento B se

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$$

Notemos que se A é independente de B então B é independente de A de modo que dizemos A e B são **eventos independentes**. .

Se os eventos têm probabilidade não nula então decorre da definição acima que a independência de A e B equivale a $\mathbb{P}(A \mid B) = \mathbb{P}(A)$ e $\mathbb{P}(B \mid A) = \mathbb{P}(B)$. É imediato da definição o seguinte fato.

Proposição 1.29 *Todo evento A de um modelo probabilístico é independente do evento certo Ω e do evento impossível $\varnothing$.* □

A independência dos eventos A e B resulta na independência entre seus complementos, como enunciado a seguir.

Proposição 1.30 *Se A e B são eventos independentes de um modelo probabilístico, então A e $\overline{B}$ são eventos independentes, $\overline{A}$ e B são eventos independentes e $\overline{A}$ e $\overline{B}$ são eventos independentes.*

Demonstração. Deduzimos de $\mathbb{P}(A) = \mathbb{P}(A \cap B) + \mathbb{P}(A \cap \overline{B})$, usando a independência de A e B, que

$$\mathbb{P}(A \cap \overline{B}) = \mathbb{P}(A) - \mathbb{P}(A \cap B) = \mathbb{P}(A) - \mathbb{P}(A)\mathbb{P}(B) = \mathbb{P}(A)(1 - \mathbb{P}(B)) = \mathbb{P}(A)\mathbb{P}(\overline{B})$$

portanto são eventos independentes. Os outros casos são demonstrados de modo análogo. □

Para investigar o caso de três eventos, voltemos ao experimento de um dado lançado duas vezes. Sejam A o evento "a soma do dois lançamentos é 7", B o evento "o primeiro lançamento resulta 4" e C o evento "o segundo lançamento resulta 2". Como vimos, A e B são eventos independentes. Por razão análoga A e C são independentes. Porém $\mathbb{P}(A \mid B \cup C) = 2/11 \neq \mathbb{P}(A)$ e $\mathbb{P}(A \mid B \cap C) = 0 \neq \mathbb{P}(A)$, ou seja, A não é independente de [B *ou* C] e não é independente de [B *e* C].

Para três eventos, digamos A, B e C, queremos que A seja independente do par de eventos $\{B, C\}$ quando o conhecimento de qualquer informação a respeito da ocorrência de B, de C, ou de uma combinação deles pelas operações elementares de conjuntos não altere a probabilidade de ocorrer A.

Exercício 1.31. Assuma, como definição de "A *é independente de* $\{B, C\}$" se vale a equação equação (1.22) a seguir

$$\mathbb{P}(A \mid B \cap C) = \mathbb{P}(A \mid B) = \mathbb{P}(A \mid C) = \mathbb{P}(A). \tag{1.22}$$

Prove que se A é independente de $\{B, C\}$ então A é independente de cada um dos eventos da família

$$\left\{\varnothing, B, C, \overline{B}, \overline{C}, B \cup C, B \cap C, B \cup \overline{C}, B \cap \overline{C}, \overline{B} \cup C, \overline{B} \cap C, \overline{B} \cup \overline{C}, \overline{B} \cap \overline{C}, \Omega\right\}$$

que chamamos de **espaço de eventos gerado** por $\{B, C\}$.

Em vista disso, definimos que A **é independente de** $\{B, C\}$ se for independente de todo evento do espaço de eventos gerado por $\{B, C\}$. Tal definição é equivalente a (veja a equação (1.22)): A *é independente de* $\{B, C\}$ *se, e somente se, é independente de* B, *é independente de* C *e é independente de* $B \cap C$, isto é,

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B),\ \mathbb{P}(A \cap C) = \mathbb{P}(A)\mathbb{P}(C) \text{ e } \mathbb{P}(A \cap B \cap C) = \mathbb{P}(A)\mathbb{P}(B \cap C).$$

Ademais, notemos que essa definição é compatível com a definição de "A independente de B" dada anteriormente pois, pelas proposições 1.29 e 1.30, o evento A é independente de todo evento do espaço de eventos gerado por $\{B\}$, o qual é $\{\varnothing, B, \overline{B}, \Omega\}$.

Em geral, estamos interessados no caso em que cada evento é independente dos outros dois eventos restantes e, nesse caso, dizemos que os eventos A, B e C são **mutuamente independentes** o que é equivalente a

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B),\ \mathbb{P}(A \cap C) = \mathbb{P}(A)\mathbb{P}(C),\ \mathbb{P}(B \cap C) = \mathbb{P}(B)\mathbb{P}(C) \text{ e } \mathbb{P}(A \cap B \cap C) = \mathbb{P}(A)\mathbb{P}(B)\mathbb{P}(C).$$

Consideremos três lançamentos de uma moeda equilibrada e os eventos E_{12} dado por "o resultado do primeiro e do segundo lançamentos coincidem", E_{13} dado por "o resultado do primeiro e do terceiro coincidem" e E_{23} dado por "o resultado do segundo e do terceiro coincidem". Cada um desses eventos tem probabilidade 1/2. Os eventos são independentes quando tomados dois-a-dois: $\mathbb{P}(E_{12} \cap E_{13}) = \mathbb{P}(\{(Ca, Ca, Ca), (Co, Co, Co)\}) = 1/4$ e, analogamente, os eventos $E_{12} \cap E_{23}$ e

$E_{13} \cap E_{23}$ têm probabilidade 1/4. Entretanto esses eventos não são mutuamente independentes pois $\mathbb{P}(E_{12} \cap E_{13} \cap E_{23}) = 1/4$ enquanto que $\mathbb{P}(E_{12})\mathbb{P}(E_{13})\mathbb{P}(E_{23}) = 1/8$.

Agora, num lançamento de dados tomamos os eventos $A = \{1,2,3,4\}$ e $B = C = \{4,5,6\}$. Os eventos B e C não são independentes. Também A e B não são independentes pois $\mathbb{P}(A \cap B) = 1/6$ enquanto que $\mathbb{P}(A)\mathbb{P}(B) = 1/3$, logo A e C não são eventos independentes. Porém $\mathbb{P}(A \cap B \cap C) = \mathbb{P}(A)\mathbb{P}(B)\mathbb{P}(C)$.

Uma coleção enumerável de eventos $\mathcal{E} = \{E_n\}$ é dita **mutuamente independente** se para todo subconjunto finito $J \subset \mathbb{N}$ vale que

$$\mathbb{P}\left(\bigcap_{\ell \in J} E_\ell\right) = \prod_{\ell \in J} \mathbb{P}(E_\ell).$$

Para $k \in \{2,\ldots,n\}$ fixo, dizemos que a coleção $\mathcal{E}$ é *k*-**a**-*k* **independente** se todo subconjunto de índices $J \subset \mathbb{N}$ com $|J| \leqslant k$ define uma subcoleção de eventos mutuamente independentes.

Independência condicional. Dizemos que A_1 e A_2 são **condicionalmente independentes** dado B se

$$\mathbb{P}(A_2 \cap A_1 \mid B) = \mathbb{P}(A_1 \mid B)\mathbb{P}(A_2 \mid B).$$

Essa definição estende-se naturalmente, como acima, para um coleção com mais que dois eventos.

As equações (1.20) e (1.21) no exemplo para filtros anti-*spam* pede que as características $C_1,\ldots,C_n$, que são usadas para classificar as mensagens, sejam independentes quando condicionamos à ocorrência de *spam* e quando condicionamos à ocorrência de não *spam*, respectivamente.

No contexto do exemplo 1.27, página 31, qual a probabilidade de um motorista se envolver num acidente no segundo ano dado que tenha se envolvido em acidente no primeiro ano de contrato? Denotemos por A_2 o evento "acidente no 2º ano de contrato". Assumiremos que A_1 e A_2 são condicionalmente independentes dado A^+, ou seja, $\mathbb{P}(A_2 \cap A_1 \mid A^+) = \mathbb{P}(A_1 \mid A^+)\mathbb{P}(A_2 \mid A^+)$. Se definirmos a medida de probabilidade $\mathbb{Q}(X) := \mathbb{P}(X \mid A_1)$, queremos determinar $\mathbb{Q}(A_2)$. Pelo teorema de probabilidade total $\mathbb{Q}(A_2) = \mathbb{Q}(A_2 \mid A^+)\mathbb{Q}(A^+) + \mathbb{Q}(A_2 \mid \overline{A^+})\mathbb{Q}(\overline{A^+})$. Mas

$$\mathbb{Q}(A_2 \,\big|\, A^+) = \frac{\mathbb{Q}(A_2 \cap A^+)}{\mathbb{Q}(A^+)} = \frac{\mathbb{P}(A_2 \cap A^+ \mid A_1)}{\mathbb{P}(A^+ \mid A_1)} = \mathbb{P}(A_2 \,\big|\, A^+ \cap A_1) = \mathbb{P}(A_2 \,\big|\, A^+)$$

em que a última igualdade segue da independência condicional assumida (verifique). Lembremos que os motoristas propensos a acidentes se envolvem em acidente no período de um ano com probabilidade 0,4, logo $\mathbb{P}(A_2 \mid A^+) = 0{,}4$. Ainda, calculamos no exemplo 1.27 que $\mathbb{Q}(A^+) = 6/13$. Desse modo temos que

$$\mathbb{Q}(A_2) = \mathbb{Q}(A_2 \,\big|\, A^+)\mathbb{Q}(A^+) + \mathbb{Q}(A_2 \mid \overline{A^+})\mathbb{Q}(\overline{A^+}) = 0{,}4 \cdot \frac{6}{13} + 0{,}2 \cdot \frac{7}{13} \approx 0{,}29.$$

Exemplo 1.32. Suponhamos que numa caixa há duas moedas, uma delas com duas caras e a outra é uma moeda comum. Uma moeda é sorteada e lançada duas vezes. Sejam A e B os eventos "o primeiro

lançamento é cara" e "o segundo lançamento é cara", respectivamente. Condicionados ao evento C definido por "a moeda normal foi a escolhida" os eventos A e B são independentes:

$$\mathbb{P}(A \cap B \mid C) = \frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2} = \mathbb{P}(A \mid C) \cdot \mathbb{P}(B \mid C).$$

Entretanto os eventos A e B não são independentes pois

$$\mathbb{P}(A) = \mathbb{P}(A \mid C)\mathbb{P}(C) + \mathbb{P}(A \mid \overline{C})\mathbb{P}(\overline{C}) = \frac{1}{2} \cdot \frac{1}{2} + 1\frac{1}{2} = \frac{3}{4}$$
$$\mathbb{P}(B) = \mathbb{P}(B \mid C)\mathbb{P}(C) + \mathbb{P}(B \mid \overline{C})\mathbb{P}(\overline{C}) = \frac{1}{2} \cdot \frac{1}{2} + 1\frac{1}{2} = \frac{3}{4}$$

porém

$$\begin{aligned}\mathbb{P}(A \cap B) &= \mathbb{P}(A \cap B \mid C)\mathbb{P}(C) + \mathbb{P}(A \cap B \mid \overline{C})\mathbb{P}(\overline{C}) \\ &= \mathbb{P}(A \mid C)\mathbb{P}(B \mid C)\mathbb{P}(C) + \mathbb{P}(A \mid \overline{C})\mathbb{P}(B \mid \overline{C})\mathbb{P}(\overline{C}) = \frac{5}{8}\end{aligned}$$

por causa da independência condicional, usada para deduzir a segunda linha da equação acima. ◊

Agora, retomemos o exemplo do vírus da *probabilite*, dado na página 35, que é um vírus que contamina 0,5% da população universitária e que o teste para detectar o vírus tem 1% de chance de acusar falsos positivos e falsos negativos. Vimos que $\mathbb{P}(B \mid A) \approx 0{,}34$ onde A é o evento "foi detectado *probabilite*" e B o evento "tem *probabilite*". Agora, suponha que o estudante estava com dor de cabeça (o teste foi feito em véspera de prova). É sabido que 95% dos indivíduos que tem *probabilite* apresentam dor de cabeça, enquanto que 10% da população não contaminada apresenta dor de cabeça; também é sabido que o evento "ter dor de cabeça", que denominamos C, não afeta a precisão do teste no sentido de que A e C são condicionalmente independentes $\mathbb{P}(A \cap C \mid B) = \mathbb{P}(A \mid B)\mathbb{P}(C \mid B)$. Com esse fato, a probabilidade de ter *probabilite* dado que o teste deu positivo e o estudante tem dor de cabeça é

$$\begin{aligned}\mathbb{P}(B \mid A \cap C) &= \frac{\mathbb{P}(A \cap C \mid B)\mathbb{P}(B)}{\mathbb{P}(A \cap C \mid B)\mathbb{P}(B) + \mathbb{P}(A \cap C \mid \overline{B})\mathbb{P}(\overline{B})} \\ &= \frac{\mathbb{P}(A \mid B)\mathbb{P}(C \mid B)\mathbb{P}(B)}{\mathbb{P}(A \mid B)\mathbb{P}(C \mid B)\mathbb{P}(B) + \mathbb{P}(A \mid \overline{B})\mathbb{P}(C \mid \overline{B})\mathbb{P}(\overline{B})} \\ &= \frac{0{,}99 \cdot 0{,}95 \cdot 0{,}005}{0{,}99 \cdot 0{,}95 \cdot 0{,}005 + 0{,}01 \cdot 0{,}1 \cdot 0{,}995} \approx 0{,}82.\end{aligned}$$

1.4.1 ESPAÇO PRODUTO

Dados os espaços de probabilidade discretos $(\Omega_i, \mathbb{P}_i)$, para $1 \leqslant i \leqslant n$, podemos definir um espaço de probabilidade discreto cujo espaço amostral é dado pelas sequências $(\omega_1, \omega_2, \ldots, \omega_n)$ do produto cartesiano $\Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$ e a medida de probabilidade em cada ponto amostral é

$$\mathbb{P}(\omega) := \mathbb{P}_1(\omega_1)\mathbb{P}_2(\omega_2)\cdots\mathbb{P}_n(\omega_n)$$

para todo $\omega = (\omega_1, \omega_2, \ldots, \omega_n) \in \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$, a qual se estende do modo usula para todo $A \subset \Omega$. Esse espaço de probabilidade é chamado **espaço produto** e é o modelo probabilístico de um experimento sendo repetido n vezes sob condições idênticas. Não é difícil verificar que

$$\sum_{\omega\in\Omega} \mathbb{P}(\omega) = \sum_{\omega_1\in\Omega_1} \sum_{\omega_2\in\Omega_2} \cdots \sum_{\omega_n\in\Omega_n} \mathbb{P}_1(\omega_1)\mathbb{P}_2(\omega_2)\cdots\mathbb{P}_n(\omega_n) = 1$$

o que garante que o espaço produto é um espaço de probabilidade. Além disso, para $A_i \subset \Omega_i$, $1 \leqslant i \leqslant n$, temos (verifique) $\mathbb{P}(A_1 \times A_2 \times \cdots \times A_n) = \mathbb{P}_1(A_1)\mathbb{P}_2(A_2)\cdots\mathbb{P}_n(A_n)$.

Um modelo probabilístico para n lançamentos de uma moeda equilibrada em que os resultados dos lançamentos são mutuamente independentes é dado pelo espaço produto $(\Omega^n, \mathbb{P}^n)$, em que $(\Omega, \mathbb{P})$ é o modelo para um lançamento dado no exemplo 1.6.

Exemplo 1.33. Sortear um número inteiro entre 0 e 999 é um experimento aleatório cujo espaço amostral é o conjunto dos números naturais até 999 e cuja medida de probabilidade é a uniforme, isto, é cada número ocorre com probabilidade 1/1.000. Se temos disponível os algarismos $0, 1, \ldots, 9$ podemos gerar uniformemente um número entre 0 e 999 se sortearmos $d_1, d_2, d_3 \in \{0, 1, \ldots, 9\}$, com os resultados mutuamente independentes, e tomarmos $n := d_1 \times 10^2 + d_2 \times 10^1 + d_3 \times 10^0$, então o espaço amostral é dado pelas ternas de algarismos (d_1, d_2, d_3) e a probabilidade de n é $(1/10)^3 = 1/1.000$. Há uma correspondência bijetiva (dada por $n = n(d_1, d_2, d_3)$) entre esses dois espaços amostrais e que preserva a probabilidade dos pontos amostrais, com isso os eventos aleatórios têm a mesma probabilidade no dois modelos e, nesse sentido, sortear uniformemente um número de três algarismos e sortear uniformemente cada um de três algarismos são experimentos aleatórios equivalentes. ◊

Exercício 1.34. Considere n repetições de um experimento modelado por $(\Omega, \mathbb{P})$. Sejam $A_1, A_2, \ldots, A_n$ eventos de Ω^n tais que a j-ésima rodada sozinha determina se A_j ocorre, ou seja, existe um $E_j \subset \Omega$ tal que $A_j = \Omega^{j-1} \times E_j \times \Omega^{n-j}$. Se em Ω^n tomarmos a medida produto, então os eventos $A_1, A_2, \ldots, A_n$ são mutuamente independentes. (Dica: comece com a prova de que os eventos são dois a dois independentes.)

Repetições independentes de um algoritmo. Uma técnica importante na utilidade dos algoritmos probabilísticos é a possibilidade de reduzir o erro das respostas executando o algoritmo com a mesma entrada várias vezes: se um algoritmo erra com probabilidade ε, então em duas execuções errará com probabilidade ε^2, em $r \in \mathbb{N}$ execuções a probabilidade de erro é ε^r.

Nos algoritmos probabilísticos assumimos independência dos resultados nos sorteios. Primeiro, assumimos que todos os sorteios feitos durante uma rodada do algoritmo são independentes, isto é, o resultado de um ou mais sorteios não altera a probabilidade do resultado de um outro sorteio. Também, assumimos que os sorteios feitos durante uma execução não altera a probabilidade dos

sorteios nas outras execuções de modo que se justifica e decaimento exponencial no erro descrito no parágrafo anterior.

Lembremos que o algoritmo 2 erra quando declara um polinômio não nulo como nulo, o que pode ter ocorrido pela escolha de uma raiz do polinômio pelo algoritmo. Fixada uma instância do problema, suponhamos r rodadas independentes desse algoritmo (com a mesma instância). Se em alguma dessas rodadas o o algoritmo 2 responde *não*, então essa é a resposta definitiva para o problema com essa instância. Se todas as r respostas forem *sim* então a resposta definitiva é *sim* e essa resposta estará errada se todas as r respostas de cada rodada estiverem erradas, o que ocorre com probabilidade 4^{-r}, pela independência dos eventos "resposta errada na i-ésima execução". Assim, se precisamos de uma garantia na resposta para o problema, por exemplo com probabilidade de erro menor que ε, para algum $\varepsilon > 0$ fixo, então basta escolher r de modo que $4^{-r} < \varepsilon$, ou seja, $r > \log_2 \sqrt{1/\varepsilon}$ rodadas.

Proposição 1.35 *Dado um real positivo* ε*, o problema* teste de identidade de polinômios em uma variável *pode ser resolvido por um algoritmo aleatorizado com probabilidade de erro menor que* ε*.* □

1.4.2 GERADOR DE NÚMEROS ALEATÓRIOS

Suponhamos que temos disponível uma fonte que gera bits aleatórios de modo uniforme e independente e queremos projetar um algoritmo que recebe um inteiro positivo M e nos devolve uma escolha aleatória em $\{0, 1, \ldots, M-1\}$. Se M é uma potência de 2, digamos que $M = 2^k$, então a resposta é simples: basta sortearmos k bits aleatórios $d_0, d_1, \ldots d_{k-1} \in \{0,1\}$ que o resultado é o número $\sum_{i=0}^{k-1} d_i 2^i$ no domínio desejado com probabilidade $1/M$. No caso em que M não é potência de 2, digamos que $2^{k-1} < M < 2^k$ (o que significa que precisamos de k bits aleatório) usamos o mesmo processo descrito no parágrafo anterior com a exceção de que se o resultado for maior ou igual a M, o processo é reiniciado e é repetido até que um número entre 0 e $M-1$ seja obtido.

Instância : inteiro positivo $M \geqslant 2$.
Resposta : uma escolha aleatória uniforme em $\{0, 1, \ldots, M-1\}$.

1 **repita**
2 **para cada** $i \in \{0, \ldots, \lfloor \log_2 M \rfloor\}$ **faça** $d_i \leftarrow_{\mathcal{U}} \{0,1\}$;
3 $N \leftarrow \sum_i d_i 2^i$;
4 **até que** $N < M$;
5 **responda** N.

Algoritmo 3: gerador de números aleatórios.

O resultado das escolhas aleatórias na linha 2 do algoritmo 3, a sequência $d_{k-1} d_{k-2} \ldots d_0$, é um evento elementar do espaço produto $(\{0,1\}^k, \mathbb{P}^k)$, em que $\mathbb{P}(0) = \mathbb{P}(1) = 1/2$, que tem probabilidade

$\mathbb{P}^k(d_{k-1}\,d_{k-2}\ldots d_0) = (1/2)^k$. Essa sequência é a representação binária do número $\sum_{i=0}^{k-1} d_i 2^i$ que pertence a $\{0,1,\ldots,2^k-1\}$.

Por exemplo, se $M = 7$ então $k = 3$. Com três bits $d_2 d_1 d_0$ temos as representações binárias dos naturais de 0 a 7. O laço da linha 1 gera qualquer um desses números com a mesma probabilidade, a saber $1/2^3 = 1/8$. Porém o algoritmo só termina se o sorteio for diferente de 7, isto é, não ocorre o evento $d_2 d_1 d_0 = 111$. Dado que esse evento não ocorre, qual a probabilidade do algoritmo responder 4? Usando probabilidade condicional $\mathbb{P}[\,N = 4 \mid N \neq 7\,] = (1/8)/(7/8) = 1/7$ e, de fato, o algoritmo escolhe $N \in \{0,\ldots,6\}$ com probabilidade 1/7.

No caso geral, definimos o evento $A = \{0,1,\ldots,M-1\}$ e para qualquer $t \in A$ a probabilidade do algoritmo responder t é dada por

$$\mathop{\mathbb{P}}_{N\in\{0,\ldots,2^k-1\}}[\,N = t \mid N \in A\,] = \frac{\mathbb{P}(\{t\}\cap A)}{\mathbb{P}(A)} = \frac{(1/2)^k}{M/2^k} = \frac{1}{M}.$$

Portanto, se o algoritmo termina, ou seja, dado que o sorteio N satisfaz $N < M$, então ele responde com um número entre 0 e $M-1$ de modo uniforme. Resta provarmos que o algoritmo termina, isto é, eventualmente a condição $N < M$ na linha 4 e satisfeita. Sempre assumimos que os sorteios e os eventos que eles definem em rodadas diferentes de um laço como o da linha 1 são independentes.

Fixamos uma instância M com $2^{k-1} < M < 2^k$ e $k = \lfloor \log_2 M \rfloor + 1$ é o número de bits sorteados. A probabilidade com que uma rodada do laço da linha 1 resulte em um inteiro N que pertença ao conjunto $\overline{A} = \{M,\ldots,2^k-1\}$ é

$$\mathbb{P}(\overline{A}) = \frac{2^k - M}{2^k} = 1 - \frac{M}{2^k}.$$

Definimos para todo $n \geqslant 1$ o evento A_n por "o algoritmo leva mais que n rodadas para terminar". Tal evento ocorre se nas n primeiras tentativas do laço na linha 1 ocorre um sorteio em $\overline{A}$ e daí pra diante pode ocorrer qualquer um dos dois casos, A ou $\overline{A}$, logo $\mathbb{P}(A_n) = (1 - M/2^k)^n$ pela independência da ocorrência dos eventos $\overline{A}$ em cada rodada do laço.

Os eventos A_n formam uma sequência decrescente $A_n \supset A_{n+1}$ e $\lim_{n\to\infty} A_n = \bigcap_{n\geqslant 1} A_n$ é o evento "o algoritmo não termina", cuja probabilidade é, por continuidade (equação (1.5) na página 21),

$$\mathbb{P}[\text{o algoritmo não termina}\,] = \mathbb{P}\Big(\lim_{n\to\infty} A_n\Big) = \lim_{n\to\infty} \mathbb{P}(A_n) = \lim_{n\to\infty}\Big(1 - \frac{M}{2^k}\Big)^n = 0$$

pois $M < 2^k$, portanto, o algoritmo termina com probabilidade 1.

Notemos que, diferente do exemplo do algoritmo 2, nesse caso a resposta está sempre correta e a aleatoriedade influencia na duração das rodadas, isto é, no tempo que leva para o algoritmo terminar. Esse algoritmo não só termina como, de fato, termina rápido, em poucas rodadas do laço da linha 1 com alta probabilidade. De $M > 2^{k-1}$ temos $\mathbb{P}(\overline{A}) < 1/2$, portanto, em n rodadas do laço todos os inteiros sorteados pertencem a $\overline{A}$ com probabilidade menor que 2^{-n}. A probabilidade de não terminar em, por exemplo, 4 rodadas é menor que 0,07, em 10 rodadas é menor que 0,00098.

1.5 EXERCÍCIOS

Exercício 1.36. Sejam A, B e C eventos aleatórios. Determine expressões que envolvem somente conjuntos e operações sobre conjuntos para

1. somente A ocorre;
2. A e B mas não C ocorrem;
3. os três eventos ocorrem;
4. pelo menos um evento ocorre;
5. pelo menos dois eventos ocorrem;
6. exatamente um evento ocorre;
7. exatamente dois eventos ocorrem;
8. nenhum evento ocorre;
9. não mais que dois eventos ocorrem.

Exercício 1.37. Considere o lançamento repetido de uma moeda equilibrada até sair coroa, como descrito no exemplo 1.13. Com que probabilidade o número de lançamentos é par?

Exercício 1.38 (Princípio da inclusão–exclusão). Prove que para eventos $A_1, A_2, \ldots, A_n$ vale

$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \mathbb{P}(A_i) - \sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathbb{P}\big(A_i \cap A_j\big) + \sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n} \mathbb{P}\big(A_i \cap A_j \cap A_k\big) + \cdots + (-1)^n \mathbb{P}\left(\bigcap_{i=1}^{n} A_i\right).$$

Exercício 1.39. Prove que o corolário 1.5, página 14, admite a seguinte extensão: para qualquer conjunto enumerável $\{E_i : i \geqslant 1\}$ de eventos num espaço discreto vale

$$\mathbb{P}\left(\bigcup_{i\geqslant 1} E_i\right) \leqslant \sum_{i\geqslant 1} \mathbb{P}(E_i).$$

Exercício 1.40. Seja p primo e tome $\Omega := \{1, 2, \ldots, p\}$ com medida uniforme $\mathbb{P}(A) = |A|/p$, para todo $A \subset \Omega$. Prove que se A e B são independentes então pelo menos um desses eventos deve ser $\varnothing$ ou Ω.

Exercício 1.41. Defina um sistema de codificação com $\mathcal{P} = \{\alpha, \beta\}$, $\mathcal{C} = \{a, b\}$ e $\mathcal{K} = \{0, 1\}$ tais que $\mathbb{P}_{\mathcal{K}}(0) = 1/10$ e $\mathbb{P}_{\mathcal{K}}(1) = 9/10$. A codificação $E_k(m)$, para cada $m \in \{\alpha, \beta\}$ é dada na tabela 1.3 abaixo. Prove que o sistema não tem sigilo perfeito.

	$E_0(m)$	$E_1(m)$
α	a	b
β	b	a

Tabela 1.3: função de codificação.

Exercício 1.42 (Teorema de Shannon). Prove o seguinte resultado: dados um sistema de codificação com $\mathcal{P}, \mathcal{K}, \mathcal{C}$ finitos e $|K| = |C|$ e dada uma medida de probabilidade $\mathbb{P}_{\mathcal{P}}$ sobre $\mathcal{P}$ tal que $\mathbb{P}_{\mathcal{P}}(P) > 0$, para todo $P \in \mathcal{P}$, esse sistema tem sigilo perfeito se e somente se as chaves são equiprováveis e se para todo $P \in \mathcal{P}$ e todo $C \in \mathcal{C}$ existe um único $K \in \mathcal{K}$ tal que $E_K(P) = C$.

Exercício 1.43. Considere $\Omega = \{0,1\}^n$ e suponha que x é o resultado de um sorteio uniforme em Ω e y é o resultado de um sorteio em Ω em que os resultados ocorrem de acordo com alguma medida de probabilidade, possivelmente diferente da uniforme. Os sorteios são independentes. Mostre que $y \oplus x$ (ou-exclusivo coordenada-a-coordenada) é qualquer elemento de Ω com probabilidade $(1/2)^n$.

Exercício 1.44. Vamos provar uma generalização do exercício 1.43. Seja $(G, \circ)$ um grupo abeliano finito, T um conjunto finito e $f\colon T \to G$ uma função qualquer. Suponha que x é o resultado de um sorteio uniforme em G e y é o resultado de um sorteio em T em que os resultados ocorrem de acordo com alguma medida de probabilidade, possivelmente diferente da uniforme, e os sorteios são independentes. Prove que $x \circ f(y)$ é qualquer elemento do grupo G com probabilidade uniforme. Prove também que para quaisquer $i \in T, j \in G$ os eventos definidos por "$y = i$" e "$x \circ f(y)$=j" são independentes.

Exercício 1.45. No exercício anterior, suponha que f seja invertível e T um conjunto de textos legíveis. A codificação de um texto legível $m \in T$ é feita transformando-o num elemento do grupo com $f(t)$, sorteando uma chave $k \in G$ uniformemente e calculando $c = k \circ f(t)$. A decodificação é feita conhecendo-se a chave k e calculando $f^{-1}(c \circ (-k))$, em que $-k$ é o elemento inverso de k em G. Verifique que tal sistema de codificação tem sigilo perfeito.

Exercício 1.46. Suponha que você tem três moedas e uma delas é viciada de modo que $\mathbb{P}(\text{cara}) = 2/3$. Escolhendo uma delas ao acaso a probabilidade de acertar qual é a viciada é um terço. Agora, suponha que o resultado do lançamento de cada uma delas, sem conhecer qual é a viciada, resulta em (cara, cara, coroa). Mostre, usando o Teorema de Bayes, que a probabilidade da primeira moeda ser a viciada é 2/5.

Exercício 1.47 (Saldanha, 1997). Dois amigos querem decidir quem pagará a conta da pizzaria com uma aposta. Cada um deles escolhe uma sequência de três resultados do lançamento de uma moeda honesta, em seguida eles jogam uma moeda até que saia uma das duas sequências: aquele que tiver escolhido a primeira sequência a sair ganhou a aposta. Por exemplo, André é o primeiro e fica com a sequência (coroa, coroa, cara) enquanto Renato responde com (cara, coroa, coroa). Eles jogam a moeda obtendo coroa, cara, coroa, cara, coroa, cara, coroa, coroa e neste momento Renato é o vencedor. Mostre que nesse caso a probabilidade do Renato ganhar o jogo é 3/4. Prove que o segundo jogador sempre tem uma escolha mais vantajosa pra ele.

Exercício 1.48. Três convidados chegaram numa festa vestindo chapéu e os entregaram na recepção. O funcionário, pouco cuidadoso, não identificou os chapéus e no final da festa os entregou aleatoriamente para as mesmas três pessoas. Use o princípio da inclusão–exclusão para mostrar que ninguém recebe o próprio chapéu com probabilidade 1/3. Generalize o resultado para n convidados e use a série de potências para a função exponencial (veja (s.8) do apêndice) para mostrar que a probabilidade de ninguém pegar o próprio chapéu converge, quando $n \to \infty$, para 1/e.

Exercício 1.49. Em um treino de paraquedistas um grupo de n paraquedistas estão enfileirados e um paraquedista é escolhido ao acaso no seu grupo. O paraquedista escolhido cumprimenta todos os paraquedistas do seu grupo e salta da avião; o grupo fica então dividido em dois: um grupo formado pelos paraquedistas que se encontravam a esquerda daquele que pulou e o outro grupo formado pelos paraquedistas a direita. O procedimento é repetido nos grupos restantes até sobrarem grupos de um único paraquedista, que pulam um a um. Note que paraquedistas que em algum momento ficam em grupos diferentes não se cumprimentaram e não se cumprimentarão desse momento em diante. A ordem da fila dentro de cada grupo é sempre mantida. Prove que os paraquedistas das posições i e j, sem perda de generalidade $j > i$, se cumprimentam com probabilidade $2/(j - i + 1)$.

Exercício 1.50. Considere uma moeda que resulta em cara com probabilidade $p \in (0, 1)$. Prove que a probabilidade de que em n lançamentos (independentes) temos mais que k caras é no máximo $\binom{n}{k} p^k$.

Exercício 1.51. Considere uma moeda que resulta em cara com probabilidade 1/5 e coroa com probabilidade 4/5. Justifique que a probabilidade de sair menos que k coroas em $2k$ lançamentos (independentes) é

$$\sum_{0}^{k-1} \binom{2k}{i} (4/5)^i (1/5)^{2k-i} < (1/5)^{2k} 4^k \sum_{0}^{k-1} \binom{2k}{i}.$$

Use o teorema do binômio de Newton e prove que a probabilidade de sair menos que k coroas em $2k$ lançamentos dessa moeda é menor que $(4/5)^{2k}$.

Exercício 1.52. Considere o seguinte procedimento para gerar uma permutação da sequência $a = (1, 2, \ldots, n)$, para qualquer $n \in \mathbb{N}$:

- para cada coordenada $i = 1, 2, \ldots, n$ do vetor a
  - sorteie uniformemente uma coordenada $j \in \{1, 2, \ldots, n\}$
  - troque os componentes das coordenadas: coloque o número da coordenada j na coordenada i e o da coordenada i na coordenada j.

O vetor resultante é uma permutação do vetor inicial com probabilidade uniforme?

Exercício 1.53. Considere um algoritmo que recebe um inteiro positivo $M > 0$ e devolve um inteiro escolhido aleatoriamente em $\{0, 1, \ldots, M-1\}$ da seguinte forma: (1) gere um número N de k bits como no algoritmo acima; (2) devolva o resto da divisão inteira de N por M. Isso evitaria a execução de mais que uma iteração do laço do algoritmo acima. Mostre que isso não resulta em uma resposta em $\{0, \ldots, M-1\}$ com probabilidade uniforme.

Exercício 1.54. O seguinte gerador de números aleatórios é adaptado do exercício anterior.

Instância : inteiros positivos M e t.
Resposta : uma escolha aleatória uniforme em $\{0, 1, \ldots, M-1\}$.

1 seja k o número de bits de M;
2 $(d_0, d_1, \ldots, d_{k+t-1}) \leftarrow_{\mathcal{U}} \{0,1\}^{k+t}$;
3 $N \leftarrow \sum_i d_i 2^i$;
4 **responda** N mod M.

No caso $t = 0$ a probabilidade da resposta não é uniforme (exercício 1.53 acima). Prove que quanto maior é t mais próximo a probabilidade de N está da uniforme, no seguinte sentido

$$\sum_{n=0}^{M-1} \left| \mathbb{P}[N = n] - \frac{1}{M} \right| \leqslant \frac{1}{2^{t-1}}.$$

Exercício 1.55. Distribuímos uniformemente e independentemente m bolas idênticas em n caixas distintas. Qual é a probabilidade com que a i-ésima caixa fica vazia? Qual é a probabilidade com que a j-ésima e a i-ésima caixas ficam vazias? Qual é a probabilidade com que nenhuma fica vazia? E de exatamente uma ficar vazia?

Prove que no caso $n = m$ o maior número de bolas em qualquer caixa é no máximo $2e \log_2 n$ com probabilidade $1 - n^{-4}$ (dica: estime a probabilidade de uma caixa ter muitas bolas, a fórmula de Stirling (d.3) pode ser útil nos cálculos, e use a desigualdade de Boole, corolário 1.5 na página 14).

Exercício 1.56. Considere o conjunto dos números naturais e defina para todo subconjunto E e cada $n \geqslant 1$ a densidade relativa de E

$$P_n(E) := \frac{|\{1, 2, \ldots, n\} \cap E|}{n}.$$

Defina $p(E) := \lim_{n\to\infty} \mathbb{P}_n(E)$ quando o limite existe e seja $\mathcal{A}$ a família de subconjunto E para os quais o limite existe. Prove que se A e B são elementos disjuntos de $\mathcal{A}$ então $A \cup B \in \mathcal{A}$ e $p(A \cup B) = p(A) + p(B)$ e que esse não é o caso se os eventos não são disjuntos. Prove que p não é enumeravelmente aditiva. Finalmente, prove que p é invariante por translação, ou seja, se $p(A)$ existe então $p(\{a + 1 : a \in A\}) = p(A)$.

Exercício 1.57. Prove que o seguinte algoritmo não termina com probabilidade 1.

```
1 j ← 0;
2 repita
3 |   j ← j + 1;
4 |   para cada i ∈ {1, 2, ..., j} faça d_i ←_U {0, 1};
5 até que d_i = 1 para todo i.
```

Exercício 1.58. Alice e Bob têm, cada um, um enorme banco de dados que eles querem saber se são iguais. Podemos assumir, sem perda de generalidade, que cada banco de dados é um vetor de n bits $a_1 a_2 \ldots a_n$ para Alice e $b_1 b_2 \ldots b_n$ para Bob. Alice e Bob podem enviar mensagens um ao outro. Uma saída trivial é: Alice envia os n bits para Bob, então Bob verifica se os dois vetores são os mesmos e envia o resultado (sim ou não) para Alice. Toda a comunicação usa $n+1$ mensagens de um bit.

O seguinte protocolo pressupõe-se mais econômico:

1. Alice escolhe um primo $p \in [n^2, 2n^2]$ e manda para Bob;
2. Alice constrói o polinômio $\mathsf{A}(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_n x^{n-1}$;
3. Bob constrói o polinômio $\mathsf{B}(x) = b_1 + b_2 x + b_3 x^2 + \cdots + b_n x^{n-1}$;
4. Alice sorteia $\alpha \in \mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$, calcula $\mathsf{A}(\alpha) \bmod p$ e manda α e $\mathsf{A}(\alpha) \bmod p$ para Bob;
5. Bob calcula $\mathsf{B}(\alpha)$ e manda para Alice se $\mathsf{A}(\alpha) = \mathsf{B}(\alpha)$ ou não.

Tal protocolo erra se $\mathsf{A} \neq \mathsf{B}$, porém $\mathsf{A}(\alpha) = \mathsf{B}(\alpha)$. Verifique que

$$\mathbb{P}[\,\text{erro}\,] \leqslant \mathop{\mathbb{P}}_{\alpha \in \mathbb{Z}_p}[(\mathsf{A} - \mathsf{B})(\alpha) = 0] \leqslant \frac{n-1}{p}$$

e conclua que $\mathbb{P}[\,\text{erro}\,] < 1/n$. Mostre que são usados $O(\log n)$ bits.

Exercício 1.59 (Lema do isolamento, (Mulmuley, Vazirani e Vazirani, 1987)). O seguinte resultado diz que, independentemente da natureza de uma família $\mathcal{F}$ de conjuntos, uma atribuição aleatória de pesos aos elementos de $\bigcup_{\mathsf{F} \in \mathcal{F}} \mathsf{F}$ isola o elemento da família menos pesado com grande probabilidade. Este lema tem muitas aplicações na teoria da computação, em particular, Mulmuley e seus coautores o usaram para projetar um algoritmo aleatorizado paralelizável para encontrar emparelhamento de peso máximo em um grafo (exercícios 2.65 e 2.71).

Sejam E um conjunto finito e $\mathcal{F}$ uma família de subconjuntos de E. Uma *m-ponderação* de E é uma função $p\colon \mathsf{E} \to \{1, \ldots, m\}$ que atribui pesos inteiros para os elementos de E. O peso de um subconjunto não vazio $\mathsf{S} \subset \mathsf{E}$ é $p(\mathsf{S}) = \sum_{e \in \mathsf{S}} p(e)$. A ponderação p é *isolante para* $\mathcal{F}$ se o peso mínimo, $\min_{\mathsf{S} \in \mathcal{F}} p(\mathsf{S})$, for alcançado em um único elemento de $\mathcal{F}$. O lema do isolamento é o seguinte resultado

Dado $m \in \mathbb{N}$, para todo conjunto finito E e toda família $\mathcal{F} \subset 2^{\mathsf{E}}$ temos

$$\mathop{\mathbb{P}}_{p\colon \mathsf{E}\to\{1,\dots,m\}}[p \text{ é isolante para } \mathcal{F}] \geqslant \left(1 - \frac{1}{m}\right)^{|\mathsf{E}|}.$$

Para provar esse resultado, suponha que nenhum elemento de $\mathcal{F}$ é um superconjunto de outro elemento de $\mathcal{F}$. Considere P o conjunto de todas as m-ponderações de E e $\mathsf{P}^{>1}$ o conjunto de todas as m-ponderações de E que não atribuem o peso 1 a qualquer elemento de E. Defina $\phi\colon \mathsf{P}^{>1} \to \mathsf{P}$ da seguinte forma: dada a ponderação $p \in \mathsf{P}$ tome algum $\mathsf{S}_p \in \mathcal{F}$ de peso mínimo de acordo com p e, fazendo $p' = \phi(p)$, defina

$$p'(i) = \begin{cases} p(i) - 1 & \text{se } i \in \mathsf{S}_p \\ p(i) & \text{se } i \notin \mathsf{S}_p. \end{cases}.$$

1. Prove que se $p \in \mathsf{P}^{>1}$ então p' é isolante em $\mathcal{F}$.

2. Prove que ϕ é injetiva.

3. Prove que $\mathbb{P}_p[p \text{ é isolante em } \mathcal{F}] \geqslant |\phi(\mathsf{P}^{>1})|/|\mathsf{P}|$.

4. Conclua a demonstração do lema do isolamento.

2 | ALGORITMOS ALEATORIZADOS

Uma definição precisa de algoritmo é importante para entender os processos computacionais, conhecer seus limites e estabelecer sua eficiência na resolução de problemas. Neste capítulo apresentamos uma discussão informal, discutiremos mais dos aspectos formais em outro capítulo, apresentamos exemplos clássicos de algoritmos que usam bits aleatórios em suas computações e como limitamos a chance de erro e estimamos seu tempo de execução. Também, aproveitamos essa introdução para aprendermos alguns algoritmos em Teoria do Números que serão úteis mais aditante no texto. Na última seção apresentamos um exemplo de problema computacional cuja solução probabilística envolve um modelo contínuo.

2.1 ANÁLISE DE ALGORITMOS

Um algoritmo define sem ambiguidade uma sequência de passos para resolver um problema computacional. Um *problema computacional* é caracterizado por um conjunto de *instâncias* (ou entradas), um conjunto de *respostas* e uma *relação* que associa instâncias a respostas. Por exemplo, o problema "multiplicar dois inteiros positivos" tem como instâncias pares (n, m) de inteiros positivos e como respostas inteiros positivos. A relação que se quer computar é definida pelos pares $((n, m), z)$ de instâncias e respostas tais que $n \cdot m = z$. Notemos que entre instâncias e respostas temos uma relação e não uma função pois é possível que uma instância esteja associada a mais de uma resposta. Por exemplo, se as instância são fórmulas da lógica proposicional e as respostas são valorações das variáveis com verdadeiro ou falso e que tornam a fórmula verdadeira, então a instância x_1 *ou* x_2 tem três respostas possíveis.

Os algoritmos são, geralmente, descritos no que costumamos chamar de *pseudocódigo*, uma mistura de algumas palavras-chave em português com sentenças construídas como em uma linguagem de programação estruturada como a linguagem C.

Um algoritmo executado sobre qualquer instância do problema produz uma resposta que deve estar correta, isto é, o par instância/resposta deve fazer parte da relação do problema. Além da correção do algoritmo, devemos conhecer o comportamento do algoritmo com respeito ao consumo de recursos para resolver o problema. Alguns recursos para os quais podemos querer estimar o consumo por um algoritmo são *espaço*, *tempo*, *comunicação* e *aleatorização*. Os dois primeiros são os mais comumente estudados, o primeiro está associado a quantidade de "memória" extra usada para resolver uma instância do problema e o segundo é dado em função do número de *instruções elementares* realizadas pelo algoritmo. Além desses, pode ser de interesse a quantidade de unidades de informação transmitidas e recebidas (comunicação, veja o exercício 1.58) e, quando analisamos algoritmos probabilísticos, a *quantidade de bits sorteados* pelo algoritmo (veja um caso na página 73).

A *Análise de Algoritmos* é um ramo de estudo importante da Teoria da Computação, nela aprendemos as técnicas de prova da *correção* de algoritmos e de estimativas de *eficiência* quanto ao consumo de recursos.

Tamanho da instância. Para expressar o consumo de recursos pelos algoritmos nós levamos em conta o *tamanho da instância*. Em último nível, uma instância do problema é dada por alguma *codificação* dela com uma cadeia de bits, donde definimos o tamanho de uma instância como o número de bits usados na representação da instância. Na prática, somos menos precisos e adotamos algumas simplificações que dependem muito do problema que esta sendo estudado e da representação usada, em geral o tamanho da instância é um inteiro positivo que descreve a quantidade de componentes da instância. Por exemplo, se o problema é multiplicar dois números inteiros, o tamanho é a quantidade

de algarismos desses números (em alguma base com pelo menos dois algarismos). Se o problema é multiplicar duas matrizes de números inteiros, o tamanho é, geralmente, a dimensão da matriz quando podemos supor que o tamanho da matriz é muito grande quando comparada ao tamanho dos números e se esse não é o caso então o tamanho dos números deve ser levado em conta. Para pesquisar uma lista de itens e determinar se um item particular está presente nela, o tamanho da instância é o número de itens na lista. No problema de ordenação de uma sequência numérica o tamanho é dado pelo número de elementos da sequência. Em algoritmos sobre grafos, o tamanho é ou o número de vértices, ou o número de arestas, ou a soma de ambos. Há justificativas razoáveis para tais simplificações e, mesmo que façamos escolhas concretas de codificação de instâncias, tentamos manter a discussão abstrata o suficiente para que as estimativas sejam independentes da escolha da codificação.

Tempo de execução. Como estimamos o *tempo de execução* de um algoritmo para resolver uma determinada instância? São duas as ideias preliminares. Primeiro, como já dissemos, determinamos quanto tempo o algoritmo leva em função do tamanho da instância. A segunda ideia é avaliarmos o quão rapidamente a função que caracteriza o tempo de execução aumenta com o tamanho da entrada[1] expressando a ordem de grandeza do crescimento do tempo de execução. O consumo de tempo dos algoritmos, como medida de sua eficiência, expressa o número de *instruções básicas* executadas pelo algoritmo em função do tamanho da entrada descrita em notação assintótica. Isso nos permite algumas simplificações: podemos (quase sempre) assumir que cada linha consome tempo constante, mesmo as operações aritméticas podem serem assumidas de tempo constante. Porém, isso não é regra e tem de ser feito com cuidado, essa hipótese não pode ser assumida quando as operações dependem do tamanho da entrada, como no problema de multiplicação de inteiros, por exemplo, onde as operações aritméticas tem custo proporcional ao tamanho dos operandos.

Na seção 2.1.1 veremos as notações que usamos para expressar o tempo de execução de um algoritmo. As estimativas para o consumo são feitas para as classes das instâncias que têm o mesmo tamanho e expressamos o desempenho de algoritmos em função do tamanho da representação das instâncias. É possível que entre instâncias de mesmo tamanho a quantidade de recursos usados por uma algoritmo varie. Por exemplo, ao ordenar uma lista de dez números a quantidade de instruções executadas pode variar de acordo com a disposição dos números nessa lista, eventualmente, pode ser mais barato se a lista já está quase ordenada, logo, devemos adotar alguma estratégia para resumir o tempo de execução do algoritmo na classe das instâncias de mesmo tamanho: tomamos o *pior caso* — aquele em que o algoritmo consome mais recursos — ou o *caso médio* — a média de consumo numa classe.

[1] É natural esperarmos que instâncias maiores demandem mais recurso dos algoritmos. Não vamos lidar com casos em que isso não vale.

Em Complexidade Computacional convenciona-se chamar um algoritmo de **eficiente** com respeito ao tempo de execução se o número de instruções executadas é limitado superiormente por uma função polinomial no tamanho da instância. Na teoria isso é muito conveniente pois

- a classe das funções polinomiais é fechada para soma, multiplicação e composição de funções, assim a noção de eficiência é preservada por práticas comuns de programação;
- os modelos formais tradicionais de computação são polinomialmente equivalentes, o que torna a escolha do modelo irrelevante para essa definição de eficiência;
- com algum cuidado, as várias representações computacionais de objetos abstratos, como um grafo por exemplo, têm tamanhos polinomialmente relacionados, o que faz a codificação ser irrelevante para essa definição de eficiência.

Na prática isso pode não ser representativo de eficiência pois o polinômio pode ter um grau muito alto o que torna uma implementação de um caso assim inviável para o uso.

Para o problema cuja instância é uma lista $a_1, a_2, \cdots, a_n$ de inteiros e um inteiro x e a resposta é "sim" ou "não", que significa a ocorrência ou não, respectivamente, de x na lista, uma solução é a busca linear: percorra a lista e verifique se cada elemento dela é o item procurado.

Instância : uma lista $a_1, \ldots, a_n$ de inteiros e um inteiro x.
Resposta : *sim* se x ocorre em a e *não* caso contrário.

1 $i \leftarrow 1$;
2 **enquanto** $a_i \neq x$ *e* $i < n$ **faça** $i \leftarrow i + 1$;
3 **se** $a_i = x$ **então responda** sim.
4 **senão responda** não.

Algoritmo 4: busca sequencial.

No melhor caso o elemento x ocorre na primeira posição da sequência, o algoritmo executa a atribuição na linha 1, o teste na linha 2, a comparação na linha 3 e responde. Essencialmente, um número constante de instruções. Nesse caso escrevemos que o tempo de execução é $O(1)$.

O pior caso ocorre quando o valor que está sendo procurado não está na lista. O numero de instruções executadas é: 1 atribuição na linha 1, mais $4(n-1)$ nas linhas 2 e 3 (são feitas duas comparações na linha 2, uma adição e uma atribuição na linha 3 repetidas $n-1$ vezes), mais 1 comparação na linha 4, mais 1 retorno na linha 5. No total são $4(n-1)+3$ instruções. A função $4n-1$ é linear de modo que dizemos que o seu crescimento é da ordem de n e, usando notação assintótica, dizemos que o tempo de execução de pior caso do algoritmo é $O(n)$. Nesse problema, podemos estimar a ordem de grandeza do número de instruções executadas considerando apenas o numero de comparações que são feitas, as outras instruções contribuem com uma constante multiplicativa

desse termo. Assim, se tivéssemos contado apenas o número de comparações também chegaríamos a conclusão de que o tempo de execução de pior caso do algoritmo é $O(n)$.

Resumindo, no melhor caso a quantidade de comparações é constante, não depende de n e no pior caso cresce linearmente com n. Para estimar o caso médio, vamos assumir que o item procurado está na lista. Também, vamos assumir que cada elemento da lista tem a mesma probabilidade de ser o valor buscado. Com tais hipóteses o número médio de comparações é i se a busca termina na posição i, portanto, o número médio de comparações é

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

Nesse caso, dizemos que o tempo de execução de caso médio do algoritmo é $O(n)$ pois, novamente, temos uma função de crescimento linear em n.

Agora, consideremos o problema de ordenar uma sequência de inteiros. Uma solução é o seguinte algoritmo conhecido como ordenação por inserção:

Instância : uma sequência $a_1, \ldots, a_n$ de inteiros.
Resposta : uma permutação $a_{\pi(1)}, \ldots, a_{\pi(n)}$ da entrada tal que $a_{\pi(i)} \leqslant a_{\pi(j)}$ sempre que $\pi(i) < \pi(j)$.

```
1 para i de 2 até n faça
2     x ← a_i;
3     j ← i − 1;
4     enquanto (a_j > x e j ⩾ 1) faça
5         a_{j+1} ← a_j;
6         j ← j − 1;
7     a_{j+1} ← x.
```

Algoritmo 5: ordenação por inserção.

Notemos que o tempo do algoritmo é determinado pela condição no laço da linha 4, as linhas 2, 3 e 7 são executadas $n-1$ vezes pelo laço da linha 1. Para i fixo, uma rodada completa do laço executa 2 comparações, 2 atribuições e 2 operações; no pior caso[2] o laço é executado para todo j de i ate 1, depois o laço é falso para a segunda condição ($j = 0$). Logo são $6i$ instruções mais as 2 comparações finais. No pior caso, o custo de ordenação por inserção de uma sequência com n elementos é

$$T(n) = 2 + \sum_{i=2}^{n} 6i = 2 + 6\frac{(n+2)(n-1)}{2} = 3n^2 + 3n + 5$$

[2] O pior caso para ordenação por inserção ocorrerá quando a lista de entrada estiver em ordem decrescente.

portanto $T(n)$ tem ordem de crescimento de n^2. Notemos o seguinte, a ordem de grandeza de $T(n)$ é dada pelo fato de termos dois laços aninhados e dentro desses laços o número de instruções executadas em cada rodada é constante de tal forma que, para a ordem de grandeza, não importa se contamos $j \leftarrow j-1$ como 1 ou 2 instruções, é suficiente estabelecer que seja constante.

Para estimar o caso médio observamos que o fato determinante para o número de instruções executadas é o "tipo de ordem" dos elementos da sequência e não quais são os elementos em si. Por exemplo, ordenar $(1,2,3,4)$ usa o mesmo número de comparações que $(4,7,8,9)$, assim como ordenar $(1,4,3,5,2)$ e $(11,15,14,20,13)$. Em outras palavras, o mesmo tipo de ordem significa que a mesma permutação π ordena as duas instâncias. Dito isso, assumimos que as instâncias são formadas por sequências de n inteiros distintos fixos e que qualquer uma das $n!$ permutações são igualmente prováveis.

Fixado i, com $2 \leqslant i \leqslant n$, consideremos a subsequência $(a_1,\ldots,a_i)$. Para cada i vale que no início do laço da linha 1 temos $(a_1,\ldots,a_{i-1})$ ordenado e o laço da linha 4 procura a posição de a_i em $(a_1,\ldots,a_{i-1})$ ordenado. Definimos o posto(a_i) como a posição do elemento a_i no subvetor $(a_1,\ldots,a_i)$ ordenado. Por exemplo, com entrada $(3,6,2,5,1,7,4)$ o posto de 5 (que é o a_4) é 3 pois em $(3,6,2,5)$, quando ordenado, o número 5 ocupa a terceira posição.

Exercício 2.1. Verifique que o posto de a_i é igualmente provável ser qualquer $j \in \{1,2,\ldots,i\}$.

Dado i, o subvetor $(a_1,\ldots,a_{i-1})$ está ordenado com os mesmos $i-1$ primeiros elementos do vetor original da entrada. O teste no laço é executado $i-\text{posto}(a_i)+1$ vezes. Assim, o número médio de comparações para o qual o teste do laço vale é

$$\sum_{\text{posto}=1}^{i} \frac{i-\text{posto}+1}{i} = \frac{i+1}{2}.$$

O número médio de comparações efetuadas pelo algoritmo de ordenação por inserção é

$$\sum_{i=2}^{n} \frac{i+1}{2} = \frac{(n+4)(n-1)}{2}$$

que é da ordem de n^2.

2.1.1 NOTAÇÃO ASSINTÓTICA

As estimativas para o custo de um algoritmo, como o tempo de execução, tomam a ordem de grandeza da quantidade de recursos necessários por um algoritmo em função do tamanho e são expressas usando notação assintótica. Isso garante, por exemplo, que a estimativa teórica seja representativa para as várias possíveis implementações do algoritmo, as quais dependem da máquina, da linguagem de programação e da habilidade do programador dentre outros fatores. As diferenças causadas

por esses fatores não alteram a ordem de grandeza da função. Além disso, essa consideração permite uma simplificação substancial quando contamos o número de instruções executadas e também simplifica a questão da codificação de instâncias como os números, por exemplo, que podem ser expressos em qualquer base com pelo menos dois símbolos pois, nesse caso, eles têm representação de ordem logarítmica na quantidade de dígitos.

Abaixo f e g são funções[3] de $\mathbb{R}^{\geqslant 0}$ em $\mathbb{R}$ com g *assintoticamente positiva*, ou seja $g(n) > 0$ para todo n suficientemente grande.

Dizemos que f é **assintoticamente muito menor** que g e escrevemos

$$f(n) = o(g(n)) \text{ quando } n \to \infty \tag{2.1}$$

se, e só se,

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

Por exemplo,

1. $1 = o(\log(\log(n)))$.
2. $\log(\log(n)) = o(\log(n))$.
3. $\log(n) = o(n^{\varepsilon})$ para todo $\varepsilon > 0$.
4. $n^{\varepsilon} = o(n^{c})$ para quaisquer $0 < \varepsilon < 1 \leqslant c$.
5. $n^{c} = o(n^{\log n})$ para todo $1 \leqslant c$.
6. $n^{\log n} = o(\exp(n))$.
7. $\exp(n) = o(n^{n})$.
8. $n^{n} = o(\exp(\exp(n)))$.

Também usamos a notação $f \ll g$ o que nos permite escrever, a partir do exemplo acima, a sequência monótona

$$1 \ll \log(\log(n)) \ll \log(n) \ll n^{\varepsilon} \ll n^{c} \ll n^{\log n} \ll e^{n} \ll n^{n} \ll e^{e^{n}}.$$

Dizemos que f **assintoticamente menor** que g e escrevemos

$$f(n) = O(g(n)) \text{ quando } n \to \infty \tag{2.2}$$

se existe $n_0 > 0$ e existe $c > 0$ tais que para todo $n \geqslant n_0$

$$|f(n)| \leqslant cg(n).$$

[3] Uma função $f(n)$ que expressa o consumo de um recurso por algum algoritmo é uma função de $\mathbb{N}$ em $\mathbb{N}$, mas aqui vamos tratar a notação assintótica de modo um pouco mais geral que nos permitirá usá-la em outras situações.

Na definições dadas nas equações (2.1) e (2.2) o símbolo "=" não é a igualdade no sentido usual, é um abuso da notação em troca de algumas conveniências. Temos que $n = O(n^2)$ e $n^2 + 2n + 1 = O(n^2)$ mas $n \neq n^2 + 2n + 1$. Quando usamos essas definições, em geral, omitimos o parâmetro $n \to \infty$.

Proposição 2.2 *Se* $f(n) = o(g(n))$ *então* $f(n) = O(g(n))$.

A prova é imediata da definição de limite. A recíproca da proposição 2.2 não vale, como pode ser visto tomando-se $f(n) = g(n) = n^2$.

Proposição 2.3 *Se* $f_1(n) = O(g_1(n))$ *e* $f_2(n) = O(g_2(n))$ *então*

1. $f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\})$.
2. $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$.
3. $a \cdot f_1(n) = O(g_1(n))$ *para toda constante* $a \in \mathbb{R}$.

Demonstração. Vamos provar o item 1. Digamos que $|f_1(n)| \leqslant c_1 g_1(n)$ para todo $n \geqslant n_1$ e que $|f_2(n)| \leqslant c_2 g_2(n)$ para todo $n \geqslant n_2$, onde n_1, n_2, c_1, c_2 são as constantes dadas pela definição de notação O. Então

$$\begin{aligned} |f_1(n) + f_2(n)| \leqslant |f_1(n)| + |f_2(n)| &\leqslant c(g_1(n) + g_2(n)) \text{ com } c = \max\{c_1, c_2\} \\ &\leqslant 2c(\max\{g_1(n), g_2(n)\}) \end{aligned}$$

para todo $n \geqslant \max\{n_1, n_2\}$, o que prova a afirmação.

As outras propriedades são deduzidas de modo análogo e são deixadas como exercício. □

É preciso observamos alguns cuidados com o teorema acima pois, por exemplo, não vale que $1^k + 2^k + \cdots + (n-1)^k + n^k = O(\max\{1^k, 2^k, \ldots, (n-1)^k, n^k\}) = O(n^k)$. O problema aqui é que o máximo só pode ser tomado sobre um número de termos que não dependa de n. De fato, temos que $1^k + \cdots + n^k = O(n^{k+1})$ e que $1^k + \cdots + n^k \neq O(n^k)$.

Fica como exercício a verificação do seguinte resultado.

Proposição 2.4 *Se* $f(n) = O(g(n))$ *e* $g(n) = O(h(n))$ *então* $f(n) = O(h(n))$.

Alguns exemplos são dados a seguir.

1. $an^2 + bn + c = O(n^2)$ para toda constante $a > 0$.

   Primeiro, observamos que $|an^2 + bn + c| \leqslant |a|n^2 + |b|n + |c|$ e agora usamos a proposição 2.3 em cada operando das somas, de $an^2 = O(n^2)$, $|b|n = O(n)$ e $|c| = O(1)$ temos $an^2 + |b|n + |c| = O(\max\{n^2, n, 1\}) = O(n^2)$. Analogamente, para todo $k \in \mathbb{N}$

$$\sum_{i=0}^{k} a_i n^i = O(n^k).$$

2. $n\log(n!) = O(n^2 \log n)$.

   Primeiro, temos $n = O(n)$. Depois, $n! = \prod_{i=1}^{n} i < \prod_{i=1}^{n} n = n^n$. Como log é crescente $\log(n!) < \log(n^n) = n\log(n)$, portanto $\log(n!) = O(n\log(n))$. Pela proposição 2.3 $n\log(n!) = O(n^2 \log n)$.

3. Para toda constante $a > 1$, $\log_a(n) = O(\log(n))$. De fato,

$$\log_a(n) = \frac{1}{\log a} \log n$$

   porém $\frac{1}{\log a} = O(1)$ e $\log n = O(\log n)$ e pela proposição 2.3 $\log_a(n) = O(\log(n))$.

Convenções de uso da notação assintótica. Ao usar notação assintótica nós desconsideramos os coeficientes, por exemplo, usamos $O(n^2)$ ao invés de $O(3n^2)$ e $O(1)$ ao invés de $O(1024)$ ainda que, como classes de funções, $O(n^2) = O(3n^2)$ e $O(1) = O(1024)$.

Escrevemos no argumento de $O(\cdot)$ somente o termo mais significativo, por exemplo, usamos $O(n^2)$ ao invés de $O(2n^2 + 5n\log n + 4)$. Nesse caso, da proposição 2.3 vale que $2n^2 + 5n\log n + 4 = O(\max\{n^2, n\log n, 1\}) = O(n^2)$.

Um algoritmo eficiente com respeito ao tempo de execução é um algoritmo que nas instâncias de tamanho n tem tempo de execução $O(n^k)$ para algum inteiros positivo k fixo.

Quando notação assintótica aparece em equações, na forma "expressão 1 = expressão 2" onde "expressão" são expressões algébricas que envolvem notação assintótica, os termos assintóticos em "expressão 1'são quantificados universalmente, enquanto que os termos assintóticos em "expressão 2" são quantificados existencialmente. Por exemplo, em $n^3 + O(n^2) = O(n^3) + n^2 + n$ entendemos como

$$\text{para todo } f(n) = O(n^2), \text{ existe } g(n) = O(n^3), \text{ tal que } n^3 + f(n) = g(n) + n^2 + n$$

para todo n suficientemente grande.

Notação Ω e Θ. A notação Ω foi introduzida por Donald Knuth e escrevemos $f(n) = \Omega(g(n))$ se, e somente se, $g(n) = O(f(n))$. Escrevemos $f(n) = \Theta(g(n))$ se, e somente se, $f(n) = O(g(n))$ e $g(n) = O(f(n))$.

Exercício 2.5. Suponha que $\lim_{n\to\infty} \frac{f(n)}{g(n)} = L$. Verifique se valem as firmações

(a) $n^{1,5} = O(n^2)$.

(b) $\frac{n^2}{10} = O(n)$.

(c) $n^2 - 100n = O(n^2)$.

(d) $n\log(n) = O(n^2)$.

(e) $n = O(n \log n)$.

(f) $2^n = O(n)$.

(g) $2^n = O(2^{n-1})$.

(h) Se $L > 0$, $f(n) = \Theta(g(n))$.

(i) Se $L = 0$, $f(n) = O(g(n))$ mas $f(n) \neq \Theta(g(n))$.

(j) Se $L = \infty$, $f(n) = \Omega(g(n))$ mas $f(n) \neq \Theta(g(n))$.

(k) Se $a_k > 0$, então $\sum_{i=0}^{k} a_i n^i = \Theta(n^k)$.

O algoritmo de Euclides. O seguinte algoritmo é conhecido como Algoritmo de Euclides (circa 300 aC). Ele computa o maior divisor comum de dois inteiros quaisquer. É um algoritmo recursivo baseado no fato de que $\text{mdc}(a, b) = \text{mdc}(b, a \bmod b)$ onde $a \bmod b$ é o resto na divisão de a por b.

Instância : um par de inteiros (a, b).
Resposta : $\text{mdc}(a, b)$.

1 $a \leftarrow |a|$
2 $b \leftarrow |b|$;
3 **se** $b = 0$ **então responda** a.
4 **senão responda** $\text{mdc}(b, a \bmod b)$.

Algoritmo 6: $\text{mdc}(a, b)$.

O algoritmo de Euclides está correto: $\text{mdc}(a, b) = \text{mdc}(|a|, |b|)$ e $\text{mdc}(a, 0) = |a|$, para todo $a \in \mathbb{Z}$, portanto, podemos assumir que $a > b > 0$. Também, notemos que o algoritmo termina pois $0 \leqslant a \bmod b < b$, pelo Teorema da Divisão, logo o valor da variável b decresce estritamente a cada iteração. Para concluir, observamos que se a e $b > 0$ são inteiros então

$$d|a \text{ e } d|b \Leftrightarrow d|b \text{ e } d|a \bmod b$$

donde deduzimos que se a e $b > 0$ são inteiros então $\text{mdc}(a, b) = \text{mdc}(b, a \bmod b)$.

O algoritmo de Euclides computa $\text{mdc}(a, b)$ em tempo $O(\log(|a|)\log(|b|))$. Consideremos a seguinte sequência dos parâmetros das chamadas recursiva do algoritmo de Euclides, começando com $(a, b) = (r_0, r_1)$

$$\begin{aligned}
(r_1, r_0 \bmod r_1) &= (r_1, r_2) \\
(r_2, r_1 \bmod r_2) &= (r_2, r_3) \\
&\vdots \\
(r_{\ell-2}, r_{\ell-3} \bmod r_{\ell-2}) &= (r_{\ell-2}, r_{\ell-1}) \\
(r_{\ell-1}, r_{\ell-2} \bmod r_{\ell-1}) &= (r_{\ell-1}, r_\ell)
\end{aligned}$$

e $r_{\ell+1} = 0$. O custo em cada linha é o de uma divisão, ou seja, na linha i, $1 \leqslant i < \ell$, o custo é $O(\log(r_i)\log(q_i))$, onde $r_{i-1} = r_i q_i + r_{i+1}$. O custo total é

$$\sum_{i=1}^{\ell} \log(r_i)\log(q_i) \leqslant \log(b) \sum_{i=1}^{\ell} \log(q_i) = \log(b)\log(q_1 q_2 \cdots q_\ell) \leqslant \log(b)\log(a)$$

pois $a = r_0 \geqslant r_1 q_1 \geqslant r_2 q_2 q_1 \geqslant \cdots \geqslant r_\ell q_\ell \cdots q_2 q_1 \geqslant q_\ell \cdots q_2 q_1$.

Teorema 2.6 *O tempo de execução de pior caso do algoritmo* Euclides(a, b) *é* $O(\log(|a|)\log(|b|))$. □

Exercício 2.7 (o pior caso do Algoritmo de Euclides). No que segue, f_k é o k-ésimo número de Fibonacci, definido recursivamente por $f_1 = 1$, $f_2 = 1$ e $f_k = f_{k-1} + f_{k-2}$ para todo $k \geqslant 3$. Mostre que o algoritmo de Euclides com entradas f_{k+2} e f_{k+1} executa k chamadas recursivas. Prove o seguinte resultado: *se* (a, b) *é o menor par de inteiros positivos que faz o algoritmo de Euclides executar* k *chamadas recursivas então* $(a, b) = (f_{k+1}, f_{k+2})$.

O algoritmo de Euclides estendido. O algoritmo de Euclides Estendido determina uma solução (x, y) inteira da equação

$$ax + by = \text{mdc}(a, b)$$

para $a,\ b \in \mathbb{Z}$ quaisquer. Esse algoritmo é bastante útil na seguinte situação. Se $\text{mdc}(a, n) = 1$ para inteiros a e $n > 1$, então a equação $ax \equiv c \pmod{n}$ tem solução, ou seja, existem x e y inteiros tais que $ax + ny = c$ e ao algoritmo estendido os encontra. No caso $c = 1$ a solução é um **inverso multiplicativo de** a **módulo** n.

Instância : (a, b) par de inteiros não negativos.
Resposta : uma terna (d, x, y) tal que $d = \text{mdc}(a, b) = ax + by$.

1 **se** $b = 0$ **então responda** $(a, 1, 0)$.
2 $(d, x, y) \leftarrow$ Euclides_estendido$(b, a \bmod b)$
3 **responda** $(d, y, x - \lfloor a/b \rfloor y)$.

Algoritmo 7: Euclides_estendido(a, b).

Uma execução do algoritmo acima com entradas 86 e 64 resulta nos seguintes valores

a	b	$\lfloor a/b \rfloor$	x	y	d
86	64	1	3	−4	2
64	22	2	−1	3	2
22	20	1	1	−1	2
20	2	10	0	1	2
2	0	—	1	0	2

Exercício 2.8 (tempo de execução do algoritmo euclidiano estendido). Verifique que o tempo de execução do algoritmo 7 é $O(\log(|a|)\log(|b|))$.

Exercício 2.9 (correção do algoritmo euclidiano estendido). Prove que o algoritmo 7 responde corretamente. Para isso, suponha que (d', x', y') são os valores atribuídos na linha 2

$$d' = bx' + (a \bmod b)y' = bx' + (a - \lfloor a/b \rfloor b)y' = ay' + b(x' - \lfloor a/b \rfloor y').$$

A prova segue por indução.

Exponenciação modular. Computar 2^n no modo tradicional é extremamente custoso quando n é grande; com 100 dígitos isso daria cerca de 10^{100} passos o que é impossível de realizar manualmente sem atalhos. Em geral a^b avaliado por multiplicações repetidas tem tempo de execução $\Omega(b(\log a)^2)$. Um jeito mais esperto é "elevar ao quadrado" repetidas vezes, por exemplo, para calcular 2^{24} podemos começar com $2^3 = 8$, elevá-lo ao quadrado, o que resulta $2^6 = 62$, elevá-lo ao quadrado, o que resulta $2^{12} = 4.096$, e elevá-lo ao quadrado, o que resulta $2^{24} = 16.777.216$. Para calcular 2^{29} com esse método, recursivamente, $2^{29} = 2 \cdot 2^{28}$, a raiz de 2^{28} é 2^{14} cuja raiz é 2^7 que é $2 \cdot 2^6$ que por sua vez é $2^2 \cdot 2^3$. Em resumo, a^b é avaliado com base na observação de que

$$a^b = \begin{cases} (a^{b/2})^2 & \text{se } b \text{ é par,} \\ a \cdot a^{b-1} & \text{se } b \text{ é ímpar.} \end{cases}$$

O seguinte algoritmo é uma versão iterativa da recursão acima para calcular $a^b \pmod n$. Seja $b_k b_{k-1} \dots b_1 b_0$ a representação binária de b e defina

$$c_i := b_k 2^{i-1} + b_{k-1} 2^{i-2} + \cdots + b_{k-i+1} 2^0$$
$$d_i := a^{c_i} \bmod n$$

para $i \geqslant 1$ com $c_0 = 0$ e $d_0 = 1$. Computamos d_{i+1} a partir de d_i da seguinte forma

$$c_{i+1} = \begin{cases} 2c_i, & \text{se } b_{k-i} = 0 \\ 2c_i + 1, & \text{se } b_{k-i} \neq 0 \end{cases};$$

e

$$d_{i+1} = \begin{cases} {d_i}^2 \bmod n = a^{c_{i+1}} \bmod n = (a^{c_i})^2 \bmod n & \text{se } b_{k-i} = 0 \\ a \cdot {d_i}^2 \bmod n = a^{c_{i+1}} \bmod n = a \cdot (a^{c_i})^2 \bmod n, & \text{se } b_{k-i} \neq 0. \end{cases}$$

Portanto, $c_{k+1} = b_k 2^k + \cdots + b_1 2 + b_0 = b$ e $d_{k+1} = a^b \bmod n$.

Exemplo 2.10. Vamos usar essa estratégia para calcular $2^{24} \bmod 25$. Primeiro, 24 em base 2 fica $b = 11000$.

i	0	1	2	3	4	5
b_{4-i}	1	1	0	0	0	
c_i	0	1	3	6	12	24
d_i	1	2	8	14	21	16

Portanto $2^{24} \equiv 16 \pmod{25}$. $\Diamond$

Notemos que em toda iteração os valores de d têm no máximo tantos dígitos quanto n, ou seja, têm $O(\log n)$ dígitos, portanto, a multiplicação e o resto têm tempo de execução $O(\log^2 n)$. O número de iterações é a quantidade de bits na representação de b, logo $O(\log b)$. Isso prova o seguinte resultado.

Teorema 2.11 *O algoritmo 8 abaixo com* $a = O(\log n)$ *determina* $a^b \bmod n$ *em tempo* $O(\log(b)\log^2(n))$.

Instância : inteiros não negativos a, b e $n > 1$.
Resposta : $a^b \bmod n$.

```
1  c ← 0;
2  d ← 1;
3  Seja b_k b_{k−1} ... b_1 b_0 a representação binária de b;
4  para i de k até 0 faça
5  |  c ← 2 · c;
6  |  d ← d · d mod n;
7  |  se b_i = 1 então
8  |  |  c ← c + 1;
9  |  |  d ← d · a  mod n;
10 responda d.
```

Algoritmo 8: exponenciação modular.

Observação 2.12 (*sobre o custo computacional das operações aritméticas*). O custo das operações aritméticas usados acima são os custos dos algoritmos escolares, não os dos mais eficientes. Se $M(n)$ é o custo para multiplicar dois números de até n bits, então temos os seguintes tempos de execução.

Multiplicação	$M(n)$
Divisão	$O(M(n))$ (Newton–Raphson)
MDC	$O(M(n)\log n)$ (Stehlé–Zimmermann)
Exponenciação modular	$O(kM(n))$, k é o tamanho do expoente

Tabela 2.1: custo das operações aritméticas.

O tempo de uma multiplicação do algoritmo escolar é $M(n) = O(n^2)$. Atualmente, o algoritmo mais usado é o algoritmo de Schönhage–Strassen de 1971. O tempo de execução de pior caso é $O(n \log n \log\log n)$ para dois números de n dígitos. Esse algoritmo foi o método de multiplicação mais rápido até 2007, quando o algoritmo de Fürer foi anunciado. Entretanto, o algoritmo de Fürer só alcança uma vantagem para valores astronomicamente grandes e não é usado na prática.

Harvey e Van Der Hoeven (2019) publicaram um algoritmo de tempo $O(n \log n)$ para a multiplicação de inteiros. Como Schönhage e Strassen conjecturam que $n\log(n)$ é mínimo necessário para a multiplicação esse pode ser o "melhor possível", porém, até o momento esse algoritmo também não é útil na prática pois os autores observam que as estimativas de custo valem para números com pelo menos $2^{4.096}$ bits. O número de átomos estimado no universo é $10^{80} \approx 2^{266}$.

2.2 ALGORITMOS ALEATORIZADOS

Nos algoritmos aleatorizados o tempo de execução depende do tempo para os sorteios realizados. Formalmente, consideramos que os algoritmos sorteiam bits de modo uniforme e independente, com cada sorteio em tempo constante, de modo que o sorteio de $a \in \Omega$ consome tempo proporcional ao tamanho da representação binária dos elementos de Ω, isto é, $O(\log|\Omega|)$. Em geral, se $\log_2|\Omega|$ for polinomial no tamanho da entrada então um sorteio não afeta a ordem do tempo de execução de um algoritmo eficiente e podemos considerar o tempo de um sorteio como sendo constante.

No teste de identidade polinomial, algoritmo 2 na página 25, o tempo de execução depende do tempo para o sorteio do número a e do tempo para computar $f(a)$. O tempo para computar $f(a)$ depende da representação de f e será eficiente se for feito em tempo polinomial no tamanho da representação de f. Isso se verifica quando o polinômio é dado explicitamente e, assim, o algoritmo 2 é um algoritmo probabilístico de tempo polinomial que erra com probabilidade limitada. Esse tipo de algoritmo, com tempo de execução determinístico e chance de errar, é chamado na literatura de algoritmo de *Monte Carlo*.

O algoritmo gerador de números aleatórios, algoritmo 3, página 42, sempre responde certo. Em contrapartida o tempo de execução pode ser diferente em execuções distintas com a mesma instância. Esse tipo de algoritmo é chamado de um algoritmo de *Las Vegas*. Uma execução do algoritmo 3 com entrada M e com uma única rodada do laço tem tempo de execução $O(\log M)$ para sortear os bits e realizar a soma; outra execução com a mesma entrada M pode mais azarada e precisar de 2 rodadas, mas o tempo de execução continua $O(\log M)$; outra execução com a mesma entrada M pode ser muito azarada e precisar de M rodadas e o tempo de execução será $O(M \log M)$, que é exponencial no tamanho da entrada! O que nos tranquiliza é que sabemos, do capítulo 1, que isso é muito pouco provável. Ainda, se tomamos a média do número de rodadas ponderada pela probabilidade como uma medida representativa do número de rodados do laço que, tipicamente, o algoritmo executa, então temos que serão no máximo 2 rodadas. De fato, se $p = M/2^k$ é a probabilidade com que o algoritmo executa exatamente uma rodada, $(1-p)p$ é a probabilidade com que o algoritmo executa exatamente duas rodadas, $(1-p)^2p$ para três rodadas e assim por diante, o algoritmo executa exatamente k rodadas com probabilidade $(1-p)^{k-1}p$, portanto, o número médio de rodadas é (veja (s.6)

do apêndice)

$$\sum_{k \geqslant 1} k(1-p)^{k-1}p = \frac{1}{p} = \frac{2^k}{M} \leqslant 2 \tag{2.3}$$

rodadas, onde $k = \lfloor \log_2 M \rfloor + 1$. Portanto, o *tempo médio de execução* é $2 \cdot O(\log M)$, ou seja, $O(\log M)$.

Um fato importante a ser ressaltado neste momento é que esse tempo médio que calculamos é sobre os sorteios do algoritmo, diferente do que fizemos com os algoritmos de busca sequencial e de ordenação por inserção, no início deste capítulo, onde a média foi feita sobre o tempo de execução nas diferentes entradas para o algoritmo. A única fonte de aleatoriedade nos algoritmos probabilísticos são os bits aleatórios que ele usa, não há uma medida no conjunto de instâncias.

Dado um algoritmo A e uma instância x para A, podemos representar as possíveis computações de A com x com uma árvore binária $T_{A,x}$ (figura 2.1) em que cada ramificação significa que um sorteio

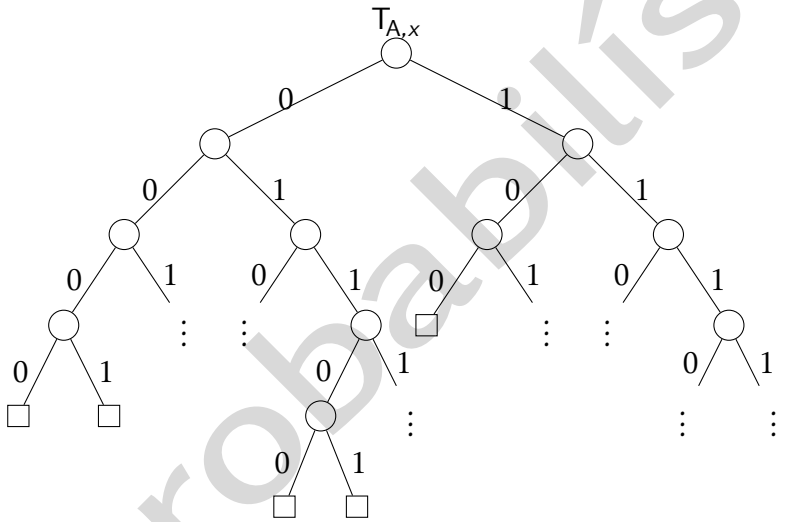


Figura 2.1: árvore de execução de A com instância x.

foi realizado, uma computação c específica está associada a um caminho da raiz até alguma folha (□) e ela ocorre com probabilidade $\mathbb{P}(c) = 2^{-|c|}$ onde $|c|$ é o comprimento (número de arestas) no caminho.

Para termos um exemplo simples de uma árvore de execução, vamos modificar ligeiramente o algoritmo 2 para que faça até dois sorteios: se no primeiro $f(a) \neq 0$ então pode parar e responder *não*, senão faça mais um sorteio. Além disso, as instâncias são polinômios de grau no máximo 2.

```
1 a ←_U {1,2,3,4};
2 se f(a) ≠ 0 então responda não.
3 senão
4 |   a ←_U {1,2,3,4};
5 |   se f(a) ≠ 0 então responda não.
6 |   senão responda sim.
```

As computações desse algoritmo com entrada $(x-1)(x-3)$ em função dos sorteios são caracterizadas pelas sequências: $(1,1,\textit{não})$, $(1,2,\textit{não})$, $(1,3,\textit{sim})$ e $(1,4,\textit{não})$, $(2,\textit{não})$, $(4,\textit{não})$, $(3,1,\textit{não})$, $(3,2,\textit{não})$, $(3,3,\textit{sim})$ e $(3,4,\textit{não})$, esquematizadas na figura 2.2. Dos sorteios independentes decorre que a probabilidade de uma computação é o produto das probabilidades no ramo daquela computação, logo

$$\mathbb{P}[\,\text{erro}\,] = \mathbb{P}((1,3,\textit{sim})) + \mathbb{P}((3,3,\textit{sim})) = \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2}.$$

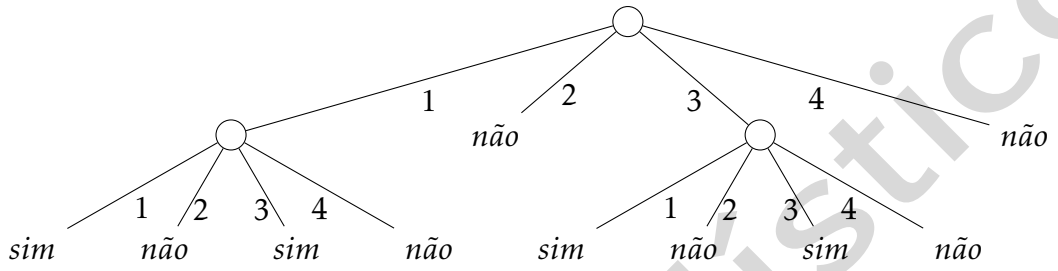


Figura 2.2: árvore de execução do algoritmo acima com entrada $(x-1)(x-3)$.

Definimos o modelo probabilístico discreto $(\Omega_{A,x}, \mathbb{P})$ com o espaço amostral formados pelos caminhos c na árvore de execução e a medida $2^{-|c|}$ em que $c \in \Omega_{A,x}$ é um ramo da árvore e $|c|$ o número de arestas em c.

Exercício 2.13. Verifique que para quaisquer dois ramos distintos de $T_{A,x}$ vale que a sequência binária de um ramo não pode ser prefixo da sequência de outro ramo. Prove que $\rho = \sum_c 2^{-|c|} \leqslant 1$. Nessa situação, ρ é a probabilidade com que A com instância x termina a computação.

Por fim, registramos que há, ainda, algoritmos aleatorizados que têm probabilidade de errar e têm probabilidade de demorar muito pra terminar e que em algumas referências são chamados de *Atlantic City*.

2.2.1 CORTE-MÍNIMO EM GRAFOS

Um *grafo* G é dado por um par de conjuntos (V,E) em que V é finito, é o conjunto dos *vértices* de G, e $E \subset \binom{V}{2}$. O *grau* de um vértice $x \in V$ em G é a quantidade de arestas de E a que x pertence. Se contamos o número de pares $(v,e) \in V \times E$ tais que $v \in e$ temos, pela definição de grau, que a quantidade de pares é a soma dos graus dos vértices. Por outro lado, cada aresta é composta por dois vértices de modo que a quantidade de pares é $2|E|$. Esse resultado quase sempre é o primeiro teorema nos textos de Teoria dos Grafos: *em todo grafo, a soma dos graus dos vértices é duas vezes o número de arestas do grafo*. Nesta seção assumimos, sem perda de generalidade, que os grafos são sobre os vértices $V = \{1,2,\ldots,n\}$ para algum n.

Um subconjunto de arestas de um grafo $G = (V, E)$ da forma

$$\nabla(A) := \left\{ \{u, v\} \in E \colon u \in A \text{ e } v \in \overline{A} \right\}$$

é chamado de **corte definido por** A em G.

Exemplo 2.14. A figura 2.3 abaixo mostra um grafo G e um corte de arestas $\nabla(A)$ definido por $A = \{0,1,2,7,8\}$, o qual é formado pelas arestas (em azul na figura) $\{0,4\}$, $\{0,5\}$, $\{1,3\}$, $\{1,6\}$, $\{8,3\}$,

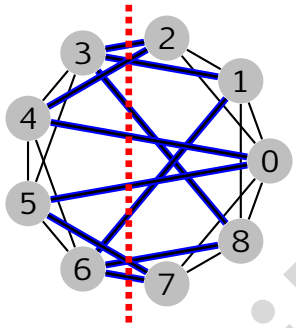


Figura 2.3: o corte definido por $\{0,1,2,7,8\}$ são as arestas em azul.

$\{6,8\}$, $\{5,7\}$, $\{6,7\}$, $\{2,3\}$, $\{2,4\}$ que cruzam a reta vertical pontilhada. ◊

Um **corte mínimo** em G é um corte com

$$\text{mincut}(G) := \min\left\{ \left|\nabla(A)\right| \colon \varnothing \neq A \subsetneq V \right\}$$

arestas.

No grafo do exemplo 2.14, o corte definido por $\{2\}$ é mínimo, assim como o definido por $\{7\}$. O problema em que estamos interessados é enunciado como segue.

Problema computacional do corte mínimo em um grafo (MINCUT):

Instância : um grafo G e um inteiro positivo k.

Resposta : *sim* se $\text{mincut}(G) \leqslant k$, *não* caso contrário.

Para explicar um algoritmo probabilístico para esse problema, precisaremos de uma definição mais geral de grafo. Em um *multigrafo* as arestas formam um multiconjunto no qual entre um par de vértices pode haver mais de uma aresta. Seja M um multigrafo. Para qualquer aresta $e \in E(M)$ em M definimos por *contração da aresta* e a operação que resulta no multigrafo com os extremos de e identificados e as arestas com esses extremos removidos, o multigrafo resultante é denotado por M/e (veja uma ilustração na figura 2.5 abaixo).

A ideia do algoritmo para decidir se $\text{mincut}(G) \leqslant k$ é repetir as operações

1. sortear uniformemente uma aresta,

2. contrair a aresta sorteada,

até que restem 2 vértices no multigrafo. As arestas múltiplas que ligam esses 2 vértices são arestas de um corte no grafo original. Os próximos parágrafos ilustram a ideia do algoritmo que será apresentado em seguida; para facilitar a compreensão mantemos nos rótulos dos vértices todas as identificações realizadas. Considere o grafo G representado pelo diagrama da figura 2.4. A figura

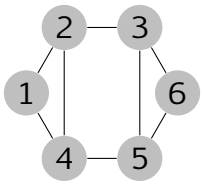


Figura 2.4: exemplo de um grafo.

2.5 abaixo representa uma sequência de três contrações de arestas, a aresta que sofre a contração está em vermelho. Se no multigrafo final da figura 2.5 tem como a próxima contração de aresta a que

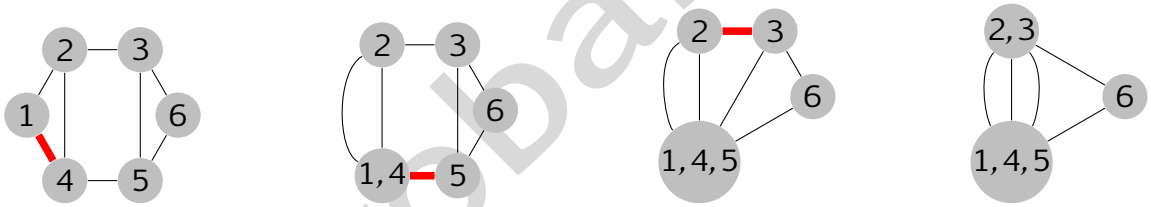
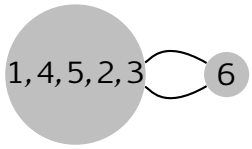


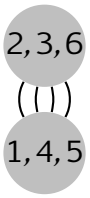
Figura 2.5: uma sequência de três contrações de aresta. Uma contração na aresta em vermelho resulta no multigrafo à direita na sequência. Para manter o registro das contrações acumulamos os rótulos nos vértices.

identifica os vértices representados por $1,4,5$ e por $2,3$ então o multigrafo resultante dessa contração é mostrado na figura 2.6(a) que corresponde ao corte definido por $A = \{6\}$ no grafo original G. Esse corte em G tem duas arestas e é um corte mínimo. Por outro lado, se identificarmos $2,3$ com 6 então o multigrafo obtido corresponde ao corte definido por $A = \{2,3,6\}$ em G e que tem 4 arestas, como na figura 2.6(b).

Exercício 2.15. Seja G um grafo. Prove que após uma sequência qualquer de contrações de arestas de G, um corte no multigrafo resultante corresponde a um corte no grafo original. Conclua que a sequência de operações realizadas, sortear aresta e contrair a aresta sorteada até que restem 2 vértices, determina um corte em G.



(a)



(b)

Figura 2.6: dois resultados possíveis para contração a partir do último multigrafo da figura 2.5. Em (a) o resultado da contração da aresta de extremos $1,4,5$ e $2,3$. Em (b) o resultado da contração da aresta $2,3$ com 6. Ambos correspondem a um corte no grafo original G.

Sejam $G = (V,E)$ um grafo com n vértices e $C = \nabla(A)$ um corte mínimo em G. Vamos mostrar que a probabilidade do algoritmo que descrevemos encontrar o corte C é pelo menos $\binom{n}{2}^{-1}$. De $\text{mincut}(G) = |C|$ o grau mínimo de um vértice em G é pelo menos $|C|$, portanto, G tem pelo menos $|C|n/2$ arestas.

O espaço amostral nesse caso é dado pelas sequências de $n-2$ arestas distintas que correspondem as escolhas aleatórias do algoritmo. O algoritmo executado sobre G encontra o corte mínimo $C = \nabla(A)$ se nas $n-2$ rodadas somente contrai arestas com ambos os extremos em A, ou com ambos extremos em $\overline{A}$.

Denotemos por B_i o evento "a i-ésima escolha aleatória, a aresta e_i, não está em C". A probabilidade de escolher uma aresta de C na primeira escolha aleatória é

$$\frac{|C|}{|E(G)|} \leqslant \frac{|C|}{|C|n/2} = \frac{2}{n}$$

logo $\mathbb{P}(B_1) \geqslant 1 - \frac{2}{n}$. Agora, denotemos por G_1 o grafo resultante da primeira rodada de contrações. A probabilidade de escolher uma aresta de C na segunda escolha, dado que na primeira escolha não ocorreu uma aresta de C é

$$\frac{|C|}{|E(G_1)|} \leqslant \frac{|C|}{|C|(n-1)/2} = \frac{2}{n-1}$$

pois o multigrafo tem $n-1$ vértices e grau mínimo pelo menos $|C|$, logo

$$\mathbb{P}(B_2 \mid B_1) \geqslant 1 - \frac{2}{(n-1)}$$

e, genericamente, na i-ésima escolha a probabilidade de escolher uma aresta de C dado que até agora não foi escolhida uma aresta de C é

$$\mathbb{P}\left(B_i \;\middle|\; \bigcap_{j=1}^{i-1} B_j\right) \geqslant 1 - \frac{2}{n-i+1} = \frac{n-i-1}{n-i+1}.$$

A probabilidade de nenhuma aresta escolhida ser de C é $\mathbb{P}(B_1 \cap B_2 \cap \cdots \cap B_{n-2})$ e pelo exercício 1.24, página 30, temos que

$$\mathbb{P}\left(\bigcap_{i=1}^{n-2} B_i\right) \geqslant \prod_{i=1}^{n-2}\left(\frac{n-i-1}{n-i+1}\right) = \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}.$$

Este algoritmo, devido a Karger (1993), recebe um grafo G com pelo menos 3 vértices e um inteiro positivo k, e responde *sim* ou *não*. Quando o algoritmo responde *sim* é porque foi descoberto um corte com até k arestas, portanto, o corte mínimo tem tamanho no máximo k. Por outro lado, a resposta *não* significa que o algoritmo não achou um corte com até k arestas, o que não significa que o grafo não o tenha, portanto, a resposta *não* pode estar errada.

Instância : um grafo G com $n \geqslant 3$ vértices e $k \in \mathbb{N}$.
Resposta : *sim* caso $\text{mincut}(G) \leqslant k$, senão *não* com probabilidade de erro $< 1/2$.

```
1  repita
2      i ← 0;
3      G_0 ← G;
4      repita
5          e ←_U E(G_i);
6          G_{i+1} ← G_i/e;
7          i ← i + 1;
8      até que i = n − 2;
9      se |E(G_{n−2})| ⩽ k então responda sim.
10 até que complete (n 2) rodadas;
11 responda não.
```

Algoritmo 10: corte mínimo.

Acima provamos o seguinte resultado.

Proposição 2.16 *Seja* G *um grafo com pelo menos três vértices. Fixado um corte mínimo* C *em* G, *a probabilidade do algoritmo 10 determinar* C *no laço da linha 4 é pelo menos* $1/\binom{n}{2}$. □

Agora, vamos determinar a probabilidade de erro.

Teorema 2.17 *Supondo que as rodadas do laço da linha 1 sejam independentes temos*

$$\mathbb{P}[\,\text{erro}\,] = \mathbb{P}[\,\text{resposta } \textit{não} \mid \text{mincut}(G) \leqslant k\,] < \frac{1}{e}.$$

Demonstração. Se G tem um corte com no máximo k arestas, então a probabilidade do algoritmo não encontrar um tal corte em nenhuma das iterações do laço na linha 1 é no máximo

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}}.$$

Da sequência decrescente $(1-(1/n))^n$ convergir para e^{-1} (veja (s.8) no apêndice) temos que qualquer subsequência converge para o mesmo valor

$$\left(1-\frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}} \leqslant \frac{1}{e}$$

e isso prova o teorema. □

O número de instruções executadas no laço mais interno desse algoritmo é $O(|E|) = O(n^2)$. Contando as execuções do laço externo são $O(n^4)$ instruções executadas.

Notemos que se o número de rodadas na linha 10 for $\ell\binom{n}{2}$ então a probabilidade de erro é menor que $e^{-\ell}$, para $\ell = c\log n$ a probabilidade de erro é n^{-c}. Se executarmos o laço externo $c\log(n)\binom{n}{2}$ vezes, então o custo de tempo é $O(n^4 \log n)$.

2.2.2 VERIFICAÇÃO DO PRODUTO DE MATRIZES

Nesta seção veremos um algoritmo que recebe as matrizes A, B e C e verifica o produto $A\cdot B = C$ realizando menos operações aritméticas que o próprio produto $A\cdot B$ realiza usando o melhor algoritmo conhecido até hoje.

Problema computacional do teste de produto de matrizes:
Instância : matrizes A, B, C quadradas de ordem n.
Resposta : *sim* se $AB = C$, caso contrário *não*.

Esse teste pode ser feito usando o algoritmo usual (escolar) para o produto de matrizes, o qual realiza $O(n^3)$ operações aritméticas. Um dos algoritmos mais eficientes conhecidos é o de Coppersmith–Winograd (veja em Knuth, 1981), que realiza o produto de duas matrizes $n\times n$ perfazendo da ordem de $n^{2,376}$ operações aritméticas. O algoritmo aleatorizado devido a Freivalds (1977) apresentado a seguir decide se $AB = C$ com $O(n^2)$ operações aritméticas, mas pode responder errado caso $AB \neq C$.

A ideia do algoritmo de Freivalds para esse problema é que se $AB = C$ então $(vA)B = vC$ para todo vetor v e esse último teste tem custo da ordem de n^2 operações aritméticas. Porém, se $AB \neq C$ então é possível termos $(vA)B = vC$, por exemplo, caso v seja nulo. O que conseguimos garantir é que se o vetor v é aleatório então tal igualdade ocorre com probabilidade pequena.

Por exemplo, sejam

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \text{ e } B = \begin{pmatrix} 3 & 6 & 9 \\ 1 & 2 & 1 \\ 3 & 1 & 3 \end{pmatrix}, \; AB = \begin{pmatrix} 14 & 15 & 13 \\ 35 & 40 & 39 \\ 56 & 67 & 63 \end{pmatrix} \text{ e } C = \begin{pmatrix} 14 & 15 & 13 \\ 10 & 20 & 10 \\ 56 & 67 & 63 \end{pmatrix}$$

de modo que $AB \neq C$. Consideremos v um vetor binário. Para $v = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$ temos

$$vAB = 0\begin{pmatrix} 14 & 15 & 13 \end{pmatrix} + 1\begin{pmatrix} 35 & 40 & 39 \end{pmatrix} + 0\begin{pmatrix} 56 & 67 & 63 \end{pmatrix} = \begin{pmatrix} 35 & 40 & 39 \end{pmatrix}$$
$$vC = 0\begin{pmatrix} 14 & 15 & 13 \end{pmatrix} + 1\begin{pmatrix} 10 & 20 & 10 \end{pmatrix} + 0\begin{pmatrix} 56 & 67 & 63 \end{pmatrix} = \begin{pmatrix} 10 & 20 & 10 \end{pmatrix}$$

portanto, $vABC \neq vC$, enquanto que para $v = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$ temos

$$vAB = 0\begin{pmatrix} 14 & 15 & 13 \end{pmatrix} + 0\begin{pmatrix} 35 & 40 & 39 \end{pmatrix} + 1\begin{pmatrix} 56 & 67 & 63 \end{pmatrix} = \begin{pmatrix} 56 & 67 & 63 \end{pmatrix} \quad (2.4)$$
$$vC = 0\begin{pmatrix} 14 & 15 & 13 \end{pmatrix} + 0\begin{pmatrix} 10 & 20 & 10 \end{pmatrix} + 1\begin{pmatrix} 56 & 67 & 63 \end{pmatrix} = \begin{pmatrix} 56 & 67 & 63 \end{pmatrix} \quad (2.5)$$

portanto, $vABC = vC$. Notemos que $vAB = vC$ para todo $v \in \{0,1\}^3$ cuja segunda coluna seja 0, isto é, para

$$v \in \left\{\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}\right\}$$

vale a igualdade, para qualquer outro vetor binário vale a diferença $vAB \neq vC$. Nesse exemplo a probabilidade de erro quando sorteamos v é $4/8 = 1/2$. Se escolhermos as coordenadas do vetor v dentre $\{0,1,2\}$ então 9 dos 27 vetores farão essa estratégia falhar, ou seja, a probabilidade de erro é $1/3$.

Instância : matrizes A, B, C quadradas de ordem n.
Resposta : *sim* se $AB = C$, caso contrário *não* com probabilidade de erro no máximo $1/2$.

1 $v \leftarrow_{\mathcal{U}} \{0,1\}^n$;
2 **se** $(vA)B = vC$ **então responda** *sim*.
3 **senão responda** *não*.

Algoritmo 12: teste de produto de matrizes.

O produto $v(AB)$ é uma combinação linear das linhas de AB com coeficientes em v, assim como vC, como pode ser visto em (2.4) e (2.5), de modo que se $AB \neq C$ então há k linhas em que AB e C diferem, para algum $k \in \{1, \ldots, n\}$ e se as coordenadas do vetor v que são os coeficientes correspondentes a tais linhas forem 0, então teremos $v(AB) = vC$ e isso ocorre com probabilidade $(1/2)^k \leqslant 1/2$, pois $k > 0$.

Proposição 2.18 *Sejam* A, B, C *matrizes* $n \times n$. *Se* $AB \neq C$ *então* $\mathbb{P}_{v \in \{0,1\}^n}[(vA)B = vC] \leqslant 1/2$. □

A técnica a seguir (Mitzenmacher e Upfal, 2005) nos dá o cálculo da probabilidade de erro desse algoritmo e é útil em outras situações.

Princípio da decisão adiada. Muitas vezes um experimento probabilístico é modelado como uma sequência de escolhas aleatórias independentes. O princípio da decisão adiada diz que podemos optar pela ordem com que as escolhas são feitas, adiando as escolhas mais relevantes para efeito de cálculos. Aqui, ao invés de uma escolha uniforme $v \in \{0,1\}^n$ podemos considerar uma escolha de cada coordenada de v de modo uniforme e independente em $\{0,1\}$. Com a hipótese de que $AB \neq C$ a última escolha fica definida.

Outra demonstração da proposição 2.18. Assumamos que cada coordenada de $\mathsf{v} \in \{0,1\}^n$ é sorteada com probabilidade 1/2 e independentemente uma das outras. Sejam A, B e C matrizes como acima e $\mathsf{D} := \mathsf{AB} - \mathsf{C}$ matriz não nula. Queremos estimar $\mathbb{P}[\,\mathsf{vD} = 0\,]$.

Se $\mathsf{D} \neq 0$ e $\mathsf{vD} = 0$, então existem ℓ e c tais que $d_{\ell,c} \neq 0$ com $\sum_{j=1}^{n} v_j d_{j,c} = 0$, assim podemos escrever

$$v_\ell = -\frac{1}{d_{\ell,c}} \sum_{\substack{j=1 \\ j\neq\ell}}^{n} v_j d_{j,c} \tag{2.6}$$

e se consideramos que cada coordenada de v foi sorteada independentemente, podemos assumir que v_i, para todo $i \neq \ell$, foi sorteado antes de v_ℓ de modo que o lado direito da igualdade (2.6) acima fica determinado e a probabilidade de sortear v_ℓ que satisfaça a igualdade é ou 0, caso o valor da direita não esteja em $\{0,1\}$, ou 1/2 caso contrário. Portanto, $\mathbb{P}[\,\mathsf{vD} = 0\,] = \mathbb{P}[\,\mathsf{vAB} = \mathsf{vC}\,] \leqslant \frac{1}{2}$. □

Desaleatorização. Os bits aleatórios que um algoritmo usa para resolver um problema é um recurso que queremos otimizar, diminuir a quantidade utilizada sem penalizar muito os outros recursos, como por exemplo, o número de operações aritméticas realizadas. A isso chamamos de ***desaleatorização***. Por ora, vejamos uma versão do teste de produto de matrizes que usa menos bits aleatórios (devida a Kimbrel e Sinha, 1993).

Instância : matrizes A, B, C quadradas de ordem n.
Resposta : *sim* se $\mathsf{AB} = \mathsf{C}$, caso contrário *não* com probabilidade de erro no máximo 1/2.

1 $x \leftarrow_{\mathcal{U}} \{1,2,\ldots,2n\}$;
2 $\mathsf{v} \leftarrow \begin{pmatrix} 1 & x & x^2 & \ldots & x^{n-1} \end{pmatrix}$;
3 **se** $(\mathsf{vA})\mathsf{B} = \mathsf{vC}$ **então responda** *sim*,
4 **senão responda** *não*.

Algoritmo 13: teste de Kimbrel–Sinha para produto de matrizes.

Vamos assumir que $\mathsf{AB} \neq \mathsf{C}$ e supor que existam n escolhas em $\{1,2,\ldots,2n\}$, denotadas $x_1, x_2, \ldots, x_n$, para as quais $(\mathsf{vA})\mathsf{B} = \mathsf{vC}$. Com essas escolhas formamos a matriz de Vandermonde

$$\mathsf{V} = \mathsf{V}(x_1, x_2, \ldots, x_n) = \begin{pmatrix} 1 & x_1 & {x_1}^2 & \ldots & {x_1}^{n-1} \\ 1 & x_2 & {x_2}^2 & \ldots & {x_2}^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & {x_n}^2 & \ldots & {x_n}^{n-1} \end{pmatrix}.$$

Se para cada linha $\mathsf{v}(x_i) = \begin{pmatrix} 1 & x_i & x_i^2 & \ldots & x_i^{n-1} \end{pmatrix}$ dessa matriz vale que $(\mathsf{v}(x_i) \cdot \mathsf{A}) \cdot \mathsf{B} = \mathsf{v}(x_i) \cdot \mathsf{C}$, então $\mathsf{VAB} = \mathsf{VC}$, o que implica $\mathsf{AB} = \mathsf{C}$ pois V é invertível, contrariando $\mathsf{AB} \neq \mathsf{C}$. Portanto, no caso em que $\mathsf{AB} \neq \mathsf{C}$, temos que o algoritmo responde errado em no máximo $n-1$ escolhas de $x \in \{1,2,\ldots,2n\}$. A

probabilidade de erro é a probabilidade de escolher uma das no máximo $n-1$ escolhas ruins descritas no parágrafo acima e é menor que $n/m \leqslant 1/2$.

O número de bits aleatórios utilizados é $\lceil \log_2(2n) \rceil$, necessários para x. O algoritmo de Freivalds usa n bits aleatórios, portanto, exponencialmente mais.

2.2.3 IDENTIDADE POLINOMIAL REVISITADA

O problema computacional que nos interessa é: dado um polinômio p de n variáveis sobre um corpo $\mathbb{F}$, determinar se p é identicamente nulo.

Para descrevermos o caso geral do problema de testar a igualdade de polinômios começamos com algumas definições para evitar ambiguidades. O termo "polinômio" tem, usualmente, dois significados. No cálculo, geralmente significa uma função, cujas variáveis podem ser instanciadas. Na álgebra, é simplesmente uma soma formal de termos, com cada termo sendo um produto de uma constante e um monômio. Um *polinômio* formal sobre um corpo $\mathbb{F}$ com indeterminadas $x_1, x_2, \ldots, x_n$ é uma expressão finita da forma

$$\sum_{(i_1, i_2, \ldots, i_n)} c_{i_1, i_2, \ldots, i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \tag{2.7}$$

em que $c_{i_1, i_2, \ldots, i_n} \in \mathbb{F}$ são os coeficientes do polinômio e $i_1, i_2, \ldots, i_n$ são inteiros não negativos. O conjunto de todos esses polinômios é denotado por $\mathbb{F}[x_1, x_2, \ldots, x_n]$. O *grau total* do polinômio f dado pela equação (2.7), denotado por ∂f, é o maior valor de $i_1 + i_2 + \cdots + i_n$ dentre todos os índice para os quais $c_{i_1, i_2, \ldots, i_n} \neq 0$ e o *grau em* x_j, denotado $\partial_{x_j} f$, é o maior valor de i_j tal que $c_{i_1, i_2, \ldots, i_n} \neq 0$.

Quando usamos o significado funcional, polinômio não nulo significa que a função não é identicamente igual a zero. Quando usamos a definição algébrica, polinômio não nulo significa que pelo menos um coeficiente não é igual a zero. Essa distinção muitas vezes passa desapercebida porque se o polinômio é avaliado em um corpo infinito, como os números reais ou complexos (ou mesmo num domínio de integridade como $\mathbb{Z}$), então as duas noções coincidem, um polinômio como combinações lineares de termos com ao menos um coeficiente diferente de zero induz uma função que não é constante igual a zero e vice-versa.

Nos corpos finitos pode acontecer de polinômios distintos determinarem a mesma função polinomial. Por exemplo, no corpo finito com 7 elementos os polinômios 1 e $x^7 - x + 1$ são diferentes, mas como funções de $\mathbb{F}_7$ em $\mathbb{F}_7$ são iguais pois $x^7 \equiv x \pmod{n}$ pelo Pequeno Teorema de Fermat (teorema 2.41, página 89). No corpo finito com 3 elementos os polinômios 0 e $x^3 - x$ são diferentes, mas como funções de $\mathbb{F}_3$ em $\mathbb{F}_3$ são iguais pelo mesmo motivo (ou, porque $x^3 - x = x(x-1)(x-2)$).

Temos de fato dois problemas computacionais: dado um polinômio p, (1) decidir se p vale zero, como função, em todo elemento do corpo e (2) decidir se p na forma canônica tem todos os coeficientes nulos. Certamente, (2) implica (1) mas a inversa nem sempre vale. Em corpos infinitos como

$\mathbb{Q}$, $\mathbb{R}$ e $\mathbb{C}$ e em corpos finitos que contenham uma quantidade suficientemente grande de elementos ($|\mathbb{F}| > \partial p$) vale que (1) implica (2) e isso segue do teorema 2.19 abaixo. Portanto, sob a hipótese de que $|\mathbb{F}|$ é suficientemente grande os problemas (1) e (2) descritos acima são equivalentes.

Vamos fixar que para dois polinômios $p, q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ escrevemos $p = q$ se os polinômios são iguais *quando escritos* na forma canônica, como na equação (2.7). Por exemplo, a seguinte identidade entre expressões algébricas que correspondem a polinômios é verdadeira (ambas são o determinante da matriz de Vandermonde)

$$\sum_{\sigma \in \mathbb{S}_n} \text{sinal}\left(\prod_{1 \leqslant i < j \leqslant n} (\sigma(j) - \sigma(i))\right) \prod_{i=1}^{n} {x_i}^{\sigma(i)-1} = \prod_{1 \leqslant i < j \leqslant n} (x_j - x_i)$$

onde $\mathbb{S}_n$ é o conjunto de todas as permutações $\sigma \colon \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$.

Quando nos referirmos a uma instância do problema computacional, "dado p" significa que

- ou polinômio é dado como uma *caixa-preta* para a qual fornecemos $a \in \mathbb{F}^n$ e recebemos $p(a)$;
- ou é dado explicitamente por uma expressão como um determinante, por exemplo, ou, mais concretamente, como um circuito aritmético, que definiremos no capítulo 5. Nesse caso, faz parte do algoritmo avaliar $p(a)$.

Claramente, polinômios dados na representação canônica tornam o problema trivial. Ademais, obter o polinômio na forma canônica a partir do circuito aritmético ou reescrevendo uma expressão algébrica está fora de questão pois é uma tarefa que pode ser muito custosa se procuramos por um algoritmo eficiente para o problema. Por exemplo, o polinômio $\prod_{1 \leqslant i < j \leqslant n}(x_i - x_j)$ escrito como combinação linear de monômios resulta numa expressão da forma

$$\sum z_1 z_2 \cdots z_{\binom{n-1}{2}} z_{\binom{n}{2}}$$

onde $z \in \{x_i, x_j\}$ para cada par $1 \leqslant i < j \leqslant \binom{n}{2}$, com a soma de 2^n parcelas. Cada parcela tem tamanho da ordem de n^2 termos, totalizado uma expressão com tamanho da ordem de $n^2 2^n$, contra a expressão original com tamanho da ordem de n^2 termos.

Os dois modelos derivados da forma como é dado o polinômio são bem estudados porque o problema da identidade polinomial é muito importante. O primeiro é o chamado *modelo caixa preta* e o segundo é chamado de modelo *caixa branca*. A vantagem da representação implícita é que o polinômio pode ser avaliado eficientemente em qualquer entrada, desde que as operações do corpo possam ser feitas eficientemente, mas o problema é mais fácil no modelo caixa branca (Shpilka e Yehudayoff, 2010).

Problema computacional do teste de identidade polinomial (PIT):
Instância : circuito aritmético que computa um polinômio p de n variáveis sobre $\mathbb{F}$.
Resposta : *sim* se p é identicamente nulo, *não* caso contrário.

Problema 1 (*Identidade polinomial*). Existe um algoritmo determinístico para PIT que executa um número de operações em $\mathbb{F}$ que é polinomial no tamanho do circuito aritmético?

A estratégia do algoritmo probabilístico para esse problema é idêntica ao caso de uma variável, sorteamos n números e avaliamos o polinômio nessa n-upla. Notemos que não há uma generalização imediata do Teorema Fundamental da Álgebra pois, por exemplo, um polinômio sobre um corpo como $\mathbb{Q}$ e com várias variáveis pode ter um número infinito de raízes, como é o caso de $x_1 x_2$. A estratégia é baseada no seguinte teorema.

Teorema 2.19 (teorema de Schwartz–Zippel) *Sejam $\mathbb{F}$ um corpo, $p \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ um polinômio não nulo com grau total d e $S \subset \mathbb{F}$ finito. Então*

$$\mathop{\mathbb{P}}_{(x_1,\ldots,x_n)\in S^n}[p(x_1,\ldots,x_n) = 0] \leqslant \frac{d}{|S|}.$$

Esse resultado foi (re)descoberto várias vezes e de modo independente (DeMillo e Lipton, 1978; Schwartz, 1979; Zippel, 1979, dentre outros). Com o teorema 2.19 nós podemos resolver probabilisticamente o problema de identidade polinomial. Suponha que nos seja dado um polinômio $p(x_1,\ldots,x_n)$ de grau total $d < 2|\mathbb{F}| \leqslant |S|$. O algoritmo sorteia $r_1,\ldots,r_n$ em $S \subset \mathbb{F}$ (ou em $\mathbb{F}$) finito e suficientemente grande, computa $p(r_1,\ldots,r_n)$ e decide, de modo que a probabilidade de erro é no máximo $d/|S| \leqslant 1/2$.

Seja $p \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ um polinômio de grau positivo e $S \subset \mathbb{F}$ finito. Se $n = 1$, então p tem no máximo ∂p raízes em S pois em $\mathbb{F}[x_1]$, que é um domínio de fatoração única, vale o teorema da divisão de modo que se $r \in \mathbb{F}$ é raiz de p, então existe $q \in \mathbb{F}[x_1]$ tal que $p(x) = (x - r)q(x)$. Portanto são no máximo ∂p raízes.

Se $n = 2$, tome $k = \partial_{x_2} p$ (se $k = 0$ então caímos no caso anterior). Podemos reescrever p como

$$p(x_1, x_2) = a_k(x_1) \cdot x_2^k + a_{k-1}(x_1) \cdot x_2^{k-1} + \cdots + a_0(x_1)$$

com polinômios $a_i \in \mathbb{F}[x_1]$ para todo i. Tome o conjunto das raízes de p em S^2

$$R_p := \{(x_1, x_2) \in S^2 \colon p(x_1, x_2) = 0\}$$

e o conjunto dos pares com a primeira coordenada raiz de $a_k(x_1)$

$$R_{a_k} := \{(r, x_2) \in S^2 \colon a_k(r) = 0\}.$$

Como a_k é polinômio em uma variável há $\leqslant \partial a_k$ raízes em $\mathbb{F}$, logo há $\leqslant \partial a_k \cdot |S|$ pares em R_{a_k}, isto é, $|R_{a_k}| \leqslant \partial a_k \cdot |S|$.

Os pares em $R_p \setminus R_{a_k}$ são aqueles que anulam p, mas a primeira coordenada não anula a_k. Assim se $(r_1, r_2) \in R_p \setminus R_{a_k}$, então $p(r_1, x_2) \in \mathbb{F}[x_2]$ e $\partial p(r_1, x_2) = k$, logo, é anulado para $\leqslant k$ valores $x_2 \in S$. Daí, $|R_p \setminus R_{a_k}| \leqslant k|S|$ e

$$|R_p| \leqslant |R_{a_k}| + |R_p \setminus R_{a_k}| \leqslant (\partial a_k + k)|S| \leqslant \partial p \cdot |S|$$

Essa é uma boa estimativa, o polinômio $(x_1 - x_2)^2 - 1$ de grau total 2 no corpo $\mathbb{F}_3$ tem raízes $(0,1)$, $(1,0)$, $(2,1)$, $(1,2)$, $(0,2)$ e $(2,0)$, ou seja, $6 = 2 \cdot 3 = 2 \cdot |\mathbb{F}_3|$ raízes em $\mathbb{F}_3$.

O argumento que acabamos de descrever é indutivo, provamos o caso de duas variáveis usando o caso de uma variável. Podemos, sem muita dificuldade, generalizar o passo acima para provar o seguinte resultado.

Lema 2.20 (lema de Schwartz) *Sejam $\mathbb{F}$ um corpo, $S \subset \mathbb{F}$ finito, $n \geqslant 1$, e $p \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ um polinômio não nulo. Então, a quantidade de n-uplas $(r_1, \ldots, r_n) \in S^n$ tais que $p(r_1, \ldots, r_n) = 0$ é no máximo $\partial p \cdot |S|^{n-1}$.*

Demonstração. Por indução em $n \geqslant 1$. Pela dedução acima a base, $n = 1$, vale. Basta verificarmos o passo da indução. Suponhamos que para $n \geqslant 2$, todo $q \in \mathbb{F}[x_1, \ldots, x_{n-1}]$ tem no máximo $\partial q \cdot |S|^{n-2}$ raízes em S^{n-1}. Vamos mostrar que $p \in \mathbb{F}[x_1, \ldots, x_n]$ tem no máximo $\partial p \cdot |S|^{n-1}$ raízes em S^n.

Suponhamos, sem perda de generalidade, que $\partial_{x_n} p = k > 0$ e com isso podemos escrever

$$p(x_1, \ldots, x_{n-1}, x_n) = \sum_{j=0}^{k} g_j(x_1, \ldots, x_{n-1}) \cdot x_n^j.$$

Definimos os conjuntos

$$R_p := \{(x_1, \ldots, x_{n-1}, x_n) \in S^n : p(x_1, \ldots, x_{n-1}, x_n) = 0\}$$

e

$$R_{g_k} := \{(a_1, \ldots, a_{n-1}, x_n) \in S^n : g_k(a_1, \ldots, a_{n-1}) = 0\}.$$

Pela hipótese indutiva g_k tem no máximo $\partial g_k \cdot |S|^{n-2}$ raízes em S^{n-1}, portanto, $|R_{g_k}| \leqslant \partial g_k \cdot |S|^{n-1}$. Em $R_p \setminus R_{g_k}$ temos os pontos $(a_1, \ldots, a_{n-1}, x_n) \in R_p$ tais que $g_k(a_1, \ldots, a_{n-1}) \neq 0$, portanto $p(a_1, \ldots, a_{n-1}, x_n)$ é um polinômio em x_n de grau k, logo tem $\leqslant k$ raízes. Daí, $|R_p \setminus R_{g_k}| \leqslant k|S|^{n-1}$. Portanto,

$$|R_p| = |R_{a_k}| + |R_p \setminus R_{a_k}| \leqslant (\partial g_k + k)|S|^{n-1} \leqslant \partial p \cdot |S|^{n-1}$$

é a estimativa procurada para o número de raízes de p em S^n. □

O algoritmo para polinômios de várias variáveis é uma adaptação simples do algoritmo 2 e é como segue.

Instância : $d > 0$ e $f(x_1, \ldots, x_n)$ de grau total no máximo d.
Resposta : *não* se $f \neq 0$, caso contrário *sim* com probabilidade de erro no máximo 1/4.

1 **para cada** $i \in \{0, \ldots, n-1\}$ **faça** $x_i \leftarrow_{\mathcal{U}} \{1, 2, \ldots, 4d\}$;
2 **se** $f(x_1, x_2, \ldots, x_n) \neq 0$ **então responda** *não*.
3 **senão responda** *sim*.

Algoritmo 15: identidade entre polinômios.

A probabilidade de erro do algoritmo 15 segue do teorema de Schwartz–Zippel. Para S e f como acima, um algoritmo que escolhe aleatoriamente $x_1, \ldots, x_n$ em S e decide "$f = 0$?" baseado no teste $f(x_1, \ldots, x_n) = 0$ erra com probabilidade no máximo $d/|S| = 1/4$. Repetindo k vezes o algoritmo, se $f \neq 0$ então o algoritmo responde *sim* somente se nas k iterações (independentes) foi sorteada uma raiz de f cuja probabilidade é

$$\mathbb{P}[\,\text{erro}\,] \leqslant \left(\frac{1}{4}\right)^k.$$

Exercício 2.21. Qual é a probabilidade de resposta errada em k repetições (independentes) do algoritmo se as escolhas aleatórias são garantidas ser sem repetição.

2.2.4 RAÍZES PRIMITIVAS

Um grupo multiplicativo $(G, \cdot)$ é *cíclico* se possui um *gerador* g, isto é, para todo $h \in G$ existe um inteiro positivo l tal que $g^l = h$. Nesses casos, o expoente l é o *logaritmo discreto* de h em G na base g. Vários algoritmos importantes na criptografia de chave pública baseiam sua segurança na suposição de que o problema do logaritmo discreto (dado$h$, determinar l) com G e g cuidadosamente escolhidos não tem algoritmo eficiente (veja o exemplo 5.1, página 120). Em Criptografia é frequente o uso do grupo multiplicativo dos inteiros módulo um primo p como, por exemplo, no protocolo Diffie–Hellman–Merkle para troca pública de chaves criptográficas (Diffie e Hellman, 1976), onde precisamos determinar de modo eficiente um gerador desse grupo (veja o protocolo na página 121).

Um elemento que gera o grupo multiplicativo dos inteiros invertíveis módulo n, denotado $\mathbb{Z}_n^*$, é chamado de **raiz primitiva módulo** n. É um resultado conhecido que raízes primitivas existem se, e só se, $n = 1, 2, 4, p^k, 2p^k$ com p primo e $k \in \mathbb{N}$. O problema que estamos interessado aqui é o seguinte.

Problema computacional da raiz primitiva módulo p:

Instância : $p > 2$ primo.

Resposta : uma raiz primitiva módulo p.

Não se conhece algoritmo eficiente para determinar uma raiz primitiva módulo p a menos que seja dado a fatoração de $p-1$. Lembramos que, atualmente, não é conhecido algoritmo eficiente para o problema da fatoração (problema 5, página 120).

> *Problema* 2 (*Raiz primitiva módulo* p). Dado um primo p é possível determinar em tempo polinomial em $\log p$ uma raiz primitiva módulo p?

A seguir, $n \geqslant 2$ é inteiro, $\mathbb{Z}_n$ denota o conjunto das classes dos restos da divisão identificado com $\{0, 1, \ldots, n-1\}$ que munido da soma e do produto módulo n é um anel comutativo com unidade. Um elemento a do anel tem inverso multiplicativo se, e só se, $\text{mdc}(a, n) = 1$ e $\mathbb{Z}_n^*$ denota o conjunto $\{a \in \mathbb{Z}_n : \text{mdc}(a, n) = 1\}$ que munido do produto módulo n é um grupo comutativo, chamado grupo das unidades do anel $\mathbb{Z}_n$.

A **ordem (multiplicativa)** de $a \in \mathbb{Z}_n^*$, denotada $\mathrm{ord}_*(a)$, é o menor inteiro positivo k tal que $a^k \equiv 1 \pmod{n}$.

O algoritmo trivial para determinar a ordem de um elemento $a \in \mathbb{Z}_n^*$ calcula todas as potências $a^k \pmod{n}$, o que é inviável se $|\mathbb{Z}_n^*|$ é grande como, por exemplo, no caso das aplicações em criptografia. Mesmo $\varphi(n) := |\mathbb{Z}_n^*|$ é difícil de computar quando n não é primo ou potência de primo. A função $\varphi(n)$ é conhecida como função totiente de Euler, voltaremos a ela no final da seção.

Exercício 2.22. Prove que, se $a \in \mathbb{Z}_n^*$ e sua ordem é k, então

1. Teorema de Euler: $a^{\varphi(n)} \equiv 1 \pmod{n}$;
2. $a^m \equiv a^\ell \pmod{n}$ se e só se $m \equiv \ell \pmod{k}$. Em particular, se $a^m \equiv 1 \pmod{n}$ então k divide m;
3. $\mathrm{ord}_*(a^m) = k/\mathrm{mdc}(m,k)$. (Dica: verifique que, para quaisquer inteiros a e x diferentes de 0 vale que $ax \equiv 0 \pmod{n}$ se, e só se, $x \equiv 0 \pmod{n/\mathrm{mdc}(a,n)}$.)

O seguinte resultado deu origem ao algoritmo 17, probabilístico, para raiz primitiva módulo p.

Lema 2.23 *Dado $a \in \mathbb{Z}_n^*$, se m tem fatoração em primos $m = p_1^{m_1} p_2^{m_2} \cdots p_k^{m_k}$ é tal que $a^m \equiv 1 \pmod{n}$, então*

$$\mathrm{ord}_*(a) = \prod_{i=1}^{k} p_i^{m_i - f_i},$$

onde f_i é o maior inteiro não negativo tal que $a^{m/p_i^{f_i}} \equiv 1 \pmod{n}$, para todo $i \in \{1, 2, \ldots, k\}$.

Demonstração. Seja $o = \mathrm{ord}_*(a)$. Pelo exercício 2.22 temos que o divide m, portanto, $o = p_1^{r_1} p_2^{r_2} \cdots p_k^{r_k}$, com $0 \leqslant r_i \leqslant m_i$ para todo i. Para determinar r_1 tomemos a sequência

$$a^{m/p_1^{m_1}},\ a^{m/p_1^{m_1-1}},\ a^{m/p_1^{m_1-2}},\ \ldots,\ a^{m/p_1}, a^m$$

módulo n e seja f_1 o maior inteiro não negativo tal que $a^{m/p_1^{f_1}} \equiv 1 \pmod{n}$ (a primeira ocorrência de 1). Então temos

$$a^{m/p_1^{f_1}} \equiv 1 \pmod{n} \text{ e } a^{m/p_1^{f_1+1}} \not\equiv 1 \pmod{n} \text{ ou } f_1 = m_1.$$

De $a^{m/p_1^{f_1}} \equiv 1 \pmod{n}$ temos que o divide $m/p_1^{f_1}$, ou seja,

$$o \text{ divide } p_1^{m_1 - f_1} p_2^{m_2} \cdots p_k^{m_k}$$

mas de $a^{m/p_1^{f_1+1}} \not\equiv 1 \pmod{n}$ temos que o não divide $m/p_1^{f_1+1}$, ou seja

$$o \text{ não divide } p_1^{m_1 - f_1 - 1} p_2^{m_2} \cdots p_k^{m_k},$$

isso só pode ocorrer se $r_1 = m_1 - f_1$. Analogamente, $r_i = m_i - f_i$ para todo $i \in \{2, \ldots, k\}$. □

Corolário 2.24 *Se $a^m \equiv 1 \pmod n$ e $a^{m/p} \not\equiv 1 \pmod n$ para todo primo p divisor de m, então a tem ordem m.* □

Exercício 2.25. Prove que se p é primo e no $\mathbb{Z}_p^*$ a ordem de a_1 é n_1, a ordem de a_2 é n_2 e $\text{mdc}(n_1, n_2) = 1$ então a ordem de $a_1 a_2 \in \mathbb{Z}_p^*$ é $n_1 n_2$.

Exemplo 2.26. O $\mathbb{Z}_{11}^*$ é um grupo de ordem $10 = 2 \cdot 5$. Pelo corolário 2.24, se $a^{10} \equiv 1 \pmod{11}$ (que vale pelo Teorema de Euler) e $a^2 \not\equiv 1 \pmod{11}$ e $a^5 \not\equiv 1 \pmod{11}$, então a tem ordem 10, logo é um gerador.

a	1	2	3	4	5	6	7	8	9	10
$a^2 \bmod 11$	1	4	9	5	3	3	5	9	4	1
$a^5 \bmod 11$	1	10	1	1	1	10	10	10	1	10

Os geradores são $2, 6, 7, 8$. Se usarmos o fato de que, certamente, 1 e 10 não são geradores, então a probabilidade de escolher um gerador num sorteio em $\{2, \ldots, 9\}$ é $1/2$. Se escolhemos um elemento uniformemente e repetimos a escolha de modo independente, o número médio (ponderado pelas probabilidades) de rodadas até sair um gerador é $1\frac{1}{2} + 2\frac{1}{4} + 3\frac{1}{8} + \cdots = 2$. ◊

Proposição 2.27 *Para $p > 2$ primo, o $\mathbb{Z}_p^*$ admite $\varphi(p-1)$ geradores.*

Demonstração. Se $a \in \mathbb{Z}_p^*$ é um gerador, $\mathbb{Z}_p^* = \{a \bmod p, a^2 \bmod p, \ldots, a^{p-1} \bmod p\}$ e a ordem de $a^k \bmod p$ é, pelo exercício 2.22, $(p-1)/\text{mdc}(k, p-1)$ logo a^k é gerador se, e só se, $\text{mdc}(k, p-1) = 1$. Com isso, concluímos que são $\varphi(p-1)$ geradores de $\mathbb{Z}_p^*$ □

Portanto, $\varphi(p-1)/(p-1)$ é a probabilidade de sortear um gerador no $\mathbb{Z}_p^*$. Podemos usar os limitantes conhecidos para a função φ (veja Bach e Shallit, 1996, teorema 8.8.7) para ter, por exemplo,

$$p = \frac{\varphi(n)}{n} > \frac{1}{6 \log\log(n)} \tag{2.8}$$

para todo $n > 10$ (veja o exercício 2.67 no final do capítulo para um limitante). O número médio (como na equação (2.3)) de sorteios até sair um gerador é $\sum_{k \geqslant 1} k(1-p)^{k-1} p < 6 \log\log(n)$.

O resultado a seguir mostra que uma escolha uniforme em $\mathbb{Z}_p^*$ implica numa escolha uniforme nos elemento do $\mathbb{Z}_p^*$ da forma $x^{(p-1)/p_i}$ para p_i primo que divide $p-1$.

Proposição 2.28 *Sejam p um primo e p_i um fator primo de $p-1$. Se $f : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ é o homomorfismo de grupos definido por $f(x) = x^{(p-1)/p_i} \bmod p$, então $|f^{-1}(a)| = |f^{-1}(b)|$ para quaisquer $a \neq b \in \text{Im}(f)$; ademais $|\text{Im}(f)| = p_i$.*

Demonstração. Sejam r uma raiz primitiva e a e b elementos distintos do $\mathbb{Z}_p^*$. Sejam $k, \ell \in \{1, 2, \ldots, p-1\}$ distintos tais que $b = r^k$ e $a = r^\ell$. Pelo exercício 2.22 temos $f(b) = f(a)$ se e só se $k\frac{p-1}{p_i} \equiv \ell\frac{p-1}{p_i} \pmod{p-1}$. De $1 \leqslant k, \ell \leqslant p-1$ distintos temos $k \not\equiv \ell \pmod{p-1}$. Portanto $k \equiv \ell \pmod{p_i}$.

Agora, fixado k a quantidade de inteiros ℓ com $k \equiv \ell \pmod{p_i}$ é $(p-1)/p_i$, logo, a pré-imagem $f^{-1}(b)$, para qualquer b, tem cardinalidade $(p-1)/p_i$. Finalmente $|\mathrm{Im}(f)| = (p-1)/|f^{-1}(b)| = p_i$. □

Corolário 2.29 *Com as hipóteses acima*

$$\mathop{\mathbb{P}}_{x\in\mathbb{Z}_p^*}\left[x^{\frac{p-1}{p_i}} \equiv 1 \pmod{p}\right] = \frac{1}{p_i}$$

para todo p_i fator primo de $p-1$. □

O algoritmo 17 abaixo já era conhecido de Gauss que, como exemplo, determinou um gerador de $\mathbb{Z}_{73}^*$ no seu famoso trabalho *Disquisitiones Arithmeticæ*.

Instância : um primo p e a fatoração $p-1 = p_1^{m_1} p_2^{m_2} \cdots p_k^{m_k}$.
Resposta : raiz primitiva módulo p.

1 **para** i *de* 1 *até* k **faça**
2 **repita**
3 $a \leftarrow_{\mathcal{U}} \{2, 3, \ldots, p-1\}$;
4 $b \leftarrow a^{(p-1)/p_i} \bmod p$;
5 **até que** $b \neq 1$;
6 $q_i \leftarrow a^{(p-1)/p_i^{m_i}} \bmod p$;
7 **responda** $\prod_{i=1}^{k} q_i \bmod p$.

Algoritmo 17: raiz primitiva.

A prova de que o algoritmo 17 está correto segue da observação de que no final da i-ésima iteração do **para** na linha 1 vale que

$$q_i^{p_i^{m_i}} \equiv 1 \pmod{p} \text{ e } q_i^{p_i^{m_i-1}} \not\equiv 1 \pmod{p}$$

e nesse caso, pelo corolário 2.24 a ordem de q_i no $\mathbb{Z}_p^*$ é $p_i^{m_i}$. Usando uma generalização natural do exercício 2.25 para um produto com mais que 2 fatores, temos que a ordem de $q_1 q_2 \cdots q_k$ em $\mathbb{Z}_p^*$ é $p-1$, portanto, é um gerador do grupo.

O tempo de execução do algoritmo depende do número de rodadas do **repita** na linha 2 que por sua vez depende dos sorteios. Para cada i no laço da linha 1 a probabilidade de sair do laço na linha 2 é $1-(1/p_i)$ pelo corolário acima, portanto como em (2.3), o número médio de rodadas é $1/(1-(1/p_i)) = p_i/(p_i-1) \leqslant 2$. As exponenciações na linha 4 podem ser feita em tempo $O(\log^3 p)$ (algoritmo 8, página 62). Assim, o laço da linha 2 tem custo médio $O(\log^3 p)$. As exponenciações na linha 6 também são feitas em tempo $O(\log^3 p)$. Assim, o laço da linha 1 tem custo final $O(k \cdot \log^3 p)$ em média. Os $k-1$ produtos módulo p na linha 7 envolvem números da ordem de $\log p$ dígitos, o que custa $O(k \cdot \log^2 p)$. Portanto, em média, o tempo de execução do algoritmo é $O(k \cdot \log^3 p)$.

O parâmetro k é a quantidade de divisores primos distintos de $p-1$, usualmente denotada nos livros de teoria dos números por $\omega(p-1)$, cuja ordem de grandeza é (veja Bach e Shallit, 1996, teorema 8.8.10)

$$\omega(n) = O(\log n/\log\log n). \tag{2.9}$$

Dito isso, podemos concluir o seguinte resultado.

Teorema 2.30 *O tempo médio de execução para o algoritmo 17 é* $O(\log^4 p/\log\log p)$. □

A função φ de Euler. A função φ de Euler associa a cada inteiro positivo n a quantidade de inteiros positivos menores que n que são coprimos com n

$$\varphi(n) := |\{a \in \mathbb{N} : \mathrm{mdc}(a,n) = 1 \text{ e } 1 \leqslant a \leqslant n\}| = |\mathbb{Z}_n^*|.$$

Decorre da definição que se p é primo então $\varphi(p) = p-1$. Para potências de primo temos que dentre $1, 2, \ldots, p^k$ não são coprimos com p^k aquele que têm p como fator primo, a saber $p, 2p, 3p, \ldots, p^{k-1}p$, portanto $p^k - p^{k-1}$ são coprimos, isto é,

$$\varphi(p^k) = p^k\left(1 - \frac{1}{p}\right).$$

Para determinarmos o valor da função de Euler nos naturais que não são potência de primo o seguinte é fundamental: a função φ é multiplicativa.

Teorema 2.31 *Se* $n, m \in \mathbb{Z}^+$ *são coprimos então* $\varphi(nm) = \varphi(n)\varphi(m)$.

Demonstração. O caso $m = 1$ ou $n = 1$ é imediato. Sejam $n, m > 1$ inteiros. Para todo inteiro a vale (verifique)

$$\mathrm{mdc}(a, nm) = 1 \Leftrightarrow \mathrm{mdc}(a,n) = 1 \text{ e } \mathrm{mdc}(a,m) = 1$$

Desse modo, $\varphi(nm)$ é a quantidade de naturais entre 1 e nm que são coprimos com n e com m concomitantemente. Em

$$\begin{array}{ccccccc}
1 & 2 & \ldots & i & \ldots & m \\
m+1 & m+2 & \ldots & m+i & \ldots & 2m \\
2m+1 & 2m+2 & \ldots & 2m+i & \ldots & 3m \\
\vdots & \vdots & \ldots & \vdots & \ldots & \vdots \\
(n-1)m+1 & (n-1)m+2 & \ldots & (n-1)m+i & \ldots & nm
\end{array}$$

há $\varphi(m)\varphi(n)$ coprimos com n e m pois: há na tabela acima $\varphi(m)$ colunas que começam com um inteiro coprimo com m. Se um divisor de m divide i, então divide todos os números na coluna i. Portanto, os coprimos com m em $\{1, \ldots, nm\}$ aparecem nas $\varphi(m)$ colunas dos coprimos com m.

Os inteiros $\{1,\ldots,nm\}$ com m e n também aparecem nas $\varphi(m)$ colunas dos coprimos com m. Porém, cada uma dessas colunas é um sistema completo de restos módulo n, portanto, há em cada $\varphi(n)$ coprimos com n. Assim, há $\varphi(m)\varphi(n)$ coprimos com n e m. □

Exercício 2.32. Use indução para mostrar que se $\mathrm{mdc}(n_i, n_j) = 1$ para todo $i \neq j$ então $\varphi(n_1 n_2 \cdots n_k) = \varphi(n_1)\varphi(n_2)\cdots\varphi(n_k)$.

Corolário 2.33 *Se $n = p_1^{m_1} \cdots p_k^{m_k}$ é a fatoração canônica de n então*

$$\varphi(n) = n \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$$

para todo inteiro $n > 1$. □

Exercício 2.34. Mostre que $\varphi(n) = n - 1$ se e só se n primo.

Então, concluímos que é fácil determinar $|\mathbb{Z}_n^*|$ se temos a fatoração em primos de n. Por outro lado, tomando, por exemplo, o caso $n = pq$ temos $\varphi(n) = (p-1)(q-1)$ logo, de n e $\varphi(n)$ temos $p + q = n + 1 - \varphi(n)$. Também, $(p-q)^2 = (p+q)^2 - 4n$. De $p+q$ e $p-q$ é fácil determinar p e q. Esse fato parece indicar que o problema de determinar $\varphi(n)$ é tão difícil computacionalmente quanto fatoração de n.

> *Problema* 3 (*Problema da função φ de Euler*). Dado um inteiro $n > 1$, é possível determinar $|\mathbb{Z}_n^*|$ em tempo polinomial em $\log n$?

2.2.5 POLINÔMIOS IRREDUTÍVEIS E ARITMÉTICA EM CORPOS FINITOS

Os corpos finitos têm um papel importante em Computação. Veremos adiante neste texto várias construções que usam tal estrutura algébrica para, por exemplo, desaleatorizar algoritmos; também são úteis na Teoria dos Códigos, em Complexidade Computacional e na Criptografia. Nos algoritmos sobre esses corpos precisamos de uma descrição explícita deles, assim como das operações aritméticas do corpo. De fato, assumimos que a aritmética é dada e a medida de custo de tempo de um algoritmo é em função do número de operações no corpo. Veremos abaixo como fazer isso e mais detalhes do que mostramos aqui são encontrados em Gathen e Gerhard (2013) e Lidl e Niederreiter (1997).

Para todo primo p existe um corpo com p^d elementos, para todo $d \in \mathbb{N}$, que é único a menos de isomorfismos e é denotado por $\mathbb{F}_{p^d}$. Reciprocamente, todo corpo finito tem p^d elementos para algum p primo e algum d natural. No caso $d = 1$ temos os corpos mais simples dados por $\mathbb{Z}_p$ com p primo e as operações módulo p, mas em geral $\mathbb{F}_{p^d} \neq \mathbb{Z}_{p^d}$ se $d > 1$.

Polinômios. Denotamos por $\mathbb{Z}_n[x]$, para $n \geqslant 2$, o conjunto dos polinômios

$$a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$$

com indeterminada x e com coeficientes $a_0, a_1, \ldots$ em $\mathbb{Z}_n$ que com a soma e o produto usual de polinômios é um anel comutativo com unidade. O coeficiente a_0 de x^0 é o **termo constante** e o coeficiente a_m da maior potência de x é o **coeficiente principal** do polinômio. O grau de um polinômio não nulo f, denotado ∂f, é o maior expoente de x com coeficiente *não nulo*, polinômio de grau 0 é chamado de **polinômio constante** e o grau do polinômio nulo não está definido. Um polinômio é **mônico** se o coeficiente principal é 1.

Exercício 2.35 (divisão com resto para polinômios). Sejam $n \geqslant 2$ inteiro e $h, f \in \mathbb{Z}_n[x]$ polinômios com h mônico. Prove que existem únicos $q, r \in \mathbb{Z}_n[x]$ com $f = hq + r$ e $\partial r < \partial h$. Verifique que tal divisão pode ser realizada com $O(\partial f \cdot \partial h)$ operações do $\mathbb{Z}_n$.

No caso $r = 0$ dizemos que h **divide** f. Agora, se $f = h \cdot q$ com $0 < \partial h < \partial f$ então h é um **divisor próprio** de f.

Se $h \in \mathbb{Z}_n[x]$ é mônico então dizemos que $f, g \in \mathbb{Z}_n[x]$ são **congruentes módulo** $h(x)$, denotado $f \equiv g \pmod{h}$, se existe $q \in \mathbb{Z}_n[x]$ tal que $qh = f - g$, ou seja, $f - g$ é divisível por h. Então $\mathbb{Z}_n[x]/(h)$ denota o conjunto de todos os polinômios em $\mathbb{Z}_n[x]$ de grau menor que ∂h que, munido das operações $+_h$ e $\cdot_h$ definidas por

$$f +_h g \coloneqq (f + g) \bmod h$$
$$f \cdot_h g \coloneqq (f \cdot g) \bmod h$$

é um anel com unidade. Nos algoritmos podemos considerar os elementos de $\mathbb{Z}_n[x]/(h)$ como vetores de comprimento $d = \partial h$. Somar dois deles pode ser feito facilmente usando $O(d)$ adições em $\mathbb{Z}_n$. A multiplicação e a divisão podem ser feitas do modo usual com $O(d^2)$ multiplicações e adições no anel dos coeficientes. Finalmente, calculamos $fg \bmod h$ pelo algoritmo de divisão polinomial, resultando no final um tempo de execução de $O(d^2)$ e o mdc de dois polinômios de grau no máximo d pode ser computado em $O(d^2 \log d)$. De fato, é possível fazer melhor, como na observação 2.12; multiplicação em tempo $M(d) = O(d \log d \log\log d)$ usando a técnica de Schönhage–Strassen (transformada rápida de Fourier), o mdc em tempo $O(\log(d)M(d))$ e $f^k \bmod h$ em tempo $O(\log(k)M(d))$.

Polinômios irredutíveis. Um polinômio $f \in \mathbb{F}[x]$ com grau positivo é dito **irredutível** em $\mathbb{F}[x]$ se em toda fatoração $f = h \cdot q$ ou h ou q é polinômio constante. No caso de polinômios com coeficientes sobre um corpo $\mathbb{F}$ o resultado do exercício acima, a divisão com resto para polinômios, vale para qualquer $h \in \mathbb{F}[x]$ não nulo. Em $\mathbb{F}[x]$ vale ainda a **fatoração única**: *todo polinômio não nulo $f \in \mathbb{F}[x]$ pode ser escrito de forma única a menos da ordem dos fatores como um produto $a \cdot h_1 \cdot h_2 \cdots h_s$ com $a \in \mathbb{F}$ e*

$h_i \in \mathbb{F}[x]$ *mônicos, irredutíveis e de grau positivo.* Notemos que $\mathbb{F}[x]$ não é corpo pois, por exemplo, o polinômio x não tem inverso multiplicativo.

Uma aplicação importante dos polinômios irredutíveis é a construção explícita de corpos finitos. Essas construções são baseadas no seguinte resultado. Quando $h \in \mathbb{F}_p[x]$, para p primo, é um polinômio mônico, irredutível e com grau $\partial h = d$, o conjunto $\{g \in \mathbb{F}_p[x] \colon \partial g < d\}$ munido da adição e multiplicação módulo h é um corpo com p^d elementos, portanto (isomorfo a) o $\mathbb{F}_{p^d}$. Todos os corpos finitos são obtidos desse modo e não há uma escolha canônica para h, escolhas diferentes resultam em corpos diferentes, porém isomorfos.

> *Problema* 4 (*Problema de computar um polinômio irredutível*). Existe algoritmo determinístico eficiente para computar um polinômio irredutível $h \in \mathbb{F}[x]$ de grau d?

Corpos binários. Um caso particularmente interessante em Computação são os **corpos binários**, corpos com 2^n elementos para $n \in \mathbb{N}$. Os elementos de $\mathbb{F}_{2^n}$ podem ser identificados com as sequências binárias $\{0,1\}^n$ da seguinte forma. Para $n = 1$, temos $\mathbb{F}_2 = \mathbb{Z}_2$, ou seja, o conjunto $\{0,1\}$ com as operações binárias usuais de soma ("ou" exclusivo lógico) e multiplicação módulo 2 ("e" lógico). Como já sabemos, $\mathbb{F}_2[x]$ denota o anel dos polinômios com coeficientes em $\mathbb{F}_2$ com soma e multiplicação usuais de polinômios e as operações nos coeficientes são módulo 2. O quociente $\mathbb{F}_2[x]/(h)$ é um corpo com 2^n elementos para qualquer $h \in \mathbb{F}_2[x]$ irredutível em $\mathbb{F}_2[x]$ de grau n. Os elementos desse corpo são dados por todos os polinômios $p \in \mathsf{F}_2[x]$ de grau no máximo n os quais são representados pelas sequências $(b_0, b_1, \ldots, b_{n-1}) \in \{0,1\}^n$ de seus coeficientes de maneira natural, $p = b_0 x^{n-1} + \cdots + b_{n-2}x + b_{n-1}$.

Por exemplo, o corpo com 4 elementos $\mathbb{F}_{2^2}$ é dado pelos polinômios $\mathbb{Z}_2[x]/(x^2+x+1) = \{0, 1, x, x+1\}$ e pode ser representado por $\{(0,0),(0,1),(1,0),(1,1)\}$. A representação polinomial de $\mathbb{F}_{2^3}$ com $h(x) = x^3 + x + 1$ é dada pelos polinômios $\{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$. O corpo com dezesseis elemento é dado por $\mathbb{Z}_2[x]/(x^4+x+1) = \{0, 1, x, x^2, x^3, x+1, x^2+1, x^3+1, x^2+x, x^3+x, x^3+x^2, x^3+x^2+x, x^2+x+1, x^3+x+1, x^3+x^2+1, x^3+x^2+x+1\}$ ou por $\mathbb{Z}_2[x]/(x^4+x^3+x^2+x+1)$.

A soma nesses conjuntos é feita como o usual para polinômios, no caso das sequências binárias é coordenada a coordenada. Por exemplo, em $\mathbb{F}_{2^3}$ temos $(x^2+1)+(x^2+x+1) = (1+1)x^2+x+(1+1) = x$ e o produto é o produto usual tomado módulo h, por exemplo temos $(x^2+x)(x^2+x+1) = x^4+x$ que módulo $h(x)$ é x^2.

Corpos finitos. De modo geral, um elemento do corpo finito $\mathbb{F}_{p^n}$, para $n > 1$, é dado por um vetor com n elementos de $\mathbb{F}_p$ e eficiência numa operação significa tempo polinomial em $n \log p$, assumindo tempo constante para as operações em $\mathbb{F}_p$. Assim, se h é dado, a aritmética em $\mathbb{F}_{p^n}$ é fácil. Uma caracterização de polinômios irredutíveis é dada no execício 2.39, ela é consequência do seguinte

resultado sobre polinômios irredutíveis que enunciaremos sem prova (Lidl e Niederreiter, 1997, teorema 3.20).

Teorema 2.36 *Seja P_k o produto de todos os polinômios mônicos, irredutíveis no corpo $\mathbb{Z}_p[x]$ de grau k. Então o produto de todos os P_k para k que divide n é*

$$\prod_{k|n} P_k = x^{p^n} - x$$

para todo n.

Como vimos, polinômios irredutíveis em $\mathbb{F}_p[x]$ são usados para transportar a aritmética do corpo $\mathbb{F}_p$ para extensões do $\mathbb{F}_p$, o corpo $\mathbb{F}_{p^n}$ é isomorfo a $\mathbb{F}_p[x]/(h)$ qualquer que seja o polinômio $h \in \mathbb{F}_p[x]$ irredutível e de grau n. Esse isomorfismo permite a construção das tabelas aritméticas a partir das operações em polinômios. O problema que precisamos resolver é encontrar tal h. A ideia para se obter polinômios irredutíveis é a mais óbvia possível sortear o polinômio e testar a irredutibilidade.

Instância : inteiro positivo n.
Resposta : polinômio mônico irredutível em $\mathbb{Z}_p[x]$ de grau n.

1 **repita**
2 | **para** i *de* 1 *até* n **faça** $a_i \leftarrow_{\mathcal{U}} \{0, 1, \ldots, n-1\}$
3 **até que** $x^n + \sum_{i=0}^{n-1} a_i x^i$ *seja irredutível.*

Algoritmo 18: gerador de polinômios irredutíveis.

Para estimar a probabilidade de achar polinômios mônicos irredutíveis vamos usar o seguinte resultado (veja Lidl e Niederreiter, 1997, exercício 3.27) que pode ser deduzido do teorema 2.36 acima usando a formula de inversão de Möbius

$$f(n) = \sum_{k|n} g(k) \Leftrightarrow g(n) = \sum_{k|n} \mu(k) f\left(\frac{n}{k}\right)$$

em que $\mu(1) = 1$, $\mu(n) = 0$ se n é divisível por um quadrado maior que 1, senão $\mu(n) = (-1)^\omega$ onde $\omega = \omega(n)$ é a quantidade de primos distintos que dividem n.

Do teorema, $x^{p^n} - x = \prod_{k|n} P_k$, portanto, tomando os graus

$$p^n = \sum_{k|n} \partial P_k \Leftrightarrow \partial P_n = \sum_{k|n} \mu(k) p^{\frac{n}{k}}$$

e se $I(n)$ é o número de polinômios mônicos irredutíveis de grau n no $\mathbb{Z}_p[x]$, então $\partial P_n = nI(n)$, logo

$$I(n) = \frac{1}{n} \sum_{k|n} \mu(k) p^{\frac{n}{k}}.$$

Também, $\partial P_n \leqslant \sum_{k|n} \partial P_n = p^n$ donde tiramos

$$I(n) \leqslant \frac{p^n}{n}.$$

Por outro lado,

$$p^n - \partial P_n = \sum_{\substack{k|n \\ k<n}} \partial P_k \leqslant \sum_{\substack{k|n \\ k<n}} p^k \leqslant \sum_{k=1}^{n/2} p^k = \frac{p^{n/2+1}-p}{p-1} \leqslant \frac{p}{p-1} p^{n/2} \leqslant 2p^{n/2}$$

portanto, $\partial P_n \geqslant p^n - 2\sqrt{p^n}$.

Lema 2.37 *Se p é primo então*

$$\frac{p^n}{n} \geqslant I(n) \geqslant (p^n - 2\sqrt{p^n})/n$$

é uma estimativa para a quantidade de polinômios mônicos irredutíveis de grau n no $\mathbb{Z}_p[x]$. □

O tempo de execução do algoritmo 18 depende dos sorteios e do custo do teste de irredutibilidade. Este último deixamos pra depois, por ora temos, usando o lema 2.37 acima, que

$$\frac{1}{n} \geqslant \frac{I(n)}{p^n} \geqslant \frac{1}{n}\left(1 - \frac{2}{\sqrt{p^n}}\right) > \frac{1}{2n}$$

para $p^n > 16$. Da equação (2.3) com a probabilidade acima, o número médio de rodadas do **repita** é menor que $2n$, portanto o custo médio do laço é $O(nT(n))$ em que T é o tempo para testar irredutibilidade de um polinômio de grau n.

Teste de Rabin para irredutibilidade de polinômios. O teste de irredutibilidade é baseado no seguinte resultado.

Teorema 2.38 (Rabin, 1980b) *Sejam n um natural e p um primo. Um polinômio $h \in \mathbb{Z}_p[x]$ de grau n é irredutível em $\mathbb{Z}_p$ se, e só se,*

$$h(x) \Big| \left(x^{p^n} - x\right) \tag{2.10}$$

$$\operatorname{mdc}\left(h(x), x^{p^{n/q}} - x\right) = 1 \text{ para todo fator primo } q \text{ de } n. \tag{2.11}$$

Demonstração. Se h é irredutível então (2.10) e (2.11) seguem do teorema 2.36. Para a recíproca, novamente pelo teorema 2.36, temos que se f é um fator irredutível de h com grau d então d divide n por (2.10). Suponhamos que $d < n$, então $d|(n/q)$ para algum fator primo q de n. Mas se esse é o caso, então f divide $x^{p^{n/q}} - x$, contradizendo (2.11), portanto $d = n$. □

Instância : $h \in \mathbb{Z}_p[x]$ mônico de grau n e os fatores primos $q_1, q_2, \ldots, q_k$ de n.
Resposta : *sim*, h é irredutível ou *não*, h é redutível.

1 **para** i *de* 1 *até* k **faça**
2 **se** $\operatorname{mdc}(h, x^{p^{n/q_i}} - x \pmod h) \neq 1$ **então responda** *não*.
3 **se** $x^{p^n} - x \pmod h = 0$ **então responda** *sim*.
4 **senão responda** *não*.

Algoritmo 19: teste de irredutibilidade de Rabin.

O $\mathrm{mdc}(h, x^{p^{n/q_i}} - x \pmod h)$ é computado calculando-se a potência $x^{p^n} \pmod h$, o que pode ser feito com $O(n \log p)$ operações de custo $O(n^2)$ realizadas $k = O(\log n)$ vezes, portanto, $O(n^3 \log(n) \log(p))$ operações do corpo. De fato, é possível melhorar essa estimativa, o tempo de execução desse algoritmo é $O(n\mathsf{M}(n) \log\log(n) \log(p))$ (Gao e Panario, 1997). Ademais, Gao e Panario (1997) propuseram uma melhoria no algoritmo de Rabin que resulta em tempo de execução $O(n\mathsf{M}(n) \log p)$.

Teste de Ben-Or para irredutibilidade de polinômios. O teste de irredutibilidade é baseado no seguinte resultado que segue do teorema 2.36.

Exercício 2.39. Um polinômio mônico h de grau n é irredutível se, e somente se, para cada $i = 1, 2, \ldots, \lfloor n/2 \rfloor$ vale que $\mathrm{mdc}\left(x^{p^i} - x \pmod h, h\right) = 1$.

Instância : $h \in \mathbb{Z}_p[x]$ mônico de grau n.
Resposta : *sim*, h é irredutível ou *não*, h é redutível.

1 **para** i *de* 1 *até* $n/2$ **faça**
2 | **se** $\mathrm{mdc}(h, x^{p^i} - x \pmod h) \neq 1$ **então responda** *não*.
3 **responda** *sim*.

Algoritmo 20: teste de irredutibilidade de Ben-Or.

Esse algoritmo, proposto em Ben-Or (1981), no pior caso computa $n/2$ vezes uma exponenciação modular com potência p e um mdc com polinômios de grau no máximo n, resultando no tempo de execução $O(n^3 \log(pn))$; refinando a análise chegamos a $O(n\mathsf{M}(n) log(pn))$. Embora o resultado assintótico seja pior do que o caso anterior, o algoritmo de Rabin, o algoritmo de Ben-Or é na prática muito eficiente, o seu desempenho bate o de Rabin porque polinômios aleatórios têm fatores irredutíveis de grau baixo o que é detetado muito rapidamente pelo algoritmo de Ben-Or (Gao e Panario, 1997).

2.3 TESTES DE PRIMALIDADE ALEATORIZADOS

Recordemos que um natural $n > 1$ é **primo** se os únicos divisores positivos de n são 1 e n. Um natural $n > 1$ é **composto** se admite divisores positivos além de 1 e n. O problema de decidir se um dado inteiro $n \geqslant 2$ é primo é muito antigo, o crivo de Eratóstenes (200 aC) é um dos primeiros algoritmos para teste de primalidade. Esse algoritmo lista os inteiros de 2 até n e, sucessivamente, toma o menor elemento da lista e apaga todos os seus múltiplo; isso é repetido enquanto a lista não está vazia ou o menor elemento é no máximo $\sqrt{n}$. No final se n está na lista é primo, senão é composto.

Problema computacional do teste da primalidade:
Instância : dado $n \geqslant 2$ inteiro ímpar.
Resposta : *sim* se n é primo, *não* caso contrário.

A busca por soluções eficientes parece ter sido proposta pela primeira vez por Gödel em 1956, uma solução para esse problema é eficiente se a decisão é tomada tempo polinomial em $\log n$. Do ponto de vista computacional o crivo de Eratóstenes não é eficiente, tem custo exponencial em $\log n$. Em 2002 foi anunciado que decidir se um número é primo pode ser resolvido em tempo polinomial, os indianos Agrawal, Kayal e Saxena (2004) mostraram um algoritmo que determina se n é primo em tempo $O(\log^{12+\varepsilon} n)$, para qualquer $\varepsilon > 0$. Em seguida esse algoritmo foi melhorado, o expoente no logaritmo diminuiu para $6 + \varepsilon$ (Lenstra, 2002) mas ainda está longe de ser mais viável que os algoritmos probabilísticos.

Os algoritmos probabilísticos são extremamente simples e eficientes, como é o caso do teste de Miller–Rabin que com k rodadas independentes tem custo $O(k \log^2(n) \log\log(n) \log\log\log(n))$ (Monier, 1980). A probabilidade de erro é exponencialmente pequena em k. Para $k = 100$ a probabilidade do algoritmo responder *primo* todas as vezes quando a entrada é um número composto é menor que 10^{-30}, que é um número muito pequeno (aproximadamente a massa do elétron).

Exemplo 2.40. O seguinte texto do manual do Maxima, um sistema de computação algébrica sob a licença pública GNU é um exemplo típico de como a primalidade de um número é testada na prática.

> "`primep (n)` Teste de primalidade. Se primep(n) retorna false, n é um número composto, e se ele retorna true, n é um número primo com grande probabilidade.
>
> Para n menor que 10^16 uma versão deterministica do teste de Miller-Rabin é usada. Se primep(n) retorna true, então n é um número primo.
>
> Para n maior do que 10^16 primep realiza primep_number_of_tests testes de pseudo-primalidade de Miller-Rabin e um teste de pseudo-primalidade de Lucas. A probabilidade com que n passe por um teste de Miller-Rabin é inferior a 1/4. Usando o valor padrão 25 para primep_number_of_tests, a probabilidade de n ser composto é muito menor do que 10^-15."

O *Mathematica* também implementa seu teste de primalidade usando algoritmos probabilísticos.

> "`PrimeQ` primeiro testa divisibilidade por primos pequenos, em seguida usa o teste de Miller-Rabin para bases 2 e 3, e em seguida usa o teste de Lucas."

No Python, a biblioteca SymPy usada para computação simbólica implementa um teste de primalidade; a documentação descreve

"`sympy.ntheory.primetest.isprime(n)` Test if n is a prime number (True) or not (False). For n < 10ˆ16 the answer is accurate; greater n values have a small probability of actually being pseudoprimes.

Negative primes (e.g. -2) are not considered prime.

The function first looks for trivial factors, and if none is found, performs a safe Miller-Rabin strong pseudoprime test with bases that are known to prove a number prime. Finally, a general Miller-Rabin test is done with the first k bases which, which will report a pseudoprime as a prime with an error of about 4ˆ-k. The current value of k is 46 so the error is about 2 x 10ˆ-28."

Notemos que em todos os casos as probabilidades de erro são muito pequenas. ◊

2.3.1 OS TESTES DE FERMAT E LUCAS

Um resultado célebre da teoria dos números, o Pequeno Teorema de Fermat enunciado a seguir, garante que para inteiros n e a, *se* $a^{n-1} \not\equiv 1 \pmod{n}$ *então* n *é composto.* O seguinte teorema decorre do Teorema de Euler pois, de acordo com o exercício 2.34, temos $\varphi(p) = p-1$ para todo p primo.

Teorema 2.41 (Pequeno Teorema de Fermat) *Se* $a \in \mathbb{Z}\setminus\{0\}$ *e* p *é primo que não divide* a*, então* $a^{p-1} \equiv 1 \pmod{p}$. □

Esse teorema nos dá o teste de primalidade

$$p \text{ é primo se, e somente se, } a^{p-1} \equiv 1 \pmod{p} \text{ para todo } 1 \leqslant a \leqslant p-1.$$

Dado um n, para qual queremos testar primalidade, se existe um inteiro $a \in \{1,2,\ldots,n-1\}$ tal que $a^{n-1} \not\equiv 1 \pmod{n}$, então esse teorema nos garante que n é composto. Chamamos tal a de uma **testemunha de Fermat** para o fato de n ser composto.

A ideia do teste de Fermat é, para dado n, sortear a e testar se a é testemunha para n, se for testemunha então n é composto, senão possivelmente um primo.

Instância : um inteiro ímpar $n \geqslant 4$.
Resposta : *sim* se n é primo, *não* caso contrário.

1 $a \leftarrow_{\mathcal{U}} \{2,3,\ldots,n-2\}$;
2 **se** $a^{n-1} \not\equiv 1 \pmod{n}$ **então responda** não.
3 **senão responda** sim.

Algoritmo 22: teste de Fermat.

Usando o algoritmo de exponenciação modular (algoritmo 8, página 62), o tempo de execução do teste de Fermat é $O(\log^3 n)$, logo de tempo polinomial em $\log_2 n$. O algoritmo só erra ao declarar que n é primo, a resposta *não* está sempre correta. Vamos estimar a probabilidade de erro.

Uma testemunha fixa não serve para todo número composto. Por exemplo, é possível verificar que 2 é testemunha para todo número natural composto até 340, mas que não é testemunha para $341 = 11 \cdot 31$ pois $2^{340} \equiv 1 \pmod{341}$.

Dizemos que um natural n é um **pseudoprimo de Fermat para a base** a quando vale que

$$n \text{ é ímpar, composto e } a^{n-1} \equiv 1 \pmod{n}.$$

A base $a \in \{1, 2, \ldots, n-1\}$ para o qual n é pseudoprimo é uma testemunha **mentirosa** para n.

Podemos descobrir que 341 é composto testando-o contra outras bases e nesse caso $3^{340} \equiv 54 \pmod{341}$ o que atesta que 341 é composto. É possível estender essa estratégia e testar se n contra toda base $a \in \{2, 3, \ldots, n-2\}$, porém isso não resulta em um algoritmo viável, o tempo de execução seria exponencial no tamanho da representação de n. A proposição abaixo garante que se incrementamos a base a e fazemos o teste "$a^{n-1} \equiv 1 \pmod{n}$?", então o mais longe que iremos é até o menor divisor primo de n, mas isso pode não ser muito melhor do que usar crivo de Eratóstenes.

Proposição 2.42 *Todo n ímpar e composto é pseudoprimo para as bases 1 e $n-1$. Ademais, não há $n \geqslant 4$ pseudoprimo para toda base $a \in \{1, 2, \ldots, n-1\}$.*

Demonstração. A primeira afirmação, que todo n ímpar e composto é pseudoprimo para as bases 1 e $n-1$ segue trivialmente de que valem $1^{n-1} \equiv 1 \pmod{n}$ e

$$(n-1)^{n-1} = \sum_{i=0}^{n-1} \binom{n-1}{i} n^i (-1)^{n-i-1} \equiv (-1)^{n-1} \pmod{n}$$

logo $(n-1)^{n-1} \equiv 1 \pmod{n}$ pois n é ímpar.

Se n é impar, composto e $a^{n-1} \equiv 1 \pmod{n}$, então $a^{(n-1)/2} a^{(n-1)/2} \equiv 1 \pmod{n}$, ou seja, $a^{(n-1)/2}$ tem inverso multiplicativo módulo n, portanto $\operatorname{mdc}(a^{(n-1)/2}, n) = 1$ donde deduzimos que $\operatorname{mdc}(a, n) = 1$. Então, para n ser pseudoprimo para todo $a \in \{2, \ldots, n-2\}$ ele deve ser primo, uma contradição. □

Como argumentamos na demonstração acima, se $a^k \equiv 1 \pmod{n}$ para algum $k > 1$, então $aa^{k-1} \equiv 1 \pmod{n}$, ou seja, a tem inverso multiplicativo módulo n, portanto, $a \in \mathbb{Z}_n^*$. Nesse caso, o conjunto das bases mentirosas para $n \geqslant 3$ ímpar e composto

$$\mathsf{M}_n := \left\{ a \in \{1, 2, \ldots, n-1\} \colon a^{n-1} \equiv 1 \pmod{n} \right\}$$

é um subconjunto do $\mathbb{Z}_n^*$. Ademais, pela proposição acima $1 \in \mathsf{M}_n$. Se $a, b \in \mathsf{M}_n$, então $a \cdot b \pmod{n} \in \mathsf{M}_n$ pois $(ab)^{n-1} \equiv a^{n-1} b^{n-1} \equiv 1 \pmod{b}$, portanto M_n é subgrupo de $\mathbb{Z}_n^*$. Pelo Teorema de Lagrange[4] temos, para algum inteiro m, que $m|\mathsf{M}_n| = |\mathbb{Z}_n^*|$.

[4] Em grupos finitos, a cardinalidade de um subgrupo divide a cardinalidade do grupo.

Se M_n for subgrupo próprio temos $m \geqslant 2$, portanto uma escolha na linha 1 do algoritmo causa erro na resposta com probabilidade

$$\frac{|\mathsf{M}_n \setminus \{1, n-1\}|}{|\{2,\ldots,n-2\}|} \leqslant \frac{|\mathsf{M}_n|}{n-1} \leqslant \frac{|\mathbb{Z}_n^*|/2}{n-1} < \frac{1}{2}$$

Senão, $m = 1$ e a igualdade $\mathsf{M}_n = \mathbb{Z}_n^*$ ocorre quando n é um **número de Carmichael**

$$n \text{ é impar, composto e } a^{n-1} \equiv 1 \pmod{n} \text{ para todo } a \in \mathbb{Z}_n^*.$$

Esses números são bastante raros. Sobre a distribuição, sabemos que se $c(n)$ é a quantidade de números de Carmichael até n, então

$$n^{0{,}332} < c(n) < ne^{-\frac{\log n \log\log\log n}{\log\log n}}.$$

A cota inferior é de Harman (2005) e dela deduz-se que há infinitos números de Carmichael, a cota superior é de Erdős (1956). Note que desse fato podemos concluir que a densidade dos números de Carmichael até 10^{256} é menor que $0{,}8 \times 10^{-58}$.

Nesse caso, quando um número de Carmichael n é dado como entrada no teste de Fermat, o algoritmo responde certo somente se escolher $a \in \{2, 3, \ldots, n\}$ tal que $\mathrm{mdc}(a, n) > 1$. A probabilidade de erro é

$$\frac{\varphi(n)-2}{n-3} > \frac{\varphi(n)}{n} = \prod_p \left(1 - \frac{1}{p}\right)$$

onde o produto é sobre todo primo p que divide n (corolário 2.33). Se n tem poucos e grandes fatores primo, resulta num número próximo de 1.

Exercício 2.43 (critério de Korselt). Prove que n é um número de Carmichael se, e só se, é composto, livre de quadrado (não é divisível por um quadrado maior que 1) e $p-1$ divide $n-1$ para todo p que divide n.

Exercício 2.44. Prove que um número de Carmichael n tem pelo menos 3 fatores primos distintos. (Dica: assuma $n = pq$ e derive uma contradição usando o exercício anterior.)

Segundo Garfinkel (1994) o teste Fermat é usado pelo PGP[5], versão 2.6.1, para gerar números primos. A chance do PGP gerar um número de Carmichael é menor que 1 em 10^{50}. O PGP escolhe um primo para seus protocolos de criptografia e assinatura digital da seguinte maneira: escolhe $b \in_{\mathcal{U}} \{0,1\}^{t-2}$ e acrescenta 11 como os 2 bits mais significativos, com isso temos um natural m de t bits; verifica se m é divisível por algum dos 100 menores primos; se m não é primo então recomeça-se com o próximo ímpar; caso contrário, usa 4 rodadas do teste de Fermat.

[5] *Pretty Good Privacy*, é um software de criptografia que fornece autenticação e privacidade criptográfica para comunicação de dados desenvolvido por Phil Zimmermann em 1991.

O teste de Lucas. Acabamos de ver um método probabilístico que ao declarar que um número é primo, tal número ou de fato é primo ou tivemos muito azar em tentar provar que o número é composto. Uma vez que não esperamos uma sequência de eventos ruins, depois de algumas repetições aceitamos que o número é primo. No entanto, não temos uma prova de primalidade do número. Esta seção é dedicada a um teste que pode provar que um número é primo.

O teste de Lucas é baseado no Pequeno Teorema de Fermat e na fatoração de $n-1$. Existem alguns testes para primalidade de n baseados na fatoração de $n+1$ ou de $n-1$ e que funcionam bem quando essas fatorações são fáceis como é o caso dos números de Fermat e de Mersenne.

Teorema 2.45 *Um inteiro $n > 2$ é primo se, e somente se, existe $a \in \{1, 2, \ldots, n-1\}$ tal que*

(1) $a^{n-1} \equiv 1 \pmod{n}$ *e*

(2) $a^{(n-1)/p} \not\equiv 1 \pmod{n}$ *para todo p primo e divisor de $n-1$.*

Antes de provar esse teorema, verifiquemos o seguinte fato que será usado: se $a^{n-1} \equiv 1 \pmod{n}$ então $\operatorname{mdc}(a, n) = 1$. Assuma que $n \mid a^{n-1} - 1$ e que $d \mid n$ e $d \mid a$. De $d \mid n$ e $n \mid a^{n-1} - 1$ temos $d \mid a^{n-1} - 1$, mas $d \mid a^{n-1}$ portanto $d \mid 1$, logo $d = \pm 1$. Tomando $d = \operatorname{mdc}(a, n)$, temos $d = 1$.

Demonstração. Se valem (1) e (2), então pelo corolário 2.24 a tem ordem multiplicativa $n-1$. De (1) temos que $\operatorname{mdc}(a, n) = 1$, isso e o Teorema de Euler implicam que $n-1$ divide $\varphi(n)$ (exercício 2.22) e como $\varphi(n) = |\mathbb{Z}_n^*| \leqslant n-1$ temos que $\varphi(n) = n-1$, ou seja, n é primo (exercício 2.34).

Se n é primo, tome para a uma raiz primitiva. O item (1) decorre do teorema de Fermat. O item (2) decorre de a ser raiz primitiva, logo $\operatorname{ord}_*(a) = n-1$, portanto, $a^k \not\equiv 1 \pmod{n}$ para $k < n-1$ positivo. □

Instância : inteiros positivos $n > 1$, k e os fatores primos distintos de $n-1$.
Resposta : *sim* n é primo, *não* se n é composto.

1 **repita**
2 $a \leftarrow_{\mathcal{U}} \{2, 3, \ldots, n-1\}$;
3 **se** $a^{n-1} \equiv 1 \pmod{n}$ **então**
4 **para cada** *fator primo p de $n-1$* **faça**
5 **se** $a^{(n-1)/p} \not\equiv 1 \pmod{n}$ **então responda** *sim.*
6 **até que** *completar k iterações;*
7 **responda** *não.*

Algoritmo 23: teste de Lucas.

A quantidade de divisores primos distintos de $n-1$ é $k = O(\log n)$ (veja equação (2.9)). Cada iteração no algoritmo calcula $k+1$ exponenciais e uma exponenciação modular custa $O(\log^3 n)$, portanto, o tempo de execução é $kO(\log^4 n) = O(\log^5 n)$.

O teste de Miller–Rabin surgiu de duas contribuições, Miller (1975) propôs um teste de primalidade determinístico baseado na validade da hipótese de Riemann generalizada (ainda uma conjectura) e Rabin (1980a) introduziu aleatoriedade no teste tornando-o independente dessa hipótese.

O teste de Miller–Rabin é baseado no fato de que se $n > 2$ é primo, então a equação $x^2 \equiv 1 \pmod{n}$ tem exatamente duas soluções inteiras, como demonstramos a seguir. Claramente, 1 e -1 satisfazem a equação $x^2 \equiv 1 \pmod{n}$. Ainda, $-1 \equiv n-1 \pmod{n}$ e se $n > 2$ então $-1 \not\equiv 1 \pmod{n}$. Portanto, são pelo menos duas soluções. De $x^2 \equiv 1 \pmod{n}$ temos que n divide $x^2 - 1 = (x+1)(x-1)$, portanto, n divide $x-1$ ou n divide $x+1$, ou seja, $x \equiv 1 \pmod{n}$ ou $x \equiv -1 \pmod{n}$.

Uma solução inteira $1 \leqslant x < n$ de $x^2 \equiv 1 \pmod{n}$ é uma **raiz quadrada de 1 módulo** n. Dizemos que o 1 e $n-1$ são as raízes triviais de 1 módulo n. Assim, uma raiz não trivial de algum inteiro prova que n é composto.

Assim, para $n > 2$ ímpar podemos escrever $n - 1 = 2^s r$ com $s \geqslant 1$ e r ímpar, pelo teorema fundamental da aritmética, e, dado a, podemos calcular $a^{n-1} \bmod n$ através da sequência $(b_0, b_1, \ldots, b_s)$ dada por

$$\left(a^{2^0 r} \bmod n, a^{2^1 r} \bmod n, a^{2^2 r} \bmod n, \cdots, a^{2^s r} \bmod n\right).$$

Se b_i é 1 ou $n-1$, para algum i, então $b_j = 1$ para todo $j > i$, pois $1^2 \equiv (n-1)^2 \equiv (-1)^2 \pmod{n}$.

Exemplo 2.46. Para $n = 561$ e $a = 2$, temos $560 = 2^4 \cdot 35$ e a sequência

$$(2^{35} \bmod 561, 2^{2\cdot 35} \bmod 561, 2^{4\cdot 35} \bmod 561, 2^{8\cdot 35} \bmod 561) = (263, 166, 67, 1).$$

Para $a = 6$

$$(2^{35} \bmod 561, 2^{2\cdot 35} \bmod 561, 2^{4\cdot 35} \bmod 561, 2^{8\cdot 35} \bmod 561) = (318, 144, 540, 441).$$

Nesse caso 2 é uma testemunha mentirosa e 6 uma testemunha de Fermat para 561.

Para $n = 325$ e $a = 2$, temos $324 = 2^2 \cdot 81$ e a sequência

$$(2^{81} \bmod 325, 2^{2\cdot 81} \bmod 325, 2^{4\cdot 81}) = (252, 129, 66).$$

O 2 é uma testemunha de Fermat para 325. Para $a = 7$

$$(2^{81} \bmod 325, 2^{2\cdot 81} \bmod 325, 2^{4\cdot 81}) = (307, 324, 1)$$

de modo que 7 é uma testemunha mentirosa para 325. Para $a = 49$ a sequência é $(324, 1, 1)$, ou $(-1, 1, 1)$, e para $a = 126$ a sequência é $(1, 1, 1)$. ◊

Se n é ímpar e a sequência de inteiros módulo n

$$\left(a^{2^0 r}, a^{2^1 r}, a^{2^2 r}, \cdots, a^{2^s r}\right)$$

é da forma, onde "×" significa $\notin \{1,-1\}$,

$$(\times,\times,\ldots,\times) \text{ ou } (\times,\ldots,\times,-1) \text{ ou } (\times,\times,1,1,\ldots,1) \text{ ou } (\times,\times,\ldots,\times,1)$$

então n é composto. Nos dois primeiros casos a é uma testemunha para n composto, nos outros dois $b_{i-1} \neq 1, n-1$ e $b_i = 1$ de modo que b_{i-1} é uma raiz não trivial de 1 módulo n. Em resumo, se em algum momento temos uma sequência onde a primeira ocorrência de 1, se houver, não é precedida por -1 então n é composto. Resta as seguintes formas para a sequência

$$(\pm 1,1,\ldots,1,1,1) \text{ ou } (\times,\times,\ldots,\times,-1,1,\ldots,1)$$

e nesses casos não temos informação sobre a primalidade de n.

Caso seja verdadeiro que a sequência não comece com 1 e que o -1 não ocorra, exceto possivelmente na última posição, ou seja,

$$a^r \not\equiv \pm 1 \pmod{n} \quad \text{e} \quad a^{2^j r} \not\equiv -1 \pmod{n} \quad (\forall j \in \{1,\ldots,s-1\})$$

então n é composto. A prova desse fato está no lema a seguir. Um $a \in \{1,\ldots,n-1\}$ para o qual vale a equação acima é uma **testemunha forte** para o fato de n ser composto.

Lema 2.47 *Sejam n um primo ímpar e r,s inteiros tais que $n-1 = 2^s r$ com r ímpar. Então, para todo inteiro a não divisível por n vale ou $a^r \equiv 1 \pmod{n}$ ou $a^{2^j r} \equiv -1 \pmod{n}$ para algum j, $0 \leqslant j \leqslant s-1$.*

Demonstração. Seja a como no enunciado e consideremos $(a^{2^0 r}, a^{2^1 r}, a^{2^2 r}, \cdots, a^{2^s r})$ a sequência das potências de a tomadas módulo n. Chamamos de t o menor expoente tal que $a^{2^t r} \equiv 1 \pmod{n}$. Como $a^{2^s r} = a^{n-1} \equiv 1 \pmod{n}$, o número t está bem definido. Então, ou $t = 0$ e temos $a^r \equiv 1 \pmod{n}$ ou $1 \leqslant t \leqslant s$ e $a^{2^t r} - 1 \equiv 0 \pmod{n}$, mas $a^{2^t r} - 1 = (a^{2^{t-1} r} + 1)(a^{2^{t-1} r} - 1)$. Como n divide $a^{2^t r} - 1$ e, pela minimalidade de t, não divide $a^{2^{t-1} r} - 1$, necessariamente n divide $a^{2^{t-1} r} + 1$, ou seja, $a^{2^{t-1} r} \equiv -1 \pmod{n}$. □

O algoritmo de Miller citado no começo desta seção assume a hipótese generalizada de Riemann e um teorema que garante que, se n é composto, então existe uma testemunha forte de tamanho $O(\log^2 n)$. Esse limitante foi, mais tarde, especializado para $2\log^2 n$, ou seja, o teste determinístico verifica se há uma testemunha forte para $a = 2,3,\ldots,\lfloor 2(\log n)^2 \rfloor$, caso não encontre então declara, corretamente, que n é primo e usando $O(\log^3 n)$ operações aritméticas.

A proposta de Rabin foi sortear as bases para teste. Dado $n = 2^s r + 1$, $s \geqslant 1$ e r ímpar, o algoritmo sorteia $a \in \{2,\ldots,n-2\}$, testa se $a^r \equiv 1 \pmod{n}$ ou se $a^{2^t r} \equiv -1 \pmod{n}$ para algum $t = 0,\ldots,s-1$ e caso positivo declara n composto. A vantagem da teste de Miller–Rabin com relação ao teste de Fermat é que agora não existe o efeito análogo ao dos números de Carmichael no teste de Fermat,

Instância : inteiro $n \geqslant 3$ ímpar.
Resposta : *verdadeiro* se encontrou testemunha do fato "n é *composto*", *falso* caso contrário.

1 Determine s e r tal que $n-1 = 2^s r$ com r ímpar;
2 $a \leftarrow_{\mathcal{U}} \{2,3,\ldots,n-2\}$;
3 $x \leftarrow a^r \bmod n$;
4 **se** $x \in \{1, n-1\}$ **então responda** *falso*.
5 **para** i *de* 1 *até* $s-1$ **faça**
6 $\quad x \leftarrow x^2 \bmod n$;
7 $\quad$ **se** $x = n-1$ **então responda** *falso*.
8 **responda** *verdadeiro*.

Algoritmo 24: teste de Miller–Rabin.

como demonstraremos mais tarde, as bases que não são testemunhas fortes para o fato de n ser composto formam um subgrupo próprio de $\mathbb{Z}_n^*$.

Cada divisão por 2 na linha 1 custa $O(\log n)$, no total são s divisões. No restante, o custo é o de uma sorteio, $O(\log n)$, mais uma exponenciação modular, $O(\log r \log^2 n)$, mais $s = O(\log n)$ multiplicações módulo n, cada uma custa $O(\log^2 n)$, portanto $O((s+1)\log n + (\log r + s)\log^2 n)$, como $\log_2 r + s = \log_2(n-1)$, o tempo de execução é $O(\log^3 n)$.

Se o teste de Miller–Rabin responde *falso* é porque para os parâmetros n, r, s e o sorteio a valem

$$a^r \equiv \pm 1 \pmod{n} \text{ ou} \tag{2.12}$$

$$a^{2^j r} \equiv -1 \pmod{n} \text{ para algum } j,\ 1 \leqslant j \leqslant s-1. \tag{2.13}$$

e o inteiro a *não é testemunha forte* para a composição de n, nesse caso n pode ser primo ou composto.

Se n é ímpar e composto então a é uma **base mentirosa forte** para n. Para todo composto $n = 2^s r + 1 \geqslant 3$ com $s \geqslant 1$ e r ímpar definimos

$$M_n := \{a \in \{1,2,\ldots,n-1\}:\ \text{vale (2.12) ou (2.13)}\}$$

o conjunto das bases mentirosas, aquelas que enganam o teste de primalidade de n fazendo-o responder primo. Em qualquer um dos casos (2.12) ou (2.13) temos $a^{n-1} \equiv 1 \pmod{n}$, portanto, como já provamos, $\operatorname{mdc}(a,n) = 1$, logo $M_n \subset \mathbb{Z}_n^*$.

Dizemos que um inteiro ímpar e composto n é um **pseudoprimo forte** para a base a, $1 \leqslant a < n$, se vale a conclusão do lema 2.47. Os pseudoprimos fortes são pseudoprimos de Fermat, mas são mais raros que os de Fermat.

Exemplo 2.48. Para $n = 91$, temos $n - 1 = 90 = 2^1 \cdot 45$. Como $9^r = 9^{45} \equiv 1 \pmod{91}$ temos que 91 é pseudoprimo forte para a base 9. Ainda, 1, 9, 10, 12, 16, 17, 22, 29, 38, 53, 62, 69, 74, 75, 79, 81, 82, 90 são todas as bases para as quais 91 é um pseudoprimo forte. ◊

A seguir limitaremos em duas abordagens a probabilidade de erro do algoritmo. A primeira é mais simples e o resultado é mais fraco. Na segunda há maior exigência de pré-requisitos em Teoria dos Grupos. Ambas mostram que o número de testemunhas no caso composto é abundante, o que propicia um ambiente favorável para um teste aleatorizado.

O teste de Miller–Rabin pode ser iterado para diminuirmos a probabilidade de erro. Vamos considerar um parâmetro de entrada k que controla a probabilidade de erro do algoritmo realizando k testes independentes, quanto maior k, menor a chance de responder errado. O tempo de execução é $O(k \log^3 n)$.

Instância : um inteiro ímpar $n \geqslant 3$ e o número de rodadas k.
Resposta : *não* se n não é primo, caso contrário *sim* com probabilidade de erro $\leqslant (1/4)^k$.

1 **repita**
2 | **se** teste de Miller–Rabin(n) **então responda** *não*.
3 **até que** *complete k rodadas;*
4 **responda** *sim*.

Algoritmo 25: Algoritmo de Miller–Rabin.

Teorema 2.49 *O algoritmo de Miller–Rabin responde errado com probabilidade no máximo* $(1/2)^k$.

Demonstração. Vamos assumir que n seja um número de Carmichael: $n = 2^s r + 1$ um inteiro ímpar e composto. Definimos

$$t := \max\{j \in \{0, 1, \ldots, s-1\} : \exists b \in \mathsf{M}_n,\ b^{2^j r} \equiv -1 \pmod{n}\}.$$

Notemos que $(-1)^{2^0 r} \equiv -1 \pmod{n}$ e, como n é Carmichael, $b^{n-1} = b^{2^s r} \equiv 1 \pmod{n}$, portanto $t \in \{0, 1, \ldots, s-1\}$ está definido.

Definimos $m := 2^t r$ e

$$\mathsf{K} := \{a \in \mathbb{Z}_n : a^m \equiv \pm 1 \pmod{n}\}.$$

Se $a \in \mathsf{M}_n$ então $a \in \mathsf{K}$: se $a^r \equiv 1 \pmod{n}$ então $a^{2^t r} \equiv 1 \pmod{n}$; se $a^{2^j r} \equiv -1 \pmod{n}$ para algum j, então $j \leqslant t$. Ademais, se $a \in \mathsf{K}$, então de $t < s$ deduzimos que $a^{n-1} \equiv 1 \pmod{n}$, portanto, $\mathrm{mdc}(a, n) = 1$. Dessa forma

$$\mathsf{M}_n \subset \mathsf{K} \subset \mathbb{Z}_n^*.$$

Resta mostrarmos que K é um subgrupo próprio. Certamente, $1 \in \mathsf{K}$. Agora, tomemos $a, b \in \mathsf{K}$. Então $(ab)^m \equiv a^m b^m \equiv (\pm 1)(\pm 1) \equiv \pm 1 \pmod{n}$.

Para finalizar, vamos mostrar um elemento de $\mathbb{Z}_n^*$ que não está em K. Para isso, vamos usar o exercício 2.44 que garante que n tem pelo menos três divisores primos distintos, logo podemos escrever $n = n_1 n_2$ com n_1 e n_2 ímpares e coprimos.

Definimos $a_1 \coloneqq b \pmod{n_1}$ e considere a única solução $x \in \mathbb{Z}_n$ de

$$\begin{cases} x \equiv a_1 & \pmod{n_1} \\ x \equiv 1 & \pmod{n_2} \end{cases} \tag{2.14}$$

dada pelo Teorema Chinês do Resto (seção A.4 do apêndice, página 127). Módulo n_1 temos que $x^m \equiv a_1^m \equiv b^m \equiv -1 \pmod{n_1}$. Módulo n_2 temos que $x^m \equiv 1^m \equiv 1 \pmod{n_2}$. Portanto, $x^m \not\equiv \pm 1 \pmod{n}$, ou seja, $x \notin K$. Por outro lado, $x^{m^2} \equiv 1 \pmod{n_1}$ e $x^{m^2} \equiv 1 \pmod{n_2}$, portanto $x^{m^2} \equiv 1 \pmod{n}$ pelo teorema chinês do resto, logo $\operatorname{mdc}(x, n) = 1$, ou seja, $x \in \mathbb{Z}_n^*$.

Agora, se n não é número de Carmichael, então

$$M_n \subset F \subsetneq \mathbb{Z}_n^*$$

com os mentirosos de Fermat F subgrupo próprio de $\mathbb{Z}_n^*$.

Em ambos os casos, existe um X subgrupo próprio de $\mathbb{Z}_n^*$ com $M_n \subset X \subset \mathbb{Z}_n^*$ e, pelo Teorema de Lagrange, $|X| = \varphi(n)/m \leqslant \varphi(n)/2$ pois $m \geqslant 2$, e $|M_n| \leqslant |X|$. Assim, a probabilidade de erro em uma rodada é

$$\frac{|M_n \setminus \{1, n-1\}|}{\{2, 3, \ldots, n-2\}} \leqslant \frac{|M_n|}{\{1, 2, \ldots, n-1\}} \leqslant \frac{|X|}{\varphi(n)} \leqslant \frac{1}{2}$$

e em k rodadas independentes $(1/2)^k$. □

Com mais esforço conseguimos um resultado melhor. A seguir vamos dar uma demonstração que requer do leitor conhecimento de alguns resultados básicos da Teoria dos Grupos.

Vamos definir uma sequência ordenada por inclusão de subgrupos de $\mathbb{Z}_n^*$ que contêm as não testemunhas M_n (que não é subgrupo)

$$M_n \subset K \subset L \subset F \subset \mathbb{Z}_n^*$$

e das três últimas inclusões, se duas forem próprias temos $|M_n| \leqslant 4\varphi(n)$, logo a probabilidade de sortear um mentiroso é $\leqslant 1/4$. Nas inclusões acima F são os mentirosos de Fermat

$$F = \{a \in \mathbb{Z}_n : a^{n-1} \equiv 1 \pmod{n}\}.$$

Como já vimos acima, se n não for um número de Carmichael, então pelo menos um $a \in \mathbb{Z}_n^*$ não é um mentiroso de Fermat logo a última inclusão é estrita.

Suponha que n é um número de Carmichael fatorado como $n = p_1^{m_1} p_2^{m_2} \cdots p_k^{m_k}$ com $k \geqslant 3$ e $p_i > 2$ para todo i.

Exercício 2.50. Assuma que $n = p_1^{m_1} p_2^{m_2} \cdots p_k^{m_k}$ e prove que para quaisquer inteiros x e y, se $x \equiv y \pmod{n}$ então $x \equiv y \pmod{p_i^{m_i}}$ para todo i.

Escrevemos $n = 2^s r + 1$, com r ímpar, e definimos

$$t := \max\{j \in \{0,1,\dots,s-1\} : \exists b \in \mathbb{Z}_n^*,\ b^{2^j} \equiv -1 \pmod{n}\}.$$

Definimos $m := 2^t r$ e os grupos $\mathsf{K} \subset \mathsf{L}$ dados por

$$\mathsf{L} := \{a \in \mathbb{Z}_n : a^m \equiv \pm 1 \pmod{p_i^{m_i}} \text{ para todo } i\},$$
$$\mathsf{K} := \{a \in \mathbb{Z}_n^* : a^m \equiv \pm 1 \pmod{n}\}$$

e, finalmente, definimos o homomorfismo de grupos

$$f : \mathsf{L} \to \{\pm 1 \bmod p_1^{m_1}\} \times \{\pm 1 \bmod p_2^{m_2}\} \times \cdots \times \{\pm 1 \bmod p_k^{m_k}\}$$
$$a \bmod n \mapsto \left(a^m \bmod p_1^{m_1}, a^m \bmod p_k^{m_k}, \dots, a^m \bmod p_k^{m_k}\right).$$

O *kernel* do homomorfismo é o subgrupo de L dado pelos elementos de L cuja imagem por f é $(1,1,\dots,1)$. Pelo teorema do isomorfismo de grupos temos $\mathsf{L}/\ker f$ é isomorfo a $\mathrm{Im}(f) = \prod_i \{\pm 1 \bmod p_i^{m_i}\}$ pois f é sobrejetora. Ademais $f(\mathsf{K}) = \{(-1,-1,\dots,-1),(1,1,\dots,1)\}$.

Do parágrafo acima concluímos que $f(\mathsf{L})$ tem ordem 2^k e $\mathsf{F}(\mathsf{K})$ tem ordem 2, portanto $|\mathsf{L}|/|\ker f| = 2^k$ e $|\mathsf{K}|/|\ker f| = 2$, então $|\mathsf{L}|/|\mathsf{K}| = 2^{k-1} \geqslant 4$ pois $k \geqslant 3$.

Se para o número de fatores primos distintos de n vale $k = 2$ então n não é um número de Carmichael, logo $\mathsf{F} \neq \mathbb{Z}_n^*$. Vamos mostrar que $\mathsf{F} \neq \mathsf{L}$. Escrevemos $n = n_1 n_2$ com n_1 e n_2 ímpares e coprimos e tomamos x como em (2.14) de modo que $x \notin \mathsf{K}$, porém $x \in \mathsf{F}$ pois $x^{m^2} \equiv 1 \pmod{n}$ e $m^2 \mid n-1$, portanto $x^{n-1} \equiv 1 \pmod{n}$. Com isso,

$$\mathsf{K} \subsetneq \mathsf{F} \subsetneq \mathbb{Z}_n^*$$

de modo que $|\mathsf{K}| \leqslant \varphi(n)/4$.

Finalmente, se $k = 1$ então $n = p^e$ com $e \geqslant 2$ dado que n não é primo. Nesse caso, sabemos que $\mathbb{Z}_n^*$ é cíclico (tem raiz primitiva), portanto, temos um isomorfismo entre os grupos $(\mathbb{Z}_n^*, \cdot \bmod n)$ e $(\mathbb{Z}_{\varphi(n)}, + \bmod n)$. O número de soluções módulo n de $x^{n-1} \equiv 1 \pmod{n}$ é igual ao número de soluções de $(n-1)x \equiv 0 \pmod{\varphi(n)}$, portanto $|\mathsf{F}| = \mathrm{mdc}(n-1, \varphi(n)) = \mathrm{mdc}(p^e - 1, (p-1)p^{e-1}) = p-1$. Portanto vale

$$\frac{|\mathbb{Z}_n^*|}{|\mathsf{F}|} = p^{e-1} \geqslant 4$$

para todo $n > 9$. Quando $n = 9$, verifica-se que $\mathsf{M}_n = \{1, n-1\} = (n-1)/4$. De fato, $n - 1 = 8 = 2^3$, logo $e = 3$ e $k = 1$. A sequência de teste do algoritmo de Miller–Rabin para a é $(a \bmod 9, a^2 \bmod 9, a^4 \bmod 9)$

$a \bmod 9$	1	2	3	4	5	6	7	8
$a^2 \bmod 9$	1	4	0	7	7	0	4	1
$a^4 \bmod 9$	1	7	0	4	4	0	7	1

portanto 1 e 8 são mentirosos para 9. Em todos os casos, para todo $n \geqslant 9$ vale que $|M_n| \leqslant \varphi(n)/4$. Assim, a probabilidade de erro em uma rodada é

$$\frac{|M_n \setminus \{1, n-1\}|}{\{2,3,\ldots,n-2\}} \leqslant \frac{|M_n|}{\{1,2,\ldots,n-1\}} \leqslant \frac{\varphi(n)/4}{\varphi(n)} \leqslant \frac{1}{4}$$

e em k rodadas independentes $(1/4)^k$.

Teorema 2.51 *Para todo $n \geqslant 9$, o algoritmo de Miller–Rabin responde errado com probabilidade no máximo $(1/4)^k$.* □

Para muitos inteiros compostos n a quantidade de bases para as quais n é pseudoprimo forte é bem pequeno, muito menor que a estimativa $\varphi(n)/4$. Por exemplo, para $n = 105$ temos $M_{105} = \{1, 104\}$. Entretanto, existe inteiro n tal que $|M_n|/\varphi(n) = 1/4$, como é o caso do 9 e do 91, onde temos $|M_n| = 18$, como vimos no exemplo 2.48, e $\varphi(n) = 72$.

2.3.3 TESTE PRIMALIDADE DE AGRAWAL–BISWAS

O algoritmo aleatorizado para primalidade proposto por Agrawal e Biswas (2003) é menos eficiente que o teste de Miller–Rabin e a probabilidade de erro é maior, entretanto esse algoritmo tem a sua importância histórica pois foi o ponto partida para Agrawal, Kayal e Saxena construírem o algoritmo determinístico de tempo polinomial: "o novo algoritmo é simplesmente uma desaleatorização do nosso algoritmo. Isto pode ser feito da seguinte maneira. Nosso algoritmo aleatorizado é baseado em testes de identidade $(1+x)^n = 1 + x^n$ modulo um polinômio g escolhido aleatoriamente, de grau $\lceil \log n \rceil$ e acerta com probabilidade pelo menos 2/3. O espaço amostral de g é claramente de tamanho exponencial $(n^{\lceil \log n \rceil})$. Foi mostrado em Agrawal et al. [2002] que o espaço amostral pode ser reduzido para $O(\log^4 n)$, sem reduzir a probabilidade de sucesso!" (Agrawal e Biswas, 2003).

Os algoritmos probabilísticos eficientes conhecidos até o momento para teste de primalidade baseiam-se no pequeno teorema de Fermat. O presente algoritmo é basado numa generalização do teorema de Fermat

$$n \text{ é primo se, e somente se, } (x+a)^n \equiv x^n + a \pmod{n} \text{ para todo } 1 \leqslant a \leqslant n-1.$$

Por exemplo, $(x+1)^5 = x^5 + x^4 + 10x^3 + 10x^2 + 5x + 1$ que módulo 5 fica $x^5 + 1$. Agora, $(x+1)^6 = x^6 + 6x^5 + 15x^4 + 20x^3 + 15x^4 + 6x^2 + 1$ que módulo 6 fica $x^6 + 3x^4 + 2x^3 + 3x^2 + 1 \neq x^6 + 1$.

Teorema 2.52 *Sejam a e n inteiros coprimos. Então, $n \geqslant 2$ é primo se e somente se*

$$(x+a)^n = x^n + a \text{ no anel } \mathbb{Z}_n[x]. \tag{2.15}$$

Demonstração. Se n é primo, então n divide $\binom{n}{i}$ para todo $i \in \{1, 2, \ldots, n-1\}$, portanto, usando o

binômio de Newton,

$$(x+a)^n = \sum_{i=0}^{n} \binom{n}{i} x^n a^{n-i} = \binom{n}{0} x^n + \binom{n}{n} a^n = x^n + a^n$$

no $\mathbb{Z}_n[x]$ (2.15) segue do teorema de Fermat.

Por outro lado, se $n > 1$ é composto, seja p^k a maior potência de um fator primo p de n. Então $n = p^k c$ e

$$\binom{n}{p} = \frac{n}{p}\binom{n-1}{p-1} = p^{k-1} c \binom{n-1}{p-1}.$$

Supondo que p^k divide o lado direito dessa igualdade, temos que $p|\binom{n-1}{p-1}$, um absurdo, portanto, $\binom{n}{p} \not\equiv 0 \pmod{p^k}$, logo $\binom{n}{p} \not\equiv 0 \pmod{n}$. Como $n \nmid a$, também $n \nmid a^{n-p}$, ou seja, o coeficiente $\binom{n}{p} a^{n-p}$ de x^p em $(x+a)^n$ é não nulo, logo, $(x+a)^n \neq x + a^n$ nesse anel. □

O teste dado pelo teorema 2.52 não é prático porque computar os coeficientes da expansão de $(x+1)^n$ na equação (2.15) toma tempo $\Omega(n)$. Uma alternativa seria usar o teste probabilístico para identidade de polinômio. No entanto, isso falha por dois motivos. Primeiro é que se n não é primo, então $\mathbb{Z}_n$ não é um corpo como assumimos em nossa análise na seção 2.2.3. O segundo é que o grau do polinômio é $n = |\mathbb{Z}_n|$ e, portanto, muito grande pois o teorema de Schwartz–Zippel requer que os valores sejam escolhidos de um conjunto de tamanho estritamente maior que o grau.

Outra alternativa é testar a igualdade módulo um polinômio com grau cuidadosamente escolhido. A alternativa proposta por Agrawal e Biswas (2003) é sortear um polinômio de baixo grau $h \in \mathbb{Z}_n[x]$ e usar o teste

$$\text{se } n \text{ é primo, então } (x+a)^n \equiv x^n + a \pmod{x^r - 1, n} \text{ para todo } 1 \leqslant n \leqslant n-1 \text{ e todo } r \in \mathbb{N},$$

ou seja, testamos $(x+a)^n \equiv x^n + a$ no anel quociente $\mathbb{Z}_n[x]/(h)$. Tomando $h = x^r - 1$ comparamos $(x+a)^n \pmod{x^r-1}$ com $x^n + a \pmod{x^r-1}$. No cálculo da expansão da potência $(x+a)^n$ módulo h os coeficientes não excedem n e x^t é trocado por $x^{t \bmod r}$, portanto, os graus dos polinômios podem ser mantidos baixo no desenvolvimento do cálculo da potência $(x+a)^n$. Para o tempo de execução ser polinomial devemos ter $r = \log^{O(1)} n$.

Instância : inteiro $n > 3$.
Resposta : n é *composto*, ou *primo* com probabilidade de erro $< 1/2$.

1 **se** n *é da forma* a^b **então responda** *composto*.
2 $h \leftarrow_U \{p\colon p \text{ é um polinômio mônico de grau } \lceil \log n \rceil \text{ em } \mathbb{Z}_n[x]\}$;
3 **se** h *divide* $(x+1)^n - (x^n+1)$ *no* $\mathbb{Z}_n[x]$ **então responda** *primo*.
4 **senão responda** *composto*.

Algoritmo 26: teste de primalidade Agrawal–Biswas.

Vamos agora analisar o algoritmo. Na linha 1 testamos

1 **para** b *de* 2 *até* $\log_2 n$ **faça** $\mathsf{P} \leftarrow \lfloor n^{\frac{1}{b}} \rfloor$
2 **se** $\mathsf{P}^b = 1$ **então responda** *sim*.
3 **senão responda** *não*.

pois se n é da forma a^b, então $2 \leqslant b \leqslant \log_2(n)$. Há várias maneiras de determinar uma raiz b de n, uma delas usa busca binária e é descrita no exercício 2.59, página 112, com tempo de execução[6] $\mathsf{O}(\log^3 n)$. Na linha 2, escolhemos $\lceil \log n \rceil$ coeficientes no $\mathbb{Z}_n$ uniforme e independentemente, cada um de tamanho $\mathsf{O}(\log n)$, portanto a linha contribui com $\mathsf{O}(\log^2 n)$ para o tempo de execução do algoritmo. Na linha 3, $x^n + a$ em $\mathbb{Z}_n[x]/(h)$ pode ser representado por $x^{n \bmod r} + a$ (verifique), onde $r = \lceil \log n \rceil$ é o grau de h. Para computar a expansão de $(x+a)^n$ módulo h podemos adaptar facilmente o algoritmo 8 para exponenciação modular da seguinte forma

$\mathsf{P}(x) \leftarrow 1$;
$n = b_{\ell-1} \dots b_0$ em binário;
para k *de* $\ell - 1$ *até* 0 **faça**
  $\mathsf{P}(x) \leftarrow \mathsf{P}(x)^2$;
  se $b_k = 1$ **então** $\mathsf{P}(x) \leftarrow \mathsf{P}(x)(x + a)$;
  $\mathsf{P}(x) \leftarrow \mathsf{P}(x) \bmod (h(x), n)$;
responda $\mathsf{P}(x)$.

Temos $\ell = \mathsf{O}(\log n)$ rodadas do laço. No caso $b_k = 0$, como $\mathsf{P}(x)$ tem grau no máximo $r - 1$ temos

$$\mathsf{P}(x)^2 = \sum_{j=0}^{2r-2} a_j x^j = \sum_{j=0}^{r-1} a_j x^j + \sum_{j=r}^{r-2} a_j x^{q_j r + j \bmod r} = \sum_{j=0}^{r-1} a_j x^j + \sum_{j=0}^{r-2} a_{j+r} x^j (x^r)^{q_j}$$

O produto $\mathsf{P}(x)^2$ custa $\mathsf{O}((\log n)^4)$, são $\mathsf{O}(r^2)$ multiplicações e somas de números de tamanho $\mathsf{O}(\log n)$. Como $x^r \equiv 1 \pmod h$, se fizermos $a_{2r-1} = 0$ então podemos escrever

$$\mathsf{P}(x)^2 \equiv \sum_{j=0}^{r-1} (a_j + a_{j+r}) x^j \pmod h.$$

O resto módulo n custa $\mathsf{O}(\log^2 n)$, portanto custo para determinar o lado direito na equação acima é $\mathsf{O}(r \log^2 n) = \mathsf{O}((\log n)^3)$. O tempo da linha 3 é $\ell \mathsf{O}((\log n)^4) = \mathsf{O}((\log n)^5)$, portanto, do teste de Agrawal–Biswas tem tempo de execução $\mathsf{O}(\log^5 n)$.

Quanto à correção, se n é primo então $(x + a)^n = x^n + a$ de modo que qualquer h divide a diferença, logo algoritmo responde *primo* com probabilidade 1. Para limitar a probabilidade de erro,

[6] De fato, esse teste pode ser realizado em tempo menor que $c \log^{1+\varepsilon} n$ para qualquer $\varepsilon > 0$ e alguma constante positiva c, ou seja, praticamente linear em $\log(n)$ (Bernstein, 1998).

primeiro assumimos que todo fator primo de n é maior que 16. Isso não estraga o projeto porque o algoritmo pode ser facilmente modificado para testar divisibilidade de n por primos pequenos. Uma modificação mais drástica é que para simplificar provaremos um limitante pior.

Instância : inteiro $n > 3$.
Resposta : n é *composto*, ou *primo* com probabilidade de erro $< 1 - 0{,}49/\log n$.

1 **se** $n \in \{2,3,5,7,11,13\}$ **então responda** *primo*.
2 **se** $a \in \{2,3,5,7,11,13\}$ *divide* n **então responda** *composto*.
3 **se** n *é da forma* a^b **então responda** *composto*.
4 $h \leftarrow_{\mathcal{U}} \{p\colon p$ é um polinômio mônico de grau $\lceil \log n \rceil$ em $\mathbb{Z}_n[x]\}$;
5 **se** h *divide* $(x+1)^n - (x^n+1)$ *no* $\mathbb{Z}_n[x]$ **então responda** *primo*.
6 **senão responda** *composto*.

Em $10\lceil \log n \rceil$ testes independentes a probabilidade de sucesso é maior que $1 - 1/e^5 \approx 0{,}99$ de sucesso.

Agora, consideramos n composto que não é potência de primo e não tem divisor primo menor que 16, ou seja, a execução passou pelas três primeiras linhas do algoritmo acima.

Como n é composto, o polinômio

$$P(x) = (x+1)^n - (x^n+1)$$

é não nulo no $\mathbb{Z}_n[x]$ e também é não nulo no $\mathbb{Z}_p[x]$ para todo p que divide n. De fato, para a maior potência do primo p que divide n, digamos que p^m, o coeficiente de x^{p^m} na expansão de $P(x)$ é não nulo no $\mathbb{Z}_p$, pois p não divide $\binom{n}{p^m}$ (verifique).

Fixemos $p > 16$ um fator primo de n. Vamos mostrar que há uma probabilidade positiva de, ao escolher h como acima, termos $P(x) \not\equiv 0 \pmod{h}$ no $\mathbb{Z}_p[x]$ e, portanto, $P(x) \not\equiv 0 \pmod{h}$ no $\mathbb{Z}_n[x]$.

Primeiro, observemos que se escolhemos h mônico de grau $\lceil \log n \rceil$ em $\mathbb{Z}_n[x]$ e em seguida realizamos operações módulo p, então o resultado é o mesmo que escolher h mônico de grau $\lceil \log n \rceil$ em $\mathbb{Z}_p[x]$. Ainda, cada polinômio mônico de grau r do $\mathbb{Z}_p[x]$ ocorre no $\mathbb{Z}_n[x]$ exatamente $(n/p)^r$ vezes (por quê?), logo o sorteio dá um polinômio em $\mathbb{Z}_p$ com probabilidade uniforme.

O polinômio mônico não nulo $P(x) \in \mathbb{Z}_p[x]$ de grau n tem uma fatoração única como produto de polinômios mônicos irredutíveis e no máximo n/r fatores podem ter grau r. Se h for irredutível, então ou é igual a qualquer um desses n/r fatores e nesse caso o algoritmo responde errado ou é diferente do fatores e $P(x) \pmod{h}$ não é zero, nesse último caso o algoritmo responde corretamente, n é *composto*.

A probabilidade de h ser uma testemunha de que n é composto é a probabilidade do evento

$$h(x) \text{ é irredutível e não é um fator irredutível de } P(x)$$

cuja probabilidade é $\mathbb{P}[\,h(x)$ é irredutível$\,]-\mathbb{P}[\,h(x)$ não é um fator irredutível de $\mathsf{P}(x)\,]$. Do lema 2.37, página 86, e $p>16$ temos probabilidade $\mathsf{I}(r)/p^r > 1/(2r)$ de sortear h irredutível. A probabilidade de que h é um fator irredutível de $\mathsf{P}(x)$ é no máximo $(n/r)/p^r$ e assim a probabilidade de encontrar um h que seja irredutível e não divida $\mathsf{P}(x)$ é pelo menos

$$\frac{1}{2r}-\frac{n}{rp^r}\geqslant\frac{1}{r}\left(\frac{1}{2}-\frac{n}{16^{\log n}}\right)>\frac{49}{100r}=\Omega\left(\frac{1}{\log n}\right)$$

pois $n>p>16$ e $r=\lceil\log n\rceil\geqslant 3$.

Estimativa com probabilidade de erro constante para $n>100$. Assuma $n>p>100$. Assumimos que n é testado na força bruta contra os cem primeiros números primos: 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.

Seja Q um polinômio mônico irredutível do $\mathbb{Z}_p[x]$ com grau entre $1+r/2$ e r e seja C_Q o conjunto dos polinômios de grau r que têm Q como fator, logo $|\mathsf{C}_\mathsf{Q}|=p^{r-\partial\mathsf{Q}}$. Ademais, $\mathsf{C}_\mathsf{Q}\cap\mathsf{C}_{\mathsf{Q}'}=\varnothing$ para polinômios $\mathsf{Q}\neq\mathsf{Q}'$, caso contrário seriam dois fatores com grau maior que $r/2$ cada, um absurdo. Somando sobre todo $\mathsf{Q}\in\mathbb{Z}_p[x]$ irredutível mônico com grau entre $1+r/2$ e r, a quantidade de tais polinômios é, usando o lema 2.37,

$$\sum_{\mathsf{Q}}|\mathsf{C}_\mathsf{Q}|=\sum_{i=1+r/2}^{r}\mathsf{I}(i)p^{r-i}\geqslant\sum_{i=1+r/2}^{r}\left(\frac{p^i-2\sqrt{p^i}}{i}\right)p^{r-i}\geqslant\sum_{i=1+r/2}^{r}\left(\frac{1}{i}-\frac{1}{\sqrt{p^i}}\right)p^r$$

Usando que o N-ésimo número harmônico $\mathsf{H}_\mathsf{N}:=\sum_{i=1}^{\mathsf{N}}1/i$ satisfaz (Graham, Knuth e Patashnik, 1994, seção 6.3)

$$\mathsf{H}_\mathsf{N}=\log(\mathsf{N})+\gamma+\frac{1}{2\mathsf{N}}-\frac{1}{12\mathsf{N}^2}+\frac{\varepsilon_\mathsf{N}}{120\mathsf{N}^4},$$

com $0<\varepsilon_\mathsf{N}<1$ e $\gamma\approx 0{,}5772156649$, a constante de Euler–Mascheroni[7], deduzimos que

$$\sum_{i=1+r/2}^{r}\frac{1}{i}=\mathsf{H}_r-\mathsf{H}_{r/2}>\log 2-\frac{1}{2r}+\frac{3}{12r^2}-\frac{16}{120r^4}\geqslant\log 2-\frac{3383}{37500}$$

para $r\geqslant 5$. Agora, temos que estima a soma de uma progressão geométrica

$$\sum_{i=1+r/2}^{r}\left(\frac{1}{\sqrt{p}}\right)^i=\frac{(1/\sqrt{p})^{r/2}-(1/\sqrt{p})^r}{\sqrt{p}-1}.$$

De $n>p>100$ temos $r=\lceil\log n\rceil\geqslant 5$ e

$$\sum_{i=1+r/2}^{r}\left(\frac{1}{\sqrt{p}}\right)^i<\frac{1}{9}\left(\left(\frac{1}{10}\right)^{r/2}\right)-\left(\frac{1}{10}\right)^r<\frac{1}{9}\left(\frac{1}{10^{5/2}}-\frac{1}{10^5}\right)$$

[7] Essa constante é o limite de $\mathsf{H}_\mathsf{N}-\log\mathsf{N}$ quando $\mathsf{N}\to\infty$.

Logo, a quantidade de polinômios de grau r com algum fator irredutível de grau entre $r/2+1$ e r é

$$\sum_{Q} |C_Q| > \left(\log 2 - \frac{3383}{37500} - \frac{1}{9}\left(\frac{1}{10^{5/2}} - \frac{1}{10^5}\right)\right)p^r > 0{,}6p^r$$

e a probabildade de sortear h como um deles é $> 0{,}6$. Como o grau de Q é pelo menos $r/2+1$ temos $|C_Q| \leqslant p^{(r/2)-1}$ e no máximo $(n/(r/2))|C_Q|$ desses polinômios dividem $P(x)$ e a probabilidade de sortear um deles é menor que

$$\frac{1}{p^r}\frac{2n}{r}p^{(r/2)-1} = \frac{2n}{rp^{(r/2)+1}} \leqslant \frac{2n}{5\cdot 10^{\log n}} < 0{,}001$$

para todo $n > 100$. Assim, a probabilidade com que sorteamos h como um desses Q que não divide $P(x)$, portanto uma testemunha de que n é composto, é pelo menos $0{,}6 - 0{,}001 = 0{,}599$.

2.3.4 GERADOR DE NÚMEROS PRIMOS

O nosso próximo exemplo é um algoritmo aleatorizado para escolher um número primo. Para primos pequenos, com a tecnologia do início do século 21, números até 10^{12} são rapidamente gerados por boas implementações do Crivo de Eratóstenes. Para gerar primos grandes, podemos usar o seguinte algoritmo.

Instância : inteiro positivo n.
Resposta : um primo escolhido uniformemente em $\{n+1,\ldots,2n\}$.

1 **repita** $m \leftarrow_{\mathcal{U}} \{n+1, n+2, \ldots, 2n\}$ **até que** Teste_Primo(m) =*sim*;
2 **responda** m.

Algoritmo 27: gerador de números primos aleatórios.

O postulado de Bertrand garante que, para todo $n > 3$, existe pelo menos um primo em $\{n+1,\ldots,2n-1\}$. Esse postulado foi provado por Chebyshev que garantiu que se $\pi(n)$ é a quantidade de primos menores ou iguais a n, então

$$\frac{1}{3}\frac{n}{\log n} < \pi(2n) - \pi(n) < \frac{7}{5}\frac{n}{\log n} \tag{2.16}$$

onde $\pi(n)$ é a quantidade de primos menores ou iguais a n (Ribenboim, 1996). Logo uma escolha aleatória em $\{n+1,\ldots,2n\}$ tem probabilidade $\rho(n) = (\pi(2n) - \pi(n))/n$ de resultar um primo e

$$\frac{1}{3\log n} < \rho(n) < \frac{7}{5\log n}. \tag{2.17}$$

Nas aplicações em criptografia, por exemplo, queremos números primos cada vez maiores em função da tecnologia da época. No início do século 21 precisávamos de primos com 2.048 bits pelo menos. Nesses casos, os algoritmos aleatorizados para primalidade é a melhor opção, senão a única viável, para os teste. A equação (2.16) garante que existem $\approx 2^k/k$ primos com k bits para todo $k \geqslant 2$.

Assim, a probabilidade de um sorteio (uniforme) no intervalo $\{2^{k-1},\ldots,2^k-1\}$ produzir um primo é $\approx 1/(2k)$, logo o número esperado de sorteios até encontrar um provável primo é $O(k)$.

No caso do teste de Miller–Rabin para primalidade, vimos que a probabilidade de erro em declarar um número composto como primo é $< (1/4)^t$. Isso não significa que a probabilidade de erro do algoritmo acima é $< (1/4)^t$ porque devemos levar em conta a distribuição dos primos. Numa iteração do laço, seja C o evento "m, sorteado na linha 2, é composto" e E o evento "Teste_Primo(m), na linha 3, declara m primo". Então $\mathbb{P}(\,E \mid C\,) < (1/4)^t$ e queremos $\mathbb{P}(\,C \mid E\,)$. Pelo Teorema de Bayes

$$\mathbb{P}(\,C \mid E\,) = \frac{\mathbb{P}(\,E \mid C\,)\mathbb{P}(C)}{\mathbb{P}(E)} \leqslant \frac{\mathbb{P}(\,E \mid C\,)}{\mathbb{P}(E)} < \frac{(1/4)^t}{1/(3k)} = \frac{3k}{4^t}$$

pois $\mathbb{P}(E) > 1/(3k)$ por (2.17).

Observação 2.53. Na prática, após sortear m é mais eficiente testar divisibilidade de m por inteiros primos até uma constante k antes de testar primalidade, pois é relativamente grande a quantidade de números que tem um divisor pequeno. De imediato números pares podem ser descartados. Testar divisibilidade pelos primos até 256 descarta 80% dos números ímpares m candidatos a primo (Menezes, Oorschot e Vanstone, 1997).

2.4 O JANTAR DOS FILÓSOFOS, UM CASO NÃO-ENUMERÁVEL

Nessa seção vamos considerar um problema de computação distribuída cuja solução probabilística tem espaço amostral não enumerável; o tratamento que daremos a ambos os tópicos, computação distribuída e espaço não enumerável, será informal.

O jantar dos filósofos é um problema originalmente proposto por Dijkstra em 1965 e ilustra o problema de alocação de recursos em sistemas distribuídos. Esse problema não tem solução determinística, entretanto apresentaremos uma solução probabilística devida a Lehmann e Rabin (1981).

A seguinte formulação do problema representa toda uma classe de problemas em computação distribuída:

> Cinco filósofos estão reunidos para um jantar em torno de uma mesa circular. A vida de um filósofo consiste basicamente em pensar. Enquanto está pensando, um filósofo não interage com os outros filósofos, entretanto, o filósofo acaba por sentir fome em algum momento. Para se alimentar, ele dispõe de um prato de macarrão que nunca se esvazia, um garfo a sua esquerda e um garfo a sua direita, mas o macarrão encontra-se de tal forma oleoso que é impossível comê-lo com apenas um garfo, sendo necessários dois para tal. Um filósofo pode pegar apenas um garfo de cada vez e, obviamente, é impossível utilizar um garfo que esteja sendo utilizado por um vizinho. Uma vez que um filósofo tenha dois garfos ele se alimenta, devolve os dois garfos e volta a pensar. Um filósofo faminto e que

não seja capaz de pegar seus dois garfos todas as vezes que tente pegá-lo (pois o garfo sempre está em posse de seu vizinho) entra em inanição.

Em suma, um filósofo opera indefinidamente no ciclo: pensar, tentar comer, comer. Para comer um filósofo necessita de acesso exclusivo a dois recursos, cada um deles é compartilhado com um vizinho. O problema computacional consiste em projetar um protocolo que represente os filósofos e os garfos de maneira apropriada, para que se comportem como as entidades descritas no enunciado. O protocolo deve garantir que os filósofos comam.

O modelo probabilístico para esse problema envolve um espaço amostral equivalente ao de lançar uma moeda infinitas vezes. Se temos dois espaços de probabilidade $(\Omega_1, \mathcal{A}_1, \mathbb{P}_1)$ e $(\Omega_2, \mathcal{A}_2, \mathbb{P}_2)$ o espaço produto tem espaço amostral $\Omega_1 \times \Omega_2$ mas o espaço de eventos não é simplesmente $\mathcal{A}_1 \times \mathcal{A}_2$, mas sim a menor[8] σ-álgebra que contém todos os produtos de eventos $A_1 \times A_2$, que denotamos por $\mathcal{A}_1 \otimes \mathcal{A}_2$. No produto, a medida de probabilidade é tal que $\mathbb{P}(A_1 \times A_2) = \mathbb{P}_1(A_1)\mathbb{P}_2(A_2)$ para todos $A_1 \in \mathcal{A}_1$ e $A_2 \in \mathcal{A}_2$. O espaço de probabilidade $(\Omega_1 \times \Omega_2, \mathcal{A}_1 \otimes \mathcal{A}_2, \mathbb{P})$ é o *espaço produto*. Essa definição pode ser estendida para o produto de vários espaços, até uma quantia infinita enumerável deles, com $\Omega = \prod_n \Omega_n$ e $\mathcal{A} = \otimes_n \mathcal{A}_n$ é a menor σ-álgebra que contém os eventos $A_1 \times A_2 \cdots A_k \times \Omega_{k+1} \times \Omega_{k+2} \times \cdots$ para todo k, para todo $A_i \in \mathcal{A}_i$, para todo i. É possível mostrar que há uma única medida de probabilidade $\mathbb{P}$, tal que $\mathbb{P}(A_1 \times A_2 \cdots A_k \times \Omega_{k+1} \times \Omega_{k+2} \times \cdots) = \mathbb{P}_1(A_1)\mathbb{P}_2(A_2)\cdots\mathbb{P}_k(A_k)$.

Exemplo 2.54. Consideremos o espaço amostral formado por todas as sequências binárias

$$\{0,1\}^{\mathbb{N}} := \{(b_0, b_1, b_2, \ldots) \colon b_i \in \{0,1\}\ (\forall i)\}.$$

Denotemos por $\mathcal{C}_k$ a família de todos os eventos de $\{0,1\}^{\mathbb{N}}$ cuja ocorrência é decidida pelos k primeiros bits das sequências. Por exemplo, as sequências tais que $b_1 \neq b_2$ e $b_3 = 0$ é um elemento de $\mathcal{C}_3$; o evento "dois zeros nos sete primeiros lançamentos" é um elemento de $\mathcal{C}_7$. Dado subconjunto $B \subset \{0,1\}^k$ definimos $B_\Omega \subset \{0,1\}^{\mathbb{N}}$ por

$$B_\Omega := \{(b_0, b_1, b_2, \ldots) \colon (b_1, b_2, \ldots, b_k) \in B\}.$$

O conjunto B_Ω pode ser identificado com $B \times \{0,1\}^{\mathbb{N}}$, temos $B_\Omega \in \mathcal{C}_k$ e todo elemento de $\mathcal{C}_k$ pode ser escrito dessa forma para algum $B \subset \{0,1\}^k$. A família $\mathcal{C}_k$ é uma σ-álgebra de subconjuntos de $\{0,1\}^{\mathbb{N}}$, para cada inteiro positivo k, e no jargão de Probabilidade, esses são chamados de *eventos cilíndricos*. Notemos que $\mathcal{C}_k \subset \mathcal{C}_{k+1}$ e que o evento "não ocorre 1" não pode ser expresso por nenhuma dessas famílias.

Agora, fazemos

$$\mathcal{C} := \bigcup_{k \geqslant 1} \mathcal{C}_k$$

[8] É a intersecção de todas as σ-álgebras de $\Omega_1 \times \Omega_2$. Note-se que a intersecção de σ-álgebras de $\Omega_1 \times \Omega_2$ é uma σ-álgebra de $\Omega_1 \times \Omega_2$.

a família dos eventos cuja ocorrência é decidida por um número fixo de bits iniciais. A família $\mathcal{C}$ não é uma σ-álgebra de subconjuntos de $\{0,1\}^{\mathbb{N}}$ pois se tomamos B_k o conjunto das sequências com $b_k = 1$ então temos $B_k \in \mathcal{C}_k$ mas $\overline{\bigcup_k B_k} \notin \mathcal{C}$. Entretanto, $\mathcal{C}$ é uma *álgebra* de subconjuntos de $\{0,1\}^{\mathbb{N}}$, isto é, satisfaz: (i) $\varnothing$ é um elemento da família, (ii) o complemento de um elemento da família também pertence a família e (iii) a união de dois elementos da família pertence a família. Um teorema famoso, conhecido como *Teorema de Extensão de Carathéodory* nos diz que, nesse caso, uma função $P\colon \mathcal{C} \to [0,1]$ que satisfaz (i) $P(\{0,1\}^{\mathbb{N}}) = 1$ e (ii) $P(\bigcup_n B_n) = \sum_n P(B_n)$, para $\{B_n\}_{n\in\mathbb{N}}$ elementos disjuntos da álgebra, pode ser estendida de maneira única para uma medida de probabilidade sobre a menor σ-álgebra que contém a álgebra $\mathcal{C}$.

O próximo passo é definir P de acordo com as hipóteses do parágrafo anterior. Todo $A \in \mathcal{C}$ é da forma $B \times \{0,1\}^{\mathbb{N}}$ para algum $B \subset \{0,1\}^k$, para algum natural k. Definimos

$$P(A) := \frac{|B|}{2^k}. \tag{2.18}$$

Essa definição é consistente (veja exercício 2.79 no final desse capítulo) e para tal P vale $P(\{0,1\}^{\mathbb{N}}) = 1$ (por quê?) e é enumeravelmente aditiva (isso é bastante difícil de provar, veja o exercício 2.80 no final do capítulo) portanto, como vimos no parágrafo anterior, pode ser estendida para uma medida de probabilidade $\mathbb{P}$ sobre a menor σ-álgebra que contém $\mathcal{C}$.

Nesse espaço de probabilidade os pontos amostrais têm probabilidade zero. Dado $(b_0, b_1, b_2, \dots) \in \{0,1\}^{\mathbb{N}}$, definimos o evento $E_k := \{(\omega_0, \omega_1, \dots) \in \{0,1\}^{\mathbb{N}} \colon \omega_j = b_j \text{ para todo } j \leqslant k\}$ para todo natural k, logo $(b_0, b_1, b_2, \dots) = \bigcap_{k\in\mathbb{N}} E_k$ e a probabilidade do ponto amostral é o limite de $\mathbb{P}(E_k)$ quando $k \to \infty$ pela continuidade de $\mathbb{P}$. Usando a equação (2.18) essa probabilidade é $\lim_{k\to\infty}(1/2)^k = 0$. Como consequência da aditividade, eventos enumeráveis têm probabilidade 0.

Esse espaço de probabilidade que acabamos de definir é equivalente a distribuição uniforme no intervalo $[0,1]$, descrito no exemplo 1.9, página 16. Para os detalhes dessa construção e da equivalência convidamos o leitor a consultar o capítulo 1 de Billingsley (1979). ◊

Definições preliminares. No modelo que usaremos para representar computação distribuída é a computação é realizada pela execução de um conjunto de *processos* concorrentes, cada processo executa um algoritmo. Cada filósofo corresponde a um processo e seu algoritmo define as ações dos filósofos. Uma *ação atômica* é qualquer conjunto de instruções de um algoritmo distribuído executadas de modo indissociável, nenhum outro processo executa instrução enquanto uma ação atômica não termina. *Variáveis* representam os garfos e são compartilhadas, sendo cada garfo modelado por um espaço de memória acessível apenas aos processos que representam os filósofos que o compartilham; pegar e devolver um garfo são mudanças no valor de uma variável. O acesso às variáveis é uma ação atômica, um filósofo verifica se um garfo está disponível e, caso disponível, o pega sem

que seja incomodado por algum de seus vizinhos nesse ínterim. Ademais,

> *é garantido que sempre que um processo requisita o conteúdo de uma variável compartilhada, ele acabará por recebê-lo em algum momento futuro.* (†)

Um *escalonamento* é uma função que define, a partir do comportamento passado de todos os processos, o próximo processo a efetuar uma ação atômica. Conhecer o passado dos processos inclui conhecer os resultados de sorteios aleatórios passados, as memórias compartilhadas e privadas dos processos. Não há nenhum tipo de hipótese em relação às taxas de atividade de cada processo. Não está excluída a possibilidade de que o escalonamento seja malicioso e trabalhe contra a solução, fazendo o máximo possível para impedir que os filósofos se alimentem. Um escalonamento é *justo* se

$$\textit{todos os processos são ativados um número infinito de vezes} \qquad (\ddagger)$$

qualquer que sejam os resultados de sorteios aleatórios. Daqui em diante só consideramos escalonamentos justos.

Uma *solução* para o problema do jantar dos filósofos deve ser

- *distribuída*: não há um processo controlador ou uma memória central com a qual todos os outros processos possam se comunicar;
- *simétrica*: todos os processos devem executar o mesmo algoritmo e todas as variáveis têm a mesma inicialização. Além disso, os processos ignoram suas identidades.

O objetivo é encontrar um protocolo de ação que, respeitando as restrições acima, garanta que os filósofos se alimentem. Uma computação em *deadlock* é uma computação em que existe um instante t no qual um filósofo está tentando comer, mas a partir do qual nenhum filósofo come.

Não há solução determinística. Para esse problema não há uma solução que seja implementada por algoritmos distribuídos determinísticos. Suponha que exista uma solução distribuída e simétrica e vamos definir um escalonamento que impeça os filósofos de se alimentarem. Sem perda de generalidade, podemos enumerar os processos de 1 a n. Basta que o escalonamento ative cada um dos processos por uma ação atômica, ordenadamente, e repita essa ordem de ativação indefinidamente. Considerando-se que os processos se encontram inicialmente no mesmo estado, a simetria é preservada a cada rodada e é impossível que todos os filósofos estejam se alimentando simultaneamente, logo temos um escalonamento que impede que todos os filósofos se alimentem.

Solução probabilística. O que impede uma solução determinística para o problema do jantar dos filósofos é a simetria entre os processos. Para quebrar a simetria, vamos equipar os filósofos com moedas, permitindo que escolham aleatoriamente qual dos dois garfos tentarão pegar. A cada instante

t o processo ativo tem a sua disposição um bit aleatório b_t com probabilidade 1/2 de ser qualquer um dos dois valores e de modo que em instantes distintos os valores dos bits são independentes. O espaço amostral $\{0,1\}^{\mathbb{N}}$ é formado de todas as sequências binárias $\omega = (b_0, b_1, b_2, \dots)$. Esse espaço não é enumerável e usaremos o tratamento descrito acima.

Consideraremos o caso de $n \geqslant 3$ filósofos, denotados por P_i, para $1 \leqslant i \leqslant n$, mas sem que eles reconheçam qualquer identidade e dispostos na ordem (cíclica) $\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3, \dots, \mathsf{P}_n$ no sentido anti-horário. Os filósofos se comportam da maneira descrita pelo algoritmo 28, no qual representamos por 0 o garfo da esquerda, por 1 o garfo da direita e as linhas são instruções atômicas.

```
1 enquanto verdadeiro faça
2     pense;
3     ℓ ←_U {0,1};
4     se garfo ℓ disponível então pegue o garfo ℓ, senão vá para linha 4;
5     se garfo 1 − ℓ disponível então pegue o garfo 1 − ℓ e vá para linha 7;
6     devolva o garfo ℓ;
7     coma;
8     devolva um garfo;
9     devolva o outro garfo.
```

Algoritmo 28: Algoritmo dos Filósofos

Um escalonamento S e uma sequência infinita de bits $\omega = (b_i : i \in \mathbb{N})$ de $\{0,1\}^{\mathbb{N}}$ definem uma, e só uma, computação, que é uma sequência infinita de ações atômicas do algoritmo 28

$$\textsc{comp}(\mathsf{S}, \omega) := ((\alpha, \mathsf{P}, b)_t : t \in \mathbb{N})$$

em que α é a ação atômica efetuada pelo filósofo P no instante t para a qual há a disposição um bit aleatório $b = b_t \in \{0,1\}$ que pode ser usado ou não.

Para um escalonamento S fixo comp induz uma distribuição de probabilidade no espaço de todas as computações. O objetivo é demonstrar que no sistema dos filósofos com algoritmos aleatorizados a probabilidade de ocorrência *deadlock* é zero. Em particular, fixado S temos que $\omega \in \{0,1\}^{\mathbb{N}}$ define se a computação está ou não em *deadlock* de modo que $\{\omega \in \{0,1\}^{\mathbb{N}} : \textsc{comp}(\mathsf{S}, \omega) \text{ em } \textit{deadlock}\}$ é um evento aleatório pois depende de uma quantidade finita de bits iniciais de ω.

Em uma computação em *deadlock* não é possível que todos os filósofos peguem algum garfo um número finito de vezes pois, nesse caso, se os dois vizinhos de um filósofo P pegam seus garfos apenas um número finito de vezes, então para P os garfos estarão sempre disponíveis a partir de um determinado momento, logo ele pega seus garfos um número infinito de vezes já que a computação é justa, pela hipótese (‡), de modo que a partir de um determinado instante P come sempre que deseja.

Em uma computação em *deadlock* não é provável que algum filósofo pegue seus garfos apenas um número finito de vezes enquanto seu vizinho pegue seus garfos infinitas vezes. Assumamos, sem perda da generalidade, que P_2 é um filósofo que pega seus garfos apenas um número finito de vezes e P_1 um filósofo que pega seus garfos infinitas vezes. Se P_2 pega seus garfos apenas um número finito de vezes, então existe um instante t_0 a partir do qual P_2 não pega mais seus garfos. Em particular, o garfo a direita de P_1 estará disponível para P_2. Logo, para todo $t > t_0$, caso P_1 sorteie o garfo a sua esquerda ele certamente se alimentará pois o garfo da esquerda estará disponível em algum momento futuro, por (†), e o garfo da direita está sempre disponível. Se P_1 sorteia o garfo esquerdo um número finito de vezes, podemos enumerar os casos em que isso acontece, identificando cada caso pela sequência de sorteios usados por P_1 até a última vez que pega seu garfo esquerdo, ou seja, temos um evento enumerável F donde concluímos que não-F ocorre com probabilidade 1, ou seja, P_1 sorteia o garfo esquerdo infinitas vezes e, portanto, se alimenta infinitas vezes.

A partir dos dois parágrafos acima concluímos o seguinte.

Proposição 2.55 *Numa computação em* deadlock *todos os filósofos pegam algum dos seus garfos um número infinito de vezes com probabilidade* 1. □

Lembremos que em cada instante da computação um bit aleatório pode ou não ser usado pelo processo da vez. Para um instante t fixo temos um sequência formada pelos bits aleatórios que foram de fato usados por algum dos processos (no sorteio de um garfo). Chamemos essa sequência de *configuração de sorteios aleatórios* e chamemos duas configurações A e uma posterior B de *disjuntas* caso entre A e B todos os filósofos utilizaram pelo menos um sorteio.

Lema 2.56 *Numa computação em* deadlock, *se para um dado instante* t *a configuração de sorteios aleatórios já efetuados é* A, *então com probabilidade* 1 *haverá num momento futuro uma configuração* B *disjunta de* A *em que o último sorteio de algum filósofo foi o garfo esquerdo e o último sorteio de seu vizinho à direita foi o garfo direito.*

Demonstração. Numa computação em *deadlock*, todos os filósofos pegam algum dos seus garfos um número infinito de vezes com probabilidade 1 pela proposição 2.55.

Se A e B são duas configurações disjuntas e subsequentes então, no instante que ocorre B, a probabilidade com que o último sorteio de cada filósofo sejam iguais é $2(1/2)^n = 1/2^{n-1}$. Agora, se consideramos um intervalo de k configurações disjuntas subsequentes a partir de uma dada configuração, digamos $A_i, A_{i+1}, \ldots, A_{i+k}$, a probabilidade de que todos os filósofos tenham sorteado o mesmo valor em todos os respectivos últimos sorteios que antecedem imediatamente alguma configuração A_j, com $i < j \leqslant i + k$, é de $(1/2^{n-1})^k$. A probabilidade desse evento ao longo da computação é $\lim_{k\to\infty}(1/2^{n-1})^k = 0$, assim a partir de qualquer configuração A surgirá, com probabilidade 1, uma configuração disjunta B tal que, considerando o último sorteio de todos os filósofos, haverá algum

filósofo que sorteou 0 (garfo da esquerda) e seu vizinho à direita sorteou 1 (garfo da direita). □

LEMA 2.57 *Seja* F *um segmento inicial finito de uma computação composto por* t *instantes e tal que no instante* t *temos:* (*i*) *tanto* P_1 *quanto* P_2 *estão tentando comer,* (*ii*) *o último sorteio de* P_1 *foi o garfo esquerdo e o último sorteio de* P_2 *foi o garfo direito. Considere todas as computações* $C = \text{COMP}(S, \omega)$ *que sejam continuações de* F. *Nessas condições, em* C *pelo menos um dentre* P_1 *e* P_2 *se alimenta antes da próxima configuração disjunta da atual com probabilidade* 1.

DEMONSTRAÇÃO. No instante t os filósofos P_1 e P_2 estão tentando comer, P_1 sorteou 0 e P_2 sorteou 1 (estão na linha 4 do algoritmo 28), então antes do próximo sorteio cada um deles pode se encontrar em um dos seguintes estados:

1. o filósofo está esperando que o garfo sorteado seja disponibilizado, ou
2. o filósofo está em posse do garfo sorteado.

Se algum dos filósofos, dentre P_1 e P_2, está no estado 2 então um deles irá comer antes do próximo sorteio. De fato, no caso em que tanto P_1 quanto P_2 se encontram no estado 2 o próximo filósofo a ser ativado irá se alimentar antes do seu próximo sorteio pois encontrará o garfo compartilhado pelos dois disponível. No caso em que P_1 se encontra no estado 2 e P_2 no estado 1, se P_1 for o próximo dentre os dois a ser ativado, ele encontrará o garfo compartilhado disponível e comerá antes de ter feito algum sorteio; se P_2 for ativado, ele pode tanto permanecer no estado 1 e voltamos para a condição inicial, quanto progredir para o estado 2 e recaímos no caso anterior. Finalmente, o caso em que P_1 se encontra no estado 1 e P_2 no estado 2 é análogo ao anterior.

Por outro lado, no caso em que ambos os filósofos se encontram no estado 1, antes de algum sorteio de algum deles, um deverá avançar para o estado 2 e recaímos nos casos acima. □

Com as propriedades dadas nos lemas acima provaremos o resultado final desse capítulo. Seja S um escalonamento justo. Denotemos por D o evento "a computação $\text{COMP}(D, \omega)$ está em *deadlock*" e suponha que $\mathbb{P}(D) > 0$. Podemos então nos referir às probabilidades dos eventos condicionados ao *deadlock*. Pelo lema 2.56, com probabilidade 1 ocorre uma sequência infinita, digamos $A_1, A_2, \ldots, A_n, \ldots$, de configurações disjuntas de sorteios aleatórios satisfazendo as hipóteses do lema 2.57. Pelo lema 2.57, algum filósofo come entre A_n e A_{n+1}, para todo n, com probabilidade 1. Chegamos então à conclusão de que, condicionado ao evento "computação em *deadlock*", computações livres de *deadlock* têm probabilidade 1. Desta maneira, a ocorrência de deadlock deve ter probabilidade zero.

TEOREMA 2.58 *Para todo escalonamento* S *justo,* $\text{COMP}(S, \omega)$ *está em* deadlock *com probabilidade* 0. □

2.5 EXERCÍCIOS

Exercício 2.59. Um algoritmo para testar se n é da forma a^b é com segue: seja k tal que $2^{k-1} \leqslant n < 2^k$, então uma b-ésima raiz de n pertence ao intervalo $2^{\lfloor (k-1)/b \rfloor} \leqslant n^{1/b} < 2^{\lceil k/b \rceil}$, faça uma busca binária nesse intervalo. Descreva o algoritmo, verifique que usando a estratégia do algoritmo 8 a potência x^y pode ser calculada em tempo $O((y \log(x))^2)$ e conclua que testar se n é da forma a^b com a estratégia acima tem tempo de execução $O((\log n)^3)$.

Exercício 2.60. Dez dados equilibrados são lançados. Supondo que os resultados são independentes, use o princípio da decisão adiada para determinar a probabilidade da soma dos resultados ser divisível por seis.

Exercício 2.61. Um baralho comum de 52 cartas é embaralhado de modo que a disposição final é qualquer uma dentre as 52! possibilidades com igual probabilidade. Denote por E o evento "a carta do topo é de espadas", que ocorre com probabilidade 1/4. Use o princípio da decisão adiada para provar que $\mathbb{P}(F) = \mathbb{P}(E)$ para F o evento "a quarta carta a partir do topo é espada".

Exercício 2.62. Um baralho comum de 52 cartas é embaralhado de modo que a disposição final é qualquer uma as 52! possibilidades com igual probabilidade e em seguida é dividido em 13 montes de 4 cartas cada. Todo monte tem um único rótulo tomado em $\{A, 2, 3, \ldots, 9, 10, J, Q, K\}$ arbitrariamente. No primeiro movimento abrimos uma carta do monte K e o resultado indica o próximo monte donde abriremos uma carta e assim por diante seguimos. O jogo acaba quando uma jogada indica abrir a carta de um monte vazio. Use o princípio da decisão adiada para provar que a probabilidade de abrirmos todas as cartas do baralho é 1/13.

Exercício 2.63. Prove que se o laço da linha 1 no algoritmo para corte mínimo, algoritmo 10 na página 69, for executado $n^2\lceil \log(n) \rceil$ vezes, então a probabilidade do algoritmo não encontrar um corte de tamanho $\leqslant k$ é menor que $1/n$.

Exercício 2.64 (emparelhamento perfeito em grafos bipartidos). Seja $G = (A \cup B, E)$ um grafo com $|A| = |B| = n$ e todas as arestas em E tem um extremo em A e o outro em B, isto é G é um **grafo bipartido**. Defina a *matriz de adjacências* $A = (a_{i,j})$ pondo

$$a_{i,j} := \begin{cases} x_{i,j} & \text{se } \{a_i, b_j\} \in E \\ 0 & \text{caso contrário.} \end{cases}$$

Um **emparelhamento** M em G é um subconjunto de E formado por arestas não adjacentes, isto é, $e \cap d = \varnothing$ para quaisquer arestas e, $d \in M$ distintas. O emparelhamento M é dito **perfeito** se $|M| = n$.

Prove que G tem um emparelhamento perfeito se, e somente se, $\det(A) \neq 0$ (como polinômios).

Escreva um algoritmo baseado no teorema de Schwartz–Zippel que determina um emparelhamento perfeito caso exista. Analise a probabilidade de erro e o tempo de execução do algoritmo.

Exercício 2.65. Seja G um grafo bipartido e $\mathcal{F} := \{M_1, M_2, \ldots, M_k\}$ o conjunto dos emparelhamentos perfeitos de G. Tome $p\colon E(G) \to \{1, \ldots, 2|E|\}$ uma atribuição de pesos escolhidos uniformemente e independentemente para as arestas de G e defina o peso de um emparelhamento como a soma dos pesos de suas arestas. Na matriz A do exercício 2.64 tome $x_{i,j} = 2^{p(a_i,b_j)}$. Suponha, sem perda de generalidade, que M_1 é o único emparelhamento de peso mínimo. Prove que a maior potência de 2 que divide $\det(A)$ é o peso de M_1. Prove que para cada aresta $\{a_i, b_j\}$, o determinante da matriz resultante da eliminação da linha i e da coluna j de A vezes $2^{p(a_i,b_j)-p(M_1)}$ é ímpar se, e somente se, $\{a_i, b_j\} \in M_1$. Baseado no lema do isolamento (exercício 1.59), escreva um algoritmo probabilístico que ou devolve um emparelhamento perfeito ou falha. Analise seu algoritmo.

Exercício 2.66. Resolva o problema da verificação do produto de matrizes, apresentado na seção 2.2.2, usando a técnica da seção 2.2.3. Em particular, use o teorema 2.19 para provar que se sorteamos $v \in S$, para um S adequado, então $\mathbb{P}[vAB = vC] \leqslant 1/|S|$.

Exercício 2.67. Neste exercício provaremos (veja a equação (2.8))

$$\varphi(n) > \frac{n}{4\log n}. \tag{2.19}$$

Observe que se n tem k divisores primos distintos, então

$$\log(n) \geqslant k\log(2) \quad \text{e} \quad 2k\log(2) \geqslant \left(\prod_{i=1}^{k} \frac{i+1}{i}\right)\log(2).$$

Justifique tais desigualdades e verifique que se $p_1, p_2, \ldots, p_k$ são os divisores primos e distintos de n então

$$p_i \geqslant i+1 \quad \text{e} \quad \frac{i+1}{i} \geqslant \frac{p_i}{p_i - 1}.$$

Use os fatos acima para provar que

$$2\log(n) \geqslant \log(2)\frac{n}{\varphi(n)}$$

e, disso, conclua (2.19).

Exercício 2.68. Escreva um algoritmo aleatorizado que recebe um inteiro $M \geqslant 2$ e devolve uma escolha aleatória uniforme $N \in \{1, \ldots, M-1\}$ tal que $\text{mdc}(M, N) = 1$. Analise o algoritmo.

Exercício 2.69. Dados inteiros $m_1, m_2, \ldots, m_k > 1$, sejam $m = m_1 m_2 \cdots m_k$ e $m'_i = m/m_i$. Para $a \in \mathbb{Z}_n$, mostre como computar $a^{m'_1}, a^{m'_2}, \ldots a^{m'_k}$ com $O(\log(k)\log(m))$ multiplicações.

Exercício 2.70. O seguinte algoritmo é mais eficiente que o que vimos anteriormente para determinar um gerador do $\mathbb{Z}_p^*$. Suponha que são dados inteiros positivos p e $p_1, m_1, p_2, m_2, \ldots, p_k, m_k$ como nas instâncias do algoritmo 17.

1. Dado $a \in \mathbb{Z}_p^*$ mostre como computar a ordem de a em tempo $O(k \log^3 p)$ (dica: lema 2.23).
2. Use o exercício 2.69 para melhorar o tempo para $O(\log k \log^3 p)$.
3. Modifique o algoritmo do item anterior para construir um gerador do $\mathbb{Z}_p^*$ em tempo esperado $O(\log k \log^3 p)$.

Exercício 2.71 (Arvind e Mukhopadhyay (2008) e Klivans e Spielman (2001)). Prove a seguinte versão do lema do isolamento (exercício 1.59, página 48). Sejam C, ε constantes positivas e $\{\sum_i c_i x_i\}$ uma família de formas lineares distintas em que $0 \leqslant c_i \leqslant C$ são inteiros para todo i. Se cada x_i é escolhido aleatoriamente em $\{0, 1, \ldots, Cn/\varepsilon\}$, então existe uma única forma linear de valor mínimo com probabilidade $1 - \varepsilon$. Use esse resultado para dar um algoritmo probabilístico para o problema da identidade de polinômios.

Exercício 2.72. Dados naturais $a > 1$ e $p > 2$ primo que não divide $a^2 - 1$, mostre que

$$\frac{a^{2p} - 1}{a^2 - 1}$$

é pseudoprimo para a base a. Conclua que há infinitos pseudoprimos para qualquer base fixa.

Exercício 2.73 (algoritmo Las Vegas para raiz quadrada módulo p). O inteiro a é um quadrado módulo p se $x^2 \equiv a \pmod p$ tem solução. Considere o algoritmo 29 dado abaixo para achar uma raiz de um quadrado no $\mathbb{Z}_p$. Verifique que a resposta é uma raiz quadrada de a. Prove que

$$\mathop{\mathbb{P}}_{b \in \mathbb{Z}_p^*}\left[b^{\frac{p-1}{2}} \equiv -1 \pmod p\right] = \frac{1}{2}.$$

Não se conhece algoritmo determinístico eficiente para realizar a computação das linhas 3 – 5. Verifique que sem considerar as linhas 3 – 5 o algoritmo tem tempo polinomial em $\log(p)$. Determine o tempo médio de execução do algoritmo.

Exercício 2.74 (algoritmo Monte Carlo para raiz quadrada módulo p). No exercício 2.73 foi dado um algoritmo que nunca erra mas que pode executar por muito tempo. O algoritmo 30 abaixo modifica o algoritmo 29 transformando-o num algoritmo Monte Carlo, eficiente para determinar uma raiz quadrada módulo p mas que pode errar. Determine o tempo de execução do algoritmo. Prove que o algoritmo erra somente no caso em que o primeiro laço **repita**, linha 3, termina por causa do número

Instância : um primo $p > 2$ e um quadrado a módulo p.
Resposta : uma raiz quadrada de a.

se $p \equiv 3 \pmod 4$ **então responda** $a^{\frac{p-3}{4}+1} \bmod p$.
se $p \equiv 1 \pmod 4$ **então**
  repita
    $b \leftarrow_{\mathcal{U}} \{1, 2, \ldots, p-1\}$;
  até que $b^{\frac{p-1}{2}} \equiv -1 \pmod p$;
  $i \leftarrow 2 \cdot \frac{p-1}{4}$;
  $k \leftarrow 0$;
  repita
    $i \leftarrow \frac{i}{2}$;
    $k \leftarrow \frac{k}{2}$;
    se $a^i b^k \equiv -1 \pmod p$ **então** $k \leftarrow k + 2 \cdot \frac{p-1}{4}$;
  até que *i seja ímpar*
  responda $a^{\frac{i+1}{2}} b^{\frac{k}{2}} \bmod p$.

Algoritmo 29: raiz quadrada no $\mathbb{Z}_p^*$.

de rodadas t. Prove que

$$\mathbb{P}[\text{erro}] \leqslant \left(\frac{1}{2}\right)^t.$$

Exercício 2.75. Vimos que $x^2 \equiv 1 \pmod p$ tem exatamente duas soluções sempre que $p > 2$ é primo. Prove que $x^2 \equiv a \pmod p$ tem zero ou duas soluções mod p. Prove que se n é um inteiro composto e ímpar com $k > 1$ fotores primos distintos então o número de soluções de $x^2 \equiv a \pmod n$ é 0 ou 2^k (*dica*: teorema chinês do resto).

Exercício 2.76 (teste de primalidade de Pocklington, 1914). Seja $n = \mathsf{LR} + 1$, $\mathsf{L} > \mathsf{R}$ e q fator primo de L. Prove que se para todo q existe um inteiro $a > 1$ tal que $a^{n-1} \equiv 1 \pmod n$ e $\text{mdc}(a^{(n-1)/q} - 1, n) = 1$, então n é primo.

Exercício 2.77 (teste de primalidade de Proth, 1878). Uma quantidade relevante do maiores primos conhecidos são caracterizados pelo resultado descrito a seguir. Isso se deve a facilidade de implementação desse teste. Prove o seguinte: seja n um natural da forma $n = r2^s + 1$, com $2^s > r$ e r ímpar. Então n é primo se, e só se, existe um inteiro a tal que $a^{(n-1)/2} \equiv -1 \pmod n$, então n é primo. Ademais, se a não é um quadrado módulo p vale a recíproca.

Instância : um primo $p > 2$, um quadrado a módulo p e $t \in \mathbb{N}$.
Resposta : uma raiz quadrada de a.

se $p \equiv 3 \pmod 4$ **então responda** $a^{\frac{p-3}{4}+1} \mod p$.
se $p \equiv 1 \pmod 4$ **então**
  repita
    $b \leftarrow_{\mathcal{U}} \{1, 2, \ldots, p-1\}$;
  até que $b^{\frac{p-1}{2}} \equiv -1 \pmod p$ ***ou*** *complete* t *rodadas;*
  $i \leftarrow 2 \cdot \frac{p-1}{4}$;
  $k \leftarrow 0$;
  repita
    $i \leftarrow \frac{i}{2}$;
    $k \leftarrow \frac{k}{2}$;
    se $a^i b^k \equiv -1 \pmod p$ **então** $k \leftarrow k + 2 \cdot \frac{p-1}{4}$;
  até que i *seja ímpar*
  responda $a^{\frac{i+1}{2}} b^{\frac{k}{2}} \mod p$.

Algoritmo 30: raiz quadrada no $\mathbb{Z}_p^*$.

Exercício 2.78 (teste de primalidade de Micali). O teste de primalidade de Micali, algoritmo 31, testa primalidade de n e pode errar nas duas respostas. A ideia é sortear um quadrado a^2 do $\mathbb{Z}_p$ e extrair a raiz quadrada com o algoritmo Monte Carlo acima, algoritmo 30. Se n é primo, então as únicas raízes são a e $-a$; senão o algoritmo 30 computa uma raiz que pode ser diferente dessas duas.

Instância : inteiros positivos $n \geqslant 3$ e t.
Resposta : se n é primo ou composto.

1 **se** n *é par ou potência de primo* **então responda** *composto*.
2 $a \leftarrow_{\mathcal{U}} \{2, 3, \ldots, n-1\}$;
3 **se** $\mathrm{mdc}(a, n) \neq 1$ **então responda** *composto*.
4 $x \leftarrow a^2 \mod n$;
5 $y \leftarrow$ raiz quadrada determinada pelo algoritmo 30 com parâmetros (x, p, t);
6 **se** $y \neq \{-a, a\}$ *ou* $y^2 \neq x \mod n$ *ou a linha 3 do algoritmo 30 terminou por* t **então**
7 **responda** *composto*.
8 **senão responda** *primo*.

Algoritmo 31: teste de primalidade de Micali.

Determine o tempo de execução desse algoritmo. Prove que se n é primo então $\mathbb{P}[\text{erro}] \leqslant 2^{-t}$ e que se n é composto então $\mathbb{P}[\text{erro}] < 1/2$ (nesse caso o exercício 2.75 pode ser útil).

Exercício 2.79. Prove que a definição de probabilidade dada na equação (2.18), página 107, é consistente, isto é, se existem $B \subset \{0,1\}^k$ e $B' \subset \{0,1\}^{k'}$, com $k \neq k'$, tais que $A := B \times \{0,1\}^{\mathbb{N}} = B' \times \{0,1\}^{\mathbb{N}}$, então $P(A)$ definido na equação (2.18) coincide nas duas representações de A.

Exercício 2.80. Em geral, a parte difícil da aplicação do teorema de Carathéodory (descrito informalmente na página 107) é provar que a função que se quer estender é enumeravelmente aditiva. O que se faz, normalmente, é provar que a função é finitamente aditiva e contínua no sentido da seção 1.1.3. Prove que se $\mathbb{P}$ é não-negativa, finitamente aditiva e $\mathbb{P}(\Omega) = 1$ então são equivalentes

1. $\mathbb{P}$ é uma medida de probabilidade.
2. Para toda sequência decrescente $(A_n : n \geqslant 1)$ de elementos de $\mathcal{A}$ tal que $\bigcap_{n\geqslant 1} A_n = \varnothing$ vale que $\lim_{n\to\infty} \mathbb{P}(A_n) = 0$.

3 | VARIÁVEIS ALEATÓRIAS

4 | LEIS DE DESVIO E CONCENTRAÇÃO

5 | COMPUTAÇÃO PROBABILÍSTICA

problema!da fatoração FATORAÇÃO

Problema 5 (FATORAÇÃO). Dado $n \in \mathbb{N}$, existe algoritmo de tempo polinomial em $\log n$ que determina todos os divisores primos de n?

logaritmo discreto

Exemplo 5.1 (logaritmo discreto). Em geral, o grupo multiplicativo $(\mathbb{Z}_n^*, \cdot_n)$ dos resíduos invertíveis com respeito a multiplicação módulo n não é cíclico, isto é, não existe $\alpha \in \mathbb{Z}_n^*$ tal que para todo $\beta \in \mathbb{Z}_n^*$, $\beta = \alpha^k$ para algum $k \in \mathbb{N}$. Nos casos em que $\mathbb{Z}_n^*$ é cíclico[1] temos[2] um isomorfismo de grupos $\iota\colon (\mathbb{Z}_n^*, \cdot_n) \to (\mathbb{Z}_{\varphi(n)}, +_n)$, conhecido como **logaritmo discreto**, definindo $\iota(\alpha^k) = k$ para qualquer gerador α de $\mathbb{Z}_n^*$. Por exemplo, a função $\iota\colon \mathbb{Z}_{11}^* \to \mathbb{Z}_{10}$ dada na tabela 5.1 (usando só os representantes das classes dos resíduos) satisfaz $\iota(a \cdot_n b) = \iota(a) +_n \iota(b)$.

$\mathbb{Z}_{11}^*$	$\mathbb{Z}_{10}$
$1 (= 2^0)$	0
$2 (= 2^1)$	1
$3 (= 2^8)$	8
$4 (= 2^2)$	2
$5 (= 2^4)$	4
$6 (= 2^9)$	9
$7 (= 2^7)$	7
$8 (= 2^3)$	3
$9 (= 2^6)$	6
$10 (= 2^5)$	5

Tabela 5.1: função logaritmo discreto.

`geradores do Z11 no exemplo 2.26`

[1] $(\mathbb{Z}_n^*, \cdot_n)$ é cíclico se e somente se $n = 2, 4, p^t$, ou $2p^t$, onde $p > 2$ é primo e $t \geqslant 0$ inteiro (Ireland e Rosen, 1990, capítulo 4).

[2] $\varphi(n) := \left|\{a \in \mathbb{Z}\colon 0 \leqslant a < n \text{ e } \mathrm{mdc}(a, n) = 1\}\right| = n \cdot \prod_{i=1}^{k}\left(1 - \frac{1}{p_i}\right)$, onde $n = p_1^{a_1} \cdots p_k^{a_k}$, é a função de Euler.

A função $f(p, \alpha, x) = (p, \alpha, \alpha^x \bmod p)$, com p primo, α um gerador módulo p e $1 \leqslant x \leqslant p-1$ pode ser calculada eficientemente mas não se sabe inverter eficientemente, não é conhecido algoritmo eficiente que dados p, α e $\alpha^x \bmod p$, determina x. problema!do Logaritmo Discreto

> *Problema* 6 (*logaritmo discreto*). Existe um algoritmo que com entradas p primo, α um gerador módulo p e $1 \leqslant b \leqslant p-1$, compute em tempo polinomial em $\log p$ o menor inteiro positivo $0 \leqslant x \leqslant p-2$ tal que $b = \alpha^x \bmod p$?

Por exemplo, a classe do 3 é um gerador do $\mathbb{Z}_{1999}{}^*$. Sabemos computar rapidamente $3^{789} \mod 1999$ porém não sabemos resolver eficientemente a equação $3^x \equiv 1452 \pmod{1999}$.

Portanto, essa função é candidata a função unidirecional. ◊

Troca de chaves:.

6 | PASSEIOS ALEATÓRIOS

7 | DESALEATORIZAÇÃO

A | APÊNDICE

A.1 SEQUÊNCIAS E SÉRIES

(s.1) Uma série $\sum_{i\geqslant 1} x_i$ **converge** se existe o limite das somas parciais $S_n := \sum_{i=1}^{n} x_i$

$$\lim_{n\to\infty} S_n = \lim_{n\to\infty} \sum_{i=1}^{n} x_i$$

e **converge absolutamente** se $\sum_{i\geqslant 1}|x_i|$ converge. Uma prova dos seguintes resultados pode ser encontrada em Bartle, 1976.

(a) Uma série que converge absolutamente também converge.

(b) Se $\sum_n x_n$ converge e $\sum_n y_n$ é a série obtida de $\sum_n x_n$ eliminando-se os termos $x_n = 0$, então $\sum_n y_n$ converge para o mesmo valor.

(c) Se uma série converge absolutamente então qualquer série obtida rearranjando os termos dessa série converge absolutamente para o mesmo valor. Ademais, vale que se $\{A_i : i \in I\}$ é uma partição finita ou enumerável de $\mathbb{N}$ e $y_i = \sum_{j\in A_i} x_j$, então

$$\sum_n x_n = \sum_{i\in I} y_i.$$

(s.2) Se $\sum_{i\geqslant 1} x_i$ é tal que $x_i \geqslant 0$ para todo i, então a série converge ou tende ao infinito. A série converge se, e só se, converge absolutamente.

(s.3) Teste de convergência por comparação: se $0 \leqslant x_n \leqslant y_n$ para todo n suficientemente grande e $\sum_n y_n$ converge então $\sum_n x_n$ converge.

(s.4) a convergência absoluta de uma série implica que toda subsérie da série é convergente. Isso segue de (s.1b) acima e do teste de convergência por comparação.

(s.5) Seja $(x_{n,m} : n, m \in \mathbb{N})$ uma sequência duplamente indexada tal que $\sum_{(n,m)}|x_{n,m}|$ converge. Então

$$\sum_{n\in\mathbb{N}}\sum_{m\in\mathbb{N}} x_{n,m} = \sum_{(n,m)\in\mathbb{N}\times\mathbb{N}} x_{n,m} = \sum_{(m,n)\in\mathbb{N}\times\mathbb{N}} x_{n,m} = \sum_{m\in\mathbb{N}}\sum_{n\in\mathbb{N}} x_{n,m}$$

(s.6) se $|x| < 1$ então

(a) $\sum_{n\geqslant 0} x^n = (1-x)^{-1}$;

(b) $\sum_{n\geqslant k} x^n = x^k(1-x)^{-1}$;

(c) $\sum_{n\geqslant 1} nx^n = x(1-x)^{-2}$.

(s.7) $\sum_{n\geqslant 1} n^{-2} = \pi^2/6$.

A convergência dessa série foi estabelecida pela primeira vez por Euler e é um dos resultados responsáveis por lançar à notoriedade, na epóca, o matemático.

(s.8) A sequência $(1+1/n)^n$ é estritamente crescente e a sequência $(1-1/n)^{-n}$ é estritamente decrescente, ambas convergem para a constante $e = 2{,}71\ldots$ quando $n \to \infty$. Além disso, para todo x real

$$e^x = \sum_{n\geqslant 0} \frac{x^n}{n!}.$$

A.2 DESIGUALDADES

(d.1) De (s.8), para todo $n \geqslant 0$ vale

$$\left(1-\frac{1}{n}\right)^{-n} \geqslant e \geqslant \left(1+\frac{1}{n}\right)^n$$

e para qualquer $x > 0$

$$1 - x + \frac{x^2}{2} > e^{-x} > (1-x).$$

(d.2) O coeficiente binomial tem os seguintes limitantes triviais

$$0 \leqslant \binom{n}{i} \leqslant 2^n.$$

Para um limitante inferior melhor podemos escrever

$$\binom{n}{i} = \prod_{j=0}^{i-1} \frac{n-j}{j-i} \geqslant \left(\frac{n}{i}\right)^i$$

e para um limitante superior usamos o teorema binômial de Newton e (s.8)

$$e^{nx} \geqslant (1+x)^n = \sum_{i=0}^{n} \binom{n}{i} x^i \geqslant \binom{n}{i} x^i$$

para todo $x > 0$ e todo $i \in \{0, 1, \ldots, n\}$. Logo

$$\binom{n}{i} \leqslant e^{nx} x^{-i}.$$

Em particular, fazendo $x = i/n$, temos a segunda desigualdade de

$$\left(\frac{n}{i}\right)^i \leqslant \binom{n}{i} \leqslant \left(\mathrm{e}\frac{n}{i}\right)^i.$$

(d.3) A seguinte igualdade assintótica conhecida como fórmula de Stirling

$$n! = \left(1 + \mathrm{O}\left(\frac{1}{n}\right)\right)\sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n = (1 + o(1))\sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n$$

A seguinte tabela que exibe exemplos para números pequenos nos dá ideia da qualidade dessa aproximação

n	$n!$	Stirling
1	1	0.922137
3	6	5.83621
7	5040	4980.396
10	3628800	3598696
20	$2.432902e+18$	$2.422787e+18$
50	$3.041409e+64$	$3.036345e+64$
75	$2.480914e+109$	$2.478159e+109$
100	$9.332622e+157$	$9.324848e+157$
142	$2.695364e+245$	$2.693783e+245$

também valem os limitantes

$$\sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n \leqslant n! \leqslant \mathrm{e}\sqrt{n}\left(\frac{n}{\mathrm{e}}\right)^n$$

Nos limitantes em (d.2) para $\binom{n}{i}$ usamos que

$$\prod_{j=0}^{i-1}(n-j) > (n-i)^i = \left(1 - \frac{i}{n}\right)^i n^i \geqslant \left(1 - \frac{i^2}{n}\right)n^i = (1 + o(1))n^i$$

e se $i = o(\sqrt{n})$

$$\binom{n}{i} = (1 + o(1))\frac{n^i}{i!} = (1 + o(1))\frac{1}{\sqrt{2\pi i}}\left(\frac{\mathrm{e}n}{i}\right)^i.$$

(d.4) Para quaisquer números reais $x_1, x_2, \ldots$, para todo natural n

$$|x_1 + \cdots + x_n| \leqslant |x_1| + \cdots + |x_n|.$$

Ademais, $|x_1| + \cdots + |x_n| \leqslant |x_1| + |x_2| + \cdots$, portanto, $|x_1 + \cdots + x_n| \leqslant |x_1| + |x_2| + \cdots$ e pela continuidade da função valor absoluto $\lim_{n\to\infty}|x_1 + \cdots + x_n| = |x_1 + x_2 + \cdots|$ portanto

$$|x_1 + x_2 + \cdots| \leqslant |x_1| + |x_2| + \cdots$$

sempre que $\sum_n x_n$ converge.

A.3 ARITMÉTICA MODULAR

Invertíveis módulo n..

A.3.1 O ANEL DOS INTEIROS MÓDULO N

A.4 SOLUÇÕES DE EQUAÇÕES MÓDULO N.

Teorema A.1 (teorema chinês do resto[1]) *Sejam $n_1, n_2, \ldots, n_k$ inteiros maiores que 1 e tais que $mdc(n_i, n_j) = 1$ para todos $i \neq j$, e sejam $c_1, \ldots, c_k$ inteiros arbitrários. Então o sistema*

$$x \equiv c_i \pmod{n_i}, \quad \text{para todo } i \in \{1, 2, \ldots, k\}.$$

tem uma, e só uma, solução módulo $n = n_1 n_2 \cdots n_k$.

homomorfismo de grupos Sejam $(X, \circ)$ e $(Y, \times)$ grupos. Uma função $f\colon X \to Y$ é um **homomorfismo** se para quaisquer $a, b \in X$

$$f(a \circ b) = f(a) \times f(b).$$

Se, além disso, f é bijetora então f é um **isomorfismo**.

Teorema A.2 (Teorema do isomorfismo) *Sejam $(X, \circ)$ e $(Y, \times)$ grupos com identidades denotadas por 1_X e 1_Y, respectivamente, e f um homomorfismo. Então*

1. $\mathrm{Im}(f) = \{f(x) \in Y\colon x \in X\}$ *é subgrupo de* Y;
2. $\ker(f) = \{x \in X\colon f(x) = 1_Y\}$ *é um subgrupo de* X;
3. $X/\ker(f)$ *é isomorfo a* $\mathrm{Im}(f)$.

A.5 TEOREMA ESPECTRAL PARA MATRIZES REAIS

A.5.1 TEOREMA DE PERRON–FROBENIUS PARA MATRIZES SIMÉTRICAS NÃO-NEGATIVAS

Teorema A.3 (Teorema de Perron–Frobenius) *Seja A uma matriz simétrica, não-negativa, irredutível e com autovalores $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_n$. Então*

1. $\lambda_1 > 0$ *e associado a esse autovalor existe um autovetor positivo;*

[1] A forma original do teorema apareceu no livro *Sun Tzu Suan Ching* (manual de aritmética de Sun Tzu) do terceiro-século e republicado em 1247 por Qin Jiushao.

2. $\lambda_1 > \lambda_2$;

3. $|\lambda_i| \leqslant \lambda_1$ *para todo* $i \in [n]$;

4. $\lambda_1 = -\lambda_n$ *se, e só se, existe* ρ *tal que* $A_\rho = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}$ *com as matrizes nulas sendo quadradas.*

Bibliografia

Agrawal, Manindra e Somenath Biswas (2003). "Primality and identity testing via Chinese remaindering". Em: *J. ACM* 50.4, 429–443 (electronic) (ver pp. 7, 99, 100).

Agrawal, Manindra, Neeraj Kayal e Nitin Saxena (2004). "PRIMES is in P". Em: *Ann. of Math. (2)* 160.2, pp. 781–793 (ver pp. 5, 7, 88).

Arvind, V. e Partha Mukhopadhyay (2008). "Derandomizing the Isolation Lemma and Lower Bounds for Circuit Size". Em: *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*. Ed. por Ashish Goel et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 276–289 (ver p. 114).

Bach, Eric e Jeffrey Shallit (1996). *Algorithmic Number Theory*. Cambridge, MA, USA: MIT Press (ver pp. 79, 81).

Bartle, Robert G. (1976). *The elements of real analysis*. Second. John Wiley & Sons, New York-London-Sydney, pp. xv+480 (ver p. 124).

Ben-Or, Michael (1981). "Probabilistic Algorithms in Finite Fields". Em: *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*. IEEE Computer Society, pp. 394–398. DOI: 10.1109/SFCS.1981.37 (ver p. 87).

Bernstein, Daniel J. (1998). "Detecting perfect powers in essentially linear time". Em: *Math. Comput.* 67.223, pp. 1253–1283. DOI: http://dx.doi.org/10.1090/S0025-5718-98-00952-1 (ver p. 101).

Billingsley, P. (1979). *Probability and measure*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley (ver p. 107).

DeMillo, Richard A. e Richard J. Lipton (1978). "A Probabilistic Remark on Algebraic Program Testing". Em: *Inf. Process. Lett.* 7.4, pp. 193–195 (ver p. 75).

Diffie, Whitfield e Martin E. Hellman (1976). "New directions in cryptography". Em: *IEEE Trans. Information Theory* IT-22.6, pp. 644–654 (ver p. 77).

Erdős, P. (1956). "On pseudoprimes and Carmichael numbers". Em: *Publ. Math. Debrecen* 4, pp. 201–206 (ver p. 91).

Freivalds, Rusins (1977). "Probabilistic Machines Can Use Less Running Time". Em: *IFIP Congress*, pp. 839–842 (ver p. 70).

Gao, Shuhong e Daniel Panario (1997). "Tests and Constructions of Irreducible Polynomials over Finite Fields". Em: *Foundations of Computational Mathematics*. Ed. por Felipe Cucker e Michael Shub. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 346–361 (ver p. 87).

Garfinkel, Simson (1994). *PGP: Pretty Good Privacy*. O'Reilly (ver p. 91).

Gathen, Joachim von zur e Jürgen Gerhard (2013). *Modern Computer Algebra*. 3ª ed. Cambridge University Press. DOI: 10.1017/CBO9781139856065 (ver p. 82).

Gelbaum, B.R. e J.M.H. Olmsted (1964). *Counterexamples in Analysis*. Dover books on mathematics. Holden-Day (ver p. 16).

Graham, Paul (2002). *A Plan for Spam*. http://www.paulgraham.com/spam.html. Acesso em 06/04/2009 (ver p. 35).

Graham, Ronald L., Donald E. Knuth e Oren Patashnik (1994). *Concrete mathematics*. Second. A foundation for computer science. Reading, MA: Addison-Wesley Publishing Company, pp. xiv+657 (ver p. 103).

Harman, Glyn (2005). "On the Number of Carmichael Numbers up to x". Em: *Bulletin of the London Mathematical Society* 37.5, pp. 641–650. DOI: 10.1112/S0024609305004686. eprint: https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/S0024609305004686 (ver p. 91).

Harvey, David e Joris Van Der Hoeven (mar. de 2019). "Integer multiplication in time $O(n \log n)$". working paper or preprint (ver p. 63).

Ireland, Kenneth e Michael Rosen (1990). *A classical introduction to modern number theory*. Second. Vol. 84. Graduate Texts in Mathematics. New York: Springer-Verlag, pp. xiv+389 (ver p. 120).

Kabanets, Valentine e Russell Impagliazzo (2004). "Derandomizing polynomial identity tests means proving circuit lower bounds". Em: *Comput. Complexity* 13.1-2, pp. 1–46 (ver p. 6).

Karger, David R. (1993). "Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm". Em: *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (Austin, TX, 1993)*. New York: ACM, pp. 21–30 (ver p. 69).

Kimbrel, Tracy e Rakesh Kumar Sinha (1993). "A probabilistic algorithm for verifying matrix products using $O(n^2)$ time and $\log_2 n + O(1)$ random bits". Em: *Inform. Process. Lett.* 45.2, pp. 107–110. DOI: 10.1016/0020-0190(93)90224-W (ver p. 72).

Klivans, Adam R. e Daniel A. Spielman (2001). "Randomness efficient identity testing of multivariate polynomials". Em: *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pp. 216–223. DOI: 10.1145/380752.380801 (ver p. 114).

Knuth, Donald E. (1981). *The art of computer programming. Vol. 2*. Second. Seminumerical algorithms, Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley Publishing Co., Reading, Mass., pp. xiii+688 (ver p. 70).

Lehmann, Daniel e Michael O. Rabin (1981). "On the advantages of free choice: a symmetric and fully distributed solution to the dining philosophers problem". Em: *POPL '81: Proceedings of the*

8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. Williamsburg, Virginia: ACM, pp. 133–138. DOI: http://doi.acm.org/10.1145/567532.567547 (ver p. 105).

Lenstra, H. W. (2002). "Primality Testing with Gaussian Periods". Em: *FST TCS 2002: Foundations of Software Technology and Theoretical Computer Science*. Ed. por Manindra Agrawal e Anil Seth. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–1 (ver p. 88).

Lidl, Rudolf e Harald Niederreiter (1997). *Finite fields*. Second. Vol. 20. Encyclopedia of Mathematics and its Applications. With a foreword by P. M. Cohn. Cambridge: Cambridge University Press, pp. xiv+755 (ver pp. 82, 85).

Menezes, Alfred J., Paul C. van Oorschot e Scott A. Vanstone (1997). *Handbook of applied cryptography*. CRC Press Series on Discrete Mathematics and its Applications. With a foreword by Ronald L. Rivest. Boca Raton, FL: CRC Press, pp. xxviii+780 (ver p. 105).

Miller, Gary L. (1975). "Riemann's hypothesis and tests for primality". Em: *Seventh Annual ACM Symposium on Theory of Computing (Albuquerque, N.M., 1975)*. Assoc. Comput. Mach., New York, pp. 234–239 (ver p. 93).

Mitzenmacher, Michael e Eli Upfal (2005). *Probability and computing*. Randomized algorithms and probabilistic analysis. Cambridge: Cambridge University Press, pp. xvi+352 (ver p. 71).

Monier, Louis (1980). "Evaluation and comparison of two efficient probabilistic primality testing algorithms". Em: *Theoretical Computer Science* 12.1, pp. 97 –108. DOI: https://doi.org/10.1016/0304-3975(80)90007-9 (ver p. 88).

Mulmuley, Ketan, Umesh V. Vazirani e Vijay V. Vazirani (1987). "Matching is as easy as matrix inversion". Em: *Combinatorica* 7.1, pp. 105–113 (ver p. 48).

Nightingale, Edmund B., John R. Douceur e Vince Orgovan (2011). "Cycles, Cells and Platters: An Empirical Analysisof Hardware Failures on a Million Consumer PCs". Em: *Proceedings of the Sixth Conference on Computer Systems*. EuroSys '11. Salzburg, Austria: ACM, pp. 343–356. DOI: 10.1145/1966445.1966477 (ver p. 5).

O'Gorman, T. J. et al. (jan. de 1996). "Field Testing for Cosmic Ray Soft Errors in Semiconductor Memories". Em: *IBM J. Res. Dev.* 40.1, pp. 41–50. DOI: 10.1147/rd.401.0041 (ver p. 5).

Pak, Igor (2012). "Testing commutativity of a group and the power of randomization". Em: *LMS Journal of Computation and Mathematics* 15, 38–43. DOI: 10.1112/S1461157012000046 (ver p. 6).

Rabin, Michael O. (1980a). "Probabilistic algorithm for testing primality". Em: *J. Number Theory* 12.1, pp. 128–138 (ver p. 93).

— (1980b). "Probabilistic algorithms in finite fields". Em: *SIAM J. Comput.* 9.2, pp. 273–280 (ver p. 86).

Ribenboim, Paulo (1996). *The new book of prime number records*. New York: Springer-Verlag, pp. xxiv+541 (ver p. 104).

Rosenthal, Jeffrey S. (2006). *A first look at rigorous probability theory*. Second. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, pp. xvi+219 (ver p. 16).

Ross, Sheldon M. (2010). *A first course in Probability*. 8th. New Jersey: Prentice Hall (ver p. 31).

Saldanha, Nicolau (1997). "Precisa-se de alguém para ganhar muito dinheiro". Disponível em http://www.mat.puc-rio.br/~nicolau/publ/papers/otario.pdf. Acesso em 07/2018 (ver p. 45).

Schroeder, Bianca, Eduardo Pinheiro e Wolf-Dietrich Weber (2009). "DRAM Errors in the Wild: A Large-scale Field Study". Em: *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '09. Seattle, WA, USA: ACM, pp. 193–204. DOI: 10.1145/1555349.1555372 (ver p. 5).

Schwartz, Jacob T. (1979). "Probabilistic algorithms for verification of polynomial identities". Em: *Symbolic and algebraic computation (EUROSAM '79, Internat. Sympos., Marseille, 1979)*. Vol. 72. Lecture Notes in Comput. Sci. Berlin: Springer, pp. 200–215 (ver p. 75).

Shpilka, Amir e Amir Yehudayoff (2010). "Arithmetic Circuits: A Survey of Recent Results and Open Questions". Em: *Foundations and Trends® in Theoretical Computer Science* 5.3–4, pp. 207–388. DOI: 10.1561/0400000039 (ver p. 74).

Zippel, Richard (1979). "Probabilistic algorithms for sparse polynomials". Em: *Symbolic and algebraic computation (EUROSAM '79, Internat. Sympos., Marseille, 1979)*. Vol. 72. Lecture Notes in Comput. Sci. Berlin: Springer, pp. 216–226 (ver p. 75).

Lista de Algoritmos

Índice remissivo

Índice de símbolos

probabilidade

```
$\P$  --- $\P[Q]$ --- $\prob A$ --- $\probE A$  --- $\prob[Q]A$ --- $\prob[Q]{\frac AB}$ -
--
  $\prob[P_{\text{$1$}}]{\omega_1}$
```

$\mathbb{P} — \mathbb{Q} — \mathbb{P}(A) — \mathbb{P}[A] — \mathbb{Q}(A) — \mathbb{Q}(\frac{A}{B}) — \mathbb{P}_1(\omega_1)$

```
 $\Prob A$ --- $\ProbE A$  --- $\Prob[Q]A$ --- $\Prob[Q]{\frac AB}$
```

$\mathbb{P}(A) — \mathbb{P}[A] — \mathbb{Q}(A) — \mathbb{Q}\left(\frac{A}{B}\right)$

condicional

```
$\probC{A}{\frac BD}$ --- $\probC[Q]{A}{BD}$ --- $\probCE{A}{\frac BD}$ --- $\probCE[Q]{A}
```

$\mathbb{P}(A \mid \frac{B}{D}) — \mathbb{Q}(A \mid BD) — \mathbb{P}[A \mid \frac{B}{D}] — \mathbb{Q}[A \mid BD]$

```
$\ProbC{A}{\frac BD}$ --- $\ProbC[Q]{A}{BD}$ --- $\ProbCE{A}{\frac BD}$ --- $\ProbCE[Q]{A}
```

$\mathbb{P}\left(A \,\middle|\, \frac{B}{D}\right) — \mathbb{Q}(A \mid BD) — \mathbb{P}\left[A \,\middle|\, \frac{B}{D}\right] — \mathbb{Q}[A \mid BD]$

limit style

```
$\probX A{\frac CD}$ --- $\probXE AB$ ---  $\probCX ABC$ --- $\probCXE [Q]AB{\frac CD}$ -
--
 $\probX S {\omega_1}$ ---  $\displaystyle \probX S {\omega_1}$
```

$\mathbb{P}_A(\frac{C}{D}) — \mathbb{P}_A[B] — \mathbb{P}_A(B \mid C) — \mathbb{Q}_A[B \mid \frac{C}{D}] — \mathbb{P}_S(\omega_1) — \mathbb{P}_S(\omega_1)$

```
$\ProbX A{\frac CD}$ --- $\ProbXE AB$ ---  $\ProbCX ABC$ --- $\ProbCXE [Q]AB{\frac CD}$
```

$\mathbb{P}_A\left(\frac{C}{D}\right) — \mathbb{P}_A[B] — \mathbb{P}_A(B \mid C) — \mathbb{Q}_A\left[B \,\middle|\, \frac{C}{D}\right]$

---

indicador de evento

```
\DeclareRobustCommand{\1}[1]{\ensuremath \mathbf{1}_{#1}}
=> \1 X
```

$\mathbb{1}_X$

esperança

```
\DeclareRobustCommand{\E}[2][{\mathbbmss{E}}]{\ensuremath {#1}\colch*{#2} }
\DeclareRobustCommand{\EC}[3][{\mathbbmss{E}}]{\ensuremath {#1}\evec* {#2}{#3} }
=> \E X --- \EC XY
```

$\mathbb{E}X - \mathbb{E}[X|Y]$

variancia

```
\DeclareRobustCommand{\var}[1][\protect\operatorname{Var}]{\ensuremath #1 }
=> \var X
```

$\mathbb{V}[X]$

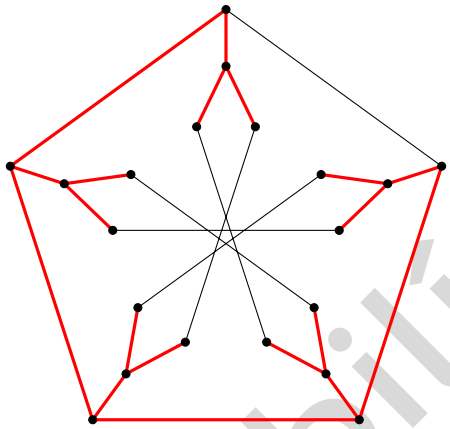


Figura A.1: exemplo de uma árvore geradora. As arestas pretas e vermelhas fazem parte do grafo, só as arestas vermelhas formam , junto com todos os vértices, um árvore geradora do grafo.