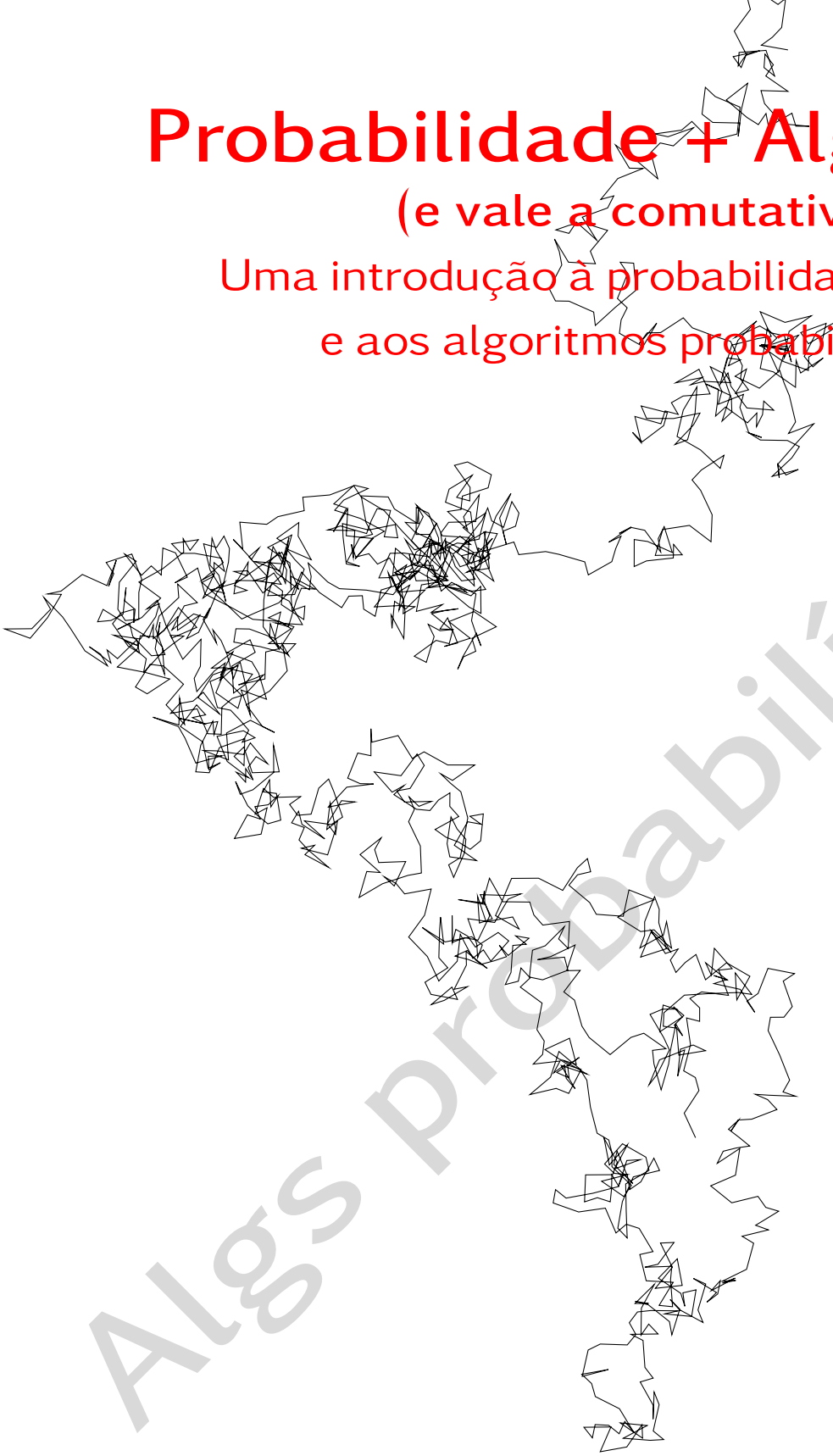
Probabilidade + Algoritmos

(e vale a comutativa)

Uma introdução à probabilidade discreta
e aos algoritmos probabilísticos



Algs probabilísticos

— J Donadelli —

última modificação 25/5/2022

A capa é um passeio aleatório. De verdade!

```
\AddToShipoutPicture*{
  \put(20,150)
  {
    \RandomWalk {
                 number = 1500,
                 length = {6pt, 12pt}}
                }
  }
}
```

compilações diferentes do LaTeX resultarão em capas diferentes (com probabilidade 0,999).

[*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

1 | ESPAÇOS DE PROBABILIDADE

Notas de aula por J. Donadelli (cmcc–ufabc)

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 e escolhendo ao acaso se houver mais de uma possibilidade. 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, $\Omega = \{\mathrm{Ca},\mathrm{Co}\}$;
3. uma moeda é lançada até sair coroa, $\Omega = \{(\mathrm{Co}),(\mathrm{Ca},\mathrm{Co}),\ldots,(\mathrm{Ca},\mathrm{Ca},\ldots,\mathrm{Ca},\mathrm{Co}),\ldots,(\mathrm{Ca},\mathrm{Ca},\ldots)\}$, 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 $(a_i \in \{\mathrm{Ca},\mathrm{Co}\}\colon\ i \geqslant 1)$ de resultados possíveis, isto é, $\Omega = \{\mathrm{Ca},\mathrm{Co}\}^{\mathbb{N}}$;
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 $\Omega = \{(x,y) \in \mathbb{R}^2\colon\ x^2 + y^2 \leqslant 1\}$.

[1] Família é usado como sinônimo de conjunto.

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, omitimos a referência ao espaço amostral e usamos E para denotar E_Ω.

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 (veja a tabela 1.1 abaixo).

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: "dicionário" de termos da Probabilidade.

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 12. Esse fenômeno só é importante quando Ω é muito grande (não enumerável).

Exercício 1.3. Descreva, de acordo com 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*: $\mathbb{P}\big(\bigcup_{i\geqslant 1} A_i\big) = \sum_{i\geqslant 1} \mathbb{P}(A_i)$ 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*
$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \mathbb{P}(A_i).$$
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
$$\mathbb{P}(\Omega) = \mathbb{P}(\Omega \cup \varnothing \cup \varnothing \cup \cdots \cup \varnothing \cup \cdots) = \mathbb{P}(\Omega) + \sum_{i\geqslant 2} \mathbb{P}(\varnothing)$$

[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.

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 esse não 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

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

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, $\mathbb{P}(\mathsf{A})$ 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,

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

para qualquer conjunto $\{\mathsf{A}_i : i \geqslant 1\}$ 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 1/63 (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 $1/3 + 1/7 - 1/63 = 29/63 \approx 0{,}46$. ◊

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 $\mathbb{P}(\mathsf{A})$ esteja definida para todo $\mathsf{A}\subset\Omega$, 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 a possibilidade de mais de uma interpretação para "ao acaso", ou "aleatório", que leva a resultados diferentes.

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

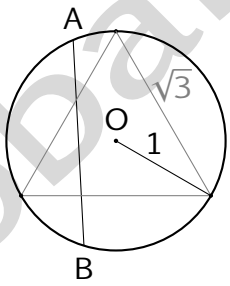
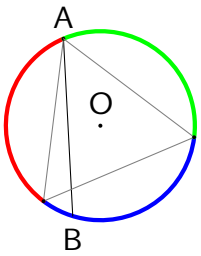


Figura 1.1: paradoxo de Bertrand.

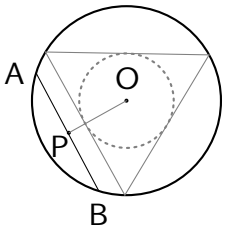
escolha da corda se dá ao tomarmos A e B escolhidos ao acaso dentre os pontos da circunferência. Consideremos o triângulo rotacionado de modo que um de seus vértices coincida com o ponto A. A corda tem comprimento maior que o lado do triângulo se B está no arco da circunferência entre os dois outros 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.

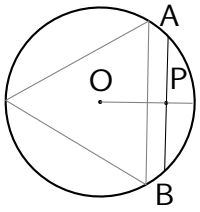
Na terceira é última interpretação para uma corda aleatória nós fixamos um raio. A corda é obtida escolhendo um ponto P no raio e tomando a corda que passa por P e perpendicular ao raio (figura 1.2(c)). A corda é maior do que um lado do triângulo, se o ponto escolhido está mais próximo do



(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.

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. O primeiro caso ($\mathcal{A} = 2^\Omega$) sempre vale em 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. A explicação desse fenômeno é muito técnica para ser dada aqui, as ferramentas necessárias vão além do escopo deste texto.

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, 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). 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.

Convencionamos a notação

$$\mathbb{P}(\omega) := \mathbb{P}(\{\omega\})$$

para os eventos elementares.

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 11, deduzimos igualmente que Ω e p definem um modelo probabilístico discreto para o experimento. ◊

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 Ω 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

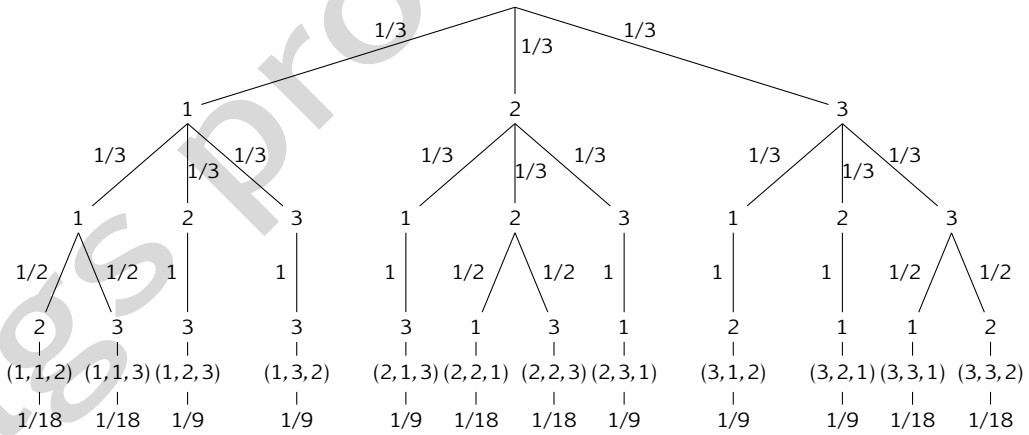


Figura 1.3: diagrama de árvore de um modelo para Monty Hall.

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 respeita 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}. Pensando ainda 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

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

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

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

[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.

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

Uma medida de probabilidade nesse caso pode ser definida do seguinte modo. Para cada natural n e cada sequência $(c_1, \ldots, c_n)$ de caras e coroas tomamos os conjuntos cilíndricos dados por essa sequência $\{(x_1, x_2, \ldots) \in \Omega\colon (x_1, \ldots, x_n) = (c_1, \ldots, c_n)\}$. O espaço de eventos $\mathcal{A}$ é a interseção de todas as σ-álgebras de Ω que contêm os cilindros. Se a probabilidade de um cilindro é a probabilidade da sequência de caras e coroas que o define, então um famoso teorema devido a Komolgorov garante que essa probabilidade pode ser estendida de modo único a todo $\mathcal{A}$. ◊

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 $(A_n\colon n \geqslant 1)$ 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 11, 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 $(B_n\colon n \geqslant 1)$ 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 descrito por 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 $\mathbb{P}(\{(c_1,\ldots,c_i)\colon i \text{ é par}\})$. 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

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

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

$$a \xleftarrow{R} \{0,1\}$$

para denotar o fato de que a é uma variável do algoritmo e que após a execução da atribuição $\xleftarrow{R}$ 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

$$a \xleftarrow{R} \Omega$$

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 $\mathsf{x} = x_1 x_2 \ldots x_\ell$ é codificada como $E_e(\mathsf{x}) = y_1 y_2 \ldots y_\ell$ com $y_i = (x_i + e) \bmod 26$, para todo i, e é decodificada como $D_e(\mathsf{x}) = y_1 y_2 \ldots y_\ell$ 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

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

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

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

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

Algoritmo 1: 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 $|E| \leqslant \text{grau}(f) \leqslant d$ pois f tem no máximo $\text{grau}(f)$ 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 1 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 uma 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 ω em Q_Ω 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 de ocorrência do evento A condicionada a ocorrência o evento E, ou como dizemos a **probabilidade condicional** de A dado 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,

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

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 9). 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 desse teorema é dado no exercício 1.24 abaixo.

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 $\mathbb{P}(T \mid B)$. Notemos que $\mathbb{P}(B \mid T)$ corresponde a sortear no segundo experimento

uma meia branca na terceira gaveta, o que ocorre com probabilidade 1/2. Usando a regra da multiplicação duas vezes escrevemos $\mathbb{P}(\,T \mid B\,) = \mathbb{P}(T \cap B)\mathbb{P}(B) = \mathbb{P}(\,B \mid T\,)\mathbb{P}(T)\mathbb{P}(B)$. Ainda, $\mathbb{P}(T)$ corresponde a probabilidade de sortear a terceira gaveta no primeiro experimento, o que ocorre com probabilidade 1/3. Logo, $\mathbb{P}(\,T \mid B\,) = 1/3$. ◊

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 definida 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. 15). O diagrama de árvore da figura 1.4 representa cada

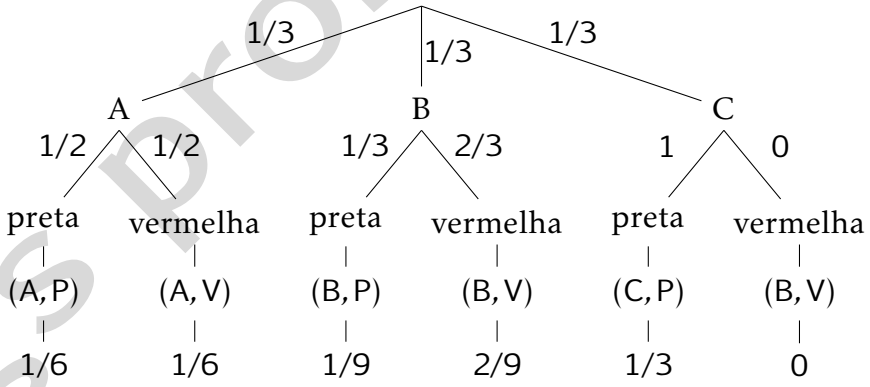


Figura 1.4: diagrama de árvore.

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 pragmática de calcular a probabilidade dada pela regra da multiplicação 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. ◊

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

$$\mathbb{P}(C \cap A \mid B) = \mathbb{P}(C \mid A \cap B)\mathbb{P}(A \mid B) \tag{1.12}$$

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,

$$\begin{aligned} \mathbb{P}(A) &= \mathbb{P}\Big((A \cap E) \cup (A \cap \overline{E})\Big) \\ &= \mathbb{P}(A \cap E) + \mathbb{P}(A \cap \overline{E}) \\ &= \mathbb{P}(A \mid E)\mathbb{P}(E) + \mathbb{P}(A \mid \overline{E})\mathbb{P}(\overline{E}). \end{aligned} \tag{1.13}$$

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 (1.13)

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

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

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

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* $\{E_i : i \in \mathbb{N}\}$ *uma partição do espaço amostral* Ω *com* $\mathbb{P}(E_i) > 0$ *para todo* i. *Então*

$$\mathbb{P}(A) = \sum_{i \geqslant 1} \mathbb{P}(A \cap E_i) = \sum_{i \geqslant 1} \mathbb{P}(A \mid E_i)\mathbb{P}(E_i) \tag{1.15}$$

para qualquer evento A.

Demonstração. Os conjuntos $A \cap E_i$, para $i \in \mathbb{N}$, são disjuntos dois a dois e da união deles resulta A. A primeira igualdade em (1.15) 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 $\mathbb{P}(\,A^{+} \mid \overline{A}_1\,) \approx 0{,}24$. $\Diamond$

A URNA DE PÓLYA Uma urna contém duas bolas, uma branca e uma preta. Em cada instante $t \in \{1,2,\ldots\}$ 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 (1.13)) $\mathbb{P}(P_2) = \mathbb{P}(\,P_2 \mid P_1\,)\mathbb{P}(P_1) + \mathbb{P}(\,P_2 \mid B_1\,)\mathbb{P}(B_1)$ 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 (1.14), 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 26), 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

$$\begin{aligned}\mathbb{P}(P_3) &= \mathbb{P}((P_1 \cap P_2) \cap P_3) + \mathbb{P}((P_1 \cap B_2) \cap P_3) + \mathbb{P}((B_1 \cap P_2) \cap P_3) + \mathbb{P}((B_1 \cap B_2) \cap P_3) \\ &= \frac{1}{2}\cdot\frac{2}{3}\cdot\frac{3}{4} + \frac{1}{2}\cdot\frac{1}{3}\cdot\frac{2}{4} + \frac{1}{2}\cdot\frac{1}{3}\cdot\frac{2}{4} + \frac{1}{2}\cdot\frac{2}{3}\cdot\frac{1}{4} = \frac{1}{2}.\end{aligned} \tag{1.16}$$

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 (1.16) 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 (1.14) é $\mathbb{P}(\mathrm{P}_1)$ e, então, $\mathbb{P}(\mathrm{P}_1) = \mathbb{P}(\mathrm{P}_3)$. Tal simetria vale para qualquer t de modo que $\mathbb{P}(\mathrm{P}_t) = \mathbb{P}(\mathrm{P}_1)$ para todo $t \geqslant 1$. Vamos demonstrar esse fato.

Consideremos uma sequência de eventos $\mathrm{E}_1, \mathrm{E}_2, \ldots, \mathrm{E}_{t-1}, \mathrm{P}_t$ em que para cada i, $1 \leqslant i < t$, temos $\mathrm{E}_t \in \{\mathrm{P}_t, \mathrm{B}_t\}$. As 2^{t-1} possíveis sequências de eventos $\mathrm{E}_1, \ldots, \mathrm{E}_{t-1}$ 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 26), a probabilidade de P_t é

$$\sum_{(\mathrm{E}_1,\ldots,\mathrm{E}_{t-1})} \mathbb{P}(\mathrm{E}_1 \cap \mathrm{E}_2 \cap \cdots \cap \mathrm{E}_{t-1} \cap \mathrm{P}_t) = \sum_{(\mathrm{E}_1,\ldots,\mathrm{E}_{t-1})} \mathbb{P}(\mathrm{E}_1) \cdot \left(\prod_{i=2}^{t-1} \mathbb{P}\left(\mathrm{E}_i \,\middle|\, \bigcap_{j=1}^{i-1} \mathrm{E}_j \right) \right) \cdot \mathbb{P}\left(\mathrm{P}_t \,\middle|\, \bigcap_{j=1}^{t-1} \mathrm{E}_j \right) \tag{1.17}$$

em que a soma é sobre todas as 2^{t-1} sequências de eventos. Os somandos no lado direito da equação (1.17) são

$$\mathbb{P}(\mathrm{E}_1)\mathbb{P}(\mathrm{E}_2 \mid \mathrm{E}_1)\mathbb{P}(\mathrm{E}_3 \mid \mathrm{E}_1 \cap \mathrm{E}_2) \cdots \mathbb{P}(\mathrm{P}_t \mid \mathrm{E}_1 \cap \cdots \cap \mathrm{E}_{t-1}) = \prod_{i=1}^{t} \frac{n_i}{i+1} \tag{1.18}$$

onde n_i é a quantidade de bolas da cor E_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 (1.18) 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 (1.17) são

$$\frac{1}{2} \cdot \frac{2}{3} \cdots \frac{m}{m+1} \cdot \frac{1}{m+2} \cdot \frac{2}{m+3} \cdots \frac{t-m}{t+1} = \frac{m!(t-m)!}{(t+1)!} = \frac{1}{(t+1)\binom{t}{m}}. \tag{1.19}$$

Há $\binom{t-1}{m}$ sequências $\mathrm{E}_1, \mathrm{E}_2, \ldots, \mathrm{E}_{t-1}$ de eventos com m posições correspondentes ao sorteio de bola branca e cada uma tem probabilidade dada pela equação (1.19), 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 (1.19) 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.

O TEOREMA DE BAYES Seja $\mathsf{E}_1,\ldots,\mathsf{E}_n$ 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

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

usando o teorema da probabilidade total obtemos

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

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

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

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

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 $\mathbb{P}(\mathsf{B} \mid \mathsf{A})$ onde A é o evento "foi detectado *probabilite*" e B o evento "tem *probabilite*". Usando o teorema de Bayes com $\mathbb{P}(\mathsf{A} \mid \mathsf{B}) = 0{,}99$, pois $\mathbb{P}(\overline{\mathsf{A}} \mid \mathsf{B}) = 0{,}01$ é a chance de ocorrer um falso negativo, e $\mathbb{P}(\mathsf{A} \mid \overline{\mathsf{B}}) = 0{,}03$ é a chance de ocorrer um falso positivo

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

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

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

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.

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

$$\mathbb{P}\left(\, S \,\middle|\, \bigcap_{i\in I} C_i \right) = \frac{\prod_{i\in I} p_i}{\prod_{i\in I} p_i + \prod_{i\in I}(1-p_i)}. \tag{1.20}$$

Para isso vamos assumir que valem

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

$$\mathbb{P}\left(\, \bigcap_{i\in I} C_i \,\middle|\, \overline{S} \right) = \prod_{i\in I} \mathbb{P}(\,C_i \mid \overline{S}\,), \tag{1.22}$$

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 (1.21) e (1.22)

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

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 (1.20).

As hipóteses assumidas nas equações (1.21) e (1.22) 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

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

entretanto

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

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 (1.23) a seguir

$$\mathbb{P}(A \mid B \cap C) = \mathbb{P}(A \mid B) = \mathbb{P}(A \mid C) = \mathbb{P}(A). \tag{1.23}$$

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 (1.23)): 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,

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

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

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

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 (1.21) e (1.22) 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 27, 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

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

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 31, 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 1 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 1 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** $d_i \xleftarrow{R} \{0,1\}$;
3 $N \leftarrow \sum_i d_i 2^i$;
4 **até que** $N < M$;
5 **responda** N.

Algoritmo 2: gerador de números aleatórios.

O resultado das escolhas aleatórias na linha 2 do algoritmo 2, 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 $d_2d_1d_0$ 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 $d_2d_1d_0 = 111$. 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

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

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 17),

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

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

Notemos que, diferente do exemplo do algoritmo 1, 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 10, 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 $|P| = |K| = |C|$ 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, com probabilidade $1/|K|$, e existe um único $K \in \mathcal{K}$ tal que $E_K(P) = C$, para todo $P \in \mathcal{P}$ e todo $C \in \mathcal{C}$.

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 se 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 (inde-

pendentes) é

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

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 $i = 1, 2, \ldots, n-1$ do vetor a
  - sorteie uniformemente uma coordenada $j \in \{i, \ldots, n\}$
  - 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 o algoritmo que recebe um inteiro positivo $M > 0$ e devolve um inteiro escolhido aleatoriamente em $\{0, 1, \ldots, M-1\}$ da seguinte forma:

Instância: inteiro positivo M.
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, ..., d_{k-1}) ←R {0,1}^k;
3 N ← Σ_i d_i 2^i;
4 responda N mod M.
```

Prove que a resposta não tem distribuição 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, ..., d_{k+t-1}) ←R {0,1}^{k+t};
3 N ← Σ_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 10).

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 no seguinte algoritmo a probabilidade p_n do laço executar pelo menos n vezes é maior que 0 e, mais que isso, essa probabilidade é maior que 0 no limite, ou seja, $\lim p_n > 0$ quando $n \to \infty$ (pode ser útil a desigualdade $1 - x \geqslant \exp(-2x)$ para $x \in [0, 1/2]$).

```
1 j ← 0;
2 repita
3 |   j ← j + 1;
4 |   para cada i ∈ {1,2,...,j} faça d_i ←R {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 é mais econômico:

1. Alice escolhe um primo $p \in [n^2, 2n^2]$ e manda para Bob;

2. Alice constrói o polinômio $A(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_n x^{n-1}$;

3. Bob constrói o polinômio $B(x) = b_1 + b_2 x + b_3 x^2 + \cdots + b_n x^{n-1}$;

4. Alice sorteia $\alpha \in \mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$, calcula $A(\alpha)$ mod p e manda α e $A(\alpha)$ mod p para Bob;

5. Bob calcula $B(\alpha)$ mod p e manda para Alice se $A(\alpha) = B(\alpha)$ ou não.

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

$$\mathbb{P}[\,\text{erro}\,] = \mathop{\mathbb{P}}_{\alpha \in_R \mathbb{Z}_p}[\,(A-B)(\alpha) = 0 \mid A - B \not\equiv 0\,] \leqslant \frac{n-1}{p}$$

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 $\bigcup_{F \in \mathcal{F}} F$ 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 2.68).

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

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

$$\mathbb{P}\Big[\, p \in_R \{1, \ldots, m\}^E \text{ é isolante para } \mathcal{F} \,\Big] \geqslant \left(1 - \frac{1}{m}\right)^{|E|}.$$

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 $P^{>1}$ o conjunto de todas as m-ponderações de E que não atribuem o peso 1 a qualquer elemento de E. Defina $\phi\colon P^{>1} \to P$ da seguinte forma: dada a ponderação $p \in P$ tome algum $S_p \in \mathcal{F}$ de peso mínimo de acordo com p e, fazendo $p' = \phi(p)$, defina

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

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

2. Prove que ϕ é injetiva.

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

4. Conclua a demonstração do lema do isolamento.

2 | ALGORITMOS ALEATORIZADOS

Notas de aula por J. Donadelli (cmcc–ufabc)

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 o *espaço*, o *tempo*, a *comunicação* e a *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 ao número de *instruções* 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 sorteios* usados pelo algoritmo (veja um caso na página 69). A *Análise de Algoritmos* é a disciplina da Teoria da Computação que trata dqas técnicas de prova da *correção* de algoritmos e de avaliação 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*. Uma instância do problema é dada por alguma *codificação* dela usando um conjunto de símbolos, em último nível uma cadeia de bits, donde definimos o tamanho de uma instância como o número de símbolos (bits) usados na representação da instância. Na prática 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 a dimensão da matriz; essa simplificação supõe que o tamanho da matriz é muito grande quando comparada ao tamanho dos números que a compõe e se esse não é o caso então o tamanho dos números deve ser levado em conta. 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 dado em função do número de vértices, do número de arestas, ou 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, contamos as instruções executadas, não usamos a noção comum de tempo pois isso dependeria do que (ou quem) executa o algoritmo. Depois, como já dissemos, determinamos quanto tempo o algoritmo demanda em função do tamanho da instância escrevendo uma função que caracteriza como o tempo de execução varia com o tamanho da entrada[1], ess função expressa a ordem de grandeza do crescimento do tempo de execução quando as instâncias crescem.

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.

Uma estimativa para o número de instruções executadas é dada para a classe das instâncias que têm o mesmo tamanho e expressamos o desempenho de algoritmos em função do tamanho da representação das instâncias. Porém, como é possível que entre instâncias de mesmo tamanho a quantidade de recursos usados por uma algoritmo varie precisamos adotar alguma medida resumo. 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. Por isso, adotamos 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 de acordo com alguma distribuição de pesos na classe das instâncias de mesmo tamanho.

Em Complexidade Computacional convenciona-se chamar um algoritmo de **eficiente** com respeito ao tempo de execução se o número de instruções executadas é, no pior caso, 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

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

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 na prática.

Terminamos essa seção com dois exemplos. Para o problema cuja instância é uma lista $a_1, a_2, \cdots, a_n$ de inteiros mais 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. Essa estratégia é expressa como abaixo.

Instância: uma lista $a_1, \ldots, a_n$ de inteiros e um inteiro x.
Resposta : *sim* se x ocorre na lista 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 3: 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 número de instruções executadas é: 1 atribuição na linha 1, mais $4(n-1)$ instruções na linhas 2 (são feitas duas comparações, uma adição e uma atribuição, repetidas $n-1$ vezes), mais 1 comparação na linha 3, mais 1 instrução na linha 4. No total são $4(n-1)+3$ instruções. A função $4n-1$ é linear em n, o tamanho da entrada, 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 número 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 da sequência com os elementos em ordem não decrescente.

```
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 4: 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 até 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

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

portanto $T(n)$ tem ordem de crescimento 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 instruções 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

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

mesma permutação 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 $(a_1, \ldots, a_i)$ da entrada. Para cada i vale que no início do laço da linha 1 temos nas $i-1$ primeiras posições a sequência $(a_1, \ldots, a_{i-1})$ ordenada e o laço da linha 4 procura a posição correta 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. Dado i e que o subvetor $(a_1, \ldots, a_{i-1})$ está ordenado, o teste no laço é executado $i - \text{posto}(a_i) + 1$ vezes.

Exercício 2.1. Verifique que, de acordo com as definições e hipóteses acima, o posto de a_i é igualmente provável ser qualquer $j \in \{1, 2, \ldots, i\}$.

Assim, o número médio de comparações é

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

que é da ordem de n^2.

2.1.1 NOTAÇÃO ASSINTÓTICA

As estimativas para o custo de um algoritmo 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 que podem influenciar o desempenho do algoritmo. 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, por exemplo, 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 $\mathbb{R}^{\geqslant 0}$, o conjunto dos reais não negativos, 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

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

[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.

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 $n \geqslant n_0$

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

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 "quando $n \to \infty$" e fica implícito que as instâncias são suficientemente grandes.

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

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

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 (de 300 aC). Ele computa o maior divisor comum de dois inteiros quaisquer baseado no fato de que $\mathrm{mdc}(a,b) = \mathrm{mdc}(b, a \bmod b)$ onde $a \bmod b$ é o resto na divisão de a por b.

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

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

Algoritmo 5: $\mathrm{mdc}(a,b)$.

O algoritmo de Euclides está correto: $\mathrm{mdc}(a,b) = \mathrm{mdc}(|a|,|b|)$ e $\mathrm{mdc}(a,0) = |a|$, 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 $\mathrm{mdc}(a,b) = \mathrm{mdc}(b, a \bmod b)$.

O algoritmo de Euclides computa $\mathrm{mdc}(a,b)$ 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 6: 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 6 é $O(\log(|a|)\log(|b|))$.

Exercício 2.9 (correção do algoritmo euclidiano estendido). Prove que o algoritmo 6 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 definimos

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

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 $2^{24} \equiv 16 \pmod{25}$. ◊

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 7 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 7: exponenciação modular.

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

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

Tabela 2.1: custo das operações aritméticas.

O tempo de uma multiplicação do algoritmo escolar é $\mathsf{M}(n) = \mathsf{O}(n^2)$. Atualmente, o algoritmo mais usado é o algoritmo de Schönhage–Strassen de 1971 cujo tempo de execução de pior caso é $\mathsf{O}(n \log n \log\log n)$ 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 $\mathsf{O}(n \log n)$ 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 estimado de átomos 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 1 na página 21, por exemplo, 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 1 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 *Monte Carlo*.

O algoritmo gerador de números aleatórios, algoritmo 2, página 39, 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 *Las Vegas*. Uma execução do algoritmo 2 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 importa 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, $(1-p)^2 p$ 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 $\mathsf{T}_{A,x}$ (figura 2.1) em que cada ramificação significa que um sorteio

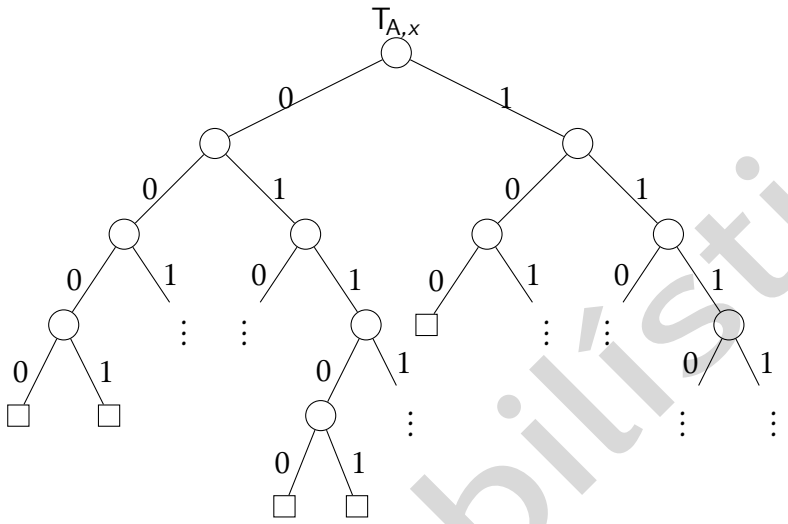


Figura 2.1: árvore de execução de A com instância x.

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 1 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 ←R {1,2,3,4};
2 se f(a) ≠ 0 então responda não.
3 senão
4 |   a ←R {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

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

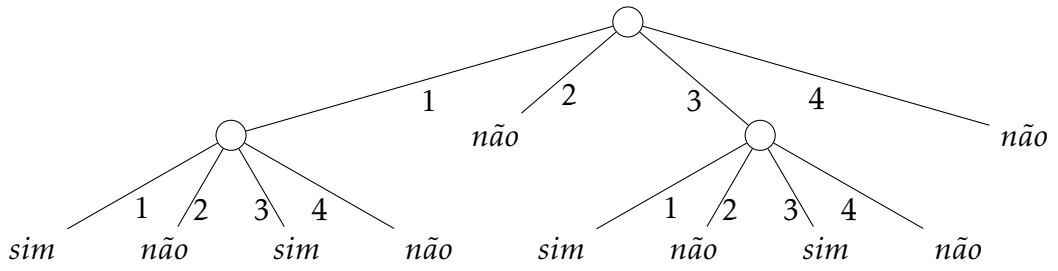


Figura 2.2: árvore de execução do algoritmo acima com entrada $(x-1)(x-3)$.

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) $\{0, 4\}$, $\{0, 5\}$, $\{1, 3\}$, $\{1, 6\}$, $\{8, 3\}$,

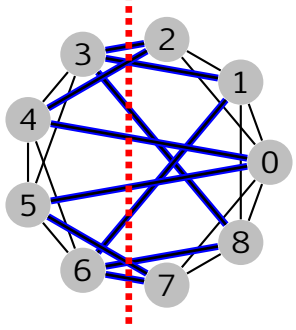


Figura 2.3: o corte definido por $\{0,1,2,7,8\}$ são as arestas em azul.

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

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

$$\text{mincut}(G) := \min\Big\{\big|\nabla(A)\big| : \varnothing \neq A \subsetneq V\Big\}$$

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 (MIN-CUT):

Instância: um grafo G.

Resposta: o tamanho do corte mínimo.

A seguir responderemos uma versão de decisão desse problema: dados um grafo G e um inteiro positivo k, responda *sim* se $\text{mincut}(G) \leqslant k$, responda *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

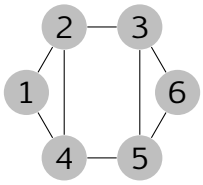


Figura 2.4: exemplo de um grafo.

identificações realizadas. Considere o grafo G representado pelo diagrama da figura 2.4. A figura 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

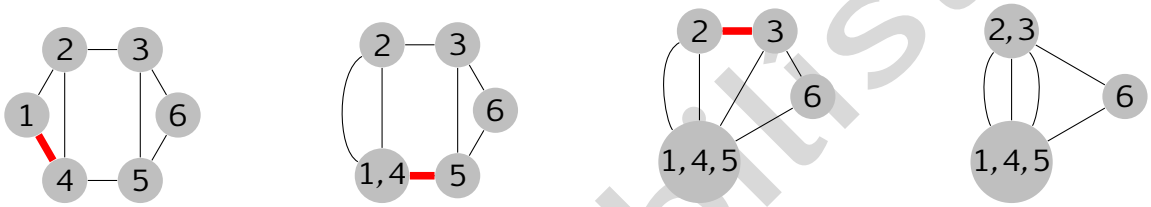


Figura 2.5: uma sequência de três contrações de aresta. Uma contração na aresta em vermelho resulta no multigrafo à direita na sequência. Para manter o registro das contrações acumulamos os rótulos nos vértices.

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).

(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.

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.

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 26, 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 ←R 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 9: 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 9 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 de inteiros.

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

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \text{ e } B = \begin{pmatrix} 3 & 6 & 9 \\ 1 & 2 & 1 \\ 3 & 1 & 3 \end{pmatrix}, \; AB = \begin{pmatrix} 14 & 13 & 20 \\ 35 & 40 & 59 \\ 56 & 67 & 98 \end{pmatrix} \text{ e } C = \begin{pmatrix} 14 & 13 & 20 \\ 10 & 20 & 10 \\ 56 & 67 & 98 \end{pmatrix}$$

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

$$vAB = 0\begin{pmatrix} 14 & 13 & 20 \end{pmatrix} + 1\begin{pmatrix} 35 & 40 & 39 \end{pmatrix} + 0\begin{pmatrix} 56 & 67 & 98 \end{pmatrix} = \begin{pmatrix} 35 & 40 & 39 \end{pmatrix}$$
$$vC = 0\begin{pmatrix} 14 & 15 & 13 \end{pmatrix} + 1\begin{pmatrix} 10 & 20 & 10 \end{pmatrix} + 0\begin{pmatrix} 56 & 67 & 98 \end{pmatrix} = \begin{pmatrix} 10 & 20 & 10 \end{pmatrix}$$

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

$$vAB = 0\begin{pmatrix} 14 & 13 & 20 \end{pmatrix} + 0\begin{pmatrix} 35 & 40 & 39 \end{pmatrix} + 1\begin{pmatrix} 56 & 67 & 98 \end{pmatrix} = \begin{pmatrix} 56 & 67 & 98 \end{pmatrix} \quad (2.4)$$
$$vC = 0\begin{pmatrix} 14 & 13 & 20 \end{pmatrix} + 0\begin{pmatrix} 10 & 20 & 10 \end{pmatrix} + 1\begin{pmatrix} 56 & 67 & 98 \end{pmatrix} = \begin{pmatrix} 56 & 67 & 98 \end{pmatrix} \quad (2.5)$$

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 ≠ 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: *não* se AB ≠ C, caso contrário *sim* com probabilidade de erro no máximo 1/2.

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

Algoritmo 11: 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 ≠ 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 ≠ 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 ≠ C a última escolha fica definida.

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

Se D ≠ 0 e 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: *não* se AB ≠ C, caso contrário *sim* com probabilidade de erro no máximo 1/2.

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

Algoritmo 12: teste de Kimbrel–Sinha para produto de matrizes.

Vamos assumir que AB ≠ C e supor que existam n escolhas em $\{1,2,\ldots,2n\}$, denotadas $x_1, x_2, \ldots, x_n$, todas distintas entre si, para as quais (vA)B = 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 VAB = VC, o que implica AB = C pois V é invertível, contrariando AB ≠ C. Portanto, no caso em que AB ≠ 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 (canônica)

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

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 nula. 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 0 e $x^7 - x$ 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 86); assim como 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 (notemos que $x^3 - x = x(x-1)(x-2)$).

Temos de fato dois problemas computacionais: dado um polinômio p, (1) decidir se, como função, p vale zero 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. Nesse caso, faz parte do algoritmo avaliar $p(a)$.

A definição precisa de circuito aritmético é dada no capítulo 4. Por ora, basta sabermos que é análogo aos circuitos lógicos, mas que nas portas "lógicas" são realizadas as operações do corpo. O tamanho do circuito é dado pela quantidade de portas mais a quantidade de ligações entre elas. O grau do polinômio dado por um circuito é no máximo 2^t, em que t é o tamanho, e esse polinômio é avaliado numa entrada x simulando-se o circuito em tempo polinomial em t.

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 bastante estudados porque o problema da identidade polinomial é muito importante na Teoria da Computação. 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 (PIT). 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 $d \geqslant 0$ e $S \subset \mathbb{F}$ finito e não vazio. Então*

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

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 $d < |\mathbb{F}| \leqslant |S|/2$. 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 estimativa justa, 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 donde o teorema 2.19 segue facilmente.

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

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

e

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

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 1 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** $x_i \xleftarrow{R} \{1, 2, \ldots, 4d\}$;
2 **se** $f(x_1, x_2, \ldots, x_n) \neq 0$ **então responda** *não*.
3 **senão responda** *sim*.

Algoritmo 14: identidade entre polinômios.

A probabilidade de erro do algoritmo 14 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 4.18, página 216). 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 221).

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 16, página 215).

> *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 16, 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 de inteiros

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

calculados 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 2.70 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*

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

para todo p_i fator primo de $p-1$. □

O algoritmo 16 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 $a \xleftarrow{R} \{2, 3, \ldots, p-1\}$;
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 16: raiz primitiva.

A prova de que o algoritmo 16 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 7, página 59). 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 16 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

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

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)

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

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 $\{g \in \mathbb{F}_p[x] : \partial g < d\}$ 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 menor que 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 elementos 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** $a_i \xleftarrow{R} \{0, 1, \ldots, n-1\}$
3 **até que** $x^n + \sum_{i=0}^{n-1} a_i x^i$ *seja irredutível.*

Algoritmo 17: 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,

$$p^n - \partial \mathrm{P}_n = \sum_{\substack{k|n \\ k<n}} \partial \mathrm{P}_k \leqslant \sum_{\substack{k|n \\ k<n}} p^k \leqslant \sum_{k=1}^{n/2} p^k = \frac{p^{n/2+1}-p}{p-1} \leqslant \frac{p}{p-1} p^{n/2} \leqslant 2p^{n/2}$$

portanto, $\partial \mathrm{P}_n \geqslant p^n - 2\sqrt{p^n}$.

Lema 2.37 *Se p é primo então*

$$\frac{p^n}{n} \geqslant \mathrm{I}(n) \geqslant (p^n - 2\sqrt{p^n})/n$$

é uma estimativa para a quantidade de polinômios mônicos irredutíveis de grau n no $\mathbb{Z}_p[x]$. □

O tempo de execução do algoritmo 17 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

$$\frac{1}{n} \geqslant \frac{\mathrm{I}(n)}{p^n} \geqslant \frac{1}{n}\left(1 - \frac{2}{\sqrt{p^n}}\right) > \frac{1}{2n}$$

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 $\mathrm{O}(n\mathsf{T}(n))$ 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,*

$$h(x) \,\Big|\, \left(x^{p^n} - x\right) \tag{2.10}$$

$$\mathrm{mdc}\left(h(x), x^{p^{n/q}} - x\right) = 1 \text{ para todo fator primo } q \text{ de } n. \tag{2.11}$$

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** $\mathrm{mdc}(h, x^{p^{n/q_i}} - x \pmod h) \neq 1$ **então responda** *não*.
3 **se** $x^{p^n} - x \pmod h = 0$ **então responda** *sim*.
4 **senão responda** *não*.

Algoritmo 18: 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 19: 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 $a \xleftarrow{R} \{2,3,\ldots,n-2\}$;
2 **se** $a^{n-1} \not\equiv 1 \pmod{n}$ **então responda** não.
3 **senão responda** sim.

Algoritmo 21: teste de Fermat.

Usando o algoritmo de exponenciação modular (algoritmo 7, página 59), 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

$$\frac{|\mathrm{M}_n \setminus \{1, n-1\}|}{|\{2,\ldots,n-2\}|} \leqslant \frac{|\mathrm{M}_n|}{n-1} \leqslant \frac{|\mathbb{Z}_n^*|/2}{n-1} < \frac{1}{2}$$

Senão, $m = 1$ e a igualdade $\mathrm{M}_n = \mathbb{Z}_n^*$ 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 $b \in_{\mathrm{R}} \{0,1\}^{t-2}$ 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.

repita
  $a \xleftarrow{R} \{2, 3, \ldots, n-1\}$;
  se $a^{n-1} \equiv 1 \pmod{n}$ **então**
    para cada *fator primo p de $n-1$* **faça**
      se $a^{(n-1)/p} \not\equiv 1 \pmod{n}$ **então responda** *sim.*
até que *completar k iterações;*
responda *não.*

Algoritmo 22: 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 $kO(\log^4 n) = O(\log^5 n)$.

2.3.2 O TESTE DE MILLER–RABIN

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 $a \xleftarrow{R} \{2,3,\ldots,n-2\}$;
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 23: 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 24: 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 338). 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 $h \xleftarrow{R} \{p\colon p \text{ é um polinômio mônico de grau } \lceil \log n \rceil \text{ em } \mathbb{Z}_n[x]\}$;
3 **se** h *divide* $(x+1)^n - (x^n+1)$ *no* $\mathbb{Z}_n[x]$ **então responda** *primo*.
4 **senão responda** *composto*.

Algoritmo 25: 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 108, 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 7 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 $h \xleftarrow{R} \{p\colon p$ 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 $\mathbb{Z}_n[x]$.

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 83, e $p>16$ temos probabilidade $\mathrm{I}(r)/p^r > 1/(2r)$ 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 $n>100$ Assumamos que $n>p>100$ e que n é testado na força bruta contra os cem primeiros números primos[7]. 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,

$$\sum_{\mathsf{Q}}|\mathsf{C}_\mathsf{Q}|=\sum_{i=1+r/2}^{r}\mathrm{I}(i)p^{r-i}\geqslant\sum_{i=1+r/2}^{r}\left(\frac{p^i-2\sqrt{p^i}}{i}\right)p^{r-i}\geqslant\sum_{i=1+r/2}^{r}\left(\frac{1}{i}-\frac{1}{\sqrt{p^i}}\right)p^r$$

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)

$$\mathsf{H}_\mathsf{N}=\log(\mathsf{N})+\gamma+\frac{1}{2\mathsf{N}}-\frac{1}{12\mathsf{N}^2}+\frac{\varepsilon_\mathsf{N}}{120\mathsf{N}^4},\tag{2.16}$$

com $0<\varepsilon_\mathsf{N}<1$ e $\gamma\approx 0{,}5772156649$, a constante de Euler–Mascheroni[8], 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

$$\sum_{i=1+r/2}^{r}\left(\frac{1}{\sqrt{p}}\right)^i<\frac{1}{9}\left(\left(\frac{1}{10}^{r/2}\right)-\left(\frac{1}{10}\right)^r\right)<\frac{1}{9}\left(\frac{1}{10^{5/2}}-\frac{1}{10^5}\right)$$

Logo, a quantidade de polinômios de grau r com algum fator irredutível de grau entre $r/2+1$ e r é

$$\sum_{\mathsf{Q}}|\mathsf{C}_\mathsf{Q}|>\left(\log 2-\frac{3383}{37500}-\frac{1}{9}\left(\frac{1}{10^{5/2}}-\frac{1}{10^5}\right)\right)p^r>0{,}6p^r$$

[7] 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.

[8] Essa constante é o limite de $\mathsf{H}_\mathsf{N}-\log\mathsf{N}$ quando $\mathsf{N}\to\infty$.

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** $m \xleftarrow{R} \{n+1, n+2, \ldots, 2n\}$ **até que** Teste_Primo(m) =*sim*;
2 **responda** m.

Algoritmo 26: 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

$$\frac{1}{3}\frac{n}{\log n} < \pi(2n) - \pi(n) < \frac{7}{5}\frac{n}{\log n} \tag{2.17}$$

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

$$\frac{1}{3\log n} < \rho(n) < \frac{7}{5\log n}. \tag{2.18}$$

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 (2.17) 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 $< (1/4)^t$ em t rodadas independentes. 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 $\mathbb{P}(\,\mathsf{E} \mid \mathsf{C}\,) < (1/4)^t$ e queremos $\mathbb{P}(\,\mathsf{C} \mid \mathsf{E}\,)$. Pelo Teorema de Bayes

$$\mathbb{P}(\,\mathsf{C} \mid \mathsf{E}\,) = \frac{\mathbb{P}(\,\mathsf{E} \mid \mathsf{C}\,)\mathbb{P}(\mathsf{C})}{\mathbb{P}(\mathsf{E})} \leqslant \frac{\mathbb{P}(\,\mathsf{E} \mid \mathsf{C}\,)}{\mathbb{P}(\mathsf{E})} < \frac{(1/4)^t}{1/(3k)} = \frac{3k}{4^t}$$

pois $\mathbb{P}(\mathsf{E}) > 1/(3k)$ por (2.18).

Observação 2.53. Na prática, após sortear m é mais eficiente testar divisibilidade de m por inteiros primos até uma constante 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[9] 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

$$\mathcal{C} := \bigcup_{k \geqslant 1} \mathcal{C}_k$$

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

[9] 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 $\Omega_1 \times \Omega_2$.

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

$$P(A) := \frac{|B|}{2^k}. \tag{2.19}$$

Essa definição é consistente (veja exercício 2.80 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 2.81 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 (2.19) 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 11. 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

todos os processos são ativados um número infinito de vezes (‡)

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 27, 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 ←R {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 27: 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 27

$$\text{COMP}(S, \omega) := ((\alpha, P, b)_t : t \in \mathbb{N})$$

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 $\{\omega \in \{0,1\}^{\mathbb{N}} : \text{COMP}(S, \omega) \text{ em } \textit{deadlock}\}$ 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 27), 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 $\textsc{comp}(D, \omega)$ 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,* $\textsc{comp}(S, \omega)$ *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 7 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 $\mathbb{P}(\mathsf{F}) = \mathbb{P}(\mathsf{E})$ 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 $\{\mathsf{A}, 2, 3, \ldots, 9, 10, \mathsf{J}, \mathsf{Q}, \mathsf{K}\}$ 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 9 na página 66, 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 $\mathsf{G} = (\mathsf{A} \cup \mathsf{B}, \mathsf{E})$ um grafo com $|\mathsf{A}| = |\mathsf{B}| = n$ e todas as arestas em E tem um vértice em A e o outro em B, isto é G é um **grafo bipartido**. Defina a *matriz de adjacências* $\mathsf{A} = (a_{i,j})$ pondo

$$a_{i,j} := \begin{cases} x_{i,j} & \text{se } \{a_i, b_j\} \in \mathsf{E} \\ 0 & \text{caso contrário.} \end{cases}$$

Um **emparelhamento** M em G é um subconjunto de E formado por arestas não adjacentes, isto é, $e \cap d = \varnothing$ para quaisquer arestas $e,\ d \in \mathsf{M}$ distintas. O emparelhamento M é dito **perfeito** se $|\mathsf{M}| = n$.

Prove que G tem um emparelhamento perfeito se, e somente se, $\det(\mathsf{A}) \neq 0$ (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 $\mathcal{F} := \{\mathsf{M}_1, \mathsf{M}_2, \ldots, \mathsf{M}_k\}$ o conjunto dos emparelhamentos perfeitos de G. Tome $p\colon \mathsf{E}(\mathsf{G}) \to \{1, \ldots, 2|\mathsf{E}|\}$ 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. 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 $\operatorname{mdc}(M, N) = 1$. Analise o algoritmo.

Exercício 2.68 (Arvind e Mukhopadhyay (2008) e Klivans e Spielman (2001)). Prove a seguinte versão do lema do isolamento (exercício 1.59, página 45). 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 2.69. 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 2.70. Neste exercício provaremos (veja a equação (2.8))

$$\varphi(n) > \frac{n}{4 \log n}. \tag{2.20}$$

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 (2.20).

Exercício 2.71. 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 2.72. 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 16.

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 2.71 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 2.73 (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 28 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 a 5. Verifique que sem considerar as linhas 3 a 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 29 abaixo modifica o algoritmo 28 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 de rodadas t. Prove que

$$\mathbb{P}[\,\text{erro}\,] \leqslant \left(\frac{1}{2}\right)^t.$$

Exercício 2.75. Vimos no texto 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 $k > 1$ fatores primos distintos então o número de soluções de $x^2 \equiv a \pmod n$ é 0 ou 2^k (*dica*: teorema chinês do resto, seção A.4 do apêndice).

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
    $b \xleftarrow{\mathrm{R}} \{1, 2, \ldots, p-1\}$;
  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 28: raiz quadrada no $\mathbb{Z}_p^*$.

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** $a^{\frac{p-3}{4}+1} \bmod p$.
se $p \equiv 1 \pmod 4$ **então**
  repita
    $b \xleftarrow{\mathrm{R}} \{1, 2, \ldots, p-1\}$;
  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 $a^{\frac{i+1}{2}} b^{\frac{k}{2}} \bmod p$.

Algoritmo 29: raiz quadrada no $\mathbb{Z}_p^*$.

Exercício 2.76. Complementando o exercício 2.75, vejamos o caso $n = pq$ para $p \equiv q \equiv 3 \pmod 4$ primos (não é necessário, apenas aumenta a eficiência do algoritmo apresentado a seguir).

Instância: inteiros c, p, q, onde $p, q \equiv 3 \pmod 4$ são primos.
Resposta : raiz quadrada de c módulo n com $n = pq$.

1 Determine a, b tais que $ap + bq = 1$
2 $r \leftarrow c^{(p+1)/4} \mod p$
3 $s \leftarrow c^{(q+1)/4} \mod q$
4 $x \leftarrow (aps + bqr) \mod n$
5 $y \leftarrow (aps - bqr) \mod n$
6 Devolva $x, -x \mod n$, y e $-y \mod n$.

Algoritmo 30: Raiz quadrada de c módulo n.

Tome $u = p^{q-1} - q^{p-1}$. Mostre que $u^2 \equiv 1 \pmod p$, que $u^2 \equiv 1 \pmod q$ e que $u^2 \equiv 1 \pmod n$. Mostre que se x_0 é solução de $x^2 \equiv a \pmod n$ então também são soluções $-x_0$, ux_0 e $-ux_0$.

Prove que o algoritmo acima está correto.

Exercício 2.77 (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 $\mathrm{mdc}(a^{(n-1)/q} - 1, n) = 1$, então n é primo.

Exercício 2.78 (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.

Exercício 2.79 (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 29. Se n é primo, então as únicas

raízes são a e $-a$; senão o algoritmo 29 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 $a \xleftarrow{R} \{2, 3, \ldots, n-1\}$;
3 **se** $\operatorname{mdc}(a, n) \neq 1$ **então responda** *composto*.
4 $x \leftarrow a^2 \mod n$;
5 $y \leftarrow$ raiz quadrada determinada pelo algoritmo 29 com parâmetros (x, p, t);
6 **se** *$y \neq \{-a, a\}$ ou $y^2 \neq x \bmod n$ ou a linha 3 do algoritmo 29 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 2.80. Prove que a definição de probabilidade dada na equação (2.19), página 104, 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 (2.19) coincide nas duas representações de A.

Exercício 2.81. Em geral, a parte difícil da aplicação do teorema de Carathéodory (descrito informalmente na página 104) 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

Notas de aula por J. Donadelli (cmcc–ufabc)

3.1 VARIÁVEIS ALEATÓRIAS DISCRETAS

Uma função definida sobre um espaço amostral de um espaço de probabilidade discreto é uma **variável aleatória discreta** e se tem contradomínio no conjunto dos números reais é chamada de **variável aleatória real**. Na maior parte deste texto estaremos interessados em variáveis aleatórias reais e omitiremos os adjetivos "discreta" e "real" ao nos referirmos às variáveis aleatórias. Usaremos, em geral, as letras maiúsculas finais do alfabeto, como por exemplo X, Y, Z, W, para denotar as variáveis aleatórias.

Um exemplo trivial de variável aleatória é dado por uma função constante, por exemplo, quando para todo $\omega \in \Omega$ há $c \in \mathbb{R}$ tal que $X(\omega) = c$. No modelo clássico para o lançamento de um dado equilibrado a variável aleatória $X\colon \{1,2,3,4,5,6\} \to \{1,2,3,4,5,6\}$ dada por $X(\omega) = \omega$, para todo ω, é

o resultado do lançamento. A resposta do algoritmo 1, página 21, e o tempo de execução do algoritmo 2, página 39, também são exemplos de variáveis aleatórias.

Exemplo 3.1 (*variável aleatória indicadora*). Este exemplo introduz uma variável aleatória que é muito útil em Probabilidade. Para qualquer evento A de um espaço de probabilidade $(\Omega, \mathbb{P})$, definimos a variável aleatória $\mathbb{1}_A \colon \Omega \to \{0,1\}$ por

$$\mathbb{1}_A(\omega) := \begin{cases} 1, & \text{se } \omega \in A \\ 0, & \text{caso contrário,} \end{cases}$$

e que chamamos de **variável aleatória indicadora** do evento A. Toda variável aleatória discreta e real pode ser representada em termos de variáveis aleatórias indicadoras. Para escrever uma representação nós tomamos uma enumeração qualquer $x_1, x_2, x_3, \ldots$ dos valores que a variável aleatória X assume e definimos uma partição do espaço amostral definida pelos eventos $A_n := \{\omega \in \Omega \colon X(\omega) = x_n\}$ de modo que

$$X(\omega) = \sum_{n \geqslant 1} x_n \mathbb{1}_{A_n}(\omega)$$

para todo $\omega \in \Omega$. ◊

DISTRIBUIÇÃO Seja $X \colon \Omega \to S$ uma variável aleatória do modelo probabilístico discreto $(\Omega, \mathbb{P})$. Denotamos por $X(\Omega)$ a imagem da variável aleatória X e definimos para cada $B \subset X(\Omega)$ o evento

$$[X \in B] = X^{-1}(B) := \{\omega \in \Omega \colon X(\omega) \in B\}$$

e $[X = t] := [X \in \{t\}] = X^{-1}(t)$. A função X não é necessariamente invertível, embora possamos fazê-la sobrejetiva restringindo S à imagem ela ainda pode não ser injetiva. No entanto, X^{-1} é bem definida para subconjuntos.

A **distribuição de** X, ou **lei** de X, é a medida de probabilidade $\mathbb{P}_X$ definida em $2^{X(\Omega)}$ por

$$\mathop{\mathbb{P}}_{X}(t) := \mathbb{P}[X = t]$$

para todo $t \in X(\Omega)$ e estendida para todo subconjunto de $X(\Omega)$ da maneira usual de modo que vale

$$\mathop{\mathbb{P}}_{X}(B) = \mathbb{P}[X \in B], \text{ para todo } B \subset X(\Omega).$$

Claramente $0 \leqslant \mathbb{P}_X(B) \leqslant 1$. Ainda $\mathbb{P}_X(S) = \mathbb{P}(\Omega) = 1$ e vale

$$\mathop{\mathbb{P}}_{X}\left(\bigcup_{n\geqslant 1} A_n\right) = \mathbb{P}\left(X^{-1}\left(\bigcup_{n\geqslant 1} A_n\right)\right) = \mathbb{P}\left(\bigcup_{n\geqslant 1} X^{-1}(A_n)\right) = \sum_{n\geqslant 1} \mathbb{P}\Big(X^{-1}(A_n)\Big) = \sum_{n\geqslant 1} \mathop{\mathbb{P}}_{X}(A_n).$$

Além disso, consideramos $(S, \mathbb{P}_X)$ com a medida de probabilidade sendo uma extensão[1] da definição acima para todo $B \subset S$ dada por $\mathbb{P}_X(B) = \mathbb{P}[X \in B \cap X(\Omega)]$.

Por exemplo, se X é o resultado do lançamento de um dado equilibrado, então a lei de X é a medida uniforme $\mathbb{P}_X(\omega) = 1/6$, para todo $\omega \in \{1,2,3,4,5,6\}$. Se A é um evento de um modelo probabilístico $(\Omega, \mathbb{P})$ então a lei de $\mathbb{1}_A$ é $\mathbb{P}_{\mathbb{1}_A}$ dada por $\mathbb{P}_{\mathbb{1}_A}(1) = \mathbb{P}(A)$ e $\mathbb{P}_{\mathbb{1}_A}(0) = 1 - \mathbb{P}(A)$.

Exemplo 3.2 (*distribuição de Bernoulli*). É comum ocorrerem situações com experimentos que interessam apenas duas características dos resultados: *sucesso* ou *fracasso*. Por exemplo, uma peça de uma linha de produção é classificada como *boa* ou *defeituosa*; o resultado de um exame médico é *positivo* ou *negativo*; um entrevistado *concorda* ou *não concorda* com a afirmação feita; a condição de um laço num algoritmo é *verdadeira* ou *falsa*; um evento A *ocorreu* ou *não ocorreu*. Esses experimentos recebem o nome de **ensaio de Bernoulli**. Nessas situações podemos modelar a observação com uma variável aleatória $X\colon \Omega \to \{0,1\}$ e definimos

$$b_p(t) := p^t(1-p)^{1-t}, \text{ para todo } t \in \{0,1\}$$

em que $p = \mathbb{P}[X = 1]$ é a probabilidade de *sucesso*. Dizemos que X tem **distribuição de Bernoulli** com parâmetro p se $\mathbb{P}_X(x) = b_p(x)$ e usamos a notação $X \in_{b_p} \{0,1\}$ ou $X \sim \text{Bernoulli}(p)$ para indicar tal fato. ◊

De um modo geral, se $\mathcal{D}$ associa a cada elemento $t \in S$ um real não-negativo $\mathcal{D}(t)$, então chamamos $\mathcal{D}$ uma **distribuição** de probabilidade sobre o conjunto enumerável S se $\sum_t \mathcal{D}(t) = 1$. A variável aleatória $X\colon \Omega \to S$ tem distribuição $\mathcal{D}$ se $\mathbb{P}_X(t) = \mathcal{D}(t)$ para todo $t \in S$ e nesse caso escrevemos

$$X \in_{\mathcal{D}} S$$

ou, o que é mais comum encontrarmos nos textos, $X \sim \mathcal{D}$. Escrevemos $x \in_{\mathcal{D}} S$ para dizer que $x \in S$ é um elemento de S escolhido de acordo com a distribuição $\mathcal{D}$.

Exemplo 3.3 (*distribuição uniforme*). Uma varável aleatória X que assume qualquer valor de um conjunto finito S com a mesma probabilidade

$$\mathcal{U}(t) = \frac{1}{|S|}$$

para todo $t \in S$, tem **distribuição uniforme** sobre S e denotamos esse fato por $X \in_R S$ ou $X \sim \mathcal{U}(S)$. ◊

Exemplo 3.4 (*distribuição geométrica*). Uma variável geométrica conta o número de realizações de ensaios de Bernoulli independentes e idênticos até que ocorra um sucesso. Por exemplo, no laço do

[1] Eventualmente temos $S = \mathbb{R}$, porém não estamos considerando aqui espaços contínuos o que torna essa consideração um abuso de notação.

algoritmo 2, página 39, que reproduzimos abaixo

repita
  para cada $i \in \{0,\ldots,\lfloor \log_2 M \rfloor\}$ **faça** $d_i \xleftarrow{R} \{0,1\}$;
  $N \leftarrow \sum_i d_i 2^i$;
até que $N < M$;

cada execução das linhas internas é um ensaio de Bernoulli e um *succeso* ocorre quando a condição $N < M$ é verdadeira, estamos interessados no número de repetições até ocorrer um sucesso. Se X é uma variável aleatória que conta o número de ensaios até que ocorra um sucesso, então a lei de X é

$$\mathcal{G}_p(t) = (1-p)^{t-1}p, \text{ para todo } t \geqslant 1$$

e dizemos que X tem **distribuição geométrica** com parâmetro p, o que denotamos por $X \sim \text{Geom}(p)$ ou $X \in_{\mathcal{G}_p} \mathbb{N}$. ◊

Exercício 3.5. Mostre que se Z tem distribuição geométrica com parâmetro p então $\mathbb{P}[Z > n-1] = (1-p)^{n-1}$ para todo $n \geqslant 1$.

Exemplo 3.6 (*distribuição binomial*). Em n ensaios de Bernoulli idênticos e independentes uma resposta $(b_1, b_2, \ldots, b_n)$ ocorre com probabilidade $p^t(1-p)^{n-t}$ sempre que ocorrerem t sucessos. A probabilidade de ocorrerem exatamente t sucessos é

$$\text{b}_{n,p}(t) := \binom{n}{t} p^t (1-p)^{n-t}, \text{ para todo } t \in \{0,1,\ldots,n\}$$

e uma variável aleatória com tal distribuição é dita ter **distribuição binomial** com parâmetros n e p, o que denotamos por $X \in_{\text{b}_p} \{0,1,\ldots,n\}$ ou $X \sim \text{b}(n,p)$. A variável com distribuição binomial de parâmetros n e p conta o número de sucessos em n ensaios de Bernoulli idênticos e independentes. ◊

Proposição 3.7 *Seja* $k := \lfloor (n+1)p \rfloor$. *A função* $\text{b}_{n,p}(t)$ *é crescente em* $\{0,1,\ldots,k\}$ *e é decrescente em* $\{k+1, k+2, \ldots, n\}$.

Demonstração. A razão de valores sucessivos é, para $t > 0$

$$\frac{\text{b}_{n,p}(t)}{\text{b}_{n,p}(t-1)} = \frac{(n-t+1)p}{t(1-p)}$$

de modo que $\text{b}_{n,p}(t)$ é crescente se, e só se, $(n-t+1)p > t(1-p)$, ou seja, $(n+1)p - t > 0$. □

Se lançamos uma moeda honesta $2n$ vezes, com que probabilidade ocorrem exatamente n caras? O número de caras é uma variável aleatória binomial $X \sim \text{b}(2n, 1/2)$ de modo que

$$\mathbb{P}[X = n] = \binom{2n}{n}\left(\frac{1}{2}\right)^{2n} = \frac{(2n)!}{(n!)^2 4^n} = (1+o(1))\frac{1}{\sqrt{\pi n}}$$

usando a aproximação de Stirling (veja (d.3)). É um probabilidade bem pequena, para $n = 50$ temos $\approx 0{,}08$ e para $n = 130$ temos $\approx 0{,}05$. Porém, de fato, com probabilidade que tende a 1 quando $n \to \infty$ o valor de X/n está no intervalo $(1/2 - \varepsilon, 1/2 + \varepsilon)$ qualquer que seja $\varepsilon \in (0, 1/2)$ pois por simetria temos

$$\mathbb{P}\left[\left|\frac{X}{n} - \frac{1}{2}\right| \geqslant \varepsilon\right] = 2\mathbb{P}\left[X \geqslant \left(\frac{1}{2} + \varepsilon\right)n\right]$$

além disso, para k como na proposição acima

$$\frac{1}{2^n}\binom{n}{k} \leqslant \mathbb{P}\left[X \geqslant \left(\frac{1}{2} + \varepsilon\right)n\right] \leqslant \frac{n+1}{2^n}\binom{n}{k}.$$

Usando a aproximação de Stirling ingenuamente

$$\binom{n}{k} \approx \frac{n^{n+\frac{1}{2}}}{(n-k)^{n-k+\frac{1}{2}}k^{k+\frac{1}{2}}}$$

tomando logaritmo e dividindo por n

$$\frac{1}{n}\log\frac{1}{2^n}\binom{n}{k} \approx -\log 2 + \frac{1}{n}\log\left(\frac{n}{n-k}\right)^{n-k} + \frac{1}{n}\log\left(\frac{n}{k}\right)^{k}$$

que, quando $n \to \infty$, converge para

$$-\log 2 - \frac{1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right) = 0.$$

Esse resultado pode ser feito rigoroso (veja o exercício 3.74, página 174) e veremos uma demonstração alternativa e mais geral (veja o exemplo 6.10, página 312). Ele já era conhecido por volta de 1700 por Jacob Bernoulli e é o primeiro resultado do que veio a ser chamado de Lei Fraca dos Grandes Números. No teorema de Jacob Bernoulli quanto maior n, menor é a incerteza sobre o valor de X/n, a frequência relativa do número de ocorrência de um evento em repetições independentes tende, conforme o número de repetições aumenta, à probabilidade da ocorrência do evento.

Exemplo 3.8 (*distribuição de Poisson*). Uma variável aleatória de Poisson expressa a probabilidade de ocorrência de um determinado número de eventos num intervalo de tempo fixo sempre que tais eventos ocorram com uma taxa média conhecida e independentemente do tempo desde a última ocorrência.

Há vários exemplos curiosos de fenômenos aleatórios com essa distribuição na literatura. Começemos com o seguinte exemplo do célebre *Introdução a Probabilidade* de Feller (1968): na segunda guerra mundial a cidade de Londres foi intensamente bombardeada pelos alemães. Para determinar se as bombas tinham um alvo ou foram lançadas aleatoriamente os ingleses dividiram o sul da cidade em pequenas regiões e determinaram a taxa de 0,9323 bombas por região. Se n_k é o número de regiões que receberam k bombas, a contagem foi

k	0	1	2	3	4	5 ou mais
n_k	229	211	93	35	7	1

ao qual o modelo de Poisson se ajusta impressionantemente bem, o que levou-os a acreditar que o bombardeio foi aleatório. Um outro exemplo, agora clássico, vem de William Sealy Gosset[2], um químico e matemático formado em Oxford e contratado, em 1899, pela famosa cervejaria *Arthur Guinness and Son* em Dublin. Sua tarefa era aperfeiçoar o processo de produção de cerveja. Gosset trabalhou com o modelo de Poisson para a contagem de células de levedura. Outra aplicação curiosa e conhecida desta distribuição é devida a Ladislau Bortkiewicz que em 1898 publicou dados sobre o número de soldados do exército da Prússia mortos por coices de cavalo, tais números seguiam uma distribuição de Poisson.

Uma variável aleatória de Poisson com parâmetro $\lambda > 0$ conta o número de ocorrências de um determinado evento que ocorre a uma taxa λ e cuja distribuição é dada por

$$\text{Poisson}_\lambda(t) := \frac{e^{-\lambda}\lambda^t}{t!}, \text{ para todo inteiro } t \geqslant 0.$$

Gosset, citado acima, observou "como a dispersão nas contagens de colonias de levedura foi semelhante ao limite exponencial da distribuição binomial". De fato, a distribuição de Poisson pode ser derivada como um caso limite da distribuição binomial quando o número de ensaios tende ao infinito e a taxa média de ocorrências permanece fixa (veja o enunciado preciso no exercício 3.75 no final deste capítulo). A seguinte estratégia pode ser transformada numa demonstração desse fato: tome o polinômio

$$\sum_{k=0}^{n} b_{n,p}(k)x^k = (xp + 1 - p)^n = (1 + (x-1)p)^n$$

pelo binômio de Newton. No limite, quando $n \to \infty$, assumindo que $p = p(n)$ é tal que $p \cdot n = \lambda$, temos (usando (s.8))

$$\sum_{k\geqslant 0} b_{n,p}(k)x^k = \lim_{n\to\infty}\left(1 + \frac{\lambda(x-1)}{n}\right)^n = e^{\lambda x}e^{-\lambda} = \sum_{k\geqslant 0}\frac{\lambda^k e^{-\lambda}}{k!}x^k$$

e, agora, comparamos os coeficientes de cada lado. Por isso, $\text{Poisson}_\lambda(t)$ pode ser usada como uma aproximação da distribuição binomial $b_{n,p}(t)$ se n for suficientemente grande e p suficientemente pequena.

Intuitivamente, podemos dizer que se λ é a taxa de ocorrência de um evento num intervalo e tomamos subintervalos suficientemente pequenos, a ponto da probabilidade de um evento ocorrer duas vezes nesse intervalo é insignificante, então a probabilidade de ocorrência do evento em cada subintervalo é λ/n. Agora, assumimos que as ocorrências do evento em todo o intervalo pode ser

[2]Publicou artigos sob o pseudônimo de *Student* porque o seu empregador proibiu as publicações por funcionários depois que segredos comerciais foram divulgados.

visto como n ensaios de Bernoulli com parâmetro λ/n. Em n ensaios independentes de Bernoulli com probabilidade de sucesso $p = p(n) = \lambda/n$, para uma constante $\lambda > 0$, a probabilidade de k sucessos é $b_{n,p}(k) \approx \lambda^k e^{-\lambda}/k!$ que é conhecido como a *aproximação de Poisson para a distribuição binomial*. ◊

Para uma variável aleatória real $X\colon \Omega \to \mathbb{R}$ os eventos do espaço amostral Ω definidos por

$$[X \leqslant r] := \{\omega \in \Omega\colon X(\omega) \leqslant r\}$$

para qualquer $r \in \mathbb{R}$, são particularmente importantes no estudo de variáveis aleatórias mais geralmente. Eles descrevem completamente o comportamento da variável aleatória, mesmo no caso geral, além do discreto. A função $F_X(r) := \mathbb{P}[X \leqslant r]$ é chamada **função de distribuição acumulada** de X. De modo análogo nós definimos os eventos como $[X < r]$ e $[X \geqslant r]$.

DISTRIBUIÇÃO CONJUNTA E INDEPENDÊNCIA Uma variável aleatória discreta $Z\colon \Omega \to \mathbb{R}^n$ $(n > 1)$ é chamada de **vetor aleatório** e usamos a notação $Z = (X_1, \ldots, X_n)$, nesse caso cada coordenada é uma variável aleatória. Sua distribuição sobre $Z(\Omega) \subset \mathbb{R}^n$ é a medida de probabilidade $\mathbb{P}_Z = \mathbb{P}_{(X_1,\ldots,X_n)}$ definida por $\mathbb{P}_Z((a_1,\ldots,a_n)) = \mathbb{P}[Z = (a_1,\ldots,a_n)] = \mathbb{P}[(X_1,\ldots,X_n) = (a_1,\ldots,a_n)]$. Essa distribuição, chamada **distribuição conjunta** das variáveis $X_1,\ldots,X_n$, *não* fica determinada pelas distribuições $\mathbb{P}_{X_i}$ a não ser com a hipótese de independência das variáveis X_i.

As variáveis aleatórias X e Y definidas em Ω com valores em S são **variáveis aleatórias independentes** se o conhecimento do valor de uma delas não altera a probabilidade da outra assumir qualquer valor, isto é, formalmente, se para quaisquer eventos A e B de Ω

$$\mathbb{P}([X \in A] \cap [Y \in B]) = \mathbb{P}[X \in A] \cdot \mathbb{P}[Y \in B].$$

Considerando as leis $\mathbb{P}_{(X,Y)}$ do vetor aleatório (X, Y), $\mathbb{P}_X$ de X e $\mathbb{P}_Y$ de Y temos que independência como definido acima é equivalente a

1. $\mathbb{P}_{(X,Y)}(A \times B) = \mathbb{P}_X(A) \cdot \mathbb{P}_Y(B)$;
2. $[X = a]$ e $[Y = b]$ são independentes, para quaisquer $a \in X(\Omega)$ e $b \in Y(\Omega)$;
3. $\mathbb{P}_{(X,Y)}((a,b)) = \mathbb{P}_X(a) \cdot \mathbb{P}_Y(b)$, para quaisquer $a \in X(\Omega)$ e $b \in Y(\Omega)$;
4. para variáveis aleatórias reais, $\mathbb{P}([X \leqslant a] \cap [Y \leqslant b]) = \mathbb{P}[X \leqslant a] \cdot \mathbb{P}[Y \leqslant b]$ para quaisquer $a, b \in \mathbb{R}$.

Naturalmente, podemos estender essa definição para qualquer quantidade finita de variáveis aleatórias. As variáveis aleatórias $X_1, X_2, \ldots, X_n$ são **independentes** se para quaisquer eventos $A_1, \ldots, A_n$

$$\mathbb{P}_{(X_1,X_2,\ldots,X_n)}(A_1 \times \cdots \times A_n) = \prod_{i=1}^{n} \mathbb{P}_{X_i}(A_i). \tag{3.1}$$

Notemos que qualquer subconjunto $X_{i_1}, \ldots, X_{i_k}$ dessas variáveis também é independente, basta tomar $A_j = X_j(\Omega)$ para todo $j \neq i_1, \ldots, i_k$ na equação (3.1) acima.

Exercício 3.9. Sejam $X_1,\ldots,X_n$ variáveis aleatórias e $S_1,\ldots,S_n$ conjuntos finitos e não vazios. Mostre que são equivalentes

1. $(X_1,\ldots,X_n) \in_R S_1 \times \cdots \times S_n$
2. $X_1,\ldots,X_n$ são independentes e $X_i \in_R S_i$ para cada i.

FUNÇÕES DE VARIÁVEIS ALEATÓRIAS Se $X : \Omega \to \mathbb{R}$ é uma variável aleatória real e $f\colon \mathbb{R} \to \mathbb{R}$ é uma função real então a função composta $f(X)$ é uma variável aleatória real Y cuja distribuição é

$$\mathop{\mathbb{P}}_{Y}(y) = \mathbb{P}[\, f(X) = y \,] = \sum_{x\colon f(x)=y} \mathop{\mathbb{P}}_{X}(x).$$

Por exemplo, se X é uma variável aleatória de $(\Omega,\mathbb{P})$, então podemos definir $Z\colon \Omega \to \mathbb{R}$ por $Z(\omega) := X(\omega)^2$ para todo $\omega \in \Omega$. A função Z também é uma variável aleatória e para todo t não negativo temos $[Z \leqslant t] = [-\sqrt{t} \leqslant X \leqslant \sqrt{t}]$. Se $a, b \in \mathbb{R}$, com $a \neq 0$, então $Y\colon \Omega \to \mathbb{R}$ dada por $Y(\omega) = a \cdot X(\omega) + b$ é uma variável aleatória tal que, para todo real t, $[Y \leqslant t] = [X \leqslant (t-b)/a]$.

Se $(X_1, X_2, \ldots, X_n)$ é um vetor aleatório de um espaço de probabilidade e $f\colon \mathbb{R}^n \to \mathbb{R}$ é função então $f(X_1, X_2, \ldots, X_n)$ é uma variável aleatória real. Logo, operações aritméticas elementares com variáveis aleatórias reais resultam em variáveis aleatórias. Em particular, se X e Y são variáveis aleatórias definidas em $(\Omega,\mathbb{P})$, a soma é a variável aleatória $X+Y\colon \Omega \to \mathbb{R}$ dada por $\{X+Y\}(\omega) = X(\omega) + Y(\omega)$ e o produto é a variável aleatória $X \cdot Y\colon \Omega \to \mathbb{R}$ dada por $\{X \cdot Y\}(\omega) = X(\omega) \cdot Y(\omega)$. A distribuição de $Z = X+Y$ é

$$\mathop{\mathbb{P}}_{Z}(z) = \mathbb{P}\Big(Z^{-1}(z)\Big) = \sum_{x \in X(\Omega)} \mathbb{P}\Big(X^{-1}(x) \cap Y^{-1}(z-x)\Big) = \sum_{x \in X(\Omega)} \mathop{\mathbb{P}}_{(X,Y)}((x, z-x)).$$

No caso particular em que X e Y são independentes

$$\mathop{\mathbb{P}}_{X+Y}(z) = \sum_{x \in X(\Omega)} \mathop{\mathbb{P}}_{X}(x) \mathop{\mathbb{P}}_{Y}(z-x).$$

Naturalmente, podemos considerar a soma e o produto para $n > 2$ variáveis, denotadas $\sum_{i=1}^n X_i$ e $\prod_{i=1}^n X_i$ respectivamente.

Por exemplo, sejam $X_1,\ldots,X_n$ variáveis aleatórias independentes e com distribuição Bernoulli com parâmetro p. Então

$$X = \sum_{i=1}^{n} X_i$$

é uma variável aleatória que conta a quantidade de sucessos nos n ensaios cuja distribuição é

$$\binom{n}{t} p^t (1-p)^{n-t} = b_{n,p}(t)$$

para todo $t \in \{0, 1, \ldots, n\}$.

Também, se $\max\colon \mathbb{R}^n \to \mathbb{R}$ é a função que calcula o maior dentre n valores reais, então temos que $\max(X_1, X_2, \ldots, X_n)$ é uma variável aleatória real.

Proposição 3.10 *Sejam X e Y variáveis aleatórias reais independentes e f e g funções de $\mathbb{R}$ em $\mathbb{R}$. Então as variáveis aleatórias $f(X)$ e $g(Y)$ são independentes.*

Demonstração. Se X e f são como no enunciado e $a \in \mathbb{R}$, então $[f(X) = a] = [X \in A]$ em que $A := \{x \in X(\Omega)\colon f(x) = a\}$. Analogamente, para Y, g e $b \in \mathbb{R}$ dados, temos $[g(Y) = b] = [Y \in B]$ em que $B := \{x \in X(\Omega)\colon g(x) = b\}$. De X e Y independentes temos $\mathbb{P}([X \in A] \cap [Y \in B]) = \mathbb{P}[X \in A] \cdot \mathbb{P}[Y \in B]$, portanto, $\mathbb{P}([f(X) = a] \cap [g(Y) = b]) = \mathbb{P}[f(X) = a] \cdot \mathbb{P}[g(Y) = b]$, ou seja, $f(X)$ e $g(Y)$ são variáveis aleatórias independentes. □

Exercício 3.11. Determine a distribuição da soma de duas variáveis de Poisson e a distribuição da soma de duas variáveis binomiais com mesmo p.

3.1.1 VALOR ESPERADO DE UMA VARIÁVEL ALEATÓRIA SIMPLES

Uma variável aleatória de $(\Omega, \mathbb{P})$ com imagem finita é chamada de **variável aleatória simples**. O **valor médio** de X é a média dos valores de $X(\Omega) = \{x_1, x_2, \ldots, x_n\}$ ponderada pela probabilidade de cada valor

$$\mathbb{E}\, X := x_1 \mathop{\mathbb{P}}_{X}(x_1) + x_2 \mathop{\mathbb{P}}_{X}(x_2) + \cdots + x_n \mathop{\mathbb{P}}_{X}(x_n) \tag{3.2}$$

e também chamado de **valor esperado** ou, ainda, **esperança** da variável aleatória simples.

Exemplo 3.12. Consideremos um jogo de azar no qual em cada aposta ou ganhamos R$1.000.000,00 com probabilidade $p \in (0, 1)$ ou perdemos R$10,00 com probabilidade $1 - p$. Se Y é o valor ganho numa aposta, então a esperança de ganho numa aposta é $\mathbb{E}\, Y = 10^6 p - 10(1 - p)$. No caso de $p = 1/2$, o prêmios são equiprováveis e o ganho médio é $\mathbb{E}\, Y = 499.995{,}00$. A probabilidade de ganharmos R$499.995,00 numa aposta é zero. Se $p = 1/100$ então a probabilidade de ganhar o valor alto é muito pequeno quando comparado com a probabilidade de perder 10 reais e o valor esperado de ganho numa única aposta é $\mathbb{E}\, Y = 9.990{,}10$. ◊

Se tomamos a partição de Ω dada por $A_k = [X = x_k]$ para todo k, então podemos escrever

$$X = \sum_{k=1}^{n} x_k \mathbb{1}_{A_k}$$

e o valor médio de X é dada em função de $\mathbb{P}$ ao invés de $\mathbb{P}_X$ por

$$\mathbb{E}\, X = x_1 \mathbb{P}(A_1) + x_2 \mathbb{P}(A_2) + \cdots + x_n \mathbb{P}(A_n). \tag{3.3}$$

O valor esperado não depende de como escrevemos X como combinação linear de variáveis indicadoras. Seja $\{B_\ell \colon 1 \leqslant \ell \leqslant m\}$ uma partição *qualquer* de Ω tal que

$$\sum_{k=1}^{n} x_k \mathbb{1}_{A_k} = \sum_{\ell=1}^{m} y_\ell \mathbb{1}_{B_\ell}$$

com possíveis valores repetidos para os y_ℓ's. Então para todo k

$$A_k = \bigcup_{\ell \colon y_\ell = x_k} B_\ell$$

logo

$$\sum_{k=1}^{n} x_k \mathbb{P}(A_k) = \sum_{k=1}^{n} x_k \sum_{\ell \colon y_\ell = x_k} \mathbb{P}(B_\ell) = \sum_{k=1}^{n} \sum_{\ell \colon y_\ell = x_k} y_\ell \mathbb{P}(B_\ell)$$

portanto

$$\sum_{k=1}^{n} x_k \mathbb{P}(A_k) = \sum_{\ell=1}^{m} y_\ell \mathbb{P}(B_\ell). \tag{3.4}$$

No teorema abaixo daremos algumas propriedades importantes do valor esperado. Na demonstração desses resultados usaremos o seguinte exercício cuja verificação é simples.

Exercício 3.13. Sejam $X = \sum_{k=1}^{n} x_k \mathbb{1}_{A_k}$ e $Y = \sum_{\ell=1}^{m} y_\ell \mathbb{1}_{B_\ell}$ duas variáveis aleatórias simples de $(\Omega, \mathbb{P})$. Verifique que valem as seguintes identidades (dica: exercício 3.52, item 3.52, página identidades (dica: exercício 3.52, item 4, página 170)

1. para quaisquer $a, b \in \mathbb{R}$
$$aX + bY = \sum_{k=1}^{n} \sum_{\ell=1}^{m} (ax_k + by_\ell) \mathbb{1}_{A_k \cap B_\ell};$$
2. $$X \cdot Y = \sum_{k=1}^{n} \sum_{\ell=1}^{m} (x_k \cdot y_\ell) \mathbb{1}_{A_k \cap B_\ell}.$$

Teorema 3.14 (propriedades do valor esperado) *Seja X uma variável aleatória simples definida no espaço amostral Ω munido da medida de probabilidade $\mathbb{P}$.*

1. *Se $X(\omega) = c$ para todo $\omega \in \Omega$ então $\mathbb{E}\, X = c$.*
2. *Para todo evento A, $\mathbb{E}\, \mathbb{1}_A = \mathbb{P}(A)$.*
3. Linearidade: *se Y é uma variável aleatória simples e a e b números reais então*
$$\mathbb{E}[aX + bY] = a\mathbb{E}\, X + b\mathbb{E}\, Y. \tag{3.5}$$

4. Monotonicidade: *se* Y *é uma variável aleatória simples e* $\mathsf{X} \leqslant \mathsf{Y}$, *isto é,* $\mathsf{X}(\omega) \leqslant \mathsf{Y}(\omega)$ *para todo* $\omega \in \Omega$ *então*

$$\mathbb{E}\,\mathsf{X} \leqslant \mathbb{E}\,\mathsf{Y}.$$

5. *Se* f *é uma função real e* $\mathsf{X}(\Omega) = \{x_1, x_2, x_3, \ldots, x_n\}$ *então*

$$\mathbb{E}[\,f(\mathsf{X})\,] = \sum_{k=1}^{n} f(x_k)\mathbb{P}_{\mathsf{X}}(x_k).$$

6. *Se* X *e* Y *são variáveis aleatórias simples e independentes então* $\mathbb{E}[\mathsf{XY}] = \mathbb{E}[\mathsf{X}]\,\mathbb{E}[\mathsf{Y}]$.

Demonstração. As demonstrações dos itens 1 e 2 são imediatas das equações (3.2) e (3.3), respectivamente. Para provarmos os itens 3, 4 e 6 consideramos X e Y tais que $\mathsf{X}(\Omega) = \{x_1, \ldots, x_n\}$ e $\mathsf{Y}(\Omega) = \{y_1, \ldots, y_m\}$, também as partições $\mathsf{A}_k = \{\omega\colon \mathsf{X}(\omega) = x_k\}$ e $\mathsf{B}_\ell = \{\omega\colon \mathsf{Y}(\omega) = y_\ell\}$ de Ω, de modo que $\mathsf{X} = \sum_{k=1}^{n} x_k \mathbb{1}_{\mathsf{A}_k}$ e $\mathsf{Y} = \sum_{\ell=1}^{m} y_\ell \mathbb{1}_{\mathsf{B}_\ell}$.

Sejam a e b reais arbitrários. Então, usando a definição (3.2), o item 1 do exercício 3.13 acima e o item 2 deste teorema, temos

$$\mathbb{E}[\,a\mathsf{X} + b\mathsf{Y}\,] = \sum_{k=1}^{n}\sum_{\ell=1}^{m} (ax_k + by_\ell)\mathbb{P}(\mathsf{A}_k \cap \mathsf{B}_\ell) = \sum_{k=1}^{n}\sum_{\ell=1}^{m} ax_k\mathbb{P}(\mathsf{A}_k \cap \mathsf{B}_\ell) + by_\ell\mathbb{P}(\mathsf{A}_k \cap \mathsf{B}_\ell)$$

e rearranjando as somas deduzimos

$$\mathbb{E}[\,a\mathsf{X} + b\mathsf{Y}\,] = \sum_{k=1}^{n}\sum_{\ell=1}^{m} ax_k\mathbb{P}(\mathsf{A}_k \cap \mathsf{B}_\ell) + \sum_{\ell=1}^{m}\sum_{k=1}^{n} by_\ell\mathbb{P}(\mathsf{A}_k \cap \mathsf{B}_\ell) = \sum_{k=1}^{n} ax_k\mathbb{P}(\mathsf{A}_k) + \sum_{\ell=1}^{m} by_\ell\mathbb{P}(\mathsf{B}_\ell)$$

onde a segunda igualdade segue do fato das famílias de eventos $\{\mathsf{A}_1, \ldots, \mathsf{A}_n\}$ e $\{\mathsf{B}_1, \ldots, \mathsf{B}_m\}$ formarem, cada uma, uma partição do espaço amostral, portanto, $\mathbb{E}[\,a\mathsf{X} + b\mathsf{Y}\,] = a\,\mathbb{E}\,\mathsf{X} + b\,\mathbb{E}\,\mathsf{Y}$.

Se $\mathsf{X} \leqslant \mathsf{Y}$ então $\mathsf{Y} - \mathsf{X} \geqslant 0$, portanto $\mathbb{E}[\,\mathsf{Y} - \mathsf{X}\,] \geqslant 0$. Usando a linearidade $\mathbb{E}[\,\mathsf{Y} - \mathsf{X}\,] = \mathbb{E}\,\mathsf{Y} - \mathbb{E}\,\mathsf{X} \geqslant 0$, donde deduzimos que $\mathbb{E}\,\mathsf{X} \leqslant \mathbb{E}\,\mathsf{Y}$.

Se X e Y são independentes então, usando o item 2 do exercício 3.13 acima,

$$\mathbb{E}[\,\mathsf{XY}\,] = \sum_{k=1}^{n}\sum_{\ell=1}^{m} x_k y_\ell \mathbb{P}(\mathsf{A}_k \cap \mathsf{B}_\ell) = \sum_{k=1}^{n}\sum_{\ell=1}^{m} x_k y_\ell \mathbb{P}(\mathsf{A}_k)\mathbb{P}(\mathsf{B}_\ell) = \sum_{k=1}^{n} x_k\mathbb{P}(\mathsf{A}_k)\sum_{\ell=1}^{m} y_\ell\mathbb{P}(\mathsf{B}_\ell) = \mathbb{E}\,\mathsf{X}\cdot\mathbb{E}\,\mathsf{Y}$$

o que prova o item 6.

Para o item 5 façamos $\mathsf{Y} := f(\mathsf{X})$ de modo que

$$f(\mathsf{X}) = \sum_{\ell=1}^{m} y_\ell\,\mathbb{1}_{\mathsf{B}_\ell}$$

com $B_\ell = [Y = y_\ell]$. Agora, notemos que para cada $k \in \{1,2,\ldots,n\}$ existe um único $\ell \in \{1,2,\ldots,m\}$ tal que $A_k \subset B_\ell$, a saber o ℓ tal que $f(x_k) = y_\ell$. Seja $I_\ell := \{k\colon 1 \leqslant k \leqslant n,\ f(x_k) = y_\ell\}$. De fato, temos (verifique)

$$B_\ell = \bigcup_{k \in I_\ell} A_k$$

sendo a união de conjuntos disjuntos, assim

$$f(X) = \sum_{\ell=1}^{m} y_\ell \mathbb{1}_{B_\ell} = \sum_{\ell=1}^{m} \sum_{k \in I_\ell} y_\ell \mathbb{1}_{A_k} = \sum_{\ell=1}^{m} \sum_{k \in I_\ell} f(x_k) \mathbb{1}_{A_k} = \sum_{k=1}^{n} f(x_k) \mathbb{1}_{A_k}$$

e pela equação (3.4) temos de $f(X) = \sum_{\ell=1}^{m} y_\ell \mathbb{1}_{B_\ell} = \sum_{k=1}^{n} f(x_k) \mathbb{1}_{A_k}$ que $\mathbb{E}\, f(X) = \sum_{k=1}^{n} f(x_k) \mathbb{P}(A_k) = \sum_{k=1}^{n} f(x_k)\, \mathbb{P}_X(x_k)$. □

Em geral, $\mathbb{E}[X \cdot Y] \neq \mathbb{E}\, X \cdot \mathbb{E}\, Y$. Por exemplo, se X é o resultado do lançamento de um dado equilibrado então, pelo item 5 do teorema acima, $\mathbb{E}[X \cdot X] = \sum_{n=1}^{6} n^2\, \mathbb{P}_X(n) = 91/6 \neq 49/4 = \mathbb{E}\, X \cdot \mathbb{E}\, X$.

Corolário 3.15 *Se $X_1,\ldots,X_n$ são variáveis aleatórias simples e $a_1,\ldots,a_n$ números reais então*

$$\mathbb{E}\left[\sum_{i=1}^{n} a_i X_i\right] = \sum_{i=1}^{n} a_i\, \mathbb{E}\, X_i.$$

Demonstração. Segue da equação (3.5) usando indução em n. □

Exemplo 3.16 (*esperança das distribuições Bernoulli e binomial*). Se X tem distribuição de Bernoulli $\mathbb{E}\, X = \mathbb{P}[X = 1] = p$ em que p.

Se $X_1,\ldots,X_n$ são variáveis Bernoulli independentes e $X = \sum_i X_i$ então X é binomial com parâmetros n e p como vimos acima. Ademais, pela linearidade da esperança, corolário acima, temos

$$\mathbb{E}\, X = \sum_{i=1}^{n} \mathbb{E}\, X_i = np$$

pois $\mathbb{E}\, X_i = p$ para todo i. ◊

Exemplo 3.17. Se $X \in_R \{1,2,3,4,5,6\}$ é o resultado de um lançamento de um dado, então

$$\mathbb{E}\, X = 1\frac{1}{6} + 2\frac{1}{6} + 3\frac{1}{6} + 4\frac{1}{6} + 5\frac{1}{6} + 6\frac{1}{6} = \frac{7}{2}$$

que também é a esperança de qualquer variável aleatória $Y \in_R S$ para qualquer conjunto S com $|S| = 6$. Agora, se Y é o resultado de outro lançamento de dado, qual é a esperança da soma $X + Y$ dos pontos no lançamento de dois dados? Pela linearidade da esperança é 7. Alternativamente, a distribuição da soma é mostrada na tabela 3.1 abaixo, donde podemos calcular o valor esperado da soma $\mathbb{E}[X + Y] =$

$X+Y$	2	3	4	5	6	7	8	9	10	11	12
$\mathbb{P}_{X+Y}$	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

Tabela 3.1: distribuição da soma de dois dados.

$X\cdot Y$	1	2	3	4	5	6	8	9	10	12	15	16	18	20	24	25	30	36
$\mathbb{P}_{X\cdot Y}$	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{4}{36}$	$\frac{2}{36}$	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{4}{36}$	$\frac{2}{36}$	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

Tabela 3.2: distribuição do produto de dois dados.

$(2\cdot 1+3\cdot 2+\cdots+11\cdot 2+12\cdot 1)/36 = 7$. O produto XY tem valor esperado $7/2\cdot 7/2$, pelo item 6 do teorema acima já que as variáveis são independentes. Alternativamente, a distribuição do produto é mostrada na tabela 3.2, donde podemos calcular o valor esperado para o produto dos resultados, $\mathbb{E}[X\cdot Y] = 49/4$. ◊

3.1.2 TABELAS DE ESPALHAMENTO

Em computação um conjunto que é modificado com passar do tempo é usualmente chamado de **conjunto dinâmico**. Uma solução bastante conhecida e estudada para a representação computacional de um conjunto dinâmico S de modo que possamos inserir elementos, remover elementos e testar pertinência é conhecida como **tabela de espalhamento** ou **tabela *hashing***. Uma maneira de implementar uma tabela de espalhamento N é construir um vetor $N[i]$ de listas ligadas indexadas por $M = \{0,1,\ldots,m-1\}$ e o acesso à tabela dá-se por uma função de *hash* $h\colon U \to M$: dado $x \in U$, a inserção, remoção ou busca por x em N é feita na lista ligada $N[h(x)]$ (veja uma ilustração na figura 3.1).

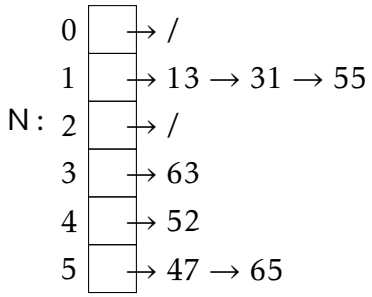


Figura 3.1: tabela de espalhamento com os elementos $\{13,31,47,52,55,63,65\}$ distribuídos de acordo com a função $h(x) = x \bmod 6$.

As operações busca, inserção e remoção de um elemento numa tabela N que representa um conjunto dinâmico $S \subset U$ com função de *hash* $h\colon U \to M$ são descritas a seguir e são chamadas de *operações*

de dicionário:

busca: dado $x \in U$, uma operação de busca por x em S responde a pergunta "$x \in S$?" e ainda, no caso positivo, retorna um apontador para x. Numa tabela de espalhamento isso é resolvido por uma busca sequencial na lista ligada $N[h(x)]$. Uma busca por x em N é dita *com sucesso* caso $x \in N$, senão é dita *sem sucesso*. O custo (ou tempo) de pior caso de uma busca é proporcional ao número de elementos na maior lista;

inserção: dado $x \in U$, uma operação de inserção acrescenta na estrutura que representa S o elemento x, caso esse ainda não faça parte de S. A inserção propriamente dita numa tabela de espalhamento é simplesmente colocar o elemento x no fim da lista ligada $N[h(x)]$ e o custo de pior caso dessa operação é constante;

remoção: dado $x \in S$, a remoção de x retira esse elemento da estrutura que representa S. A remoção propriamente dita numa tabela de espalhamento, dada a posição de x na lista ligada através de um apontador, é a remoção do elemento de uma lista ligada e o custo de pior caso dessa operação é constante.

Com essas descrições é natural constatar que para uma análise do desempenho das operações de dicionário nessa estrutura de dados é relevante o tempo de busca de um item que, no pior caso, é o tempo de busca na lista $N[i]$ com o maior número de elementos. Ademais, a pior configuração que podemos ter para o desempenho desses algoritmos ocorre quando todos os elementos de S são mapeados para a mesma lista ligada.

No emprego de tabelas de espalhamento, usualmente, temos a seguinte situação: o conjunto universo U é muito grande, S é uma fração pequena de U e o caso interessante é quando $|S| \geqslant |M|$. Como $|U| > |M|$ não há como evitar *colisões* (elementos distintos mapeados para a mesma lista). Mais que isso, se o conjunto universo U é suficientemente grande, digamos $|U| > (n-1)|M|$, então para qualquer função de *hash* $h \in M^U$ sempre haverá um conjunto S de cardinalidade n para o qual uma configuração de pior caso é inevitável.

Das funções *hashing* queremos que a probabilidade de colisão seja pequena, que a função seja fácil de computar e que a representação seja sucinta, isto é, relativamente poucos bits são necessários para armazenar a função. Essas funções, além de uma grande ferramenta prática como descrevemos abaixo, é bastante útil em Complexidade Computacional e em Criptografia e voltaremos a estudar funções de *hash* em outras seções adiante neste texto.

Uma função h escolhida uniformemente no conjunto M^U de todas as $|M|^{|U|}$ funções de U em M tem, com alta probabilidade, a propriedade de não formar listas longas (veja o exercício 1.55, página 44), entretanto, tal função pode não ter uma representação sucinta, alguma delas requer pelo menos $|U|\log(|M|)$ bits para ser representada. Idealmente, o que procuramos são funções que tenham

representação sucinta e imitam uma função aleatória na propriedade de não formar listas longas e, também, que possam ser computadas eficientemente em cada ponto do domínio.

Por ora analisaremos o caso de uma função aleatória para um conjunto S fixo porém arbitrário, nesse caso o tamanho de uma lista é uma variável aleatória. Vamos considerar o modelo probabilístico $(\mathsf{M}^{\mathsf{U}}, \mathbb{P})$ com $\mathbb{P}(h) = 1/|\mathsf{M}^{\mathsf{U}}|$ para toda função $h \in \mathsf{M}^{\mathsf{U}}$. Notemos que vale

$$\mathbb{P}[\, h(x) = i \,] = \frac{1}{|\mathsf{M}|} \tag{3.6}$$

para todo $x \in \mathsf{U}$ e todo $i \in \mathsf{M}$. De fato, sortear uniformemente uma função é equivalente a sortear uniformemente e independentemente uma imagem para cada elemento do domínio (veja o exercício 3.9, página 122), isto é, $h \in_{\mathrm{R}} \mathsf{M}^{\mathsf{U}}$ é equivalente a $h(x) \in_{\mathrm{R}} \mathsf{M}$ com as escolhas $h(x)$ independentes, para todo $x \in \mathsf{U}$.

Fixemos os parâmetros $n = |\mathsf{S}|$ e $m = |\mathsf{M}|$. A fração

$$\mu = \mu(n, m) := \frac{n}{m} \tag{3.7}$$

é denominada **carga** da tabela. A carga da tabela é o valor esperado para a quantidade de elementos de S numa mesma posição da tabela segundo a medida da equação (3.6). Tomemos

$$\mathbb{1}_{[h(x)=h(y)]} \colon \mathsf{M}^{\mathsf{U}} \to \{0, 1\}$$

a variável aleatória indicadora de colisão dos elementos $x, y \in \mathsf{U}$ sob a escolha $h \in \mathsf{M}^{\mathsf{U}}$. A probabilidade de colisão é

$$\mathbb{P}[\, h(x) = h(y) \,] = \frac{1}{m}$$

sempre que $x \neq y$, Assim, o número de itens na lista $\mathsf{N}[h(x)]$ é dado por $\sum_{y \in \mathsf{S}} \mathbb{1}_{[h(x)=h(y)]}$ e

$$\mathbb{E}\, \mathbb{1}_{[h(x)=h(y)]} = \begin{cases} \frac{1}{m}, & \text{se } y \neq x \\ 1, & \text{caso contrário,} \end{cases}$$

portanto o tamanho esperado da lista $\mathsf{N}[h(x)]$ é, pela linearidade da esperança (corolário 3.15)

$$\sum_{y \in \mathsf{S}} \mathbb{E}\, \mathbb{1}_{[h(x)=h(y)]} = \begin{cases} \frac{n}{m}, & \text{se } x \notin \mathsf{S}, \\ \frac{(n-1)}{m} + 1, & \text{caso contrário.} \end{cases} \tag{3.8}$$

O tempo esperado de uma busca em uma tabela de espalhamento com uma função *hash* h escolhida aleatoriamente em M^{U} é proporcional à carga μ da tabela.

Proposição 3.18 *O tempo esperado para uma busca sem sucesso é* $\mu + 1$ *e o tempo esperado para uma busca com sucesso é* $\mu/2 - 1/(2m) + 1$.

Demonstração. Consideremos uma busca por $x \in U$ na tabela de espalhamento N. Se $x \notin S$ então, pelo primeiro caso da equação (3.8), são necessárias $\mu + 1$ comparações em média numa busca sem sucesso. Agora, suponhamos $x \in S$ e que os elementos de S foram inseridos sequencialmente. Se x foi o i-ésimo item inserido em N, então o tamanho esperado da lista imediatamente após a inserção é $(i-1)/m + 1$ pela equação (3.8), portanto, o tempo médio da busca por x é

$$\frac{1}{n}\sum_{i=1}^{n} \frac{i-1}{m} + 1 = \frac{\mu}{2} - \frac{1}{2m} + 1$$

que é o número médio de comparações numa busca com sucesso. □

Essa análise não nos dá nenhuma pista sobre o tempo esperado para uma busca no pior caso. No caso $n = m$ a proposição acima garante que o tempo médio de busca é constante, entretanto, do exercício 1.55, página 44, a maior lista tem $O(\log n)$ elementos com alta probabilidade. Mais especificamente, provaremos na seção 3.3.3, página 168, que a maior lista tem $\Theta(\log(n)/\log(\log(n)))$ elementos.

Exemplo 3.19 (*paradoxo dos aniversários*). Para $n \leqslant m$ há $m(m-1)(m-2)\cdots(m-n+1)$ sequências sem repetições em M^n. Se $(h(x_1), h(x_2), \ldots, h(x_n))$ é uma escolha aleatória em M^n e C é o número de colisões, então nessa escolha

$$\begin{aligned}
\mathbb{P}[\,C = 0\,] &= \frac{m(m-1)(m-2)\cdots(m-n+1)}{m^n} \\
&= \left(1 - \frac{1}{m}\right)\left(1 - \frac{2}{m}\right)\cdots\left(1 - \frac{n-2}{m}\right)\left(1 - \frac{n-1}{m}\right) \\
&< \exp\left(-\frac{1}{m}\right)\exp\left(-\frac{2}{m}\right)\cdots\exp\left(-\frac{n-1}{m}\right) \\
&= \exp\left(-\frac{n(n-1)}{2m}\right)
\end{aligned}$$

na segunda linha usamos que $\exp(-x) > 1 - x$ (veja (d.1)). O conhecido *paradoxo dos aniversários* é o caso $n = 23$ e $m = 365$ na equação acima: $\mathbb{P}[\,C > 0\,] > 1 - \mathbb{P}[\,C = 0\,] > 0{,}5$, ou seja, apenas 23 pessoas são suficientes para que duas delas façam aniversário no mesmo dia com probabilidade maior que 1/2, supondo que os nascimentos ocorram uniformemente ao longo do ano.

Para todo real $x \in [0, 3/4]$ vale que $1 - x > \exp(-2x)$, portanto para $n \leqslant (3/4)m$ temos o seguinte limitante para a probabilidade de não ocorrer colisão

$$\left(1 - \frac{1}{m}\right)\left(1 - \frac{2}{m}\right)\cdots\left(1 - \frac{n-2}{m}\right)\left(1 - \frac{n-1}{m}\right) > \exp\left(-\frac{n(n-1)}{m}\right)$$

de modo que se n é aproximadamente $\sqrt{m}$ então a probabilidade de não haver colisão é maior que $\exp(-1) \approx 0{,}36$, logo com boa probabilidade uma função aleatória de espalhamento consegue espalhar $\sqrt{m}$ itens até que ocorra a primeira colisão (veja o exemplo 6.11, página 312). ◊

Notemos que para S fixo o número esperado de colisões é

$$\binom{n}{2}\frac{1}{m} = \frac{n(n-1)}{2m}$$

assim, se $m = n^2$ então $\mathbb{E}\,C < 1/2$ e, de fato, não há colisão com probabilidade pelo menos 1/2 pois

$$\frac{1}{2} > \mathbb{E}\,C = \sum_{k=0}^{\binom{n}{2}} k\mathbb{P}[C = k] = \sum_{k=1}^{\binom{n}{2}} \mathbb{P}[C = k] = \mathbb{P}[C \geqslant 1].$$

HASHING UNIVERSAL Se em vez de exigirmos que seja válida a equação (3.6) para todo x e todo i, pedirmos que numa família de funções $\mathcal{H} \subset M^U$ seja válida a condição

$$\mathbb{P}_{h\in_R\mathcal{H}}[h(x) = h(y)] \leqslant \frac{1}{m} \tag{3.9}$$

para todo $x \neq y$ ainda será verdade que o tamanho médio de uma lista é limitado pela equação (3.7), que a sentença enunciada na proposição 3.18 vale, que o número esperado de colisões é $O(n^2/2m)$ e no caso $m = n^2$ não há colisão com probabilidade pelo menos 1/2. Uma família $\mathcal{H} \subset M^U$ de funções que satisfazem a equação (3.9) é chamada de *universal*.

Como não precisamos de funções genuinamente aleatórias para termos as boas propriedades estatísticas dessas funções nos resta procurar por uma família $\mathcal{H}$ universal e não vazia de funções *hashing* com representações sucintas e que são calculadas eficientemente. Um exemplo é dado a seguir. Tomamos $U := \{0,1\}^u$ e $M := \{0,1\}^m$ e definimos uma função $h\colon U \to M$ sorteando os bits de uma matriz $A = (a_{\ell,c})$ de dimensão $m \times u$ e fazendo $h(x) := A \cdot x$ para todo $x \in U$, com as operações módulo 2. Para elementos distintos $x, y \in U$ teremos $h(x) = h(y)$ se, e somente se, $h(x-y) = A\cdot(x-y) = 0$ pois h é uma função linear. Nesse caso

$$\sum_{c=1}^{u} a_{\ell,c}(x-y)_c = 0 \text{ para cada } \ell \in \{1,\ldots,m\}.$$

Seja i uma coordenada não nula de $x-y$. Se sorteamos todos os bits de A exceto os da coluna i então há uma única escolha para cada bit da coluna i para que a equação acima se verifique, a saber

$$a_{\ell,i} = \sum_{\substack{c=1\\c\neq i}}^{u} a_{\ell,c}(x-y)_c$$

para cada $\ell \in \{1,\ldots,m\}$, o que ocorre com probabilidade $1/2^m$, ou seja, $\mathbb{P}_{h\in\mathcal{H}}[h(x) = h(y)] \leqslant 1/|M|$. Observemos que $h(0) = 0$ para qualquer escolha de A o que faz com que a equação (3.6) não seja válida para todo x e todo i.

Exemplo 3.20. Definimos uma família de funções com domínio $U = \{0, 1, \ldots, N-1\}$ e contradomínio $M = \{0, 1, \ldots, m-1\}$ com $N > m$ por

$$\left\{ h_{(a,b)} \coloneqq (ax + b \bmod p) \bmod m \colon a, b \in \{0, 1, \ldots, p-1\},\ a \neq 0 \right\}$$

para $p > N$ primo fixo.

Se $x \neq y$ então $ax + b \bmod p \neq ay + b \bmod p$, pois p é primo. Ademais, para $i, j \in \{0, 1, \ldots, p-1\}$ distintos existe exatamente um par (a, b) tal que

$$\begin{cases} ax + b \bmod p = i \\ ay + b \bmod p = j \end{cases}$$

pois do sistema acima na forma matricial

$$\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} i \\ j \end{pmatrix}$$

temos uma matriz quadrada com determinante diferente de zero, logo os valores para a e b que satisfazem o sistema linear são únicos.

Para cada $i \in \{0, 1, \ldots, p-1\}$, há $\leqslant \lceil p/m \rceil - 1$ possíveis valores para j tais que $i \equiv j \pmod m$, portanto, no máximo $p(p-1)/m$ pares. Assim, para $x \neq y$, uma escolha aleatória de função nos dá

$$\mathbb{P}\Big[h_{(a,b)}(x) = h_{(a,b)}(y) \Big] \leqslant \frac{p(p-1)/m}{p(p-1)} = \frac{1}{m}.$$

Qualquer membro dessa família pode ser representado unicamente pela tripla (a, b, p) de modo que cada função precisa de no máximo $3 \log(2|U|)$ bits. Além disso, a função contém um número constante de operações que levam tempo constante para avaliar, logo a função pode ser calculada em tempo constante ◊

3.1.3 ESPERANÇA MATEMÁTICA

Seja X uma variável aleatória real sobre o espaço discreto de probabilidade $(\Omega, \mathbb{P})$ com imagem enumerável infinita. Se estendemos de modo natural a definição de valor esperado de uma variável simples dada na equação (3.2), página 123, tomamos $X(\Omega) = \{x_1, x_2, \ldots\}$ e temos

$$\mathbb{E}\, X = \sum_{n \geqslant 1} x_n \mathbb{P}_X(x_n) \tag{3.10}$$

entretanto esse valor, caso exista já que é um limite (a definição e algumas propriedades importantes podem ser vistas na página 335), não deve depender da enumeração particular $x_1, x_2, \ldots$ da imagem de X. Uma leitura atenta nas demonstrações das propriedades de valor esperado de variável simples

revela que a reordenação dos termos da soma (sem alterar o resultado) é uma operação importante e que queremos preservar.

Seja $(x_n\colon n \geqslant 1)$ uma sequência de números reais. Se $x_n \geqslant 0$ para todo n então a série $\sum_{n\geqslant 1} x_n$ pode ser *finita*, isto é convergir para um número real, ou *infinita*, isto é tender ao infinito, o que denotamos por $\sum_n x_n = +\infty$. Em ambos os casos dizemos que a série é *bem definida*. Agora, no caso geral, definimos

$$X^+ := \sum_{n\colon x_n \geqslant 0} x_n \quad \text{e} \quad X^- := \sum_{n\colon x_n < 0} |x_n|$$

e pode acontecer de

- se ambas as séries são finitas então $\sum_n x_n = X^+ - X^-$ e a série *está bem definida*. Além disso, nesse caso a série $\sum_n x_n$ é *absolutamente convergente*, isto é, $\sum_{n\geqslant 1} |x_n|$ converge.
- Se $X^+ = +\infty$ e X^- é finita, então $\sum_n x_n := +\infty$ e, analogamente, se $X^- = +\infty$ e S^+ é finita, então $\sum_n x_n := -\infty$. Em ambos os casos a série $\sum_n x_n$ não é absolutamente convergente, porém *está bem definida* e é infinita.
- Se $X^+ = X^- = +\infty$, então a série $\sum_n x_n$ é *indefinida*.

A propriedade importante para nós é que *sempre que a série $\sum_{n\geqslant 1} x_n$ está bem definida uma permutação π qualquer na ordem dos termos da sequência não altera resultado*, isto é, $\sum_{n\geqslant 1} x_n = \sum_{n\geqslant 1} x_{\pi(n)}$. Isso não vale no caso indefinido, $\sum_n x_{\pi(n)}$ pode dar resultados diferentes para diferentes permutações π (veja equação (3.13) abaixo). Em vista disso, se a série na equação (3.10) está bem definida então a série não depende da enumeração de $X(\Omega)$ e expressa a esperança $\mathbb{E}\,X$ de X.

A **esperança matemática**, ou **valor médio** ou **valor esperado**, da variável aleatória discreta e real $X\colon \Omega \to S$ é

$$\mathbb{E}\,X := \sum_{r\in S} r\,\mathbb{P}_X(r). \tag{3.11}$$

sempre que a série (3.10) está bem definida. Ademais, a esperança é um número real quando a série converge absolutamente, isto é, quando quando $\mathbb{E}\,|X| := \sum_{r\in S} |r|\,\mathbb{P}_X(r)$ converge. Nesse último caso escrevemos $\mathbb{E}\,|X| < +\infty$ e dizemos que a variável aleatória X tem **esperança finita** ou que X é **somável**.

Nos casos em que o valor esperado da variável aleatória X está definido (convergente ou não) podemos reordenar os termos da soma de modo que podemos escrever

$$\mathbb{E}\,X = \sum_{\omega\in\Omega} X(\omega)\,\mathbb{P}(\omega) \tag{3.12}$$

que é útil na prática. Assim, X é somável sempre que $\mathbb{E}\,|X| = \sum_\omega |X(\omega)|\,\mathbb{P}(\omega) < +\infty$.

No que segue chamamos genericamente de *soma* tanto um somatório com finitos termos quanto uma série.

Exemplo 3.21 (*esperança da distribuição geométrica*). Se X é geométrica de parâmetro $p \in (0,1)$, então

$$\mathbb{E}\, X = \sum_{n \geqslant 1} n\, \mathbb{P}[X = n] = \sum_{n \geqslant 1} n(1-p)^{n-1} p = p \sum_{n \geqslant 0} n(1-p)^n = \frac{1}{p}$$

usando (s.6a). ◊

Os próximos exemplos são variáveis aleatórias com esperança infinita. Numa urna estão 1 bola branca e 1 bola preta; uma bola é escolhida ao acaso, se for preta ela é devolvida e mais uma bola preta é colocada na urna e o sorteio é repetido, se sair bola branca o experimento termina. Depois de $m \geqslant 0$ sorteios de bolas pretas há na urna $m+2$ bolas com $m+1$ delas da cor preta. No próximo sorteio, a probabilidade de sair uma bola branca, dado que saíram m bolas pretas, é $1/(m+2)$ e a probabilidade de sair uma bola preta é $(m+1)/(m+2)$, portanto, pelo teorema da multiplicação (página 26), a probabilidade do experimento terminar no $m+1$-ésimo sorteio, para $m \geqslant 1$, é

$$\left(\prod_{j=1}^{m} \frac{j}{j+1}\right) \frac{1}{m+2} = \frac{1}{2}\frac{2}{3}\frac{3}{4} \cdots \frac{m}{m+1}\frac{1}{m+2} = \frac{1}{(m+1)(m+2)}$$

e a probabilidade do experimento terminar no primeiro sorteio é $1/2$. O número esperado de sorteios no experimento é

$$1\frac{1}{2} + \sum_{m \geqslant 1} (m+1) \frac{1}{(m+1)(m+2)} = \sum_{m \geqslant 1} \frac{1}{m} = +\infty$$

essa última é a famosa série harmônica.

O *paradoxo de São Petersburgo* é sobre um jogo de aposta que consiste em pagar uma certa quantia para poder participar de uma rodada. Uma rodada consiste em jogar uma moeda até sair coroa, se o número de lançamentos for igual n então o valor pago ao jogador é 2^n reais. Quanto você estaria disposto a pagar para jogar esse jogo? O ganho esperado é $\sum_{n \geqslant 1} 2^n 2^{-n} = +\infty$. O fato de que a esperança é infinita sugere que um jogador deveria estar disposto a pagar qualquer quantia fixa pelo privilégio de participar do jogo, mas é improvável que alguém esteja disposto a pagar muito e aí está o paradoxo.

Uma série conhecida pela propriedade de convergir não-absolutamente é a série harmônica alternada

$$1 - \frac{1}{2} + \frac{1}{3} - \cdots + \frac{(-1)^{n+1}}{n} + \cdots = \log(2)$$

cuja convergência decorre da série de Taylor para o logaritmo natural. A série formada pelos valores absolutos é a série harmônica, que não converge. O seguinte rearranjo dessa série devido ao matemático Pierre Alphonse Laurent converge para um valor diferente

$$1 - \frac{1}{2} - \frac{1}{4} + \frac{1}{3} - \frac{1}{6} - \frac{1}{8} + \cdots + \frac{1}{2k-1} - \frac{1}{2(2k-1)} - \frac{1}{4k} + \cdots = \frac{\log(2)}{2}.$$

Com isso podemos moldar uma variável aleatória sem valor esperado. Tomemos uma variável aleatória X que assume os valores $x_1, x_2, \ldots$ dados por $x_i := (-1)^{i+1}/i$ para $i = 1, 2, \ldots$ com $\mathbb{P}_X(x_i) = 6/(\pi i)^2$.

Aqui usamos que $\sum_{i\geqslant 1} 1/i^2 = \pi^2/6$ (veja (s.7)) para garantir que $\mathbb{P}_X$ seja distribuição. Com essas definições

$$\sum_{i\geqslant 1} x_{\pi(i)} \mathop{\mathbb{P}}_{X}(x_{\pi(i)}) = \begin{cases} \frac{6}{\pi^2}\log(2) & \text{se } \pi \text{ é a permutação identidade} \\ \frac{3}{\pi^2}\log(2) & \text{se } \pi \text{ é a permutação de Laurent} \end{cases} \tag{3.13}$$

portanto X não tem valor médio bem definido.

Proposição 3.22 *Se X assume valores inteiros e positivos então $\mathbb{E}\, X$ está bem definida e*

$$\mathbb{E}\, X = \sum_{n\geqslant 1} \mathbb{P}[\, X \geqslant n\,].$$

Demonstração. Se X assume valores positivos então podemos rearranjar a soma

$$\sum_{r\geqslant 1} r \mathop{\mathbb{P}}_{X}(r) = \sum_{r\geqslant 1} \mathop{\mathbb{P}}_{X}(r) + \sum_{r\geqslant 2} \mathop{\mathbb{P}}_{X}(r) + \sum_{r\geqslant 3} \mathop{\mathbb{P}}_{X}(r) + \cdots$$

como $\sum_{r\geqslant j} \mathbb{P}_X(r) = \mathbb{P}[\, X \geqslant j\,]$, segue a proposição. □

Sejam $X\colon \Omega \to S$ uma variável aleatória real de $(\Omega, \mathbb{P})$ e $f\colon S \to \mathbb{R}$ uma função. No espaço de probabilidade $(S, \mathbb{P}_X)$ a função f é uma variável aleatória cuja esperança, caso esteja bem definida, é

$$\mathbb{E}\, f = \sum_y y \mathop{\mathbb{P}}_{f}(y) = \sum_y y \mathop{\mathbb{P}}_{X}[f = y] = \sum_y y\, \mathbb{P}[\, f(X) = y\,] = \sum_y y \mathop{\mathbb{P}}_{f(X)}(y) = \mathbb{E}\, f(X)$$

donde tiramos que f no modelo probabilístico $(S, \mathbb{P}_X)$ é somável se, e somente se, $f(X)$ em $(\Omega, \mathbb{P})$ é somável. Se $\mathbb{E}\, f(X)$ está definida então podemos rearranjá-la de modo que

$$\mathbb{E}\, f(X) = \sum_y y \mathop{\mathbb{P}}_{f(X)}(y) = \sum_y y \mathop{\mathbb{P}}_{X}(\{\, s\colon f(s) = y\,\}) = \sum_y \sum_{\substack{s\in S \\ f(s)=y}} y \mathop{\mathbb{P}}_{X}(s) = \sum_{s\in S} f(s) \mathop{\mathbb{P}}_{X}(s)$$

e assim provamos o seguinte resultado.

Teorema 3.23 *Se $X\colon \Omega \to S$ é uma variável aleatória e $f\colon S \to \mathbb{R}$ uma função então $f(X)\colon \Omega \to \mathbb{R}$ é variável aleatória e sua esperança satisfaz*

$$\mathbb{E}\, f(X) = \sum_{s\in S} f(s) \mathop{\mathbb{P}}_{X}(s).$$

sempre que a soma está bem definida. □

Vejamos um exemplo. Se Z é geométrica de parâmetro p, então para computar $\mathbb{E}\, Z^2$ usamos (s.6c) do seguinte modo: derivando ambos os lados de $\sum_{n\geqslant 1} nx^n = x(1-x)^{-2}$ obtemos (para $|x| < 1$)

$$\sum_{n\geqslant 1} n^2 x^{n-1} = \frac{1+x}{(1-x)^3}$$

de modo que, pelo teorema 3.23 e a série acima

$$\mathbb{E}\, Z^2 = \sum_{n\geqslant 1} n^2 \mathop{\mathbb{P}}_{Z}(n) = \sum_{n\geqslant 1} n^2 (1-p)^{n-1} p = \frac{2-p}{p^3} p = \frac{2-p}{p^2}.$$

Exemplo 3.24 (*lei de potência*). A distribuição de uma variável aleatória X sobre os inteiros positivos segue uma lei de potência com parâmetro $\alpha > 0$ se tem distribuição

$$\mathop{\mathbb{P}}_{X}(n) = \frac{1}{n^\alpha} - \frac{1}{(n+1)^\alpha}$$

ou equivalentemente, $\mathbb{P}[X \geqslant n] = 1/n^\alpha$. Distribuições de probabilidade que seguem uma lei de potência são comuns em muitas disciplinas, como a física e a biologia, e mais recentemente ganhou atenção no estudo do que se acostumou de chamar de redes complexas. Usando a proposição 3.22 temos

$$\mathbb{E}\, X = \sum_{n\geqslant 1} \mathbb{P}[X \geqslant n] = \sum_{n\geqslant 1} \frac{1}{n^\alpha}. \tag{3.14}$$

Para $\alpha = 1$ temos $\mathbb{E}\, X = \sum_{n\geqslant 1} 1/n = +\infty$. Para $\alpha = 2$ temos $\mathbb{E}\, X = \pi^2/6$ (veja (s.7)).

Se $\alpha \leqslant 1$, a esperança é infinita. Se $\alpha > 1$, a esperança é finita mas nem sempre há uma forma fechada para a soma como no caso $\alpha = 2$. A série no lado direito da equação (3.14) é a Função Zeta de Riemann[3] quando α é qualquer número complexo com parte real maior que 1.

Agora, consideramos uma variável aleatória X que assume valor nos inteiros positivos tal que $\mathbb{P}[X = n] = (cn^3)^{-1}$, com $c = \zeta(3) = \sum_{j\geqslant 1} 1/j^3$. Essa variável tem valor esperado $\mathbb{E}\, X = \sum_{n\geqslant 1} 1/cn^2 = \pi^2/6c$. Ainda, X^2 é uma variável aleatória e seu valor esperado é,

$$\mathbb{E}\, X^2 = \sum_{n\geqslant 1} i^2 \mathbb{P}[X = n] = \sum_{n\geqslant 1} \frac{1}{cn} = +\infty$$

pelo teorema acima.

A distribuição $\mathbb{P}_X(n) = (\zeta(3)n^3)^{-1}$ sobre os inteiros positivos, tem a seguinte propriedade interessante (Alexander, Baclawski e Rota, 1993): seja A_p o evento formado pelos múltiplos de p. Se p e q são distintos, os eventos A_p e A_q são independentes $\mathbb{P}_X(A_p \cap A_q) = \mathbb{P}_X(A_p)\, \mathbb{P}_X(A_q) = 1/(pq)^3$. O mesmo vale para

$$\mathop{\mathbb{P}}_{X_s}(n) = \frac{n^{-s}}{\zeta(s)}$$

e todo $s > 1$

$$\mathop{\mathbb{P}}_{X_s}(A_p) = \frac{\sum_{k\geqslant 1}(pk)^{-s}}{\sum_{n\geqslant 1} n^{-s}} = \frac{1}{p^s} \quad \text{e} \quad \mathop{\mathbb{P}}_{X_s}(A_p \cap A_q) = \mathop{\mathbb{P}}_{X_s}(A_{pq}) = \frac{1}{p^s}\frac{1}{q^s} = \mathop{\mathbb{P}}_{X_s}(A_p)\, \mathop{\mathbb{P}}_{X_s}(A_q).$$

[3] Um dos problemas matemáticos mais importantes sem solução até o momento, conhecido como Hipótese de Riemann, é uma conjectura a respeito dos zeros da função zeta; esses estariam somente nos inteiros negativos pares e nos complexos com parte real 1/2.

Vamos usar essa distribuição para dar um demonstração probabilística de que a série $\sum_{p\text{ primo}} 1/p$ diverge. Defina para todo primo q o conjunto

$$\mathrm{B}_q = \bigcap_{\substack{p\leqslant q\\ p\text{ primo}}} \overline{\mathrm{A}_p}$$

dos inteiros que não têm um divisor primo menor que q. A sequência $(\mathrm{B}_q : q \text{ primo})$ é decrescente e $\bigcap_q \mathrm{B}_q = \{1\}$. Pela continuidade da probabilidade

$$\frac{1}{\zeta(s)} = \mathbb{P}_{\mathrm{X}_s}(1) = \mathbb{P}_{\mathrm{X}_s}\left(\bigcap_{q\text{ primo}} \mathrm{B}_q\right) = \lim_{q\to\infty} \mathbb{P}_{\mathrm{X}_s}(\mathrm{B}_q) = \lim_{q\to\infty} \prod_{\substack{p\leqslant q\\ p\text{ primo}}} \mathbb{P}_{\mathrm{X}_s}(\overline{\mathrm{A}_p}) = \prod_{p\text{ primo}} \mathbb{P}_{\mathrm{X}_s}(\overline{\mathrm{A}_p}) = \prod_{p\text{ primo}} \left(1 - \frac{1}{p^s}\right) \quad (3.15)$$

que é a *fórmula do produto de Euler*, descoberta por Euler quando estava na busca por uma prova de que $\sum_{p\text{ primo}} 1/p$ diverge. Agora, tomando o logaritmo

$$-\log\zeta(s) = \sum_{p\text{ primo}} \left(1 - \frac{1}{p^s}\right)$$

e de $0 < 1/p^s < 1/2$ temos $\log(1 - 1/p^s) \geqslant -2(1/p^s)$, logo $\log\zeta(s) \leqslant 2\sum_{p\text{ primo}} 1/p^s$ de modo que

$$\frac{1}{2}\log\zeta(s) \leqslant \sum_{p\text{ primo}} \frac{1}{p^s} \leqslant \zeta(s).$$

Agora, a conclusão segue de $\lim_{s\to 1}\zeta(s) = \infty$ pois

$$\lim_{s\to 1} \sum_{k=1}^{n} \frac{1}{k^s} = \sum_{k=1}^{n} \frac{1}{k} > \mathrm{M}$$

para qualquer real $\mathrm{M} \geqslant 1$ se n for suficientemente grande. Além disso, notemos que se assumimos que a quantidade de números primos é finita, então o produto na direita da equação (3.15) está definido e é positivo para $s > 1$, fazendo $s \to 1^+$ temos 0 no lado direito e $+\infty$ no lado esquerdo da equação (3.15), uma contradição que demonstra a existência de uma quantidade infinita de números primos.

Ainda, como uma curiosidade a mais, $\lim_{s\to 1} \mathbb{P}_{\mathrm{X}_s}(\mathrm{A}) = \rho(\mathrm{A})$ em que $\rho(\mathrm{A}) = \lim_{n\to\infty} |\mathrm{A}\cap\{1,\ldots,n\}|/n$ é a densidade relativa de A (definida no exercício 1.56, página 44). ◊

PROPRIEDADES DA ESPERANÇA A seguir veremos propriedades importantes para a média das variáveis aleatórias discretas e reais, em particular veremos que valem as propriedades das variáveis aleatórias simples enunciadas no teorema 3.14.

Teorema 3.25 *Para todo evento* A, $\mathbb{E}\,\mathbb{1}_{\mathrm{A}} = \mathbb{P}(\mathrm{A})$. □

A prova desse teorema é imediata da definição de valor esperado, equação (3.11).

Teorema 3.26 (linearidade da esperança) *Se* X *e* Y *são variáveis aleatórias somáveis de* $(\Omega, \mathbb{P})$ *e* $a, b \in \mathbb{R}$ *constantes quaisquer então* $aX + bY$ *é somável e sua esperança satisfaz* $\mathbb{E}[\, aX + bY \,] = a\,\mathbb{E}\,X + b\,\mathbb{E}\,Y$.

Demonstração. Sejam X e Y variáveis aleatórias somáveis e $a, b \in \mathbb{R}$. Pela equação (3.12) acima a esperança da variável aleatória $|aX + bY|$ é

$$\mathbb{E}[\,|aX + bY|\,] = \sum_{\omega} |\{aX + bY\}(\omega)|\mathbb{P}(\omega) = \sum_{\omega} |aX(\omega) + bY(\omega)|\mathbb{P}(\omega) \leqslant \sum_{\omega} |aX(\omega)|\mathbb{P}(\omega) + \sum_{\omega} |bY(\omega)|\mathbb{P}(\omega)$$

usando a desigualdade triangular (veja (d.4)). De X e Y somáveis temos que o lado direito da equação acima é finito, portanto $aX + bY$ é somável e

$$\begin{aligned}
\mathbb{E}[\, aX + bY \,] &= \sum_{\omega} (aX + bY)(\omega)\mathbb{P}(\omega) \\
&= \sum_{\omega} (aX(\omega) + bY(\omega))\mathbb{P}(\omega) \\
&= \sum_{\omega} aX(\omega)\mathbb{P}(\omega) + \sum_{\omega} bY(\omega)\mathbb{P}(\omega) \\
&= a\,\mathbb{E}\,X + b\,\mathbb{E}\,Y
\end{aligned}$$

logo $\mathbb{E}$ é um funcional linear no conjunto das variáveis aleatórias reais de $(\Omega, \mathbb{P})$. □

Corolário 3.27 *Se* $X_1, \ldots, X_n$ *são somáveis e* $a_1, \ldots, a_n \in \mathbb{R}$ *então*

$$\mathbb{E}\left[\sum_{i=1}^{n} a_i X_i\right] = \sum_{i=1}^{n} a_i\,\mathbb{E}[\,X_i\,].$$

Demonstração. Segue do teorema por indução em $n \geqslant 2$. □

Além da linearidade, outra propriedade importante da esperança é a monotonicidade. Lembremos que $X \leqslant Y$ se $X(\omega) \leqslant Y(\omega)$ para cada $\omega \in \Omega$.

Teorema 3.28 (monotonicidade da esperança) *Se* X *e* Y *são somáveis tais que* $X \leqslant Y$ *então* $\mathbb{E}\,X \leqslant \mathbb{E}\,Y$.

Demonstração. Se $X \leqslant Y$ então $Y - X \geqslant 0$, logo $\mathbb{E}[\,Y - X\,] \geqslant 0$. Da linearidade da esperança $\mathbb{E}\,Y - \mathbb{E}\,X \geqslant 0$ donde segue o teorema. □

Corolário 3.29 *As seguintes propriedades decorrem dos teoremas acima.*

1. *Se* $X = c$ *então* $\mathbb{E}\,X = c$, *para qualquer* $c \in \mathbb{R}$.
2. *Se* $\mathbb{E}\,X$ *está definida, então* $\mathbb{E}[\, aX + b\,] = a\,\mathbb{E}\,X + b$.
3. *Se* $a \leqslant X \leqslant b$, *então* $a \leqslant \mathbb{E}\,X \leqslant b$.

4. *Se* X *é somável então* $X \cdot \mathbb{1}_A$ *é somável para todo evento* A.

5. *Se* $\mathbb{E}\, X$ *está definida então* $|\mathbb{E}\, X| \leqslant \mathbb{E}\, |X|$.

Demonstração. Vamos demonstrar apenas os dois últimos itens, os outros ficam para verificação do leitor. Notemos que $X\mathbb{1}_A \leqslant X$ e que $|X\mathbb{1}_A| = |X|\mathbb{1}_A$, logo, se X é somável, então, por monotonicidade, $X\mathbb{1}_A$ é somável.

Se X está definida e não é somável então $|\mathbb{E}\, X| = +\infty = \mathbb{E}\, |X|$ e o item 5 vale com igualdade. Senão, X é somável e de $-|x| \leqslant x \leqslant |x|$, para todo real x, temos $-|X| \leqslant X \leqslant |X|$ donde $-\mathbb{E}\, |X| \leqslant \mathbb{E}\, X \leqslant \mathbb{E}\, |X|$, ou seja, $|\mathbb{E}\, X| \leqslant \mathbb{E}\, |X|$. □

Se $X\colon \Omega \to S$ e $Y\colon \Omega \to R$ são variáveis aleatórias independentes, então

$$\mathbb{E}[\,|XY|\,] = \sum_{(x,y)\in S\times R} |xy|\, \mathop{\mathbb{P}}_{(X,Y)}((x,y)) = \sum_{(x,y)\in S\times R} |x||y| \mathop{\mathbb{P}}_{X}(x) \mathop{\mathbb{P}}_{Y}(y)$$

e como os termos são não-negativos, podemos rearranjar a soma de modo que

$$\mathbb{E}[\,|XY|\,] = \left(\sum_{x\in S} |x| \mathop{\mathbb{P}}_{X}(x)\right) \cdot \left(\sum_{y\in R} |y| \mathop{\mathbb{P}}_{Y}(y)\right) = \mathbb{E}[\,|X|\,] \cdot \mathbb{E}[\,|Y|\,]$$

Assim, se X e Y são somáveis, XY também é e a mesma dedução sem os módulos vale, ou seja, concluímos que $\mathbb{E}[\,XY\,] = \mathbb{E}\, X \cdot \mathbb{E}\, Y$.

Teorema 3.30 *Sejam* $X\colon \Omega \to S$ *e* $Y\colon \Omega \to R$ *variáveis aleatórias independentes em* $(\Omega, \mathbb{P})$ *e sejam* $f\colon S \to \mathbb{R}$ *e* $g : R \to \mathbb{R}$ *funções tais que* $f(X)$ *e* $g(Y)$ *são variáveis aleatórias somáveis. Então* $f(X) \cdot g(Y)$ *é somável e vale* $\mathbb{E}[\, f(X) \cdot g(Y)\,] = \mathbb{E}[\, f(X)\,]\, \mathbb{E}[\, g(Y)\,]$.

Demonstração. Da proposição 3.10 temos que $f(X)$ e $g(Y)$ são variáveis aleatórias independentes. Podemos replicar a dedução acima para provar que $f(X)g(Y)$ é somável e que, portanto, $\mathbb{E}[\, f(X) \cdot g(Y)\,] = \mathbb{E}[\, f(X)\,] \cdot \mathbb{E}[\, g(Y)\,]$. □

Corolário 3.31 *Se* X *e* Y *são variáveis aleatórias independentes então* $\mathbb{E}[\,XY\,] = \mathbb{E}[\,X\,]\, \mathbb{E}[\,Y\,]$. □

EQUAÇÃO DE WALD Se somamos as variáveis aleatórias $X_1, X_2, \ldots$ porém uma quantidade aleatória delas não podemos usar a linearidade da esperança diretamente (veja exercício 3.63 no final do capítulo), em geral

$$\mathbb{E} \sum_{i=1}^{N} X_i \neq \sum_{i=1}^{\mathbb{E}\, N} \mathbb{E}\, X_i\,.$$

Porém, se N e $X_1, X_2, \ldots$ são independentes, então

$$\mathbb{E}\left[\sum_{i=1}^{N} X_i \;\middle|\; N = k\right] = \mathbb{E}\left[\sum_{i=1}^{k} X_i\right] = k\,\mathbb{E}\, X_1$$

se $X_1, X_2, \ldots$ têm a mesma lei. Agora, multiplicando por $\mathbb{P}[N = k]$ e somando para todo $k \geqslant 1$

$$\mathbb{E} \sum_{i=1}^{N} X_i = \mathbb{E}\, X_1 \sum_{k \geqslant 1} k\mathbb{P}[N = k] = \mathbb{E}[X_1]\,\mathbb{E}[N].$$

Essa identidade num caso mais geral que veremos adiante neste texto é conhecida como equação de Wald.

Exemplo 3.32. Suponhamos que são recebidas $N \leqslant 10$ mensagens eletrônicas por dia, com $\mathbb{E}\, N = 5$ e o tempo gasto lendo-as e respondendo-as são dados pelas variáveis aleatórias $X_1, X_2, \ldots, X_N$ mutuamente independentes e independentes de N, com $\mathbb{E}\, X_i = 8$ minutos, para todo i. O tempo gasto em um dia com mensagens é $X = \sum_{i=1}^{N} X_i$ minutos cujo valor esperado é $8 \cdot \mathbb{E}\, N = 40$ minutos. ◊

Se N não for limitada a prova acima pode ser imitada para um soma enumerável de variáveis aleatórias com a hipótese da linearidade da esperança valer para soma enumerável. Uma condição que garante essa propriedade é $\sum_{i \geqslant 1} \mathbb{E}\,|X_i|$ converge. Com essa hipótese podemos escrever

$$\sum_{i \geqslant 1} \mathbb{E}\, X_i = \sum_{i \geqslant 1} \sum_{\omega} X_i(\omega)\mathbb{P}(\omega) = \sum_{\omega} \sum_{i \geqslant 1} X_i(\omega)\mathbb{P}(\omega) = \sum_{\omega} \left(\sum_{i \geqslant 1} X_i(\omega) \right) \mathbb{P}(\omega) = \sum_{\omega} \left(\sum_{i \geqslant 1} X_i \right) (\omega)\mathbb{P}(\omega) = \mathbb{E} \sum_{i \geqslant 1} X_i$$

e os detalhes a respeito da convergência absoluta em cada passo ficam a cargo do leitor.

Essa propriedade pode ser bastante útil em situações como a dos algoritmos Las Vegas com laços, em que o i-ésima iteração executa X_i instruções de modo que o número esperado de instruções realizadas durante uma execução é $\mathbb{E} \sum_{i \geqslant 1} X_i$.

Observação 3.33. O teorema 3.26, da linearidade, ainda vale com as hipóteses dos valores esperados $\mathbb{E}\, X$ e $\mathbb{E}\, Y$ definidos e $a\mathbb{E}\, X + b\mathbb{E}\, Y$ definido, isto é, não resulta nos indeterminados $+\infty - \infty$ ou $-\infty + \infty$. Analogamente, o teorema 3.28, da monotonicidade, ainda vale com as hipóteses de $\mathbb{E}\, X$ e $\mathbb{E}\, Y$ definidos e $\mathbb{E}\, X - \mathbb{E}\, Y$ definido. Em particular, se as duas variáveis aleatórias, X e Y, têm valor esperado estão definidos e pelo menos uma é somável, então as conclusões dos teoremas valem. Ainda, no teorema 3.28 a conclusão vale se os valores esperados de X e de Y estão definidos e $\mathbb{E}\, X < +\infty$ ou $\mathbb{E}\, Y > -\infty$. ◊

3.1.4 *QUICKSORT* PROBABILÍSTICO

O *quicksort* é um algoritmo recursivo de ordenação que, grosso modo, funciona da seguinte maneira: dado uma sequência S de números dos quais o primeiro é chamado de *pivô*, o *quicksort*(S) compara o pivô com cada outro elemento de S; os elementos menores que o pivô formam a subsequência S_1 (a esquerda do pivô no vetor) e os demais que não o pivô formam a subsequência S_2 (a direita do pivô no vetor); o algoritmo devolve a sequência (*quicksort*(S_1), pivô, *quicksort*(S_2)) ordenada recursivamente.

O *quicksort* foi inventado por C. A. R. Hoare em 1960 e sabemos que é muito rápido em geral, mas é lento em algumas raras instâncias. É um algoritmo que ordena os números em função dos resultados das comparações entre elementos da entrada e, nesse tipo de algoritmo, medimos a eficiência do algoritmo contando o número de comparações realizadas para ordenar a sequência. O algoritmo *quicksort* executa $O(n \log n)$ comparações em média e $O(n^2)$ comparações no pior caso para ordenar sequências de tamanho n. A figura 3.2 abaixo ilustra um exemplo de pior caso e um exemplo de melhor caso da árvore de recursão para uma sequência de tamanho 6.

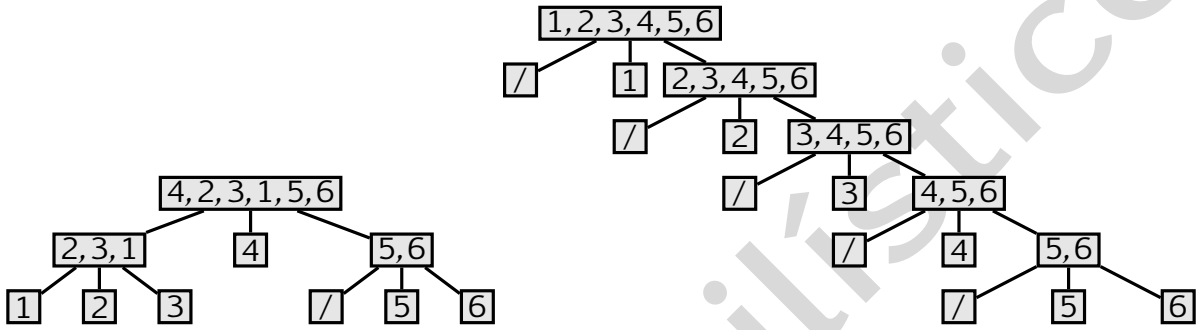


Figura 3.2: na esquerda temos uma árvore de recursão do *quicksort* onde o pivô é a mediana da sequência. A árvore da direita é o caso onde o pivô é sempre o menor elemento da sequência.

É sabido que para garantir $O(n \log n)$ comparações numa execução é suficiente garantir que as subsequências geradas no pivotamento tenham sempre uma fração constante do tamanho da sequência que as origina. Por exemplo, se S_1 (ou S_2) sempre tem 10% do tamanho de S então o número de comparações executadas pelo *quicksort* numa instância de tamanho n, que denotamos $T(n)$, é o número de comparações executadas em S_1 mais o número de comparações executadas em S_2 mais as $O(n)$ comparações executadas no particionamento que cria essas subsequências

$$T(n) = T(0{,}1n) + T(0{,}9n) + O(n)$$

cuja solução é uma função assintoticamente limitada superiormente por $n \log_{10/9} n$. Não há nada de especial na escolha de 10%. De fato, se uma proporção for mantida no particionamento, digamos que a menor parte tem uma fração α da entrada, então a recorrência $t(n) \leqslant t(\lfloor \alpha n \rfloor) + t(\lfloor (1-\alpha)n \rfloor) + cn$ para constantes $\alpha, c > 0$ tem solução $O(n \log n)$. Para conhecer esses resultados com mais detalhes sugerimos ao leitor uma consulta ao texto de Cormen, Leiserson e Rivest, 1990.

Na versão probabilística do *quicksort* a aleatoriedade é usada para descaracterizar o pior caso no sentido de que sorteando o pivô como qualquer elemento da sequência com grande chance evitamos os 10% menores e os 10% maiores elementos do vetor de modo que maior parte do tempo os pivotamentos garantem pelo menos uma fração constante da sequência original no menor lado. O seguinte

algoritmo é o *quicksort* aleatorizado.

Instância: uma sequência de números $S = (x_1, x_2, \ldots, x_n)$.
Resposta: os elementos de S em ordem crescente.

```
1  se |S| ⩽ 20 então
2      ordene S na força bruta ;
3      responda a sequência ordenada.
4  senão
5      z ←R S;
6      para cada y ∈ S, y ≠ z faça
7          se y < z então insira y em S1;
8          senão insira y em S2;
9      ordene recursivamente S1;
10     ordene recursivamente S2;
11     responda (S1, z, S2).
```

Algoritmo 32: *quicksort* aleatorizado.

Para simplificar a análise assumimos que na instância não há repetição, ou seja, todos os elementos de S são distintos. Mostraremos que, com essa estratégia, o número esperado de comparações entre elementos de S é $O(n \log n)$ com alta probabilidade.

A análise desse algoritmo é um detalhamento da seguinte ideia: consideremos um elemento x de uma instância S para o algoritmo. Sejam $S'_0 := S$ e $S'_1, S'_2, \ldots, S'_M$ as subsequências a que x pertence após cada particionamento durante uma execução. O i-ésimo particionamento é *bom* se o pivô não está entre os 10% menores e os 10% maiores elementos de S'_i, o que ocorre com probabilidade 4/5, de modo que $|S'_i|/10 \leqslant |S'_{i+1}| \leqslant 9|S'_i|/10$ e portanto x passa por no máximo $\log_{10/9}(n)$ particionamentos bons. Agora, a variável aleatória M não deve ser muito maior que $2\log_{10/9}(n)$ pois pelo exercício 1.51, página 42, a probabilidade de ocorrem menos que $\log_{10/9}(n)$ particionamentos bons em $2\log_{10/9}(n)$ particionamentos é menor que

$$\left(\frac{4}{5}\right)^{2\log_{10/9}(n)} < \left(\frac{9}{10}\right)^{2\log_{10/9}(n)} = \frac{1}{n^2}.$$

A cada particionamento o elemento não-pivô x é comparado uma única vez, com o pivô sorteado. Assim, o numero total de comparações numa execução é a soma para todo x do número de particionamentos pelos quais x passa. Portanto, uma execução realiza mais que $2n\log_{10/9}(n)$ comparações se existe um x que passa por mais que $2\log_{10/9}(n)$ particionamentos o que ocorre com probabilidade $n(1/n^2) = 1/n$. Portanto, com alta probabilidade, o *quicksort* realiza $O(n \log n)$ comparações;

ANÁLISE DO *QUICKSORT* PROBABILÍSTICO O particionamento (ou pivotamento) de S nas linhas 5, 6, 7 e 8 do algoritmo acima é considerado um *sucesso* se $\min\{|S_1|, |S_2|\} \geqslant (1/10)|S|$, caso contrário,

dizemos que o particionamento foi um fracasso. Um sucesso significa que o pivô não está entre os 10% maiores elementos da sequência particionada e nem está entre os 10% menores elementos da sequência particionada. Assim 80% dos elementos de S são boas escolhas para o pivô de modo que a escolha aleatória do pivô é um experimento de Bernoulli com parâmetro p dado por

$$p := \frac{\lfloor 0{,}8|S| \rfloor}{|S|} \tag{3.16}$$

donde

$$0{,}75 < p \leqslant 0{,}8$$

pois $0{,}8|S| - 1 < \lfloor 0{,}8|S| \rfloor \leqslant 0{,}8|S|$ e $|S| > 20$.

Consideremos uma execução do *quicksort* com entrada $S = (x_1, x_2, \ldots, x_n)$. Fixado $i \in \{1, 2, \ldots, n\}$, seja X_i o número de comparações entre o elemento x_i da entrada e algum pivô durante toda uma execução do algoritmo. Em cada particionamento x_i é comparado uma única vez (com o pivô) e se x_i é escolhido pivô então: (1) cada elemento da subsequência da qual x_i pertence é comparado com ele e tais comparações são contabilizadas pelos outros elementos e (2) a partir daí x_i nunca mais participará de um particionamento e nunca mais será comparado com outro elemento de S.

Exercício 3.34. Mostre que x_i participa de no máximo $\lfloor \log_{10/9} n \rfloor$ particionamentos com sucesso.

Denotemos por Y_k o número de particionamentos ocorridos entre o k-ésimo particionamento com sucesso do qual x_i participa (exclusive) e o próximo particionamento com sucesso do qual x_i participa (inclusive), ou seja, é o número de partições realizadas até obter um sucesso, portanto, $Y_k \sim \text{Geom}(p)$, logo $\mathbb{E}\, Y_k = 1/p \leqslant 1/0{,}75 < 2$. Pelo exercício 3.34 acima, x_i participa de no máximo $\lfloor \log_{10/9} n \rfloor$ particionamentos com sucesso e temos assim que

$$X_i \leqslant \sum_{k=0}^{\lfloor \log_{10/9} n \rfloor} Y_k$$

é o número de particionamentos pelo qual passa x_i, que é, também, o número de vezes que ele é comparado com um pivô. O número total de comparações durante toda execução é dado por

$$T := \sum_{i=1}^{n} X_i.$$

Pela linearidade e monotonicidade da esperança

$$\mathbb{E}\, X_i < 2\lfloor \log_{\frac{10}{9}} n \rfloor \quad \text{e} \quad \mathbb{E}\, T < 2n\lfloor \log_{\frac{10}{9}} n \rfloor. \tag{3.17}$$

Além disso, as no máximo $n/20$ sequências ordenadas na força-bruta na linha 2 fazem, cada uma $O(1)$ comparações, contribuindo no total com $O(n)$ comparações que somadas as $O(n \log n)$ de (3.17) totalizam $O(n \log n)$ comparações.

Lema 3.35 *O número esperado de comparações entre elementos de uma sequência com n elementos numa execução do quicksort aleatorizado é $O(n \log n)$.* □

Da disciplina Análise de Algoritmos sabemos que todo algoritmo de ordenação baseado em comparação realiza $\Omega(n \log n)$ comparações para ordenar n elementos, ou seja, em média o *quicksort* aleatorizado é o melhor possível (Cormen, Leiserson e Rivest, 1990). Isso nos leva a conjeturar que o *quicksort* faz o melhor quase sempre.

Porém, saber que um algoritmo é bom em média não nos garante que ele é bom quase sempre. Vamos mostrar que com alta probabilidade o desempenho do algoritmo está próximo da média.

Teorema 3.36 *O número de comparações numa execução do quicksort aleatorizado com uma entrada de tamanho n é $O(n \log n)$ com probabilidade $1 - O(n^{-22})$.*

Demonstração. Definimos

$$L := 12\lfloor \log_{\frac{10}{9}} n \rfloor$$

e consideramos o evento $X_i > L$, ou seja, segundo (3.17) o elemento x_i da entrada foi comparado mais do que seis vezes o valor esperado para o número de comparações. Se $X_i > L$ então em L particionamentos ocorrem menos que $\lfloor \log_{10/9} n \rfloor$ sucessos, ou seja, o número de fracassos em L particionamentos é pelo menos

$$F := 12\lfloor \log_{\frac{10}{9}} n \rfloor - \lfloor \log_{\frac{10}{9}} n \rfloor = 11\lfloor \log_{\frac{10}{9}} n \rfloor.$$

Seja Z_i o número de particionamentos com fracasso pelos quais x_i passa ao longo de L particionamentos. A variável aleatória Z_i tem distribuição binomial com parâmetros L e $1 - p$. Vamos estimar a probabilidade do evento $Z_i > F$, com isso, teremos uma limitante superior para a probabilidade de $X_i > L$. A probabilidade de ocorrerem $j > F$ fracassos em L ensaios independentes de Bernoulli é, usando (d.2) para estimar o coeficiente binomial,

$$\binom{L}{j}(1-p)^j p^{L-j} \leqslant \left(\frac{eL}{j}\frac{1-p}{p}\right)^j p^L \leqslant \left(\frac{eL}{F}\frac{1-p}{p}\right)^j p^L.$$

Ademais

$$e \cdot \frac{L}{F} \cdot \frac{1-p}{p} < e \cdot \frac{12}{11} \cdot \frac{0{,}25}{0{,}75} < 0{,}99$$

portanto,

$$\mathbb{P}[Z_i > F] = \sum_{j=F+1}^{L} \binom{L}{j}(1-p)^j p^{L-j} < \sum_{j=F+1}^{L} 0{,}99^j p^L = p^L \sum_{j=F+1}^{L} 0{,}99^j < p^L \sum_{j \geqslant 0} 0{,}99^j = 100p^L$$

no último passo usamos (s.6a), donde segue que

$$\mathbb{P}[X_i > L] < 100(0{,}8)^{12\log_{10/9}(n)-12} = \frac{100}{0{,}8^{12}}(0{,}8)^{\log_{10/9}(n^{12})} < \frac{100}{0{,}8^{12}}(0{,}8)^{c\log_{8/10}(n^{12})}$$

onde $c = 1/\log_{0,8}(10/9)$ é a constante devido a mudança de base do logaritmo para 0,8, logo

$$100p^{\lfloor} < \frac{100}{0{,}8^{12}} n^{12c} < n^{-23}$$

para todo $n > 20$.

A probabilidade de existir i com $X_i > 12\lfloor \log_{\frac{10}{9}} n \rfloor$ é

$$\mathbb{P}\left(\bigcup_{i=1}^{n} \left[X_i > 12\lfloor \log_{\frac{10}{9}} n \rfloor \right] \right) < n \cdot \frac{1}{n^{23}}$$

logo, com probabilidade $1 - O(n^{-22})$ vale que $X_i \leqslant 12\log_{10/9}(n)$ para todo i e que, com essa probabilidade, o *quicksort* aleatorizado executa $O(n \log n)$ comparações entre elementos da entrada, incluídas aí as $O(n)$ comparações feitas na linha 2 do algoritmo. □

3.2 O MÉTODO PROBABILÍSTICO, 1º MOMENTO

O método probabilístico é um método não construtivo para demonstrar a existência de objetos matemáticos e que foi popularizado pelo matemático húngaro Paul Erdős. O princípio básico é baseado no fato de que se algum objeto em uma coleção de objetos tiver uma determinada propriedade, então com probabilidade positiva uma escolha aleatória com uma distribuição apropriada nessa coleção tem essa propriedade. Em muitos casos o fato demonstrado é uma sentença que não envolve probabilidade e a demonstração usa probabilidade, como o exemplo da divergência de $\sum_p 1/p$ que apresentamos na página 136 (com a ressalva de que naquele caso existe demonstração sem usar probabilidade, o que nem sempre é o caso). Há várias técnicas, algumas sofisticadas, que se encaixam nesse método e neste momento veremos uma simples, baseada no seguinte exercício.

Exercício 3.37 (princípio de primeiro momento). Seja X uma variável aleatória real e k real. Se $\mathbb{E}\, X \geqslant k$ então $X(\omega) \geqslant k$ para algum $\omega \in \Omega$.

3.2.1 SATISFAZIBILIDADE DE FÓRMULA BOOLEANA

Uma variável booleana assume um de dois possíveis valores: *verdadeiro*, que representaremos por 1, ou *falso*, que representaremos por 0. Uma *fórmula* booleana é uma expressão que envolve variáveis $v_1, v_2, \ldots, v_n$, parênteses e os operadores lógicos $\vee$ (lê-se *ou*), $\wedge$ (lê-se *e*) e $\neg$ (lê-se *não*). Um *literal* é uma variável ou a sua negação. Uma *cláusula* é uma disjunção (*ou*) de literais. Por exemplo,

$$(v_1 \vee \neg v_2 \vee v_3) \wedge (v_2 \vee \neg v_3 \vee v_4) \wedge (v_3 \vee \neg v_4) \tag{3.18}$$

é uma fórmula booleana com três cláusulas, as duas primeiras compostas por três literais e a outra por dois. Consideramos que o *não* tem precedência sobre os outros operadores de modo que, por exemplo, $v_1 \vee \neg v_2 \vee v_3$ deve ser lido como $v_1 \vee (\neg v_2) \vee v_3$.

Uma fórmula expressa como uma conjunção de cláusulas está na Forma Normal Conjuntiva, abreviada CNF (do inglês *conjunctive normal form*). Para qualquer inteiro $k \geqslant 1$, uma *fórmula k-CNF* é da forma $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ com cada cláusula C_i sendo uma disjunção de no máximo k literais, como na equação (3.18) acima que é uma fórmula 3-CNF.

Uma *valoração* das variáveis é uma atribuição dos valores 0 ou 1 para cada uma delas e tal valoração *satisfaz* a fórmula se o resultado dessas atribuições, de acordo com as regras da lógica booleana[4], é 1. No exemplo dado na equação (3.18) a valoração $v_1 = 1$, $v_2 = 1$, $v_3 = 1$ e $v_4 = 0$ satisfaz a fórmula.

Uma valoração satisfaz uma fórmula CNF se, e somente se, satisfaz cada uma das cláusulas dessa fórmula. Vamos assumir, sem perda de generalidade, que nenhuma cláusula contém um literal e sua negação pois tal cláusula é sempre satisfeita em qualquer valoração.

O problema Satisfazibilidade é o problema de decidir se uma fórmula booleana admite uma valoração das suas variáveis que a satisfaz.

Problema computacional da satisfazibilidade booleana (SAT):

Instância: uma fórmula booleana CNF.

Resposta : *sim* se existe uma valoração que satisfaz a fórmula, *não* caso contrário.

Esse problema tem um papel central em Complexidade Computacional e não se conhece algoritmo eficiente que o resolva; foi o primeiro problema reconhecido como NP-completo, o que significa que todo problema NP pode ser codificado como uma instância de SAT. Esse notável resultado é conhecido como o Teorema de Cook–Levin.

O 3SAT é o problema de satisfazibilidade obtido quando restringimos as instâncias às fórmulas 3-CNF, também é um problema sem algoritmo eficiente conhecido. Um algoritmo eficiente para SAT também é um algoritmo eficiente para SAT e prova-se que um algoritmo eficiente para 3SAT implica em um algoritmo eficiente para SAT. Porém, quando restringimos as instâncias às fórmulas 2-CNF, o 2SAT, conhecemos um teste eficiente de satisfazibilidade (exercício 3.73 no final do capítulo).

O problema Satisfazibilidade Máxima (MAX-SAT) é um problema de otimização que consiste em determinar o maior número possível de cláusulas de uma fórmula CNF que podem ser satisfeitas por uma valoração.

Problema computacional da satisfazibilidade máxima de uma fórmula k-CNF (MAX-kSAT):

Instância: uma fórmula k-CNF com $k \geqslant 2$.

Resposta : o maior número de cláusulas que podem ser satisfeitas concomitantemente.

---

[4] $u \wedge v = \min\{u, v\}$, $u \vee v = \max\{u, v\}$ e $\neg u = 1 - u$.

Esses problemas são pelo menos tão difíceis de se resolver algoritmicamente quanto SAT e 3SAT, claramente um algoritmo eficiente para o problema MAX-SAT implica em um algoritmo eficiente para o problema SAT.

Seja $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ uma instância do SAT e $V = \{v_1, \ldots, v_n\}$ o conjunto das variáveis de $\mathcal{C}$. O conjunto de todas as valorações das variáveis é $\Omega := \{0,1\}^n$ e interpretamos a i-ésima coordenada como o valor atribuído a v_i. Em Ω consideramos a distribuição uniforme. Para cada i definimos a variável aleatória indicadora

$$\mathbb{1}_{[C_i=1]}(x_1, \ldots, x_n) := \begin{cases} 1, & \text{se a cláusula } C_i \text{ foi satisfeita por } (x_1, \ldots, x_n) \in \{0,1\}^n, \\ 0, & \text{caso contrário.} \end{cases}$$

O número de cláusulas satisfeitas por uma valoração é dado pela variável aleatória

$$X := \sum_{i=1}^{m} \mathbb{1}_{[C_i=1]}$$

e o número médio de cláusulas satisfeitas por uma valoração é, pela linearidade da esperança,

$$\mathbb{E}\, X = \sum_{i=1}^{m} \mathbb{P}[\, C_i = 1\,] \geqslant \frac{m}{2} \tag{3.19}$$

pois uma cláusula com k literais não é satisfeita se todos valem 0, o que ocorre comprobabilidade 2^{-k}, de modo $[C_i = 1]$ com probabilidade $1 - 2^{-k} \geqslant 1/2$. Na fórmula $v \wedge \neg v$ apenas metade das cláusulas pode ser satisfeita de modo que a garantia dada na equação (3.19) é a melhor possível. Assim, podemos concluir que toda fórmula boolena CNF admite uma valoração que satisfaz pelo menos metade das cláusulas.

MAX-E3SAT Vamos assumir instâncias 3-CNF em que cada cláusula tem exatamente três literais (com a hipótese de não ocorrer um literal e seu complemento), ou seja, vamos supor que as cláusulas não têm variáveis repetidas. Esse problema é rotulado como MAX-E3SAT.

Uma cláusula não é satisfeita se cada uma de seus literais não o for, o que ocorre com probabilidade 1/8, portanto, uma cláusula é satisfeita com probabilidade $\mathbb{P}[\, C_i = 1\,] = 7/8$, qualquer que seja i. O número de cláusulas satisfeitas por uma valoração é dado pela variável aleatória

$$X := \sum_{i=1}^{m} \mathbb{1}_{[C_i=1]} \tag{3.20}$$

e o número médio de cláusulas satisfeitas por uma valoração é, pela linearidade da esperança, $\mathbb{E}\, X = (7/8)m$. Se o número médio de cláusulas satisfeitas é $7m/8$, com a média sobre as valorações em Ω, então pelo exercício 3.37 deve haver uma valoração em Ω que satisfaz pelo menos $7m/8$ cláusulas. Notemos que segue desse resultado que toda instância com no máximo 7 cláusulas é satisfazível (veja o exemplo 6.4, página 308).

Teorema 3.38 *Para toda instância de* max-e3sat, *existe uma valoração que satisfaz pelo menos* 7/8 *de todas as cláusulas.* □

Se um algoritmo sorteia uma valoração e determina quantas cláusulas são satisfeitas por ela, qual o número esperado de sorteios até que seja determinada uma valoração que satisfaça pelo menos 87,5% (ou 7/8) de todas as cláusulas?

Pelo exercício 3.9, página 122, podemos escolher um valor lógico para cada variável, com as escolhas independentes, ao invés de escolher ao acaso uma valoração de V. O algoritmo pode ser descrito como abaixo.

Instância: cláusulas $\{C_1, \ldots, C_m\}$ sobre as variáveis $\{v_1, \ldots, v_n\}$ de uma fórmula 3-CNF.
Resposta: uma valoração que satisfaz $\geqslant \frac{7}{8}m$ cláusulas.

1 **repita**
2 **para cada** $i \in \{1, \ldots, n\}$ **faça** $v_i \xleftarrow{R} \{0, 1\}$;
3 avalie $C_1, \ldots, C_m$;
4 **até que** *pelo menos* $\frac{7}{8}m$ *cláusulas estejam satisfeitas*
5 **responda** $v_1, \ldots, v_n$ *e a quantidade de cláusulas satisfeitas.*

Algoritmo 35: 7/8-aproximação para max-e3sat.

Esse algoritmo aleatorizado determina uma valoração das variáveis de uma fórmula booleana de modo a satisfazer pelo menos 7/8 das cláusulas da instância. Como isso garante uma resposta para o problema de otimização que situa-se entre 7/8 do valor ótimo e o próprio valor ótimo de uma instância, dizemos que o algoritmo é uma 7/8-*aproximação* para max-3sat.

Se p é a probabilidade do evento "pelo menos $7m/8$ cláusulas são satisfeitas" por uma valoração aleatória, então o número de sorteios necessários é uma variável aleatória $Z \sim \text{Geom}(p)$. A probabilidade p é difícil de determinar pois os eventos $[C_i = 1]$ $(i = 1, \ldots, m)$ não são independentes. Vamos estimar um limitante inferior para p, assim teremos um limitante superior para $\mathbb{E}\, Z = 1/p$.

Proposição 3.39 *Se p é a probabilidade com que uma valoração satisfaz pelo menos* 7/8 *de todas as cláusulas de uma fórmula* 3*-CNF, então $p \geqslant 1/(m+8)$.*

Demonstração. Para estimar p, seja p_j é a probabilidade de uma valoração aleatória satisfazer exatamente j cláusulas. Se X é o número de cláusulas satisfeitas por uma valoração, definida na equação (3.20) acima, então

$$\frac{7}{8}m = \mathbb{E}\, X = \sum_{j=0}^{m} j p_j. \tag{3.21}$$

Para qualquer inteiro $M < m$

$$\sum_{j=0}^{m} j p_j = \sum_{j=0}^{M} j p_j + \sum_{j=M+1}^{m} j p_j \leqslant \sum_{j=0}^{M} M p_j + \sum_{j=M+1}^{m} m p_j = M \sum_{j=0}^{M} p_j + m \sum_{j=M+1}^{m} p_j \tag{3.22}$$

e se M é o maior natural estritamente menor que $7m/8$, então pela definição de p

$$\sum_{j=0}^{M} p_j = 1 - p \qquad \text{e} \qquad \sum_{j=M+1}^{m} p_j = p. \tag{3.23}$$

Das equações (3.21), (3.22) e (3.23), deduzimos que

$$\frac{7}{8}m \leqslant M(1-p) + mp, \tag{3.24}$$

logo $p \geqslant (7m-8M)/(8m-8M)$. Pela escolha de M temos $8M < 7m$ e como ambos são naturais $7m-8M \geqslant 1$. Também, de $7m/8 < M+1$ deduzimos que $8m-8M \leqslant m+8$, portanto $p \geqslant 1/(8+m)$. □

Corolário 3.40 *O número esperado de rodadas do laço do algoritmo 35 é* $\mathbb{E}\, Z \leqslant m+8$. □

O número total de instruções executadas até terminar é proporcional ao número de rodadas do **repita**, cujo valor esperado é $\leqslant 8+m$ pelo corolário acima. Supondo que a avaliação de cada cláusula pode ser feita em tempo constante, cada iteração do **repita** tem custo de sortear n bits e avaliar m cláusulas, portanto, o custo é $O(m+n)$. Esse algoritmo tem custo esperado $O(m(m+n))$, em que m é o número de cláusulas e n o de variáveis. Como consideramos somente variáveis que aparecem explicitamente nas cláusulas temos $n \leqslant 3m$, portanto, o algoritmo acima descobre uma valoração com a propriedade desejada para uma fórmula 3-CNF com m cláusulas usando $O(m^2)$ instruções.

Mais que isso pode ser dito nesse caso. A probabilidade de uma execução exceder muito o tempo esperado é pequena. De fato, usando a estimativa para uma variável geométrica dada no exercício 3.5, página 118, a probabilidade do número de iterações ultrapassar duas vezes o valor esperado é

$$\mathbb{P}[\, Z \geqslant 2m+16\,] \leqslant \left(1 - \frac{1}{m+8}\right)^{2m+15} < 0{,}14$$

para todo m. Para $m > 50$, o número de iterações do laço ultrapassa $m\log(m)$ com probabilidade menor que 0,029 e ultrapassa m^2 com probabilidade menor que 0,00000000000000000013.

ALGORITMOS APROXIMATIVOS São algoritmos que encontram soluções aproximadas para problemas de otimização. Para $0 < \alpha < 1$, uma α-aproximação é uma garantia de que o valor respondido pelo algoritmo é no máximo o valor ótimo OPT e pelo menos $\alpha \cdot$ OPT. Os algoritmos aproximativos são muito estudados em teoria da computação pois, dentre outros fatos, podem levar a resultados surpreendentes como, por exemplo, o de que *uma* $(7/8+\varepsilon)$*-aproximação em tempo polinomial para* MAX-E3SAT, *para qualquer* $\varepsilon > 0$, *implicaria em* P = NP. Assim, supondo que P ≠ NP não temos algoritmo eficiente para o MAX-E3SAT e o melhor que poderemos fazer é encontrar um algoritmo aproximativo eficiente que, no pior caso, garanta uma valoração que satisfaça 7/8 do número máximo de cláusulas que possam ser satisfeitas.

3.2.2 CORTE GRANDE EM GRAFOS

Dado um grafo $G = (V, E)$, queremos determinar um corte com muitas arestas em G. Recordemos que o corte definido por $A \subset V$ em G é o subconjunto de arestas $\nabla(A) = \{\{i,j\} \in E\colon i \in A \text{ e } j \in \overline{A}\}$. Assumimos, sem perda de generalidade, que $V = \{1, 2, \ldots, n\}$.

Consideramos o espaço amostral $\Omega := \{0,1\}^n$ munido da medida uniforme, assim um subconjunto aleatório $A \subset V$ é identificado pelo ponto amostral $(x_1, \ldots, x_n) \in \Omega$, isto é, para cada vértice $i \in V$, se $x_i = 1$ então $i \in A$, senão $i \in \overline{A}$. O número de arestas no corte é a variável aleatória

$$|\nabla(A)| = \sum_{\{i,j\} \in E} \mathbb{1}_{\{i,j\} \in \nabla(A)}$$

cuja média é $|E| \cdot \mathbb{P}[\{i,j\} \in \nabla(A)]$. Temos $\{i,j\} \in \nabla(A)$ se, e só se, $x_i \neq x_j$ o que ocorre com probabilidade $1/2$, ou seja,

$$\mathbb{E}\,|\nabla(A)| = \frac{|E|}{2}.$$

Se o valor médio do tamanho de um corte em G é $|E|/2$ então G deve conter um corte com pelo menos $|E|/2$ arestas. Com a mesma estratégia da seção anterior, podemos transformar esse resultado num algoritmo aleatorizado para determinar um corte grande em G, isto é, um corte com pelo menos metade das arestas.

Instância: grafo G sobre os vértices $V = \{1, \ldots, n\}$ e com m arestas.
Resposta: $A \subset V(G)$ tal que $\nabla(A)$ tem $\geqslant \frac{m}{2}$ arestas.

```
1 repita
2     para cada i ∈ {1,...,n} faça x_i ←R {0,1};
3     A ← {i: x_i = 1};
4     compute |∇(A)|;
5 até que |∇(A)| ⩾ m/2
6 responda A.
```

Algoritmo 36: 1/2-aproximação para MAX-CUT.

A análise desse algoritmo é análoga à da seção anterior. As linhas 2 a 4 têm custo de execução proporcional ao tamanho do grafo $n + m$. O número total de instruções executadas é uma variável aleatória que depende do número de rodadas do laço. O número de rodadas do laço até que ocorra um sucesso ($|\nabla(A)| \geqslant \frac{m}{2}$) é uma variável aleatória $Z \sim \text{Geom}(p)$, cuja esperança é $\mathbb{E}\,Z = 1/p$.

Seguindo a mesma dedução da seção anterior, equações (3.21), (3.22), (3.23) e (3.24) com M o maior natural estritamente menor que $m/2$, temos

$$p \geqslant \frac{m - M}{2m - 2M} \geqslant \frac{1}{m+2}$$

portanto, o número esperado de rodadas do laço da linha 1 é $\leqslant m + 2$.

O custo esperado para uma rodada do algoritmo é $O(m(m+n))$ para um grafo com n vértices e m arestas. A probabilidade com que o algoritmo realiza pelo menos km rodadas até parar é $\mathbb{P}[Z \geqslant km] = (1-1/(m+2))^{km-1}$ para todo k. Para $k = m$ essa probabilidade é menor que 0,00019 para todo grafo com $m \geqslant 10$ arestas. Para $m \geqslant 90$ a probabilidade é menor que 5×10^{-39}.

Problema computacional do corte máximo em grafos (MAX-CUT):

Instância: um grafo G.

Resposta: o tamanho $|\nabla(A)|$, para algum A, do corte máximo.

Para esse problema não é conhecido algoritmo eficiente e o algoritmo acima é uma 1/2-aproximação, pois determina um corte cuja quantidade de arestas está entre $(1/2)\text{OPT}$ e o valor ótimo OPT. Uma $(16/17+\varepsilon)$-aproximação em tempo polinomial para MAX-CUT, para qualquer $\varepsilon > 0$, implica em $P = NP$ (Håstad, 2001).

Existe um algoritmo determinístico eficiente de 1/2-aproximação para esse problema. Começamos com uma partição arbitrária dos vértices do grafo dado $G = (V, E)$ e movemos um vértice de cada vez de um lado da partição para o outro se isso melhora a solução, até que não haja mais movimentos que melhoram a solução. O número de iterações é no máximo $|E|$ porque o algoritmo melhora o corte em pelo menos uma aresta em cada movimento. Quando o algoritmo termina, pelo menos metade das arestas incidentes em cada vértice pertencem ao corte, pois, caso contrário, mover o vértice melhoraria o corte (verifique). Portanto, o corte inclui pelo menos metade das arestas.

3.3 DISTRIBUIÇÃO E ESPERANÇA CONDICIONAIS

Sejam $(\Omega, \mathbb{P})$ um espaço de probabilidade, A um evento desse espaço com probabilidade positiva e X um variável aleatória real e somável (ou apenas não negativa). A **esperança condicional** da variável X dado A é o valor médio de X com respeito a lei condicional

$$\mathbb{E}[X \mid A] = \sum_{\omega \in \Omega} X(\omega)\mathbb{P}_A(\omega) = \sum_{\omega \in A} X(\omega)\frac{\mathbb{P}(\omega)}{\mathbb{P}(A)} = \frac{\mathbb{E}[X \cdot \mathbb{1}_A]}{\mathbb{P}(A)}. \tag{3.25}$$

Pelo corolário 3.29, item 4, se X é somável então $X \cdot \mathbb{1}_A$ também é logo $\mathbb{E}[X \mid A] < +\infty$. Dito isso, se X tem esperança bem definida, então podemos reorganizar os termos da soma e concluir que

$$\mathbb{E}[X \mid A] = \sum_r r\mathbb{P}[X = r \mid A].$$

Por ser uma média de uma variável aleatória com respeito a uma distribuição, a esperança condicional usufrui das mesmas propriedades da esperança.

Exercício 3.41 (propriedades da esperança condicional). Prove que se X e Y são somáveis, A um evento de probabilidade positiva, B um evento qualquer e $a, b \in \mathbb{R}$, então

1. $\mathbb{E}[\,a \mid A\,] = a$.

2. $\mathbb{E}[\,\mathbb{1}_B \mid A\,] = \mathbb{P}(\,B \mid A\,)$.

3. Se $X \leqslant Y$ então $\mathbb{E}[\,X \mid A\,] \leqslant \mathbb{E}[\,Y \mid A\,]$.

4. $\mathbb{E}[\,aX + bY \mid A\,] = a\,\mathbb{E}[\,X \mid A\,] + b\,\mathbb{E}[\,Y \mid A\,]$.

5. Se X e A são independentes então $\mathbb{E}[\,X \mid A\,] = \mathbb{E}\,X$ e $\mathbb{E}[\,XY \mid A\,] = \mathbb{E}[\,X \mid A\,] \cdot \mathbb{E}[\,Y \mid A\,]$.

Nesta seção será conveniente definirmos o **suporte** da variável aleatória $X\colon \Omega \to S$ como os pontos de $(S, \mathbb{P}_X)$ com probabilidade positiva

$$R_X := \left\{ x \in S\colon \mathop{\mathbb{P}}_{X}(x) > 0 \right\}.$$

Se X e Y são duas variáveis aleatórias de $(\Omega, \mathbb{P})$ e $y \in R_Y$ então a **distribuição condicional** da variável aleatória X dado que ocorre o evento $[Y = y]$ é

$$\mathop{\mathbb{P}}_{X|Y=y}(x) := \mathbb{P}[\,X = x \mid Y = y\,]$$

para todo x. Claramente, valores diferentes de y podem resultar em distribuições condicionais diferentes. Agora, para uma variável aleatória real e somável X, a **esperança condicional de** X **dado** $[Y = y]$, para qualquer $y \in R_Y$, é dada pela equação (3.25), ou seja, vale que

$$\mathbb{E}[\,X \mid Y = y\,] = \sum_{r \in X(\Omega)} r \mathop{\mathbb{P}}_{X|Y=y}(r).$$

Se X é somável, $y \in R_Y$ e se olharmos para $\mathbb{E}[\,X \mid Y = y\,]$ como um função de y, digamos $\psi\colon \mathbb{R} \to \mathbb{R}$ dada por

$$\psi(y) := \begin{cases} \mathbb{E}[\,X \mid Y = y\,], & \text{se } y \in R_Y \\ 0, & \text{caso contrário,} \end{cases}$$

então a função

$$\mathbb{E}[\,X \mid Y\,] := \psi(Y)$$

é uma variável aleatória que passamos a chamar de **esperança condicional de** X **dado** Y e para todo $\omega \in \Omega$

$$\mathbb{E}[\,X \mid Y\,](\omega) = \psi(Y(\omega)) = \mathbb{E}[\,X \mid Y = Y(\omega)\,].$$

O valor $\mathbb{E}[\,X \mid Y\,](\omega)$ é a média ponderada dos valores $X(\upsilon)$ sobre todo ponto amostral $\upsilon \in \Omega$ tal que $Y(\upsilon) = Y(\omega)$.

Por exemplo, se um dado equilibrado é lançado duas vezes e X e Y são os valores obtidos e S := X+Y e P := X · Y, então usando a definição temos

$$\mathbb{E}[\,S \mid X = 2\,] = \sum_{r=2}^{12} r\mathbb{P}[\,S = r \mid X = 2\,] = 11/2$$

e

$$\mathbb{E}[\,P \mid Y = 1\,] = \sum_{r=1}^{36} r\mathbb{P}[\,P = r \mid Y = 1\,] = 7/2$$

embora as esperanças possam ser calculadas de maneira mais fácil usando as propriedades do valor médio. Por exemplo, para a soma temos $\mathbb{E}[\,S \mid X = 2\,] = \mathbb{E}[\,X \mid X = 2\,] + \mathbb{E}[\,Y \mid X = 2\,] = 2 + \mathbb{E}\,Y = 2 + 7/2$; para o produto temos $\mathbb{E}[\,P \mid Y = 1\,] = \mathbb{E}[\,XY \mid Y = 1\,] = \mathbb{E}[\,X \mid Y = 1\,]\mathbb{E}[\,Y \mid Y = 1\,] = \mathbb{E}[\,X \mid Y = 1\,] = 7/2$. Ainda usando a definição temos $\mathbb{E}[\,X \mid S = 5\,] = \sum_{r=1}^{6} r\mathbb{P}[\,X = r \mid S = 5\,] = 5/2$.

O valor de $\mathbb{E}[\,S \mid P\,](3,5)$ é o valor médio da soma S sobre todo par (x,y) tal que $P(x,y) = x \cdot y = 15 = P(3,5)$, a saber, os pares $(3,5)$ e $(5,3)$. Como ambos somam 8 o valor médio é $\mathbb{E}[\,S \mid P\,](3,5) = 8$. O valor de $\mathbb{E}[\,S \mid P\,](1,4)$ é o valor médio da soma S sobre todo par (x,y) tal que $P(x,y) = x \cdot y = 4 = P(1,4)$, a saber, os pares $(1,4)$, $(2,2)$ e $(4,1)$. O primeiro e o último somam 5 e o segundo par soma 4, então o valor médio é $\mathbb{E}[\,S \mid P\,](1,4) = (2/3) \cdot 5 + (1/3) \cdot 4 = 14/3$. Escrevendo de outro modo, se $P = 15$ então os resultados dos lançamentos são $(3,5)$ ou $(5,3)$, para os quais $S = 8$, de modo que

$$\mathbb{E}[\,S \mid P\,](3,5) = \psi(P(3,5)) = \psi(15) = \mathbb{E}[\,S \mid P = 15\,] = \frac{\mathbb{E}\left[\,S \cdot \mathbb{1}_{[P=15]}\right]}{\mathbb{P}[\,P = 15\,]} = \frac{8(2/36)}{2/36} = 8.$$

Se $P = 4$ então os resultados dos lançamentos são $(1,4)$, ou $(2,2)$ ou $(4,1)$, para os quais $S = 5,4,5$, respectivamente, então

$$\mathbb{E}[\,S \mid P\,](2,2) = \psi(P(2,2)) = \psi(4) = \mathbb{E}[\,S \mid P = 4\,] = \frac{5(1/36) + 4(1/36) + 5(1/36)}{3/36} = \frac{14}{3}.$$

A lei da variável aleatória $\mathbb{E}[\,S \mid P\,]$ está dada na tabela 3.3.

$\psi(P) = \mathbb{E}[\,S \mid P\,]$	2	3	4	$\frac{14}{3}$	6	7	$\frac{15}{2}$	8	9	10	11	12
$\mathbb{P}_{\psi(P)}$	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{9}{36}$	$\frac{2}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

Tabela 3.3: distribuição da soma de dois dados dado o produto.

Se $\mathbb{E}[\,X \mid Y\,]$ é uma variável aleatória então podemos calcular o seu valor médio. No exemplo acima, podemos usar a tabela 3.3 para concluir que $\mathbb{E}[\mathbb{E}[\,S \mid P\,]] = \mathbb{E}[\,\psi(P)\,] = 7$. Lembremos o exemplo 3.17, onde determinamos que $\mathbb{E}\,S = 7$ e isso não é uma coincidência. Suponhamos que X é somável de

modo que $\mathbb{E}[\,X \mid Y = y\,]$ é finita para todo $y \in R_Y$.

$$\begin{aligned}
\mathbb{E}\,X &= \sum_x x\mathbb{P}[\,X = x\,] \\
&= \sum_x x\left(\sum_y \mathbb{P}([X = x] \cap [Y = y])\right) \\
&= \sum_x \sum_y x\mathbb{P}[\,X = x \mid Y = y\,]\mathbb{P}[\,Y = y\,] \\
&= \sum_y \sum_x x\mathbb{P}[\,X = x \mid Y = y\,]\mathbb{P}[\,Y = y\,] \\
&= \sum_y \mathbb{E}[\,X \mid Y = y\,]\mathbb{P}[\,Y = y\,] \\
&= \sum_y \psi(y)\mathbb{P}[\,Y = y\,] \\
&= \mathbb{E}\,\psi(Y) = \mathbb{E}[\,\mathbb{E}[\,X \mid Y\,]\,]
\end{aligned}$$

pelo teorema de Fubini para séries (s.5). Essa dedução vale sempre que $X \geqslant 0$ ou X é somável, nesse último caso $\mathbb{E}[\,X \mid Y\,]$ também é somável.

Teorema 3.42 *Se X é uma variável aleatória somável, então a variável aleatória $\mathbb{E}[\,X \mid Y\,]$ é somável e*

$$\mathbb{E}\,X = \sum_{y \in R_Y} \mathbb{E}[\,X \mid Y = y\,]\mathbb{P}[\,Y = y\,]. \tag{3.26}$$

Ademais $\mathbb{E}[\,\mathbb{E}[\,X \mid Y\,]\,] = \mathbb{E}\,X$. □

Notemos que fazendo $X = \mathbb{1}_A$ na equação (3.26) a expressão que obtemos é o teorema da probabilidade total, teorema 1.26, página 27.

Por exemplo, podemos usar o teorema acima para calcular o número esperado de lançamentos de uma moeda equilibrada até sair coroa (já calculamos essa esperança no exemplo 3.21, página 134) condicionando no resultado do primeiro lançamento. Seja X o número de lançamentos até sair coroa, então usando o teorema 3.42

$$\mathbb{E}\,X = \mathbb{E}[\,X \mid X = 1\,]\mathbb{P}[\,X = 1\,] + \mathbb{E}[\,X \mid X > 1\,]\mathbb{P}[\,X > 1\,] = \frac{1}{2} + \mathbb{E}[\,X \mid X > 1\,]\frac{1}{2}$$

e como uma variável aleatória com distribuição geométrica não tem memória $\mathbb{P}[\,X = n+1 \mid X > 1\,] = \mathbb{P}[\,X = n\,]$ (exercício 3.53, página 170), logo

$$\begin{aligned}
\mathbb{E}[\,X \mid X > 1\,] &= \sum_{n \geqslant 1} n\mathbb{P}[\,X = n \mid X > 1\,] = \sum_{n \geqslant 2} n\mathbb{P}[\,X = n \mid X > 1\,] \\
&= \sum_{n \geqslant 1} (n+1)\mathbb{P}[\,X = n+1 \mid X > 1\,] = \sum_{n \geqslant 1} (n+1)\mathbb{P}[X = n] = \mathbb{E}[\,X\,] + 1
\end{aligned}$$

portanto $\mathbb{E}\,X = (1/2) + (1/2)(\mathbb{E}\,X + 1)$ donde concluímos que $\mathbb{E}\,X = 2$. Ainda, podemos usar a mesma estratégia para calcular $\mathbb{E}\,X^2$

$$\begin{aligned}
\mathbb{E}\,X^2 &= \mathbb{E}\Big[X^2 \;\Big|\; X > 1\Big]\frac{1}{2} + \mathbb{E}\Big[X^2 \;\Big|\; X = 1\Big]\frac{1}{2} \\
&= \mathbb{E}\,(X+1)^2\,\frac{1}{2} + \mathbb{E}\Big[X^2 \;\Big|\; X = 1\Big]\frac{1}{2} \\
&= \Big(\mathbb{E}\,X^2 + 2\,\mathbb{E}\,X + 1\Big)\frac{1}{2} + \frac{1}{2} \\
&= \mathbb{E}\,X^2\,\frac{1}{2} + \mathbb{E}\,X + \frac{1}{2} + \frac{1}{2} \\
&= \mathbb{E}\,X^2\,\frac{1}{2} + 3
\end{aligned}$$

portanto $\mathbb{E}\,X^2 = 6$.

3.3.1 *DESALEATORIZAÇÃO*: O MÉTODO DAS ESPERANÇAS CONDICIONAIS

Vimos na página 69, no caso do algoritmo para a verificação do produto de matrizes, uma estratégia que reduz a quantidade de bits aleatórios usados naquele caso específico sem comprometer significativamente o desempenho do algoritmo. Nessa seção vamos ver uma estratégia mais genérica que resulta num algoritmo determinístico mas cujo desempenho depende de conseguirmos computar esperanças condicionais de modo eficiente. A seguir nós usaremos a notação $[X_1 = x_1, X_2 = x_2, \ldots, X_i = x_i]$ com o significado de $[X_1 = x_1] \cap [X_2 = x_2] \cap \cdots \cap [X_i = x_i]$.

Seja $f\colon S^n \to \mathbb{R}$ uma função em que S é finito e que $\mathbb{E}\,f(X_1, \ldots, X_n) \geqslant \mu$. Por exemplo, no MAX-CUT $(X_1, \ldots, X_n) \in_R \{0,1\}^n$ corresponde a uma bipartição nos vértices de um grafo de n vértices e f é o número de arestas no corte, nesse caso sabemos que $\mathbb{E}\,f$ é pelo menos metade do número de arestas do grafo.

Determinar um ponto $(x_1, \ldots, x_n) \in S^n$ tal que $f(x_1, \ldots, x_n) \geqslant \mu$ fazendo uma busca exaustiva em S^n pode não ser eficiente pois o conjunto pode ser muito grande. Entretanto, pode ser possível determinar tal ponto de modo eficiente quando for possível computar as esperanças condicionais $\mathbb{E}[\,f(X_1, \ldots, X_n) \mid X_1 = x_1, \ldots, X_i = x_i\,]$, para todo i, de modo eficiente.

Para cada $i = 1, 2, \ldots, n$, dados $x_1, \ldots, x_i$ tais que $\mathbb{E}[\,f(X_1, \ldots, X_n) \mid X_1 = x_1, \ldots, X_i = x_i\,] \geqslant \mu$, temos pelo teorema 3.42

$$\mathbb{E}[\,\mathbb{E}[\,f(X_1, \ldots, X_n) \mid X_1 = x_1, \ldots, X_i = x_i, X_{i+1}\,]\,] = \mathbb{E}[\,f(X_1, \ldots, X_n) \mid X_1 = x_1, \ldots, X_i = x_i\,] \geqslant \mu$$

logo, pelo princípio de primeiro momento, deve existir algum r em S para o qual temos o limitante para o valor esperado $\mathbb{E}[\,f(X_1, \ldots, X_n) \mid X_1 = x_1, \ldots, X_i = x_i, X_{i+1} = r\,] \geqslant \mu$ assim, testamos (de modo eficiente) para cada $r \in S$ se

$$\mathbb{E}[\,f(X_1, \ldots, X_n) \mid X_1 = x_1, \ldots, X_i = x_i, X_{i+1} = r\,] \stackrel{?}{\geqslant} \mu.$$

Repetindo essa estratégia para cada i ao final teremos $(x_1,\ldots,x_n)$ tal que $f(x_1,\ldots,x_n) \geqslant \mu$.

MAX-E3SAT O problema que estudamos na seção 3.2.1 é, dado uma fórmula booleana 3-CNF $\mathcal{C} = \{C_1,\ldots,C_m\}$, determinar uma valoração $(x_1,\ldots,x_n) \in \{0,1\}^n$ para as variáveis $V = \{v_1,\ldots,v_n\}$ da fórmula tal que $(v_1,\ldots,v_n) = (x_1,\ldots,x_n)$ satisfaz pelo menos 7/8 das cláusulas $C_1,\ldots,C_m$. No algoritmo aleatorizado sorteamos a valoração uniformemente em $\{0,1\}^n$ até que descobrirmos uma valoração com a propriedade desejada.

Agora, sejam $X_1,\ldots,X_n \in_R \{0,1\}$ variáveis aleatórias independentes que representam os resultados dos sorteios. Definimos a função $f(x_1,\ldots,x_n)$ como o número de cláusulas satisfeitas pela valoração $(x_1,\ldots,x_n) \in \{0,1\}^n$. Assim, como sabemos da seção 3.2.1

$$\mathbb{E}\, f(X_1,\ldots,X_n) \geqslant \frac{7m}{8}.$$

Aplicamos a estratégia de *desaleatorização* da esperança condicional para f. Se é dado $(x_1,\ldots,x_i)$ então já sabemos os valores lógicos das variáveis $v_1, v_2, \ldots, v_i$. Precisamos computar de modo eficiente

$$\mathbb{E}[\, f(X_1,\ldots,X_n) \mid X_1 = x_1,\ldots,X_i = x_i, X_{i+1} = r\,] = \sum_{j=1}^{m} \mathbb{E}\Big[\, \mathbb{1}_{[C_j=1]} \,\Big|\, X_1 = x_1,\ldots,X_i = x_i, X_{i+1} = r\,\Big]$$

para $r = 0$ e para $r = 1$. Para r fixo, cada termo da soma no lado direito é calculada considerando quatro casos:

1. se a valoração parcial deixa 3 variáveis livres na cláusula então o valor esperado condicional de $\mathbb{1}_{[C_j=1]}$ é 7/8;
2. se a valoração parcial deixa 2 variáveis livres na cláusula então se $C_j = 1$ o valor esperado condicional é 1, senão o valor esperado condicional de $\mathbb{1}_{[C_i=1]}$ é 3/4;
3. se a valoração parcial deixa 1 variável livre na cláusula então se $C_j = 1$ o valor esperado condicional é 1, senão o valor esperado condicional de $\mathbb{1}_{[C_j=1]}$ é 1/2;
4. se a valoração parcial não deixa variável livre na cláusula então o valor esperado condicional de $\mathbb{1}_{[C_i=1]}$ é 0 ou 1, valendo 1 se, e só se, a cláusula está satisfeita;

portanto cada termo é avaliado em tempo constante e em $O(m)$ passos a soma fica determinada. Realizada a soma para $r = 0$ e $r = 1$, definimos x_{i+1} como o valor de r para o qual a esperança é pelo menos $7m/8$.

Assim, a esperança é calculada com custo linear em m e determinamos uma valoração para cada uma das n variáveis que, no final, satisfaz pelo menos $7m/8$ cláusulas com custo total $O(nm)$.

CORTE GRANDE Vejamos a aplicação desse método no problema da seção 3.2.2 de determinar um corte num grafo com pelo menos metade das arestas. Dado $G = (V, E)$ com vértices $V = \{1, \ldots, n\}$ e m arestas, o algoritmo aleatorizado sorteia $(x_1, \ldots, x_n) \in \{0,1\}^n$ para formar um subconjunto $A = \{i \in V : x_i = 1\}$, se $|\nabla(A)| \geqslant m/2$ então encontrou um corte grande, senão repete o sorteio. Esse algoritmo pode ser *desaleatorizado* com o método da esperança condicional e o resultado é um tão algoritmo eficiente quanto o aleatorizado.

Definimos a função $f(x_1, \ldots, x_n) := |\nabla(A)|$ em que $(x_1, \ldots, x_n)$ é o vetor característico do conjunto A definido acima. Sejam $X_1, \ldots, X_n$ variáveis aleatórias independentes com distribuição uniforme em $\{0,1\}$, sabemos da seção 3.2.2 que

$$\mathbb{E}\, f(X_1, \ldots, X_n) \geqslant \frac{m}{2}.$$

Agora, aplicamos a estratégia dada acima para f. Se é dado $(x_1, \ldots, x_i) \in \{0,1\}^i$, para $1 \leqslant i < n$, então já sabemos quais dos vértice dentre $1, 2, \ldots, i$ estão em A e vamos decidir se $i+1$ fará parte do conjunto A calculando as esperanças $\mathbb{E}[\, f(X_1, \ldots, X_n) \mid X_1 = x_1, \ldots, X_{i+1} = r\,]$ para $r = 0$ e para $r = 1$; para pelo menos uma dessas duas escolhas devemos ter a esperança condicional pelo menos $m/2$.

Para cada aresta com ambos os extremos em $\{1, \ldots, i\}$ já sabemos se ela está no corte ou não e toda outra aresta tem probabilidade $1/2$ de estar no corte, portanto, para $r \in \{0,1\}$ fixo

$$\mathbb{E}[\, f(X_1, \ldots, X_n) \mid X_1 = x_1, \ldots, X_i = x_i, X_{i+1} = r\,] = \sum_{\{j,k\} \in E} \mathbb{E}\Big[\, \mathbb{1}_{[\{j,k\} \in \nabla(A)]} \,\Big|\, X_1 = x_1, \ldots, X_i = x_i, X_{i+1} = r \Big]$$

e cada termos da soma é calculado considerando três casos:

1. se $j, k \in \{1, \ldots, i+1\}$ então o valor esperado condicional para $\mathbb{1}_{[\{j,k\} \in \nabla(A)]}$ é 1 ou 0, valendo 1 se, e só se, $\{j,k\} \in \nabla(A)$;
2. se $j, k \in \{i+2, \ldots, n\}$ então o valor esperado condicional para $\mathbb{1}_{[\{j,k\} \in \nabla(A)]}$ é $1/2$;
3. se $j \in \{1, \ldots, i+1\}$ e $k \in \{i+1, \ldots, n\}$, ou se $k \in \{1, \ldots, i+1\}$ e $j \in \{i+1, \ldots, n\}$, então o valor esperado condicional para $\mathbb{1}_{[\{j,k\} \in \nabla(A)]}$ é $1/2$.

Assim, a esperança $\mathbb{E}[\, f(X_1, \ldots, X_n) \mid X_1 = x_1, \ldots, X_{i+1} = x_{i+1}\,]$ é calculada com custo linear em $|E| = m$ e ao final de n passos determinamos $(x_1, \ldots, x_n)$ (ou seja, um conjunto A) tal que $f(x_1, \ldots, x_n) \geqslant m/2$ (ou seja, $|\nabla(A)| \geqslant m/2$) com custo $O(nm)$.

3.3.2 *SKIP LISTS*

Uma **skip list** também é uma estrutura de dados para representar conjuntos dinâmicos (Pugh, 1989). Essa estrutura de dados é uma coleção de listas ligadas que usa aleatoriedade para se compor-

tar, em desempenho, como uma árvore de busca balanceada[5] e com a vantagem de ser mais eficiente e mais fácil de manter que uma árvore balanceada.

Numa *skip list* S mantemos um conjunto, que por abuso de notação também denotamos por S, de elementos de um universo U dotado de ordem total. Uma *skip list* pode ser descrita da seguinte maneira: os elementos de S são mantidos em uma lista ligada ordenada chamada *lista do nível 0* e denotada por S_0. Dada a lista S_i do nível i ($i \geqslant 0$), definimos a *lista do nível* $i+1$, denotada S_{i+1}, tomando um subconjunto aleatório de S_i onde cada elemento é escolhido com probabilidade 1/2, com as escolhas independentes. A lista ligada S_{i+1} é ordenada e cada elemento seu aponta para sua cópia imediatamente abaixo no nível S_i. No último nível temos $S_\ell = \varnothing$. Além dos elementos de S mantemos dois sentinelas, o $-\infty$ no começo de todas as listas e o $+\infty$ no fim de todas as listas. A figura 3.3 descreve um exemplo de uma *skip list*.

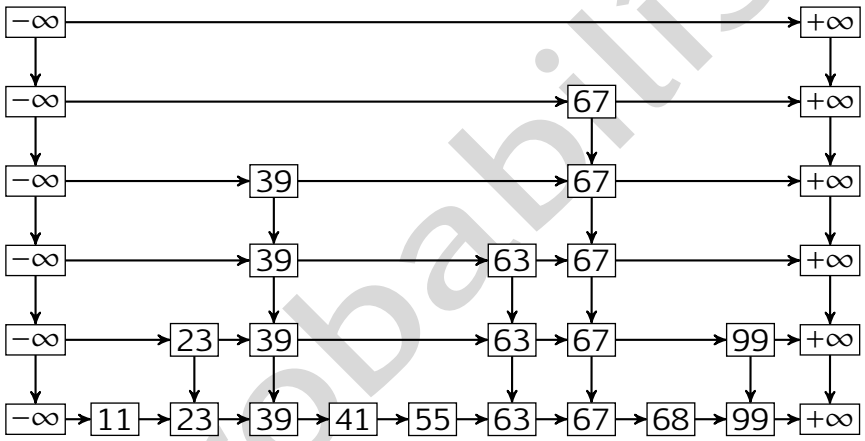


Figura 3.3: exemplo de uma *skip list* com 6 níveis que representa o conjunto $\{11, 23, 39, 41, 55, 63, 67, 68, 99\}$.

As operações de dicionário na *skip list* são descritas a seguir.

busca: dado $x \in U$, uma busca devolve um apontador para $x \in S_0$ ou para o menor elemento de S_0 maior que x, caso $x \notin S$. O início da lista é dado por um apontador S para o sentinela $-\infty$ no último nível. A busca começa em S, se o próximo elemento da lista ligada no nível atual é menor ou igual a x então a busca continua nesse nível da lista a partir desse próximo elemento, senão desce um nível se for possível ou termina caso contrário. A figura 3.4 destaca um exemplo de busca na estrutura da figura 3.3;

[5] Uma árvore binária balanceada é uma árvore com raiz r cujas subárvores esquerda e direita satisfazem algum critério de balanceamento como, por exemplo, as alturas diferem de no máximo 1. Um critério bom garante que a altura da árvore é logarítmica no número de elementos, o que faz uma busca ser rápida. Em contrapartida, toda alteração exige o esforço de recompor o balanceamento.

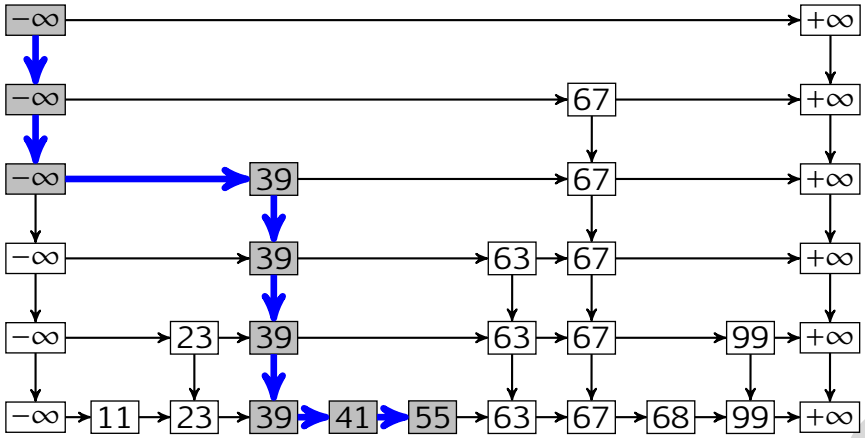


Figura 3.4: o caminho da busca por 55 na *skip list* da figura 3.3.

inserção: dado $x \in U$ a inserção de x em S coloca x no lugar apropriado da estrutura caso ele não esteja. Supondo que $x \notin S$, uma busca por x na *skip list* determina a posição do sucessor de x em S_0. A inserção é feita na lista S_0 e em seguida jogamos uma moeda, se der cara replicamos o elemento na lista do nível 1 e jogamos uma moeda para decidir se há inclusão no nível 2, e assim sucessivamente até sair coroa, quando paramos de replicar o elemento novo. Eventualmente, teremos que aumentar o número de níveis da *skip list* numa inserção;

remoção: dado um apontador para $x \in S$ uma remoção retira toda ocorrência desse elemento da estrutura; se o penúltimo nível for uma lista unitária com o elemento removido então o número de níveis diminuirá de um.

O número de comparações realizadas por uma operação de dicionário sobre uma *skip list* depende do número de níveis da estrutura, que é uma variável aleatória. Como S_{i+1} é formado a partir de S_i escolhendo ou não os elementos de S_i com probabilidade 1/2, em média S_{i+1} tem metade do tamanho de S_i, logo esperamos que o número de níveis em S de cardinalidade n seja da ordem de $\log_2(n)$. Como os níveis mais altos são esparsos esperamos caminhar pouco em cada nível de modo que, no total, esperamos que a quantidade de passos numa busca seja de ordem logarítmica no tamanho do conjunto. Vamos estudar essa estrutura de dados e para isso começamos definindo alguns parâmetros importantes. Façamos $n := |S|$, a quantidade de elementos representados na estrutura de dados.

Para todo $x \in S$ definimos a *altura* de x, denotada $h(x)$, como o número de sorteios realizados quando da inserção de x. No exemplo da figura 3.3, $h(55) = 1$ e $h(63) = 3$. A altura de x também é o número de cópias de x na estrutura. Denotamos por $H = H(S)$ a *altura* da estrutura S dada por

$$H(S) := \max\{\, h(x) \colon x \in S \,\}$$

de modo que a *skip list* S é composta pelos níveis $S_0, S_1, \ldots, S_H$. No nosso exemplo $H = h(67) = 5$.

As alturas definidas no parágrafo anterior são variáveis aleatórias. Podemos computar a esperança da variável aleatória H usando o fato dela assumir valores inteiros positivos, pela proposição 3.22, página 135,

$$\mathbb{E}\,\mathsf{H} = \sum_{i\geqslant 1} \mathbb{P}[\,\mathsf{H} \geqslant i\,] = \sum_{i=0}^{\lceil \log_2 n\rceil} \mathbb{P}[\,\mathsf{H} > i\,] + \sum_{i>\lceil \log_2 n\rceil} \mathbb{P}[\,\mathsf{H} > i\,]. \tag{3.27}$$

Claramente, $h(x) \sim \text{Geom}(1/2)$ logo $\mathbb{P}[\,h(x) = t\,] = (1/2)^{t-1}(1/2)$ para todo inteiro positivo t. Do exercício 3.5, página 118, concluímos que $\mathbb{P}[\,h(x) > t\,] = 2^{-t}$. Reunindo essas informações

$$\mathbb{P}[\,h(x) = t\,] = 2^{-t} = \mathbb{P}[\,h(x) > t\,]. \tag{3.28}$$

De volta na equação (3.27), na primeira soma no lado direto dessa equação limitaremos trivialmente a probabilidade tomando $\mathbb{P}[\,\mathsf{H} > i\,] \leqslant 1$, de modo que

$$\sum_{i=0}^{\lceil \log_2 n\rceil} \mathbb{P}[\,\mathsf{H} > i\,] \leqslant \lceil \log_2 n\rceil + 1$$

e na segunda soma, para cada i

$$\mathbb{P}[\,\mathsf{H} > i\,] \leqslant \mathbb{P}\left[\bigcup_{x\in \mathsf{S}} [h(x) > i]\right] \leqslant \sum_{x\in \mathsf{S}} \mathbb{P}[\,h(x) > i\,] \leqslant n\left(\frac{1}{2}\right)^i$$

logo

$$\begin{aligned}
\mathbb{E}\,\mathsf{H} &\leqslant \lceil \log_2 n\rceil + 1 + \sum_{i>\lceil \log_2 n\rceil} n\left(\frac{1}{2}\right)^i \\
&\leqslant \lceil \log_2 n\rceil + 1 + n\left(\sum_{i\geqslant 0}\left(\frac{1}{2}\right)^i - \sum_{i=0}^{\lceil \log_2 n\rceil}\left(\frac{1}{2}\right)^i\right) \\
&= \lceil \log_2 n\rceil + 1 + n\left(2 - \frac{1-(1/2)^{\lceil \log_2 n\rceil+1}}{1-(1/2)}\right) \\
&= \lceil \log_2 n\rceil + 1 + n\left(\frac{1}{2}\right)^{\lceil \log_2 n\rceil} \\
&< \lceil \log_2 n\rceil + 2
\end{aligned}$$

ou seja, a estrutura tem altura esperada logarítmica no número de elementos de S, como numa árvore balanceada de busca. Mais que isso, altura é logarítmica no número de elementos de S com alta probabilidade.

Proposição 3.43 *Se* S *é uma* skip list *para um conjunto com* n *elementos então para a altura* $\mathsf{H} = \mathsf{H}(\mathsf{S})$ *valem*

$$\mathbb{E}\,\mathsf{H} \leqslant \log_2(4n) \qquad e \qquad \mathbb{P}[\,\mathsf{H} > 2\log_2(n)\,] < 1/n.$$

Demonstração. Para a esperança de H basta observar que $\log_2(n)+2 = \log_2(4n)$. Da equação (3.28) deduzimos que a probabilidade de um elemento qualquer de S ter altura maior que $c\log_2 n$ é $2^{-c\log_2 n} = n^{-c}$ para qualquer constante positiva c, portanto, a probabilidade de existir algum elemento em S com altura maior que $c\log_2 n$ é

$$\mathbb{P}\left(\bigcup_{x\in S}[h(x) > c\log_2 n]\right) \leqslant \sum_{x\in S}\mathbb{P}[\,h(x) > c\log_2 n\,] = n\frac{1}{n^c} = \frac{1}{n^{c-1}}$$

portanto, fazendo $c = 2$ segue que $\mathbb{P}[\,H > 2\log_2(n)\,] \leqslant 1/n$. □

Observação 3.44 (*sobre o tamanho da estrutura*). Se $N := \sum_{i=0}^{H}|S_i|$ é o tamanho da *skip list*, então de $N = \sum_{x\in S} h(x)$, da linearidade da esperança e de $\mathbb{E}[\,h(x)\,] = 2$

$$\mathbb{E}\,N = \sum_{x\in S}\mathbb{E}[\,h(x)\,] = 2n. \tag{3.29}$$

Notemos que de $N = \sum_{i=1}^{H}|S_i|$ não é imediato valer que $\mathbb{E}\,N = \sum_{i=1}^{\mathbb{E}\,H}\mathbb{E}\,|S_i|$ por causa de independência ou não das variáveis envolvidas (veja o exercício 3.63 no final deste capítulo). Pode-se provar que o número de itens N é $O(n)$ com alta probabilidade e faremos isso na seção 6.3.1. ◊

Para determinar o número de passos esperados numa busca sobre uma *skip list* nós vamos limitar número de comparações feitas durante uma busca em dois casos: as comparações feitas nos níveis mais altos e as comparações feitas nos níveis mais baixos da *skip list*. Nos níveis mais altos o número de comparações é no máximo a quantidade total de elementos nesses níveis.

Proposição 3.45 *O número esperado de comparações feitas nos níveis mais altos de uma* skip list, *do nível* $\lceil\log_2 n\rceil + 1$ *até o nível* H, *durante uma busca é* $O(1)$.

Demonstração. Vamos mostrar que esses níveis contribuem pouco com o custo da busca limitando o custo da busca nesses níveis pelo número total de elementos neles. Façamos

$$\ell := \lceil\log_2 n\rceil + 1, \quad p := (1/2)^\ell \quad \text{e} \quad N' := \sum_{i=\ell}^{H}|S_i|.$$

Notemos que em $S_\ell, S_{\ell+1}, S_{\ell+2}, \ldots, S_H$ temos uma *skip list* para o conjunto S_ℓ logo pela equação (3.29), $\mathbb{E}[\,N' \,|\, |S_\ell| = k\,] \leqslant 2k$. Usando o teorema da esperança total, equação (3.26) na página 154, escrevemos

$$\mathbb{E}\,N' = \sum_{k=0}^{n}\mathbb{E}[\,N' \,\big|\, |S_\ell| = k\,]\mathbb{P}[\,|S_\ell| = k\,]$$

e como qualquer $x \in S$ ocorre em S_ℓ com probabilidade p, temos $|S_\ell| \sim b(n,p)$ de modo que

$$\mathbb{E}\,N' = \sum_{k=0}^{n}\mathbb{E}[\,N' \,\big|\, |S_\ell| = k\,]\binom{n}{k}p^k(1-p)^{n-k} \leqslant 2\sum_{k=0}^{n}k\binom{n}{k}p^k(1-p)^{n-k}.$$

O somatório corresponde ao valor esperado de uma variável aleatória binomial com parâmetros n e p, então $\mathbb{E}\,N' \leqslant 2np$. Ainda,

$$np = n\left(\frac{1}{2}\right)^{\ell} = n\left(\frac{1}{2}\right)^{\lceil \log_2 n\rceil+1} \leqslant n\left(\frac{1}{2}\right)^{\log_2 n} = 1$$

portanto $\mathbb{E}\,N' \leqslant 2$, ou seja, mesmo que uma busca percorra todos os elementos dos níveis $S_\ell, \ldots, S_H$, o número esperado de comparações é limitado superiormente por uma constante. □

O segundo passo da nossa análise é estimar o número de comparações nos níveis mais baixos onde se concentram a maior parte dos elementos. Nesse caso, veremos que a probabilidade da busca ficar muito tempo num nível é pequena, portanto, o tempo gasto é essencialmente determinado pela altura da *skip list*. A ideia chave é explicada no próximo parágrafo.

Seja $x_1 < x_2 < \cdots < x_n$ a lista S_0 e denotemos por y o elemento que está sendo buscado. Suponhamos que a busca tenha chegado em S_i a partir de S_{i+1} pelo elemento x_j. Notemos que $x_j \in S_{i+1}$ e seja $x_m \in S_{i+1}$ o próximo elemento de S_{i+1} depois de x_j. Com essas hipóteses $x_j \leqslant y < x_m$ e nenhum dentre os elementos consecutivos $x_{j+1}, x_{j+2}, \ldots, x_{m-1} \in S_0$ pertence a S_{i+1}. Se a busca realiza t

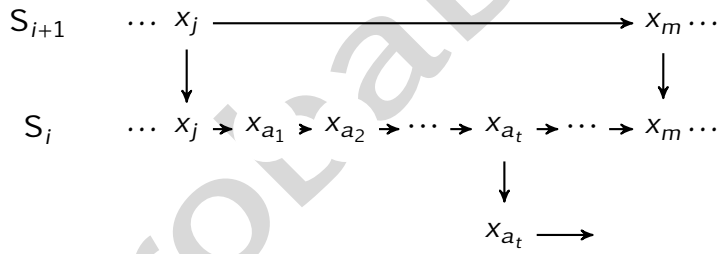


Figura 3.5: visão de uma busca no nível S_i.

passos em S_i então temos uma sequência $x_{a_1}, x_{a_2}, \ldots, x_{a_t}$ com os t maiores elementos de S_i que são menores ou iguais a y, precedidos por x_j e que não ocorrem em S_{i+1} (veja a figura 3.5); a sequência $x_j, x_{a_1}, x_{a_2}, \ldots, x_{a_t}$ com a configuração descrita acima na *skip list* corresponde a lançamentos de moeda que resultam em uma cara seguida de t coroas, já que só o primeiro elemento da sequência é replicado em S_{i+1}.

Lema 3.46 *Numa busca sem sucesso em* S, *o número esperado de comparações feitas no nível* i *é* $O(1)$ *para cada* i, $0 \leqslant i \leqslant \lceil \log_2 n \rceil$.

Demonstração. Seja y o elemento buscado, S uma *skip list* e suponhamos que $y \notin S$. Seja X_i o número de passos dados pela busca em S_i, para $0 \leqslant i \leqslant \lceil \log_2 n \rceil$. No exemplo de busca ilustrado na figura 3.4, página 159, $X_5 = X_4 = X_2 = X_1 = 0$, $X_3 = 1$ e $X_0 = 2$.

A quantidade elementos em S_i menores ou iguais a y é uma variável aleatória que denotamos por $M_i(y)$. Fixado i e condicionando a $M = M_i(y)$ o número esperado de passos no nível i é

$$\mathbb{E}\, X_i = \sum_{k=1}^{n} \mathbb{E}[\, X_i \mid M = k\,]\mathbb{P}[\, M = k\,]$$

Para todo k e todo t tal que $1 \leqslant k \leqslant n$ e $0 \leqslant t \leqslant k$ temos $X_i = t$, dado que $M = k$, se e só se os t maiores elementos menores ou iguais a y em S_i não ocorrem em S_{i+1}, mas o elemento que precede esses t ocorre em S_{i+1} (tal predecessor pode ser o sentinela), logo

$$\mathbb{P}[\, X_i = t \mid M = k\,] \leqslant \left(\frac{1}{2}\right)^t$$

de modo que usando (s.6c) obtemos

$$\mathbb{E}[\, X_i \mid M = k\,] = \sum_{j=0}^{k} j\mathbb{P}[\, X_i = j \mid M = k\,] \leqslant \sum_{j=0}^{k} \frac{j}{2^j} \leqslant 2$$

e, então,

$$\mathbb{E}\, X_i = \sum_{k=1}^{n} \mathbb{E}[\, X_i \mid M = k\,]\mathbb{P}[\, M = k\,] \leqslant \sum_{k=1}^{n} 2\mathbb{P}[\, M = k\,] = 2. \tag{3.30}$$

ou seja, o número esperado de passos no nível i é no máximo 2. □

Como nos níveis superiores vimos que são $O(1)$ comparações e nos níveis inferiores o número de comparações em cada nível é constante, o número total de comparações é, em média, proporcional a $\log_2 n$.

Teorema 3.47 *O número esperado de comparações feitas por uma busca em uma* skip list *que representa um conjunto com n elementos é $O(\log n)$.*

Demonstração. Dados uma *skip list* S e y, consideremos uma busca por y em S. Podemos assumir $y \notin S$, pois isso resulta num limitante superior para o custo de uma busca.

Com a notação do resultado anterior o número de comparações feitas pela busca no nível i, contando com a comparação que faz o apontador descer um nível (ou descobrir que $x \notin S$ achando alguém maior) é $X_i + 1$. O custo da busca nos $O(\log n)$ níveis mais baixos da *skip list* é

$$\mathbb{E} \sum_{i=0}^{\lceil \log_2 n \rceil} (X_i + 1) = \sum_{i=0}^{\lceil \log_2 n \rceil} \mathbb{E}\, X_i + 1 \leqslant 3\log_2 n$$

e o custo nos outros níveis da estrutura é $O(1)$, logo o custo total é $O(\log n) + O(1) = O(\log n)$. □

No teorema 6.29 provamos que o tempo de busca é como acima com alta probabilidade.

3.3.3 O MÉTODO PROBABILÍSTICO REVISITADO E 2º MOMENTO

Seja X uma variável aleatória que assume valores não negativos e com quadrado somável (logo X também é). Pelo teorema da esperança total,

$$\mathbb{E}[X^2] = \mathbb{E}[X^2 \mid X > 0]\,\mathbb{P}[X > 0] + \mathbb{E}[X^2 \mid X = 0]\,\mathbb{P}[X = 0].$$

No entanto $\mathbb{E}[X^2 \mid X = 0] = 0$ o que nos leva a concluir que $\mathbb{E}[X^2] = \mathbb{E}[X^2 \mid X > 0]\,\mathbb{P}[X > 0]$. Ademais, usando a linearidade da esperança condicionada nós obtemos de $\mathbb{E}[(X - \mathbb{E}[X \mid X > 0])^2 \mid X > 0] \geqslant 0$ que $\mathbb{E}[X^2 \mid X > 0] \geqslant \mathbb{E}[X \mid X > 0]^2$. Logo

$$\mathbb{E}[X^2]\,\mathbb{P}[X > 0] \geqslant (\mathbb{E}[X \mid X > 0]\,\mathbb{P}[X > 0])^2 = (\mathbb{E}\,X)^2$$

donde concluímos a seguinte desigualdade enunciada no teorema abaixo.

Teorema 3.48 (princípio de segundo momento) *Seja X uma variável aleatória que assume valores não negativos e com quadrado somável. Então*

$$\mathbb{P}[X > 0] \geqslant \frac{(\mathbb{E}\,X)^2}{\mathbb{E}[X^2]} \tag{3.31}$$

se $\mathbb{E}[X^2] \neq 0$. □

Além disso, se $X \geqslant 0$ assume valores inteiros, então $\mathbb{E}\,X = \mathbb{E}[X \mid X > 0]\,\mathbb{P}[X > 0]$ donde

$$\mathbb{P}[X > 0] \leqslant \mathbb{E}\,X \tag{3.32}$$

mais que isso, de $\mathbb{E}\,X = \sum_{i \geqslant 1} i\,\mathbb{P}[X = i]$ temos, para todo inteiro $t \geqslant 1$, que $\mathbb{E}\,X \geqslant \sum_{i \geqslant t} i\,\mathbb{P}[X = i] \geqslant t \sum_{i \geqslant t} \mathbb{P}[X = i]$, ou seja,

$$\mathbb{P}[X \geqslant t] \leqslant \frac{\mathbb{E}\,X}{t} \tag{3.33}$$

uma desigualdade de primeiro momento (exercício 3.37, página 145) que é um caso particular da **desigualdade de Markov** (eq. (6.1), pág. 306).

Exemplo 3.49 (*hashing*). Tomemos uma família de funções de hash $\mathcal{H} \subset M^U$ 2-a-2 independentes, isto é, tais que

$$\mathop{\mathbb{P}}_{h \in \mathcal{H}}([h(x) = i] \cap [h(y) = j]) = \frac{1}{|M|^2}, \tag{3.34}$$

para quaisquer $x, y \in U$ distintos e para quaisquer $i, j \in M = \{0, \ldots, m-1\}$. Sejam $S \subset M$ e X a quantidade de elementos de S que são mapeados em 0. Por (3.32) acima

$$\mathbb{P}[X \geqslant 1] \leqslant \mathbb{E}\,X = \frac{|S|}{m}.$$

Agora, se escrevemos X como soma das variáveis aleatórias indicadoras dos elementos de S serem mapeados no 0, então $\mathbb{E}\, X^2$ é a soma sobre todo par (x, y), com cada coordenada em S, da probabilidade de ambos serem mapeados em 0

$$\mathbb{E}[\,X^2\,] = \sum_{x\in S}\sum_{y\in S}\mathbb{E}\Big[\,\mathbb{1}_{x\in S}\mathbb{1}_{y\in S}\,\Big] = \sum_{x\in S}\mathbb{P}[\,h(x)=0\,]\sum_{y\in S}\mathbb{P}[\,h(y)=0 \mid h(x)=0\,]$$

usando o fato das imagens serem independentes

$$\mathbb{E}[\,X^2\,] = \sum_{x\in S}\frac{1}{m}\left(1+\sum_{\substack{y\in S\\ y\neq x}}\frac{1}{m}\right) = \frac{|S|}{m}\left(1+\frac{|S|-1}{m}\right) \leqslant \frac{|S|}{m}\left(1+\frac{|S|}{m}\right)$$

e da equação (3.31)

$$\mathbb{P}[\,X \geqslant 1\,] \geqslant \frac{(\mathbb{E}\,X)^2}{\mathbb{E}[\,X^2\,]} = \frac{|S|/m}{1+|S|/m} = \frac{|S|}{|S|+m}$$

reunindo as desigualdades

$$\frac{|S|}{|S|+m} \leqslant \mathbb{P}[\,X\neq 0\,] \leqslant \frac{|S|}{m}.$$

Na seção 6.2.2 estudaremos com mais detalhes as famílias de funções de hash. ◊

Exemplo 3.50 (*triângulos em grafos aleatórios*). Três vértices de um grafo definem um triângulo, denotado genericamente por K^3, se todas as três arestas definidas pelos vértice estão presentes. Denotamos por $\mathcal{G}_{n,p}$ o modelo binomial de grafo aleatório obtido considerando o conjunto de vértices $V = \{1, 2, \ldots, n\}$ e cada par $\{u, v\}$ está presente no conjunto de arestas com probabilidade p. Uma tripla de vértices $\{u, v, w\} \subset V$ formam um triângulo no $\mathcal{G}_{n,p}$ com probabilidade p^3.

Consideremos uma enumeração das triplas de vértices e X_i a variável aleatória indicadora de presença de triângulo na i-ésima tripla. Se X é a quantidade de triângulos no $\mathcal{G}_{n,p}$, o valor esperado para número de triângulos é

$$\mathbb{E}\,X = \sum_{i=1}^{\binom{n}{3}}\mathbb{E}\,X_i = \sum_{i=1}^{\binom{n}{3}}p^3 = \binom{n}{3}p^3 = \left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\frac{n^3p^3}{6},$$

e vamos estudar essa esperança quando p é dada em função de n. Tomemos $p = n^{-\alpha}$ para uma constante $\alpha > 0$.

Se $\alpha > 1$, então $\mathbb{E}\,X = \Theta(n^{3-3\alpha}) = o(1)$ quando $n \to \infty$, logo $\mathbb{P}[\,X > 0\,] = o(1)$ pelo primeiro momento, ou seja, quase certamente $\mathcal{G}_{n,p}$ não tem triângulo nesse caso.

Se $\alpha < 1$ então $\mathbb{E}\,X \to \infty$ quando $n \to \infty$. A condição $\mathbb{E}\,X \to \infty$ não garante que que $X > 0$ com alta probabilidade (veja o exercício 3.79 no final do capítulo), entretanto se $\mathbb{E}\,X^2 = (1 + o(1))(\mathbb{E}\,X)^2$, então pelo segundo momento temos $\mathbb{P}(\,X > 0\,) = 1 - o(1)$.

Vamos estimar a somatória

$$\mathbb{E}\,X^2 = \mathbb{E}\left[\left(\sum_{i=1}^{\binom{n}{3}} X_i\right)^2\right] = \mathbb{E}\left[\left(\sum_{i=1}^{\binom{n}{3}} X_i^2 + \sum_{i\neq j} X_i \cdot X_j\right)\right] = \sum_{i=1}^{\binom{n}{3}} \mathbb{E}[X_i^2] + \sum_{i\neq j} \mathbb{E}[X_i \cdot X_j] \tag{3.35}$$

em que o último somatório é sobre todo par de inteiros distintos $1 \leqslant i, j \leqslant m$. Para cada $i \neq j$, $\mathbb{E}[X_i \cdot X_j] = \mathbb{P}([X_i = 1] \cap [X_j = 1])$ é a probabilidade das i-ésima e j-ésima triplas de vértice formarem triângulos. Isso pode ocorrer de três modos de acordo com o número de vértices em comum entre as triplas

1. as duas triplas de vértices têm juntas 4 vértices e os triângulos uma aresta em comum, nesse caso $\mathbb{P}([X_i = 1] \cap [X_j = 1]) = p^5$;

2. as duas triplas têm juntas 5 vértices, um vértice em comum, nesse caso temos $\mathbb{P}([X_i = 1] \cap [X_j = 1]) = p^6$;

3. as duas triplas são disjuntas, nesse caso $\mathbb{P}([X_i = 1] \cap [X_j = 1]) = p^6$.

A contribuição do primeiro caso para $\sum_{i\neq j} \mathbb{E}[X_i \cdot X_j]$ é de no máximo $\binom{n}{4}$ subconjuntos de quatro vértices, em cada um há $\binom{4}{2}$ modos de escolher a aresta em comum, e as arestas ocorrem com probabilidade p^5; assintoticamente, $\binom{n}{4}\binom{4}{2}p^5 = O(n^4 p^5)$ que é muito menor que $(\mathbb{E}\,X)^2$, isto é, $o(1)(\mathbb{E}\,X)^2$. Nos outros dois casos $\mathbb{E}[X_i \cdot X_j] = \mathbb{E}\,X_i \cdot \mathbb{E}\,X_j = p^6$ pois, como não há aresta em comum, as variáveis são independentes (teorema 3.14, item 6); a contribuição para $\sum_{i\neq j} \mathbb{E}[X_i \cdot X_j]$ é no máximo $\sum_i \mathbb{E}\,X_i \sum_j \mathbb{E}\,X_j \leqslant (\mathbb{E}\,X)^2$. Para o primeiro somatório no lado direito da equação (3.35) usamos que $X_i^2 = X_i$, logo

$$\sum_{i=1}^{\binom{n}{3}} \mathbb{E}[X_i^2] + \sum_{i\neq j} \mathbb{E}[X_i \cdot X_j] \leqslant \mathbb{E}\,X + (1 + o(1))(\mathbb{E}\,X)^2$$

então do segundo momento obtemos

$$\mathbb{P}[X = 0] < 1 - \frac{(\mathbb{E}\,X)^2}{\mathbb{E}\,X^2} < 1 - \frac{(\mathbb{E}\,X)^2}{\mathbb{E}\,X + (1 + o(1))(\mathbb{E}\,X)^2} = o(1)$$

pois $\mathbb{E}\,X \to \infty$, assim $\mathcal{G}_{n,p}$ contém triângulo quase certamente.

Se $p = cn^{-1}$ ($c > 0$ constante) então $\mathbb{E}\,X \to c^3/6$, quando $n \to \infty$. Nesse caso é possível demonstrar que X converge para a distribuição de Poisson, em particular, $\lim_{n\to\infty} \mathbb{P}[X = 0] = e^{-c^3/6}$. ◊

De um modo geral, se $\mathbb{E}\,X \to 0$, então o primeiro momento garante que $\mathbb{P}[X > 0] = o(1)$ e dizemos que $X = 0$ com alta probabilidade, ou *quase certamente*. Por outro lado, $\mathbb{E}\,X \to \infty$ não significa que $X > 0$ com alta probabilidade (exercício 3.79 no final do capítulo), entretanto se $\mathbb{E}\,X^2 = (1 + o(1))(\mathbb{E}\,X)^2$, então pelo segundo momento temos $\mathbb{P}(X > 0) = 1 - o(1)$ e dizemos que $X > 0$ com alta probabilidade, ou *quase certamente*.

DISTRIBUIÇÃO DE BOLAS EM CAIXAS Suponha que em m caixas distribuímos $n = n(m)$ bolas de forma aleatória e seja X_i variável aleatória indicadora do evento "i-ésima caixa vazia", para $i = 1, 2, \ldots, m$. A quantidade de caixas vazias X tem esperança $\mathbb{E}\, X = \sum_{i=1}^{m} \mathbb{P}[X_i = 1] = m(1-1/m)^n$ logo (usando (d.1)) $\mathbb{E}\, X \approx me^{-n/m} = 1$ se $n = n^*(m) = m\log(m)$. Vamos usar as desigualdades de momento dadas acima para mostrar que se n é suficientemente menor que $n^*(m)$ então quase certamente ocorre caixa vazia e se n é suficientemente maior que $n^*(m)$ então quase certamente não ocorre caixa vazia.

Lema 3.51 *Suponha que, em m caixas, $n = n(m)$ bolas são distribuídas aleatoriamente, uniforme e independentemente. Dado qualquer $\varepsilon > 0$ constante, se $n > (1+\varepsilon)m\log(m)$ então não há caixa vazia com alta probabilidade e se $n < (1-\varepsilon)m\log(m)$ então há caixa vazia com alta probabilidade.*

Demonstração. Dado $\varepsilon > 0$, seja X a quantidade de caixas vazias e X_i variável aleatória indicadora de "i-ésima caixa vazia", para todo $i = 1, 2, \ldots, m$. O número de caixas vazias é $X = \sum X_i$ e

$$\mathbb{E}\, X = \sum_{i=1}^{m} \mathbb{P}[X_i = 1] = m\Big(1 - \frac{1}{m}\Big)^n = \Theta\Big(me^{-n/m}\Big)$$

e se $n > (1+\varepsilon)m\log(m)$ então $\mathbb{E}\, X = O(m^{-\varepsilon})$, portanto, pela equação (3.32) vale $\mathbb{P}[X > 0] \leqslant C\, m^{-\varepsilon}$ para alguma constante $C > 0$, ou seja, a probabilidade com que nenhuma caixa está vazia é alta

$$\mathbb{P}[X = 0] > 1 - Cm^{-\varepsilon}. \tag{3.36}$$

Por outro lado, se $n < (1-\varepsilon)m\log(m)$ então $\mathbb{E}\, X = \Omega(m^{\varepsilon})$ mas isso não assegura que há caixas vazias com alta probabilidade. Nesse caso usamos segundo memento

$$\mathbb{E}\, X^2 = \sum_{i=1}^{m} \mathbb{E}[X_i^2] + \sum_{i \neq j} \mathbb{E}[X_i \cdot X_j]$$

em que o último somatório é sobre todo par de inteiros distintos $1 \leqslant i, j \leqslant m$, como em (3.35). Agora, $X_i^2 = X_i$, por ser variável indicadora, logo $\mathbb{E}[X_i^2] = \mathbb{P}[X_i = 1] = (1-1/m)^n$. Também, $X_i \cdot X_j$ é uma variável aleatória indicadora do evento "ambas caixas estão vazias" e $\mathbb{E}[X_i \cdot X_j] = \mathbb{P}([X_i = 1] \cap [X_j = 1]) = (1-2/m)^n$, portanto

$$\begin{aligned}\frac{(\mathbb{E}\, X)^2}{\mathbb{E}\, X^2} &= \frac{m^2\Big(1-\frac{1}{m}\Big)^{2n}}{m\Big(1-\frac{1}{m}\Big)^{n} + m(m-1)\Big(1-\frac{2}{m}\Big)^{n}} \geqslant \frac{m\Big(1-\frac{1}{m}\Big)^{2n}}{\Big(1-\frac{1}{m}\Big)^{n} + m\Big(1-\frac{1}{m}\Big)^{2n}} \\ &= \frac{1}{\frac{1}{m}\Big(1-\frac{1}{m}\Big)^{-n} + 1} > \frac{1}{\frac{1}{m}\Big(1-\frac{1}{m}\Big)^{-m\log(m^{1-\varepsilon})} + 1} > \frac{1}{m^{-\varepsilon} + 1}\end{aligned}$$

que tende a 1 quando $m \to \infty$. Usando a desigualdade (3.31) concluímos que a probabilidade de ter não ter pelo menos uma caixa vazia é

$$\mathbb{P}[X = 0] < 1 - \frac{1}{(m^{-\varepsilon} + 1)}. \tag{3.37}$$

Em resumo, temos das equações (3.36) e (3.37) que a probabilidade de não haver caixas vazias é

$$\mathbb{P}[\,\mathsf{X}=0\,]=\begin{cases}1-o(1), & \text{se } n>(1+\varepsilon)m\log(m),\\ o(1), & \text{se } n<(1-\varepsilon)m\log(m),\end{cases}$$

onde $o(1)$ expressa uma função (não negativa) de m que tende a 0 quando m tende ao infinito. □

Do desenvolvimento acima temos que (1) se $\mathbb{E}\,\mathsf{X}=o(1)$ então $\mathbb{P}(\mathsf{X}=0)=1-o(1)$ e (2) se $\mathsf{X}_1,\ldots,\mathsf{X}_n$ são variáveis aleatórias de Bernoulli de mesmo parâmetro e $\mathsf{X}=\sum_i \mathsf{X}_i$ então

$$\mathbb{E}\,\mathsf{X}^2=\mathbb{E}\,\mathsf{X}+\sum_{i\neq j}\mathbb{E}[\,\mathsf{X}_i\cdot\mathsf{X}_j\,] \tag{3.38}$$

e se $\sum_{i\neq j}\mathbb{E}[\,\mathsf{X}_i\cdot\mathsf{X}_j\,]\leqslant(1+o(1))(\mathbb{E}\,\mathsf{X})^2$ enquanto $\mathbb{E}\,\mathsf{X}\to\infty$, então $\mathbb{P}[\,\mathsf{X}=0\,]=o(1)$. Ademais

$$\mathbb{P}[\,\mathsf{X}=0\,]=\begin{cases}1-o(1), & \text{se } \mathbb{E}\,\mathsf{X}=o(1),\\ o(1), & \text{se } \mathbb{E}\,\mathsf{X}\to\infty.\end{cases}$$

Vamos aplicar esse princípio no próximo exemplo.

CARGA MÁXIMA NO CASO $n=m$ Agora, suponhamos que n bolas são distribuídas aleatoriamente, uniforme e independentemente, em $m=n$ caixas. Seja X a quantia de caixas com pelo menos k bolas e X_i variável aleatória indicadora de ocorrência de k ou mais bolas na caixa i, para todo $i=1,2,\ldots,m$. Assim, $\mathsf{X}=\sum\mathsf{X}_i$.

Há $\binom{n}{k}$ modos de escolher um conjunto de k bolas, as quais estão numa caixa específica com probabilidade $(1/n)^k$. Para as bolas restantes ignoramos o destino, portanto pelo corolário 1.5, página 10, a probabilidade de uma caixa ter pelo menos k bolas é

$$\mathbb{E}\,\mathsf{X}_i=\mathbb{P}[\,\mathsf{X}_i=1\,]\leqslant\binom{n}{k}\left(\frac{1}{n}\right)^k\leqslant\left(\frac{\mathrm{e}n}{k}\right)^k\frac{1}{n^k}=\left(\frac{\mathrm{e}}{k}\right)^k.$$

Ainda,

$$\mathbb{P}[\,\mathsf{X}_i=1\,]\geqslant\binom{n}{k}\left(\frac{1}{n}\right)^k\left(1-\frac{1}{n}\right)^{n-k}\geqslant\frac{n^k}{k^k}\frac{1}{n^k}\frac{1}{\mathrm{e}}\geqslant\frac{1}{\mathrm{e}k^k}$$

de modo que

$$\frac{1}{\mathrm{e}k^k}\leqslant\mathbb{E}\,\mathsf{X}_i\leqslant\left(\frac{\mathrm{e}}{k}\right)^k.$$

Ingenuamente, a transição de $\mathsf{X}=0$ para $\mathsf{X}>0$ se dá quando $\mathbb{E}\,\mathsf{X}\approx n(\mathrm{e}/k)^k\approx 1$, ou seja, quando $k\log(k)\approx\log(n)$, portanto quando $k\approx\log(n)/\log(\log(n))$.

Fazendo $k=4\log(n)/(\log\log(n))$ temos $\mathbb{E}\,\mathsf{X}=o(1)$, para $n\to\infty$. De fato,

$$\left(\frac{\mathrm{e}}{k}\right)^k=\left(\frac{\mathrm{e}\log\log(n)}{4\log(n)}\right)^k\leqslant\left(\frac{\log\log(n)}{\log(n)}\right)^{\frac{4\log(n)}{\log\log(n)}}=\left(\mathrm{e}^{\log\log\log(n)-\log\log(n)}\right)^{\frac{4\log(n)}{\log\log(n)}}$$
$$=\mathrm{e}^{-4\log(n)\left(1-\frac{\log\log\log(n)}{\log\log(n)}\right)}=n^{-4+\frac{4\log\log\log(n)}{\log\log(n)}}\leqslant n^{-4(1+1/\mathrm{e})}<n^{-2}$$

pois $4\log\log\log(n)/\log\log(n)$ tem valor máximo $4/e$ em $n = e^{e^e}$. Portanto, por (3.32), $\mathbb{P}(X > 0) \leqslant \mathbb{E}\, X \leqslant n \cdot n^{-2}$, isto é,

$$\mathbb{P}[\,X = 0\,] = 1 - o(1)$$

de modo que com alta probabilidade a carga máxima numa caixa é $O(\log n/\log(\log n))$.

Agora, se $k = \log(n)/(3\log\log(n))$ então $\mathbb{E}\, X \to \infty$, para $n \to \infty$. De fato,

$$\left(\frac{1}{k}\right)^k = \left(\frac{3\log\log(n)}{\log(n)}\right)^{\frac{\log(n)}{3\log\log(n)}} \geqslant \left(\frac{\log\log(n)}{\log(n)}\right)^{\frac{\log(n)}{3\log\log(n)}} = \left(e^{\log\log\log(n) - \log\log(n)}\right)^{\frac{\log(n)}{3\log\log(n)}}$$
$$= e^{-\frac{\log(n)}{3}\left(1 - \frac{\log\log\log(n)}{\log\log(n)}\right)} = n^{-\frac{1}{3}\left(1 - \frac{\log\log\log(n)}{\log\log(n)}\right)}$$

de modo que $\mathbb{E}\, X \geqslant n \cdot n^{-1/3 + \log\log\log(n)/\log\log(n)} = n^{2/3 + o(1)}$.

Para calcular $\mathbb{E}\, X^2$ usamos a linearidade da esperança como na equação (3.38) e temos

$$\mathbb{E}\, X^2 = \mathbb{E}\, X + n(n-1)\mathbb{E}[\,X_i \cdot X_j\,]$$

para quaisquer $i \neq j$ e precisamos estimar a probabilidade

$$\mathbb{E}[\,X_i \cdot X_j\,] = \mathbb{P}\Big([\,X_i = 1\,] \cap [\,X_j = 1\,]\Big) = \sum_{k_1=k}^{n-k} \sum_{k_2=k}^{n-k_1} \binom{n}{k_1}\binom{n-k_1}{k_2}\left(\frac{1}{n}\right)^{k_1+k_2}\left(1 - \frac{2}{n}\right)^{n-k_1-k_2}.$$

Começamos com algumas simplificações

$$\sum_{k_1=k}^{n-k} \sum_{k_2=k}^{n-k_1} \binom{n}{k_1}\binom{n-k_1}{k_2}\left(\frac{1}{n}\right)^{k_1+k_2}\left(1 - \frac{2}{n}\right)^{n-k_1-k_2} \leqslant \sum_{k_1=k}^{n} \sum_{k_2=k}^{n} \binom{n}{k_1}\binom{n}{k_2}\left(\frac{1}{n}\right)^{k_1+k_2}\left(1 - \frac{1}{n}\right)^{2(n-k_1-k_2)}$$
$$= \sum_{k_1=k}^{n}\left(\binom{n}{k_1}\left(\frac{1}{n}\right)^{k_1}\left(1 - \frac{1}{n}\right)^{n-2k_1} \sum_{k_2=k}^{n}\binom{n}{k_2}\left(\frac{1}{n}\right)^{k_2}\left(1 - \frac{1}{n}\right)^{n-2k_2}\right)$$
$$= \sum_{k_1=k}^{n}\left(b_{n,\frac{1}{n}}(k_1)\left(1 - \frac{1}{n}\right)^{-k_1} \sum_{k_2=k}^{n} b_{n,\frac{1}{n}}(k_2)\left(1 - \frac{1}{n}\right)^{-k_2}\right)$$
$$\leqslant \left(1 - \frac{1}{n}\right)^{-2k} \sum_{k_1=k}^{n}\left(b_{n,\frac{1}{n}}(k_1) \sum_{k_2=k}^{n} b_{n,\frac{1}{n}}(k_2)\right).$$

Porém, $\mathbb{E}\, X_i = \sum_{x=k}^{n} b_{n,\frac{1}{n}}(x)$, logo

$$\mathbb{E}[\,X_i \cdot X_j\,] \leqslant \left(1 - \frac{1}{n}\right)^{-2k} \sum_{k_1=k}^{n}\left(b_{n,\frac{1}{n}}(k_1) \sum_{k_2=k}^{n} b_{n,\frac{1}{n}}(k_2)\right) = \left(1 - \frac{1}{n}\right)^{-2k} (\mathbb{E}\, X_i)(\mathbb{E}\, X_j)$$

e o valor esperado de X^2 é limitado por

$$\mathbb{E}\, X^2 \leqslant \mathbb{E}\, X + n^2\left(1 - \frac{1}{n}\right)^{-2k} (\mathbb{E}\, X_i)(\mathbb{E}\, X_j) = \mathbb{E}\, X + \left(1 - \frac{1}{n}\right)^{-2k} (\mathbb{E}\, X)^2.$$

Finalmente, usando a desigualdade de segundo momento, equação (3.31),

$$\mathbb{P}[\,X=0\,] < 1 - \frac{\mathbb{E}[\,X\,]^2}{\mathbb{E}\,X^2} \leqslant 1 - \frac{1}{\frac{1}{\mathbb{E}\,X} + \left(1-\frac{1}{n}\right)^{-2k}}$$

mas $1/\mathbb{E}\,X \to 0$ e $(1-1/n)^{-2k} \approx e^{2k/n} \to 1$, pela escolha de k, donde concluímos que $\mathbb{P}[\,X=0\,] = o(1)$ ou seja, a carga máxima é $\Omega(\log(n)/\log(\log(n)))$ com alta probabilidade. Resumindo, a probabilidade de não haver caixas com pelo menos k bolas é

$$\mathbb{P}[\,X=0\,] = \begin{cases} 1-o(1), & \text{se } k \geqslant 4\log(n)/\log\log(n), \\ o(1), & \text{se } k \leqslant \log(n)/3\log\log(n). \end{cases}$$

3.4 EXERCÍCIOS

Exercício 3.52. Considere os eventos A e B de um espaço de probabilidade. Verifique as identidades

1. $\mathbb{1}_{A\cup B} = \mathbb{1}_A + \mathbb{1}_B - \mathbb{1}_{A\cap B}$.
2. $\mathbb{1}_{\overline{A}} = 1 - \mathbb{1}_A$.
3. $A \subset B \Rightarrow \mathbb{1}_A \leqslant \mathbb{1}_B$.
4. $\mathbb{1}_{A\cap B} = \mathbb{1}_A \cdot \mathbb{1}_B$.

Exercício 3.53 (variáveis geométricas não têm memória). Sejam $X \sim \text{Geom}(p)$ uma variável aleatória, $t \geqslant 0$ e $n \geqslant 1$ dois números inteiros. Prove que $\mathbb{P}[\,X = n+t \mid X > t\,] = \mathbb{P}[\,X=n\,]$ e que $\mathbb{P}[\,X \geqslant n+t \mid X > t\,] = \mathbb{P}[\,X \geqslant n\,]$.

Exercício 3.54. Prove as seguintes propriedades para variáveis aleatórias discretas.

1. Se $\mathbb{P}[\,X \leqslant Y\,] = 1$ então $\mathbb{E}\,X \leqslant \mathbb{E}\,Y$.
2. Se $\mathbb{P}[\,a \leqslant X \leqslant b\,] = 1$ então $a \leqslant \mathbb{E}\,X \leqslant b$.
3. Se $X \geqslant 0$ e $\mathbb{E}\,X = 0$ então $\mathbb{P}[\,X=0\,] = 1$.

Exercício 3.55. Considere o espaço produto $(\Omega, \mathbb{P})$ obtido de $(\Omega_i, \mathbb{P}_i)$ para $1 \leqslant i \leqslant n$. Seja X_i a variável aleatória "projeção na i-ésima coordenada" definida em Ω dada por $X_i(\omega_1,\ldots,\omega_n) = \omega_i$. Prove que $X_1,\ldots,X_n$ são mutuamente independentes e que $\mathbb{P}_{X_i} = \mathbb{P}_i$.

Exercício 3.56. Prove que $X_1,\ldots,X_n$ são variáveis aleatórias independentes de Ω em S se, e somente se, a distribuição do vetor aleatório $(X_1,\ldots,X_n)$ sobre S^n é uma medida produto.

Exercício 3.57. Seja π uma permutação de $\{1, 2, \ldots, n\}$. Para todos $1 \leqslant i < j \leqslant n$ o par (i, j) determina um inversão de π se $\pi(i) > \pi(j)$. Determine o número esperado de inversões numa permutação escolhida ao acaso.

Exercício 3.58. Cada um dos $n \geqslant 1$ convidados de uma festa entrega seu chapéu na chapelaria da recepção. Quando a festa acaba o recepcionista devolve os chapéus aos convidados em uma ordem aleatória. Qual é o número esperado de convidados que recebem seu próprio chapéu de volta?

Exercício 3.59. Nesse exercício vamos deduzir a esperança de uma variável binomial sem recorrer a linearidade da esperança. Primeiro prove que vale a seguinte identidade entre coeficientes binomiais $i\binom{n}{i} = n\binom{n-1}{i-1}$. Agora, use essa identidade para provar que $\mathbb{E}[\,Y^k\,] = np\mathbb{E}[\,(X+1)^{k-1}\,]$ vale para $X \sim b(n-1, p)$ e $Y \sim b(n, p)$, com $n > 0$ e $p \in (0, 1)$. Conclua que $\mathbb{E}\, Y = np$.

Exercício 3.60. Sejam U e M conjuntos finitos. Uma família de funções $\mathcal{H} \subset M^U$ que quando munida da medida uniforme satisfaz

$$\mathop{\mathbb{P}}_{h \in_R \mathcal{H}}[h(u) = i] = \frac{1}{|M|}, \text{ para todo } u \in U \text{ e para todo } i \in M \tag{3.39}$$

não é suficiente para garantir um bom comportamento da família de funções de *hash* numa tabela de espalhamento. Verifique que a família $\mathcal{H}$ formada pelas $|M|$ funções constantes satisfaz (3.39). Nesse caso, toda $h \in \mathcal{H}$ resulta no pior caso?

Exercício 3.61. Projete um algoritmo aleatorizado que recebe uma lista de n números e devolve o k-ésimo maior elemento da lista. O número esperado de comparações deve ser $O(n)$. Determine a probabilidade com que o algoritmo faz mais que $O(n \log n)$ comparações (*dica:* use o particionamento do *quicksort*).

Exercício 3.62. Dado um conjunto de n números e um real positivo ε, tome uma amostra aleatória de $\Theta(\log n)$ elementos e ordene-os usando $O(\log n \log\log n)$ comparações. Prove que o resultado é uma ε-aproximação da mediana dos números dados com probabilidade de erro no máximo n^{-2}. Uma ε-aproximação da mediana é um elemento cujo posto (definido na página 52) está no intervalo $[(1-\varepsilon)n/2, (1+\varepsilon)n/2]$.

Exercício 3.63. Considere o espaço amostral

$$\Omega := \{(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1), (1,1,1), (2,2,2), (3,3,3)\}$$

e as variáveis aleatórias $X_i(\omega)$ que dá a i-ésima coordenada de ω, para $i = 1, 2, 3$ e todo $\omega \in \Omega$. Prove que

- para todos $i, j \in \{1, 2, 3\}$, vale $\mathbb{P}[\,X_i = j\,] = 1/3$;

- as variáveis X_i não são independentes.

Seja N uma variável aleatória sobre $\{1,2,3\}$ com a mesma distribuição de X_2. Prove que

$$\mathbb{E} \sum_{i=1}^{N} X_i \neq \sum_{i=1}^{\mathbb{E}\, N} \mathbb{E}\, X_i .$$

Exercício 3.64 (linearidade da esperança para soma enumerável). Prove que se $X_1, X_2, \ldots$ são variáveis aleatórias e $\sum_{i \geqslant 1} \mathbb{E}\,|X_i|$ converge, então $\mathbb{E} \sum_{i \geqslant 1} X = \sum_{i \geqslant 1} \mathbb{E}\, X_i$.

Exercício 3.65. Verifique que a hipótese de convergência é necessária no exercício anterior. Para tal, considere o espaço de probabilidade $(\mathbb{N}, \mathbb{P})$ com $\mathbb{P}(n) = 2^{-n}$ e as variáveis aleatórias

$$X_n(j) := \begin{cases} 2^n, & \text{se } j = n, \\ -2^{n+1}, & \text{se } j = n+1, \\ 0, & \text{caso contrário.} \end{cases}$$

1. Mostre que $\mathbb{E}\, X_n = 0$ para todo $n \geqslant 1$.
2. Tome $X = \sum_{n \geqslant 1} X_n$. Determine $X(1), X(2), X(3), \ldots$.
3. Usando o item anterior determine $\mathbb{E}\, X = \sum_{n \geqslant 1} X(n)\mathbb{P}(n)$.

Conclua que $\mathbb{E} \sum_n X_n \neq \sum_n \mathbb{E}\, X_n$.

Exercício 3.66. Na seção 3.1.2, página 131, foi dito que se um tabela de espalhamento usa um função aleatória então o número esperado de colisões para S fixo é $\binom{n}{2}(1/m) = n(n-1)/(2m)$ de modo que se $m = n^2$ então $\mathbb{E}\, C < 1/2$ e, de fato, não há colisão com probabilidade pelo menos $1/2$. Portanto, se S é um conjunto estático podemos sortear uma função até que encontremos uma que não ocasiona colisões para elementos de S. Escreva um algoritmo aleatorizado que dado S encontra uma função de *hash* h *perfeita*. Determine a complexidade desse algoritmo. É possível usar o método das esperanças condicionais nesse caso?

Exercício 3.67. Prove que para cada inteiro $n \geqslant 4$ existe uma coloração das arestas do grafo completo com n vértices com duas cores de modo que o número total de cópias monocromáticas de um grafo completo com 4 vértices é no máximo $\binom{n}{4} 2^{-5}$. Escreva um algoritmo aleatorizado de tempo polinomial em n que descobre uma coloração como descrita acima. Mostre como construir tal coloração deterministicamente em tempo polinomial usando o método de esperanças condicionais.

Exercício 3.68. Um **conjunto independente** em um grafo $G = (V, E)$ é um subconjunto de vértices $U \subset V$ tal que não há arestas de E formada só por vértices de U, isto é, U não contêm os dois vértices

de qualquer aresta do grafo. Por exemplo, no grafo da figura 2.4, na página 64, $\{2,6\}$ é um conjunto independente, assim como $\{2,5\}$, entretanto $\{1,2,5\}$ e $\{2,5,6\}$ não são conjuntos independentes.

Considere o seguinte algoritmo para computar um conjunto independente: dado $G = (V,E)$ com n vértices e $m := |E| > 0$;

1. inicie com U vazio;
2. para cada $v \in V$, acrescente v a U com probabilidade p;
3. para cada $e \in E$ com ambos extremos em U, remova um dos extremos de U.

Determine um limitante inferior para a cardinalidade esperada para U em função de p, n e m. Conclua que, para uma escolha ótima de p, todo grafo tem um conjunto independente de tamanho pelo menos $n^2/(4m)$ vértices. Escreva e analise um algoritmo Las Vegas para determinar um conjunto independente de cardinalidade maior ou igual ao valor acima.

Exercício 3.69. Considere o seguinte algoritmo para computar um conjunto independente: dado $G = (V,E)$;

1. inicie com U vazio;
2. tome um permutação π de V;
3. percorra os vértices de G na ordem definida pela permutação π e para cada $v \in V$, se v não tem vizinhos em U acrescente v a U.

Prove que U construído dessa maneira é independente em G. Escreva um algoritmo aleatorizado baseado na ideia acima e que determina um conjunto independente cuja cardinalidade esperada é

$$\sum_{u \in V} \frac{1}{d(u)+1}$$

em que $d(u)$ é o grau do vértice u em G.

Exercício 3.70. Prove as seguintes identidades para esperança condicional. Se X, Y e Z são variáveis aleatórias sobre um espaço de probabilidade discreto então

$$\mathbb{E}[\,X \mid Z\,] = \mathbb{E}[\,\mathbb{E}[\,X \mid Y,Z\,] \mid Z\,].$$

Ademais, se f e g são funções de variáveis reais

$$\mathbb{E}\,\mathbb{E}[\,f(X)g(X,Y) \mid X\,] = \mathbb{E}[\,f(X)\,\mathbb{E}\,g(X,Y) \mid X\,].$$

Exercício 3.71. Sejam X e Y variáveis aleatórias discretas e reais. Prove que para qualquer função $h\colon Y(\Omega) \to \mathbb{R}$

$$\mathbb{E}\,(X - \mathbb{E}[X \mid Y])^2 \leqslant \mathbb{E}\,(X - h(Y))^2$$

ou seja, $\mathbb{E}[X \mid Y]$ é a função de Y que melhor aproxima X no sentido de ter o menor erro quadrático médio. Agora, prove que vale a igualdade se, e só se, existe uma função $g\colon Y(\Omega) \to \mathbb{R}$ tal que $g(Y) = \mathbb{E}[X \mid Y]$ com probabilidade 1.

Exercício 3.72. Escreva um algoritmo Las Vegas que recebe uma fórmula booleana CNF $C_1 \wedge \cdots \wedge C_m$ em que cada cláusula tenha exatamente k literais e encontra uma valoração que satisfaz $m(1 - 2^{-k})$ cláusulas. Escreva uma versão desaleatorizada usando o método da esperança condicional. Analise o tempo de execução dos algoritmos.

Exercício 3.73. Um **grafo dirigido** é definido por um par de conjuntos finitos (V, E) em que V é o conjunto de vértices do grafo e E o conjunto de arestas, cada aresta é um par $(u, v) \in V \times V$. Dizemos que x **alcança** y **em** G se existe um caminho dirigido[6] dirigido de x para y no grafo G.

Dada uma fórmula booleana 2-CNF Φ, defina o grafo dirigido $G = G(\Phi)$ com $2n$ vértices dados por x e $\neg x$ para cada variável x de Φ e cada cláusula $C = \ell_1 \vee \ell_2$ contribui com exatamente duas arestas: $(\neg\ell_1, \ell_2)$ e $(\neg\ell_2, \ell_1)$.

Demonstre a seguinte afirmação: *A fórmula 2-CNF Φ não é satisfazível se, e somente se, existe uma variável x de Φ tal que x alcança $\neg x$ e $\neg x$ alcança x.*

Use a afirmação acima para projetar um algoritmo eficiente para 2SAT.

Exercício 3.74 (lei de grandes desvios para variáveis aleatórias binomiais). Sejam $X_i \sim \text{Bernoulli}(1/2)$ variáveis aleatórias independentes $1/2 < \alpha \leqslant 1$ um real. Seja $S_n = X_1 + \cdots + X_n$. Verifique que

$$\frac{1}{2^n}\frac{n!}{\lceil \alpha n\rceil!(n - \lceil \alpha n\rceil)!} \leqslant \mathbb{P}[S_n \geqslant \alpha n] \leqslant \frac{n+1}{2^n}\frac{n!}{\lceil \alpha n\rceil!(n - \lceil \alpha n\rceil)!}$$

e conclua, usando a aproximação de Stirling, que

$$\lim_{n\to\infty} \frac{1}{n}\mathbb{P}[S_n \geqslant \alpha n] = -I(\alpha)$$

onde $I(\alpha) = \alpha \log \alpha + (1 - \alpha)\log(1 - \alpha) + \log 2$.

Exercício 3.75. Sejam $\lambda > 0$ real e (p_n) uma sequência de reais em $[0, 1]$. Prove que

$$\lim_{n\to\infty} \binom{n}{k} {p_n}^k (1 - p_n)^{n-k} = \frac{e^{-\lambda}\lambda^k}{k!}.$$

[6] Um caminho dirigido é dado por uma sequência de vértices $x = v_0, v_1, v_2, \ldots, v_k = y$ em que vértices consecutivos formam aresta, $(v_i, v_{i+1}) \in E$.

Exercício 3.76. Suponha que temos uma fonte de bits aleatórias que responde 0 com probabilidade $p \in (0,1)$ e responde 1 com probabilidade $1-p$, independentemente. Escreva um algoritmo que usa essa fonte responda 0 ou 1 uniforme e independentemente. Calcule o tempo esperado de execução em função de p.

Exercício 3.77 (Raab e Steger, 1998). Assuma que n bolas são guardadas ao acaso em n caixas. Seja X a quantia de caixas com pelo k bolas e X_i variável aleatória indicadora de ocorrência de mais que k bolas na caixa i, para $i = 1, 2, \ldots, m$.

1. Prove que $\mathbb{E}\, X_i = (1 + o(1))\mathrm{b}_{n,\frac{1}{n}}(k)$. Uma sugestão é que $\mathrm{E}X_i = \mathbb{P}[\, X_i = 1\,] = \sum_{x=k}^{n} \mathrm{b}_{n,\frac{1}{n}}(x)$ e a soma acima pode ser estimada da seguinte maneira

$$\sum_{x=k}^{n} \mathrm{b}_{n,\frac{1}{n}}(x) = \mathrm{b}_{n,\frac{1}{n}}(k)\left(1 + \frac{\mathrm{b}_{n,\frac{1}{n}}(k+1)}{\mathrm{b}_{n,\frac{1}{n}}(k)} + \frac{\mathrm{b}_{n,\frac{1}{n}}(k+2)}{\mathrm{b}_{n,\frac{1}{n}}(k+1)}\frac{\mathrm{b}_{n,\frac{1}{n}}(k+1)}{\mathrm{b}_{n,\frac{1}{n}}(k)} + \cdots + \frac{\mathrm{b}_{n,\frac{1}{n}}(n)}{\mathrm{b}_{n,\frac{1}{n}}(n-1)}\frac{\mathrm{b}_{n,\frac{1}{n}}(n-1)}{\mathrm{b}_{n,\frac{1}{n}}(n-2)}\cdots\frac{\mathrm{b}_{n,\frac{1}{n}}(k+1)}{\mathrm{b}_{n,\frac{1}{n}}(k)}\right)$$

   e se $x \geqslant k$ então $\frac{\mathrm{b}_{n,\frac{1}{n}}(x+1)}{\mathrm{b}_{n,\frac{1}{n}}(x)} = \varepsilon(n,k) < 1$ (ε não depende de x).

2. Prove, usando as informações em (d.2) e (d.3) do Apêndice, que se $k = \alpha\log(n)/\log(\log(n))$ então $\mathbb{E}\, X = n^{1-\alpha+o(1)}$ e conclua que

$$\mathbb{E}\, X \to \begin{cases} \infty & \text{se } 0 < \alpha < 1 \\ 0 & \text{se } \alpha > 1 \end{cases}.$$

3. Prove se $k = \alpha\log(n)/\log(\log(n))$ então

$$\mathbb{P}[\, X > 0\,] = \begin{cases} 1 - o(1) & \text{se } 0 < \alpha < 1 \\ o(1) & \text{se } \alpha > 1 \end{cases}.$$

Exercício 3.78. Prove que se distribuímos ao acaso n bolas em m caixas e $n < (2/\mathrm{e})m\log(m)$ então com alta probabilidade o maior número de bolas em uma caixa é

$$\frac{4\log(n)}{\log\left(\frac{2n}{me}\log(n)\right)}.$$

Exercício 3.79. Para uma estrutura aleatória $\mathcal{S}_{n,p}$, a função de probabilidade $p^*(n)$ é uma **função limiar** para a propriedade $\mathcal{P}$ se

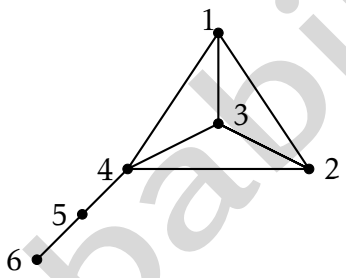
$$\mathbb{P}\Big[\, \mathcal{S}_{n,p} \text{ tem } \mathcal{P}\,\Big] = \begin{cases} o(1) & \text{se } p(n) = o(p^*(n)) \\ 1 - o(1) & \text{se } p^*(n) = o(p(n)) \end{cases}$$

ou seja, $\mathcal{S}_{n,p}$ quase certamente não tem $\mathcal{P}$ quando $p \ll p^*$ e quase certamente tem $\mathcal{P}$ quando $p \gg p^*$. Ou ainda, a função é limiar se ocorre o inverso, ou seja, $\mathcal{S}_{n,p}$ quase certamente tem $\mathcal{P}$ quando $p \ll p^*$ e quase certamente não tem $\mathcal{P}$ quando $p \gg p^*$

$$\mathbb{P}\Big[\,\mathcal{S}_{n,p} \text{ tem } \mathcal{P}\,\Big] = \begin{cases} 1 - o(1) & \text{se } p(n) = o(p^*(n)) \\ o(1) & \text{se } p^*(n) = o(p(n)). \end{cases}$$

A função p^* marca uma transição de fase entre não ter a propriedade e ter a propriedade.

No exemplo 3.50 a função $p^*(n) = n^{-1}$ é chamada de função limiar para propriedade "conter triângulo". Seguindo o exemplo 3.50, mostre que $n^{-2/3}$ é uma função limiar para "$\mathcal{G}_{n,p}$ contém subgrafo completo com 4 vértices". Em seguida, considere o grafo H dado pela figura abaixo. Prove que se $n^{-2/3} \gg p = p(n) \gg n^{-3/4}$ então o número esperado de cópias de H em $\mathcal{G}_{n,p}$ tende ao infinito, entretanto, o número esperado de subgrafos completos com 4 vértices em $\mathcal{G}_{n,p}$ tende a 0. Conclua que quase certamente $\mathcal{G}_{n,p}$ não contém H.



4 | COMPUTAÇÃO PROBABILÍSTICA

Notas de aula por J. Donadelli (cmcc–ufabc)

4.1 MODELOS DE COMPUTAÇÃO

Esta seção é dedicada a apresentação de modelos de computação de maneira *resumida* e *informal*. O leitor interessado em aprender mais sobre esses modelos que apresentaremos sugerimos o texto de Savage (1998).

A Máquina de Turing, proposta por Alan Turing, e o λ-cálculo, proposto por Alonzo Church, foram os primeiros modelos formais de computação que capturam o conceito do que intuitivamente os matemáticos chamam de algoritmo. Esses modelos pioneiros, o modelo genérico dado por uma linguagem de programação (pela linguagem C por exemplo) e outros como o modelo formal RAM, são equivalentes no sentido de que tudo que pode ser computado em um também pode ser no outro. Simulações entre modelos formais e um modelo genérico são possíveis e não há muita perda de desempenho sob condições razoáveis o que torna o estudo de classes de complexidade independente do modelo adotado.

Os modelos formais de computação ajudam na compreensão das dificuldades inerentes de se resolver problemas computacionais, têm a vantagem de serem mais simples e bem definidos que as linguagens de programação e a desvantagem de serem mais difíceis de descrever um algoritmo específico.

A especificação de um modelo de computação inclui uma descrição do que é um dispositivo (máquina ou programa) do modelo, como uma *entrada* é apresentada, o que é permitido para o dispositivo do modelo calcular para que a *saída* seja obtida a partir da entrada. Além disso, o modelo deve fornecer uma noção de *passo elementar* de computação da qual derivamos medidas de *tempo de execução* do dispositivo. O tempo de execução de um dispositivo do modelo com uma dada entrada é o número de passos elementares necessários para concluir uma computação. Em geral, o tempo de execução é dado como função do *tamanho* da entrada segundo algum critério como, por exemplo, o de *pior caso*.

Vimos que um **problema computacional** é caracterizado por uma tripla $(\mathcal{I}, \mathcal{O}, \mathcal{R})$ em que $\mathcal{I}$ denota o conjunto das instâncias (entradas) do problema, $\mathcal{O}$ o conjunto das respostas (saídas) e $\mathcal{R} \subset \mathcal{I} \times \mathcal{O}$ é uma relação que associa a cada instância uma ou mais respostas. Uma solução para um problema computacional computacional é um dispositivo de computação que computa uma função $\mathsf{A} \subset \mathcal{R}$ e nesse caso dizemos que A é uma **função computável**. Usaremos a notação funcional para denotar o resultado de um processo computacional de modo que o dispositivo C com instância $x \in \mathcal{I}$ responde $\mathsf{C}(x) \in \mathcal{O}$.

Os problemas computacionais podem ser classificados em problemas de *decisão* e problemas de *busca*. Num problema de busca cada instância x de um problema está associada a um conjunto $\mathsf{R}_x = \{y \in \mathcal{O} : (x, y) \in \mathcal{R}\}$ das soluções possíveis e que pode ser vazio. Uma *solução* do problema é uma função computável A tal que $\mathsf{A}(x) \in \mathsf{R}_x$ sempre que $\mathsf{R}_x \neq \varnothing$, caso contrário $\mathsf{A}(x) := \varnothing$ (ou algum

símbolo fora de $\mathcal{O}$ convencionado previamente).

Em um problema de decisão $\mathcal{O} = \{\text{sim}, \text{não}\}$ e, usualmente, especificamos um subconjunto $\mathsf{L} \subset \mathcal{I}$ tal que $\mathsf{A}(x) = \text{sim}$ se e somente se $x \in \mathsf{L}$; desse modo o problema é decidir se uma instância qualquer x pertence ao conjunto L. Por exemplo, o problema de decidir se um número é primo tem $\mathcal{I}$ como o conjunto dos inteiros positivos e tomamos L como o subconjunto dos inteiros positivos e primos. O teste em si é uma função computável A que responde sim se x pertence a L, caso contrário responde não, isto é, decide se x é ou não é um número primo.

A Complexidade Computacional é uma disciplina que, dentre outros objetivos, classifica problemas em classes de complexidade com relação a um modelo. Em geral, essas classes consideram problemas computacionais de decisão. Isso se justifica pela simplicidade e porque, embora pareça muito restritivo, para um grande classe de problemas de interesse os problemas de busca podem ser reduzidos a problemas de decisão (Goldreich, 2008, teoremas 2.6 e 2.10). O seguinte exemplo ilustra esse fenômeno.

Exemplo 4.1 (SAT). O problema denominado por SAT é o seguinte problema de decisão: dado uma fórmula booleana Φ, existe uma valoração das suas variáveis que a torna verdadeira? Nesse caso o conjunto $\mathcal{I}$ das instâncias é a família de todas as fórmulas booleanas sobre as variáveis $x_1, x_2, x_3, \ldots$. Como é um problema de decisão as respostas são $\mathcal{O} = \{\text{sim}, \text{não}\}$ e a relação $\mathcal{R}$ é dada pelos pares (Φ, sim) e $(\Phi', \text{não})$ para cada $\Phi \in \mathcal{I}$ satisfatível e cada $\Phi' \in \mathcal{I}$ não satisfatível ou, ainda, podemos colocar esse problema como o problema de decidir pertinência em

$$\text{SAT} := \{\Phi : (\Phi, \text{sim}) \in \mathcal{R}\}.$$

A versão de busca do problema SAT pede para determinar uma valoração das variáveis de Φ que torne a fórmula verdadeira ou determinar que tal valoração não existe. Entretanto, se $\Phi = \Phi(x_1, x_2, \ldots, x_n)$ e existe um algoritmo A para SAT então, caso Φ seja satisfatível, podemos escrever uma algoritmo A' para determinar uma valoração da seguinte maneira: A' usa A para decidir se $\Phi(1, x_2, \ldots, x_n)$ é satisfatível, se não for satisfatível então $\Phi(0, x_2, \ldots, x_n)$ é satisfatível; determinado o valor de x_1, o algoritmo A' repete o processo para determinar o valor de x_2, e assim por diante. Em n repetições desse processo descobrimos, valoração, caso exista, uma valoração ou descobrimos que não existe uma valoração que torne Φ verdadeira. Ademais, se A é um algoritmo de tempo polinomial então A' também será um algoritmo de tempo polinomial. ◊

Há, ainda, uma classe importante de problemas computacionais formada pelos problemas de *otimização*, como o MAX-SAT e o MAX-3SAT apresentados na seção 3.2.1. Cada par $(x, y) \in \mathcal{R}$ tem um custo $f(x, y) \in \mathbb{R}$ e buscamos pelas respostas que excedem um limiar ou atingem um valor máximo. No primeiro caso, dados x e c procuramos por $y \in \mathsf{R}_x$ tal que $f(x, y) \geqslant c$. No segundo caso, dado x, buscamos por y tal que $f(x, y)$ é máximo dentre todos $y \in \mathsf{R}_x$. Um problema computacional de

otimização pode ser reduzido a um problema computacional de busca (Goldreich, 2008, seção 2.2.2) que, por sua vez, pode ser reduzido a um problema computacional de decisão.

4.1.1 MÁQUINAS DE TURING

Uma **máquina de Turing** é uma sêxtupla $(Q, \Gamma, \Sigma, \delta, q_0, q_{\text{para}})$, em que Q é um conjunto finito de *estados* da máquina, Γ é o *alfabeto da fita* que contém um símbolo especial ƀ que representa *espaço em branco* e contém o conjunto Σ que é um *alfabeto* finito de modo que $\text{ƀ} \notin \Sigma$, $q_0 \in Q$ é o *estado inicial*, $q_{\text{para}} \notin Q$ é o *estado final*, e $\delta\colon Q \times \Gamma \to (Q \cup \{q_{\text{para}}\}) \times \Gamma \times \{\leftarrow, —, \rightarrow\}$ é a *função de transição*. No que segue Σ^* denota o conjunto de todas as *palavras* formadas por símbolos de Σ.

Uma **computação** de uma máquina de Turing M começa com uma entrada w, digamos que $w = w_1 w_2 w_3 \ldots w_n \in \Sigma^*$, escrita nas n primeiras posições de uma *fita* que é infinita para a direita; as outras posições contêm o símbolo ƀ, um símbolo por posição. A máquina M começa no estado q_0 lê w_1 na posição mais a esquerda da fita; essa é a **configuração inicial**. Se a máquina está num estado $q \in Q$ e lê $\gamma \in \Gamma$ na fita então o próximo passo é dado por $\delta(q, \gamma) = (q', \gamma', \ell)$ em que $q' \in Q \cup \{q_{\text{para}}\}$ é o novo estado, $\gamma' \in \Gamma$ é o símbolo que a máquina deixa na posição que estava o símbolo γ e o passo $\ell \in \{\leftarrow, —, \rightarrow\}$ diz qual posição da fita é a próxima a ser lida, respectivamente, a que está a esquerda da atual, a atual ou a posição a direita da atual. A computação continua com $\delta(q', \gamma')$ se q' não é o estado final. Se M é uma máquina de Turing e $w \in \Sigma^*$ uma entrada para M, então uma computação de M com entrada w pode

- *terminar*, o que significa que M atingiu o estado final q_{para}. Nesse caso, a resposta da computação é a sequência z de símbolos escrita na fita desde a posição mais a esquerda até a última posição antes do primeiro ƀ, e escrevemos $M(w) = z$ ou, senão, $M(w) = \text{ƀ}$;
- *não terminar*, o que significa que a computação nunca chega ao estado final. Máquinas de Turing que não terminam são úteis para classificar problemas computacionais mas não são dispositivos úteis de computação.

Exemplo 4.2. Considere o problema de decidir a paridade de uma sequência binária $x_1 x_2 x_3 x_4 \ldots x_n$ para qualquer $n \geqslant 1$, ou seja, determinar se a quantidade de 1's é ímpar, nesse caso responder 1, ou se é par, nesse caso responder 0. Uma máquina de Turing para esse problema tem estados $\{P, I, q_P, q_I\}$ além de q_0 e q_{para}. A função de transição é representada pelo diagrama de estados da figura 4.2, explicado com o exemplo da figura 4.1.

Uma computação dessa máquina com uma entrada particular fica completamente descrita por: a palavra escrita na fita, o estado da máquina e a posição de leitura/escrita a cada passo. A execução

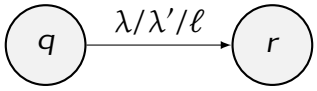


Figura 4.1: legenda para um diagrama de estados: se a máquina está no estado $q \in Q$ e lê $\lambda \in \Gamma$ na fita, então escreve $\lambda' \in \Gamma$ nessa posição e movimenta-se, de acordo com $\ell \in \{\leftarrow, —, \rightarrow\}$ ou uma posição para a esquerda, ou uma para a direita ou permanece na posição que está e, em seguida, entra no estado $r \in Q \cup \{q_{\text{para}}\}$.

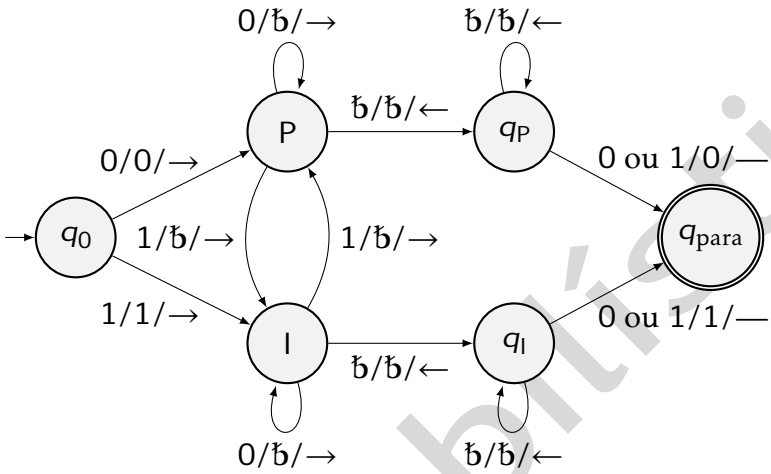


Figura 4.2: diagrama de estados da máquina de Turing para o problema da paridade.

dessa máquina de Turing com entrada 011 está mostrada na figura 4.3, que deve ser lida de cima para baixo, da esquerda para a direita. Na última configuração lemos 0 na fita, o que significa que a a quantidade de 1's nessa entrada particular é par. ◊

Uma função $f\colon \Sigma^* \to \Sigma^*$ é uma **função computável** se existe uma máquina de Turing M tal que para todo $w \in \Sigma^*$, quando M começa a computação na configuração inicial com entrada w, termina com somente $f(w)$ escrito na fita (todo outro símbolo na fita é ƀ), ou seja, $M(w) = f(w)$.

Seja M uma máquina de Turing que termina a computação com qualquer entrada e $T\colon \mathbb{N} \to \mathbb{N}$ uma função. Dizemos que M é uma **máquina de Turing com tempo de execução** T, ou simplesmente máquina de Turing de tempo T, se para todo $w = w_1 w_2 w_3 \dots w_n \in \Sigma^*$ a máquina computa $M(w)$ após no máximo $T(n)$ transições.

Um conjunto de palavras $L \subset \Sigma^*$ é uma **linguagem**. Por abuso de notação denotamos também por L a função característica[1] $L\colon \Sigma^* \to \{0,1\}$ do conjunto L. Se L é uma linguagem e M uma máquina de Turing, então dizemos que M **decide** L se para todo $w \in \Sigma^*$

$$M(w) = L(w)$$

[1] A *função característica de um conjunto* $A \subset U$ é a função $f\colon U \to \{0,1\}$ tal que $f(x) = 1$ se e somente se $x \in A$.

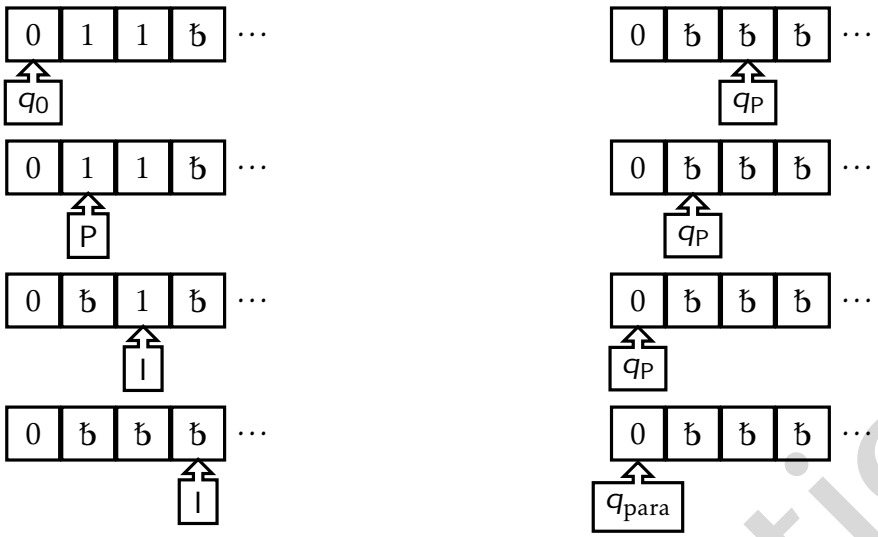


Figura 4.3: execução da máquina de Turing para a paridade com entrada 011.

ou seja, decide pertinência na linguagem L. Ainda, se M decide L então dizemos que M **aceita** as palavras $w \in L$ e **rejeita** as palavras $w \notin L$.

Vamos assumir que as instâncias e respostas dos problemas computacionais estão codificadas usando símbolos de um alfabeto Σ finito e com pelo menos dois símbolos. De fato, assumimos, sem perda de generalidade (veja o exercício 4.24, página 234) que

$$\Sigma = \{0, 1\}$$

de modo que Σ^* é o conjunto de todas as sequências binárias. O **tamanho** ou **comprimento** de uma palavra $w \in \Sigma^*$ é a quantidade de símbolos de Σ que constituem w, denotado por $|w|$.

Usamos o símbolo especial λ para a *palavra vazia* de modo que $\lambda \in \Sigma^*$ e $|\lambda| = 0$. Também, convencionamos que Σ^n denota o conjuntos das palavras de comprimento n. Assim, podemos escrever

$$\Sigma^* = \bigcup_{n \geqslant 0} \Sigma^n.$$

As computações são realizadas sobre sequências binárias, logo os elementos do conjunto das instâncias $\mathcal{I}$ de um problema computacional devem ser representados usando somente 0's e 1's por uma **codificação**

$$c \colon \mathcal{I} \cup \mathcal{O} \to \{0, 1\}^*$$

e no caso $\mathcal{O} = \{\text{sim}, \text{não}\}$ convencionamos

$$c(\text{sim}) = 1 \text{ e } c(\text{não}) = 0.$$

Por exemplo, números, grafos, fórmulas booleanas devem ser codificados em binário. Não nos preocuparemos com detalhes dessas codificações e *assumiremos que sempre é possível fixarmos uma codificação canônica*. No caso de números consideramos a sua representação em base 2 e um grafo é dado pela matriz de adjacências.

Sempre que for oportuno deixar claro que estamos usando uma representação binária (canônica) usamos a notação $\langle x \rangle$ para $c(x)$. Por exemplo, uma representação binária da fórmula booleana Φ é denotada por $\langle \Phi \rangle$ e a linguagem $\textsc{sat} \subset \Sigma^*$ é o conjunto $\textsc{sat} = \{\langle \Phi \rangle : \Phi \text{ é fórmula booleana satisfatível}\}$.

Desse modo, quando falamos em computar uma função $f : \mathcal{I} \to \mathcal{O}$, de fato nos referimos a uma função $f_c : c(\mathcal{I}) \to \{0,1\}^*$ de maneira que para uma representação binária de $x \in \mathcal{I}$, a função f_c determina uma representação binária de $f(x)$.

UNIVERSALIDADE E INDECIDIBILIDADE Um fato de grande importância a respeito de máquina de Turing, chamado de universalidade, está descrito a seguir em (4.1). Primeiro o leitor deve se convencer de que uma máquina de Turing M tem uma descrição finita e em seguida perceber que essa descrição pode ser codificada por $\langle M \rangle$, ou seja, *é possível estabelecermos uma codificação binária canônica para as máquinas de Turing* (Hopcroft, Motwani e Ullman, 2000, seção 9.1.2); denotamos por M_α a máquina cuja representação binária é $\alpha \in \{0,1\}^*$. O fato importante é que

$$\textit{existe uma máquina de Turing } \mathfrak{U} \textit{ dita } \text{universal } \textit{tal que } \mathfrak{U}(w, \alpha) = M_\alpha(w) \tag{4.1}$$

e essa simulação é bastante eficiente, se M_α é uma máquina de Turing de tempo T então $\mathfrak{U}(\cdot, \alpha)$ é uma máquina de tempo $O(T \log(T))$ (Arora e Barak, 2009, seção 1.4.1, teorema 1.9).

O conjunto de todas as máquinas de Turing é enumerável. A cardinalidade do conjunto formado pelas funções $f : \mathbb{N} \to \{0,1\}$ é a cardinalidade do conjunto $2^{\mathbb{N}}$ das partes de $\mathbb{N}$ e pelo Teorema de Cantor não há função sobrejetiva de $\mathbb{N}$ em $2^{\mathbb{N}}$, portanto existem funções de $\mathbb{N}$ em $\{0,1\}$ que não são computáveis por máquina de Turing ou, pondo de outro modo

existem linguagens que não são decidíveis

e dizemos que uma linguagem dessas é **indecidível**.

Um problema indecidível bastante conhecido é o famoso *Problema da Parada*: decidir se a computação de uma máquina de Turing M com uma entrada x termina. Consideremos uma enumeração $(M_i : i \in \mathbb{N})$ das máquinas de Turing e suponhamos que exista uma máquina de Turing H que com entrada $(\langle i \rangle, x)$ decide se a computação $M_i(x)$ termina. Tomemos P a máquina de Turing que com entrada x computa da seguinte maneira:

```
1 se H(x, x) = 0 então responda 1;
2 senão se 𝔘(x, x) = 1 então responda 0;
3 senão responda
4 1.
```

Algoritmo 38: $P(x)$

A resposta é $P(x) = 0$ se, e só se, $\mathfrak{U}(x, x) = 1$. Agora, se tomarmos como entrada para P a palavra

$x := \langle n \rangle$ de tal sorte que $\mathsf{M}_n = \mathsf{P}$ então $\mathfrak{U}(x, x) = \mathsf{M}_n(\langle n \rangle) = \mathsf{P}(x)$ e

$$\mathsf{P}(x) = 0 \text{ se, e só se, } \mathsf{P}(x) = 1$$

uma contradição. Logo, não deve existir a máquina H.

MODELOS PROBABILÍSTICOS Uma **máquina de Turing probabilística** M é definida como uma máquina que sempre termina e que cada transição tem dois movimentos legais δ_0 e δ_1, entretanto, a cada transição durante uma computação só um deles é escolhido como o próximo passo; dado um estado q e um símbolo da fita σ a máquina executa a transição $\delta_i(q, \sigma)$ para uma escolha aleatória $i \in \{0, 1\}$. A resposta da máquina para uma entrada w é uma variável aleatória que depende de w; nesse caso, consideramos a distribuição da variável aleatória $\mathsf{M}(w)$ da máquina probabilística M com entrada fixa $w \in \Sigma^*$ em que a probabilidade é tomada sobre as escolhas aleatórias feitas por M. O espaço de probabilidades é formado por todas as sequências binárias que representam as escolhas internas da máquina numa computação desde a configuração inicial no estado q_0 até alcançar q_{para}. Para uma tal sequência de comprimento m associamos a probabilidade 2^{-m}. Na computação de M com entrada w a probabilidade

$$\mathbb{P}[\, \mathsf{M}(w) = z \,]$$

é a soma das probabilidades de todas as sequências de escolhas aleatórias internas que terminam com a palavra $z \in \Sigma^*$ escrita na fita.

Notemos que para uma uma entrada w fixa o número de bits aleatórios usados numa computação com w pode variar, entretanto, para toda sequência binária r que representa as escolhas aleatórias numa computação com w, não existe uma sequência r' que também representa as escolhas aleatórias numa computação e tal que r seja prefixo de r' ou vice-versa.

Uma máquina probabilística M **decide** L **com probabilidade de erro** ϵ **em tempo** T, para $\mathsf{T} \colon \mathbb{N} \to \mathbb{N}$, se para toda palavra $w \in \Sigma^*$

$$\mathbb{P}[\, \mathsf{M}(w) \neq \mathsf{L}(w) \,] \leqslant \varepsilon$$

e $\mathsf{M}(w)$ termina a computação em no máximo $\mathsf{T}(|w|)$ transições para *qualquer que seja a sequência de bits aleatórios* usados na computação.

Uma alternativa à definição acima é dada a seguir pelo modelo chamado em algumas referências bibliográficas de máquina de Turing probabilística *off-line*: é uma máquina de Turing determinística que tem uma entrada auxiliar escrita numa fita auxiliar só de leitura, além da entrada principal escrita na fita principal. Essa fita auxiliar contém uma sequência B de bits aleatórios, cada posição tem um bit com distribuição uniforme e as ocorrências na sequência são independentes. Cada posição da fita auxiliar é usada (lida) no máximo uma única vez. Notemos que o comprimento da entrada auxiliar pode ser definido como igual ao tempo T de execução da máquina. Nesse caso, consideramos

a distribuição da variável aleatória $M(w, B)$ da máquina probabilística M com entrada fixa $w \in \Sigma^*$ e um vetor aleatório $B \in_R \{0,1\}^{T(|w|)}$. Uma máquina probabilística *off-line* M com tempo de execução T **decide** L **com probabilidade de erro** ϵ se para toda palavra $w \in \Sigma^*$

$$\mathop{\mathbb{P}}_{B \in_R \{0,1\}^{T(|w|)}}[M(w, B) \neq L(w)] \leqslant \varepsilon.$$

Os dois modelos são equivalentes: uma computação da máquina probabilística *off-line* N como definida acima pode ser simulada por uma máquina probabilística M que com entrada w seleciona internamente uma sequência de bits aleatórios b e computa $N(w, b)$. Por outro lado, a computação de uma máquina probabilística M é simulada pela máquina determinística N que realiza transições de $M(w)$ usando uma sequência de bits aleatórios b dados com entrada auxiliar como as escolhas aleatórias internas a M. De fato, mais do que o que foi dito vale, as definições são equivalentes no contexto do tempo de execução, assim, na sequência, vamos usar livremente o modelo probabilístico que for mais conveniente (com uma leve preferência ao *off-line*).

4.1.2 MODELO RAM E ALGORITMOS EFICIENTES

Um modelo mais próximo, na sua concepção, do computador (eletrônico) que conhecemos é o modelo RAM que tem um número infinito e enumerável de registradores (memória), $M_0, M_1, \ldots$, um conjunto finito de instruções para transferência de valores entre registradores, endereçamento (direto e indireto), aritmética (soma e subtração), desvio condicional e um contador de programa que indica qual instrução de um programa vai ser executada. Um *programa* RAM é uma sequência finita de instruções $N_0, N_1, \ldots, N_m$ dentre as instruções válidas do modelo.

Um conjunto típico de instruções é

`INC Mi`	Incrementa o conteúdo do registrador i de 1
`DEC Mi`	Decrementa o conteúdo do registrador i de 1
`CLR Mi`	Substitui o conteúdo de M_i por 0
`Mi` $\leftarrow$ `Mj`	Substitui o conteúdo de M_i pelo de M_j
`Mi` $\leftarrow$ `MMj`	Substitui o conteúdo de M_i pelo de M_{M_j}
`MMi` $\leftarrow$ `Mj`	Substitui o conteúdo de M_{M_i} pelo de M_j
`JMP Ni`	Atribui N_i ao contador de programa
`Mj JIfZ Ni`	Se $M_j = 0$, atribui N_i ao contador de programa
`CONTINUE`	Continue na próxima instrução, caso exista, ou pare

Cada posição de memória pode armazenar um inteiro e as instruções operam sobre inteiros. As instruções são executadas sequencialmente, a cada instruções executada o contador de programa é incrementado, exceto nas instruções de desvio. A computação começa com a entrada nas primeiras

posições de memória, digamos $M_0, M_1, \ldots, M_n$, as outras posições são nulas; o contador de programa indica N_0; a resposta é dada em M_0. Em qualquer instante, apenas uma quantidade finita dos números armazenados na memória são diferente de 0. Para obtermos um modelo RAM probabilístico acrescentamos a instrução `RAND(k)` que retorna um inteiro aleatório de k bits.

Uma diferença importante com relação às máquinas de Turing é que numa máquina RAM temos acesso direto as posições de memória[2], numa só instrução da máquina, como em um computador. Por outro lado, uma diferença importante com relação aos computadores é que assumimos que uma máquina RAM tem memória ilimitada e isso faz com que em cada posição de memória deva ser permitido armazenar um inteiro arbitrariamente grande pois precisamos guardar na memória os endereços de memória para ser possível o endereçamento direto. Exceto pelas computações que abusam do fato de ser permitido armazenar e operar um inteiro arbitrariamente grande, um computador e uma máquina RAM têm os mesmos programas.

O **tempo de execução** de um programa RAM que sempre termina é definida pelo número de instruções da RAM que o programa executa em função do tamanho da entrada. Usualmente consideramos dois modelos RAM quando levamos em conta o tempo de execução: o modelo de **custo logarítmico** que leva em conta o custo das operações em função do tamanho dos operandos, e o modelo de **custo unitário** com instruções em tempo constante, que é o mais comum quando analisamos algoritmos. A soma de dois números, por exemplo, tem custo proporcional à quantidade de dígitos no primeiro caso e no segundo caso tem custo constante. Se um programa RAM usa números grandes o suficiente para que se torne irreal assumir que a adição e outras operações podem ser executadas com custo unitário, o modelo de custo logarítmico de RAM é mais adequado. A escolha de qual o modelo é mais adequado depende da aplicação. Por exemplo, para algoritmos que lidam com números, como o que encontramos na seção 2.2.2 em que o produto vC de um vetor de dimensão n por uma matriz quadrada de ordem n foi atribuído custo (unitário) $O(n^2)$, a análise com custo unitário só é representativa quando os operandos têm tamanho significativamente menor que a instância do problema, caso as entradas da matriz e do vetor tivessem n dígitos, por exemplo, esse custo estaria muito subestimado.

Uma máquina de Turing que executa T transições pode ser simulada por um programa RAM que executa $O(T)$ instruções (Savage, 1998, teorema 3.8.1). Um programa RAM com inteiros de tamanho limitado que executa T instruções pode ser simulado por uma máquina de Turing que executa $O(T^3)$ transições (Savage, 1998, teorema 8.4.1) (veja também Papadimitriou, 1994, teoremas 2.4 e 2.5).

Um programa RAM é muito semelhante a um programa em linguagem de montagem (*assembly*), que é especifica a um processador e um sistema operacional. Há compiladores que transformam programas em linguagens como C em programas em linguagem de montagem para, em seguida, criar um código que seja executável num computador eletrônico.

[2]Esse é o motivo do nome *random access*, não tem relação com "aleatório".

ALGORITMOS EFICIENTES Tradicionalmente *computação eficiente* é sinônimo de computação feita por um dispositivo com *tempo de execução polinomial* no tamanho da entrada. Dizemos que uma máquina de Turing, ou programa RAM, tem **tempo polinomial** se existe constante inteira e positiva k tal que nas entradas de tamanho n o tempo de execução do dispositivo é $O(n^k)$. Comentamos acima que máquina de Turing e o modelo RAM são **polinomialmente equivalentes**, ou seja, tudo o que pode ser computado em tempo polinomial num modelo também pode ser computado em tempo polinomial no outro. Portanto, a classe dos problemas computacionais que podem ser resolvidos em tempo polinomial para o modelo máquina de Turing coincide com a classe dos problemas computacionais que podem ser resolvidos em tempo polinomial para o modelo RAM. De fato, essa classe é invariante para todos os modelos clássicos que são razoáveis.

Para simplificar nós dizemos, de modo genérico, **algoritmo de tempo polinomial** para nos referirmos a um dispositivo de um modelo formal tradicional que tenha tempo de execução polinomial. Ademais, temos a facilidade de descrever o algoritmo usando um pseudocódigo. Neste texto assumimos implicitamente o custo unitário quando analisamos o tempo de execução de um algoritmo e a distribuição de probabilidade da resposta, a não ser que seja dito explicitamente outra coisa. Nesse sentido, para nos mantermos independentes de um modelo formal tomamos o cuidado de o tamanho dos operandos usados nas instruções serem limitados polinomialmente no tamanho da entrada e dos sorteios serem feitos num conjunto de tamanho polinomial no tamanho da entrada de modo a garantir a equivalência polinomial (assintótica) entre os modelos de custo unitário e logarítmico.

4.1.3 CIRCUITO BOOLEANO

Um **circuito booleano** é representado por um grafo orientado acíclico em que cada vértice está associado a: (i) ou a uma porta lógica dentre *e*, *ou*, *não*; (ii) ou a uma variável x_i para $i \in \{1,2,\ldots,n\}$ para algum $n \in \mathbb{N}$; (iii) ou uma saída s_i, $i \in \{1,2,\ldots,m\}$ para algum $m \in \mathbb{N}$. O grau de entrada dos vértices pode assumir um de três valores: 0 (variáveis), 1 (porta *não* e saídas) e 2 (portas *e* e *ou*). Um circuito computa uma **função binária** (o caso $m = 1$ é chamado de **função booleana**)

$$\begin{aligned} C\colon \{0,1\}^n &\to \{0,1\}^m \\ (x_1, x_2, \ldots, x_n) &\mapsto (s_1, s_2, \ldots, s_m), \end{aligned}$$

que também denotaremos por C, do seguinte modo, uma entrada $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ valora as variáveis do circuito; sempre que as entradas de uma porta lógica estão valoradas, a porta opera sobre os valores e devolve o resultado, esse resultado é propagado no circuito até que encontra uma saída.

Exemplo 4.3. O circuito na figura 4.4 abaixo computa a paridade da sequência binária $x_1x_2x_3x_4$, ou seja, $s_1 = 1$ se e somente se o número de ocorrências de 1's na sequência é ímpar. ◊

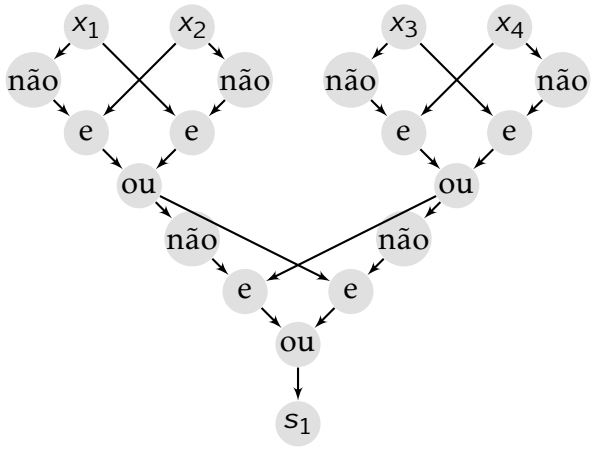


Figura 4.4: Circuito booleano que computa $s_1(x_1, x_2, x_3, x_4) = x_1 + x_2 + x_3 + x_4 \bmod 2$.

Se L é uma linguagem, então uma família de circuitos $\mathcal{C} = (\mathsf{C}_n)_{n\in\mathbb{N}}$ **decide** L se para cada $x \in \{0,1\}^*$ o circuito $\mathsf{C}_{|x|}$ com entrada x responde 1 se e somente se $x \in \mathsf{L}$, em outras palavras, se denotamos também por L a função característica $\mathsf{L}\colon \{0,1\}^* \to \{0,1\}$ do conjunto L, temos que

$$\mathsf{C}_n(x) = \mathsf{L}(x)$$

para todo $x \in \{0,1\}^n$, para todo n.

O **tamanho do circuito** C, denotado por $|\mathsf{C}|$, é o número de vértices do grafo subjacente a definição de circuito e, em geral, estamos interessado no tamanho dos circuitos de uma família $\mathcal{C}$ em função do tamanho da entrada. Outro parâmetro usado para medir a complexidade de um circuito é a sua **profundidade** que é o número de arestas no caminho mais longo através do circuito.

Seja $\mathcal{C} = (\mathsf{C}_n)_{n\in\mathbb{N}}$ uma família de circuitos e $s\colon \mathbb{N} \to \mathbb{N}$ uma função. Dizemos que $\mathcal{C}$ é uma família de **tamanho** $s(n)$ se $|\mathsf{C}_n| \leqslant s(n)$ para todo n. A **complexidade de uma linguagem** L **com respeito a circuitos** é a função $s_{\mathsf{L}} : \mathbb{N} \to \mathbb{N}$ se $s_{\mathsf{L}}(n)$ é o menor tamanho de uma família de circuitos que decide L.

Um fato conhecido sobre circuitos e que será usado várias vezes mais adiante é o seguinte resultado cuja prova é um exercício (Sipser, 1996, teorema 9.30).

Exercício 4.4 (teorema 9.30 em Sipser (1996)). Prove que se L é uma linguagem que pode ser decidida por uma máquina de Turing de tempo $\mathsf{T}(n)$ então L pode ser decidida por uma família de circuitos $(\mathsf{C}_n)_{n\in\mathbb{N}}$ de tamanho $\mathrm{O}(\mathsf{T}(n)^2)$ (dica: suponha tempo $\mathsf{T}(n)$, exatamente, considere apenas as $\mathsf{T}(n)$ primeiras posições da fita e as $\mathsf{T}(n)$ configurações da fita dadas por uma computação, uma para cada passo (veja a figura 4.3, página 182). Cada posição de uma fita, depende somente de três posições da fita da configuração imediatamente anterior).

A recíproca dessa proposição não vale pois há linguagens indecidíveis por máquina de Turing que admitem circuito de tamanho polinomial (exercício 4.25, página 234). Circuito booleano e máquina

de Turing não são computacionalmente equivalentes.

Ao contrário da máquina de Turing, com circuito booleano temos um dispositivo de computação para cada tamanho de entrada, nesse caso dizemos que circuito booleano é um **modelo não-uniforme** de computação enquanto que máquina de Turing é um **modelo uniforme** de computação. A não uniformidade faz com que seja possível *desaleatorizar* eficientemente circuitos probabilísticos, ou seja é possível evitar o uso de bits aleatórios sem aumentar substancialmente o tamanho dos circuitos (exercício 4.27, página 234).

CIRCUITO ARITMÉTICO É como um circuito booleano no qual as entradas são variáveis ou constantes, as portas *ou* são substituídas pela *soma* e as portas *e* são substituídas pela *multiplicação*. De modo mais geral, em um **circuito aritmético sobre um corpo** $\mathbb{F}$ as constantes e as operações aritméticas são as do corpo. Por exemplo, o circuito na figura 4.5 abaixo computa $x_1 x_2 + 1$. Circuito aritmético é o

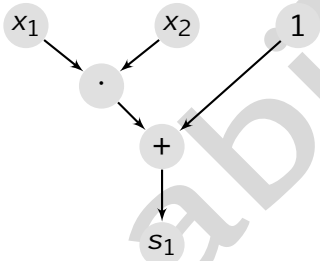


Figura 4.5: Circuito aritmético que computa $s_1(x_1, x_2) = x_1 \cdot x_2 + 1$.

modelo padrão para computação sobre polinômios, a saída de um circuito aritmético é um polinômio (ou um conjunto de polinômios) nas de variáveis entrada. As medidas de complexidade associadas com tais circuitos são *tamanho*, que é a quantidade de vértices no grafo subjacente, e *profundidade*, que é a maior distância entre uma entrada e uma saída.

Observamos que toda função booleana $f\colon \{0,1\}^n \to \{0,1\}$ pode ser expressa por uma fórmula booleana que, por sua vez, equivale a um polinômio sobre $\mathbb{F}_2$.

4.2 CLASSES DE COMPLEXIDADE DE TEMPO POLINOMIAL

Uma classe de complexidade computacional com respeito a um modelo de computação é o conjunto dos problemas de computação que são resolvidos por um dispositivo computacional naquele modelo com alguma restrição de recurso como, por exemplo, o tempo de execução. Como nos restringimos aos problemas de decisão, as classes de complexidade são apresentadas como classes de linguagens. As definições que apresentaremos abaixo são elaboradas tendo em mente máquinas de Turing como modelo formal de algoritmo, mas com alguns cuidados que já descrevemos isso significa

que *algoritmo* pode ser entendido no sentido corriqueiro, escrito em pseudocódigo.

A classe P — *Polynomial time* — é a classe das linguagens decididas por algoritmos de tempo polinomial. Por exemplo, o problema de decidir se um inteiro positivo é primo está em P (Agrawal, Kayal e Saxena, 2004). Não sabemos se SAT pertence a P pois não conhecemos nenhum algoritmo de tempo polinomial para SAT.

A classe descrita a seguir é dada por uma definição alternativa, porém equivalente, à definição clássica que normalmente aparece na literatura (e.g. Papadimitriou, 1994; Sipser, 1996), a definição clássica é dada no o exercício 4.5 a seguir.

A classe NP — *Nondeterministic Polynomial time* — é a classe das linguagens L para as quais existe um polinômio p e um algoritmo probabilístico M de tempo polinomial tal que todo $w \in \{0,1\}^*$

1. se $w \in \mathsf{L}$ então $\mathbb{P}_{\mathsf{B} \in_R \{0,1\}^{p(|w|)}}[\mathsf{M}(w,\mathsf{B}) = 1] > 0$, ou seja, se $w \in \mathsf{L}$ então existe uma *entrada auxiliar* $\mathsf{B} \in \{0,1\}^{p(|w|)}$ que faz M responder *sim*;

2. se $w \notin \mathsf{L}$ então $\mathbb{P}_{\mathsf{B} \in_R \{0,1\}^{p(|w|)}}[\mathsf{M}(w,\mathsf{B}) = 0] = 1$, nesse caso, se $w \notin \mathsf{L}$, M responde *não* qualquer que seja a *entrada auxiliar*.

Por exemplo, $\text{SAT} \in \mathsf{NP}$ pois podemos escrever um algoritmo M que com entrada (Φ, c) verifica em tempo polinomial se c é uma valoração das variáveis da fórmula booleana Φ que torna a fórmula verdadeira ou não. Como $\Phi \in \text{SAT}$ se, e somente se, existe uma valoração c das variáveis de Φ que torna a fórmula verdadeira, temos que uma escolha aleatória para c satisfaz a fórmula $\Phi \in \text{SAT}$ com probabilidade maior que zero. Ainda, qualquer escolha aleatória para c não satisfaz uma fórmula $\Phi \notin \text{SAT}$, logo o algoritmo responde 1 com probabilidade 0. Ademais, o tamanho de $\langle c \rangle$ é polinomial no tamanho de $\langle \Phi \rangle$.

Uma sequência c como no exemplo acima é chamada de **certificado**. Se $w \in \mathsf{L}$ e $\mathsf{L} \in \mathsf{NP}$ então há um certificado curto, de tamanho polinomial em $|w|$, que certifica tal fato, senão, caso $w \notin \mathsf{L}$, não há um certificado curto de que $w \in \mathsf{L}$. Por exemplo, um inteiro $d > 1$ que divide outro inteiro positivo n é um certificado curto de que n é composto. O problema de decidir se um inteiro positivo é composto está em NP pois $|\langle d \rangle| \leqslant |\langle n \rangle|$ e há um algoritmo de tempo polinomial em $|\langle n \rangle|$ que verifica se d divide n.

Não é difícil mostrar que $\mathsf{P} \subset \mathsf{NP}$: se $\mathsf{L} \in \mathsf{P}$ e D é um algoritmo polinomial para L então um algoritmo probabilístico polinomial M para L simula D e ignora os bits aleatórios de modo que $\mathsf{M}(w,\mathsf{B}) = \mathsf{D}(w)$ para todo w e todo B. Por outro lado, não é sabido se a inclusão $\mathsf{NP} \subset \mathsf{P}$ é verdadeira. Esse é um dos principais, senão o principal, problema não resolvido da Teoria da Computação e foi formulado independentemente por Stephen Cook e Leonid Levin em 1971

Problema 5. $\mathsf{P} \neq \mathsf{NP}$?

Muito do que está exposto neste capítulo nasceu das tentativas de entender melhor esse problema.

Exercício 4.5 (definição clássica de NP*).* A definição original da classe NP envolve máquinas de Turing não-determinísticas: uma sêxtupla como a que define a máquina de Turing exceto pelo função de transição que tem múltiplos valores $\delta(q,\gamma) = \{(q_1,\gamma_1,\ell_1),\ldots,(q_k,\gamma_k,\ell_k)\}$. Uma palavra é *aceita* se algum "ramo" da computação termina com aceite. Cabe ressaltar que o sentido de "não-determinístico" que aparece na definição não é o mesmo que de "probabilístico", a definição de máquina é semelhante mas a definição de computação é diferente. Essa máquina não é um modelo realista de computação (veja mais em Sipser, 1996), uma alternativa mais atual para essa definição é a seguinte. Um **verificador** de tempo polinomial para uma linguagem L é um algoritmo de tempo polinomial M tal que para todo $w \in \{0,1\}^*$

- se $w \in \mathsf{L}$ então existe $c \in \{0,1\}^{p(|w|)}$ tal que $\mathsf{M}(w,c) = 1$, para algum polinômio p e
- se $w \notin \mathsf{L}$ então $\mathsf{M}(w,c) = 0$ para todo $c \in \{0,1\}^*$.

Prove que a classe NP é a mesma para qualquer definição dentre as três descritas acima.

PROBLEMAS COMPLETOS A linguagem L é chamada de NP-**completa** se está em NP e existe um algoritmo eficiente R tal que para toda linguagem $\mathsf{I} \in \mathsf{NP}$ temos $x \in \mathsf{I}$ se e só se $\mathsf{R}(x) \in \mathsf{L}$. O algoritmo R é chamado de **redução** de tempo polinomial. Decorre dessa definição que um algoritmo eficiente para uma linguagem NP-completa L implica num algoritmo eficiente para toda linguagem em NP, e isso resolveria o problema 5.

O famoso Teorema de Cook–Levin estabelece que SAT é NP-completo. A partir de SAT várias outras linguagens foram provadas ser completas para NP mostrando-se uma redução para SAT.

A definição análoga para a classe P, a linguagem $\mathsf{L} \in \mathsf{P}$ é P-completa se para toda linguagem em P há um redução em tempo polinomial para L, não é interessante para a teoria pois todo problema em P é completo (por quê?). Nesse caso, há outras reduções de interesse como, por exemplo, as de espaço logarítmico.

COMPLEMENTO DE UMA CLASSE Se $\Pi := (\mathcal{I}, \{\text{sim}, \text{não}\}, \mathcal{R})$ é um problema computacional de decisão então $\mathcal{R}$ particiona $\mathcal{I}$ de modo natural em $\mathcal{I}_{\text{sim}}$, o conjunto das instâncias sim, e $\mathcal{I}_{\text{não}}$, o conjunto das instâncias não. Como vimos acima, a codificação dos elementos em $\mathcal{I}_{\text{sim}}$ é a linguagem L associada ao problema. O **complemento de um problema de decisão**, denotado **co**Π obtido trocando-se os papéis das intâncias *sim* e *não*, isto é, co$\mathcal{P}$ é dado por $(\mathcal{J}, \{\text{sim}, \text{não}\}, \mathcal{S})$ com $\mathcal{S}$ tal que $\mathcal{J}_{\text{sim}} = \mathcal{I}_{\text{não}}$ e $\mathcal{J}_{\text{não}} = \mathcal{I}_{\text{sim}}$. Com respeito à linguagem associada, temos que coL é a linguagem definida pela codificação de $\mathcal{I}_{\text{não}}$. O **complemento de uma linguagem** $\mathsf{L} \subset \Sigma^*$ é a linguagem $\overline{\mathsf{L}} = \Sigma^* \setminus \mathsf{L}$.

O problema SAT é o de decidir se uma fórmula booleana é satisfazível, a linguagem SAT é formada pela codificação (canônica) das fórmulas booleanas satisfazíveis. O complemento do problema, o coSAT, é o problema apelidado de UNSAT de decidir se uma fórmula booleana não é satisfazível e

linguagem UNSAT é formada pela codificação (canônica) das fórmulas booleanas não satisfazíveis. Notemos que, a rigor, $\overline{\text{SAT}}$ é diferente de UNSAT (ou coSAT), o complemento da linguagem SAT, não é a linguagem UNSAT pois podem existir sequências binárias que não codificam fórmulas booleanas, no entanto, é possível decidir de modo eficiente se uma dada palavra não codifica uma fórmula booleana de modo que tratamos elas do mesmo modo. A discussão deste parágrafo se estende para os outros problemas.

Se $\mathtt{C}$ é uma classe de complexidade de problemas de decisão o **complemento da classe de complexidade**, denotada por $\mathtt{coC}$, é a classe dos problemas de decisão formada pelos complementos dos problemas em $\mathtt{C}$.

Pode-se deduzir facilmente da definição que $\mathtt{P} = \mathtt{coP}$: se $L \in \mathtt{P}$ então existe um algoritmo polinomial M que decide L. O algoritmo $\overline{M}(w) := 1 - M(w)$ de tempo polinomial decide $\overline{L}$, portanto $L \in \mathtt{coP}$. A recíproca desse argumento vale, ou seja, $\mathtt{coP} \subset \mathtt{P}$, portanto esses conjuntos são iguais. No entanto, na definição de NP há uma assimetria entre aceitar e rejeitar uma palavra o que invalida o mesmo argumento para tentar provar que $\mathtt{NP} = \mathtt{coNP}$. De fato, atualmente, não é sabido se vale tal igualdade.

Problema 6. $\mathtt{NP} \neq \mathtt{coNP}$?

Por exemplo, a linguagem TAUT formada pelas fórmulas booleanas verdadeiras (tautologias) está em coNP. O complemento da linguagem é dado pelas fórmulas booleanas que não são tautológicas e para uma dada fórmula dessa linguagem um certificado é uma valoração que faça fórmula falsa, portanto em NP. É fácil verificar que se $\mathtt{NP} \neq \mathtt{coNP}$ então $\mathtt{P} \neq \mathtt{NP}$.

COMPLEXIDADE DE CIRCUITOS A classe $\mathtt{P/poly}$ é a classe de todas as funções booleanas $f\colon \{0,1\}^* \to \{0,1\}$ que são computáveis por uma família de circuitos de tamanho polinomial.

O principal motivo para a definição da classe $\mathtt{P/poly}$ foi a esperança de poder separar P de NP. Do exercício 4.4 acima nós deduzimos que uma cota inferior para o tamanho dos circuitos que computam um determinado problema implica numa cota inferior para o tempo de um algoritmo que computa o mesmo problema. Esse fato fornece uma estratégia de prova de que $\mathtt{P} \neq \mathtt{NP}$. Por exemplo, se soubéssemos provar que SAT não pode ser decidido por uma família de circuitos de tamanho polinomial[3] então SAT não poderia ser decidido por um algoritmo de tempo polinomial e esse fato estabeleceria que $\mathtt{P} \neq \mathtt{NP}$.

Ademais, funções booleanas que precisam de circuitos grandes abundam. O número de funções booleanas em n variáveis é 2^{2^n} e a fração delas que são computadas com um circuito com $t = 2^n/n$ portas lógicas tende a 0 quando n cresce. A quantidade de circuitos de tais pode ser estimada, de fato super estimada, com a seguinte descrição: uma sequência de t ternas em que cada terna (o, e, d) descreve o operador o da porta subjacente, uma entrada e dessa porta e a outra entrada d dessa porta

[3] Não há nada especial em SAT a não ser pelo fato de ser NP-completo; qualquer outro problema NP-completo bastaria.

lógica. Se $C(n,t)$ é a quantidade de circuitos com n entradas e t portas então

$$C(n,t) < \frac{(3(n+t)^2)^t}{t!} < \frac{(3(n+t)^2)^t}{\sqrt{2\pi t}(t/e)^t} < \frac{(3e4)^t}{\sqrt{2\pi t}} t^t = \frac{1}{\sqrt{2\pi t}} c^t t^t$$

usando um limitante inferior para o fatorial da estimativa de Stirling, (d.3) do Apêndice, $c > 0$ uma constante e $t = 2^n/n > n$ para todo $n > 4$, ainda $1/\sqrt{2\pi t} < 0{,}2$ de modo que

$$C(n, 2^n/n) < 0{,}2\left(c\frac{2^n}{n}\right)^{2^n/n} = 0{,}2\left(\frac{c}{n}\right)^{2^n/n} 2^{2^n}$$

de modo que $C(n, 2^n/n) \ll 2^{2^n}$, ou seja, a quantidade de circuitos com $2^n/n$ portas lógicas é assintoticamente muito menor que a quantidade de funções booleanas em n variáveis, assim concluímos que, assintoticamente, quase todas as funções, uma fração $(1 - o(1))$ delas, necessitam de circuitos maiores que t para serem computadas. Uma justificativa para a dificuldade de explicitarmos um problema difícil para o qual conseguimos provar uma cota inferior superpolinomial, como $2^n/n$, é dada na seção 4.5.

Também, decorre do exercício 4.4 acima que $\mathsf{P} \subset \mathsf{P/poly}$. Atualmente, não é sabido se $\mathsf{NP} \not\subset \mathsf{P/poly}$.

Problema 7. $\mathsf{NP} \not\subset \mathsf{P/poly}$?

Se esse for o caso, ou seja a resposta para o problema acima é sim, como explicamos no parágrafo acima, concluímos que $\mathsf{P} \neq \mathsf{NP}$, resolvendo o problema 5 (página 190).

4.2.1 CLASSES PROBABILÍSTICAS

A classe RP — *Randomized Polynomial time* — é formada pelas linguagens L decididas por um algoritmo probabilístico M de tempo polinomial tal que para algum polinômio p e para todo $w \in \{0,1\}^*$, a probabilidade de erro é

1. se $w \in L$, então $\mathbb{P}_{B \in_R \{0,1\}^{p(|w|)}}[M(w,B) = 0] \leqslant 1/3$, ou seja, se $w \in L$ então a M erra com probabilidade limitada, e
2. se $w \notin L$, então $\mathbb{P}_{B \in_R \{0,1\}^{p(|w|)}}[M(w,B) = 1] = 0$, ou seja, se $w \notin L$ então M não erra.

Assim, pelo item 2 uma resposta 1 (sim) do algoritmo M está sempre certa, isto é $w \in L$, enquanto que uma resposta 0 (não) pode ser um falso negativo, o que ocorre com probabilidade no máximo 1/3. O problema de determinar se um gafo G tem um corte pequeno, de tamanho menor que um dado k, está em RP pois o algoritmo 9, página 66, pode ser executado duas vezes o que limita a probabilidade de erro a 1/4, além disso se o algoritmo acha um corte pequeno responde sim (corretamente), se não acha (o que não significa que não tenha) então responde não (com chance de erro).

A constante 1/3 nos algoritmos que definem RP é arbitrária. Se realizamos k execuções independentes desse algoritmo (com a mesma entrada), então uma resposta 1 é definitiva enquanto que k respostas 0 estão todas erradas com probabilidade menor que $(1/3)^k$ e se $k = O(n^c)$ para alguma constante positiva c então as execuções terminam em tempo polinomial. O mesmo vale se trocarmos 1/3 por qualquer constante $\varepsilon \in (0,1)$ de modo que se a probabilidade de erro for ε então podemos executar o algoritmo várias vezes (com a mesma entrada) até que a probabilidade de erro fique menor que 1/3, o que certifica a pertinência em RP.

Observemos que se M decide uma linguagem L que pertence a RP então as escolhas de bits aleatórios por $M(w)$ que terminam em $M(w) = 1$ é um certificado c para $w \in L$ portanto podemos concluir que

$$RP \subset NP.$$

A classe coRP é a classe das linguagens L tais que $\overline{L} \in RP$, isto é, a classe das linguagens L para as quais existe um algoritmo probabilístico M de tempo polinomial tal que uma resposta 0 (não) está sempre certa enquanto que uma resposta 1 (sim) pode ser um falso positivo. Para algum polinômio p e para todo $w \in \{0,1\}^*$,

1. se $w \in L$, então $\mathbb{P}_{B \in_R \{0,1\}^{p(|w|)}}[M(w,B) = 0] = 0$, e
2. se $w \notin L$, então $\mathbb{P}_{B \in_R \{0,1\}^{p(|w|)}}[M(w,B) = 1] \leqslant 1/3$.

Nos algoritmos 11 e 14 (pág. 68 e seguintes) para, respectivamente, os testes de igualdade do produto de matrizes e de identidade polinomial, uma resposta *não* está sempre certa enquanto que uma resposta *sim* pode estar errada, ademais duas rodadas independentes de tais algoritmos reduzem a probabilidade de erro para o limiar da definição da classe coRP, logo as linguagens definidas por esses exemplos são linguagens em coRP. O problema de testar se um número é primo também está em coRP como atesta o algoritmo de Miller–Rabin (página 93), que também atesta que o problema de testar se um número é composto está em RP.

Um fato interessante ocorre quando consideramos a possibilidade de uma linguagem $L \in RP \cap coRP$, pois ela pode se beneficiar dos resultados exatos do algoritmo M_{RP}, o que prova que $L \in RP$, e do algoritmo M_{coRP}, o que prova que $L \in coRP$, para especificar um algoritmo probabilístico M′ que nunca erra, porém uma escolha ruim de bits aleatórios pode levar o algoritmo a executar por muito tempo: dada uma entrada $w \in \{0,1\}^*$

```
1 enquanto verdadeiro faça
2 |   se M_RP(w) = 1 então responda 1;
3 |   se M_coRP(w) = 0 então responda 0.
```

Algoritmo 39: $RP \cap coRP$

Se M_{RP} responder 1 então por definição garantimos que $w \in L$. Analogamente, se M_{coRP} responder 0 então por definição garantimos que $w \notin L$. Assim nunca obtemos uma resposta errada. Porém não

temos como saber exatamente quantas rodadas vamos esperar até que um certificado apropriado seja encontrado.

Um algoritmo probabilístico tem **tempo esperado** $T(n)$ se a variável aleatória t_w, que denota o tempo de execução do algoritmo com entrada w, tem valor esperado $\mathbb{E}\, t_w \leqslant T(|w|)$ para todo w. A esperança é computada sobre os bits aleatórios usados na computação pelo algoritmo. Nos casos em que T é um polinômio dizemos que a máquina é de **tempo esperado polinomial**. Por exemplo, o algoritmo aproximativo para o problema MAX-E3SAT, algoritmo 35 na página 148, tem tempo esperado polinomial.

No algoritmo 39 acima, o número esperado de execuções até o passo 2 ter sucesso é no máximo 2 e o mesmo vale para o passo 3, portanto o tempo esperado de M' é no máximo um número constante de simulações de $M_{RP}(w)$ e de $M_{coRP}(w)$, ou seja, o tempo esperado de execução é polinomial em $|w|$.

A classe ZPP — *Zero-error Probabilistic Polynomial time* — é a classe de complexidade de todas as linguagens para as quais existe um algoritmo probabilístico M de tempo *esperado* polinomial e tal que $\mathbb{P}[\,M(w,B) = L(w)\,] = 1$, para todo $w \in \{0,1\}^*$. Equivalentemente, ZPP é a classe das linguagens para as quais existe um algoritmo probabilístico de tempo polinomial M tal que, para todo w, $M(w) \in \{0,1,?\}$ em que 0 significa $x \notin L$ e 1 significa $x \in L$, como é usual, e ? significa "não sei" e $\mathbb{P}[\,M(w) = ?\,] \leqslant 1/2$. Intuitivamente, entendemos essa definição como que em tempo polinomial ou o algoritmo decide ou interrompe a execução.

O algoritmo 39 acima mostra que

$$\mathsf{RP} \cap \mathsf{coRP} \subset \mathsf{ZPP} \tag{4.2}$$

mas nesse caso vale a igualdade dessas classes como demonstra o seguinte resultado.

LEMA 4.6 $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$.

DEMONSTRAÇÃO. De (4.2), só precisamos provar que $\mathsf{ZPP} \subset \mathsf{RP} \cap \mathsf{coRP}$.

Seja $L \in \mathsf{ZPP}$ uma linguagem e M um algoritmo probabilístico de tempo esperado $p(n)$ que decide pertinência em L sem errar. Para provar pertinência em RP, definimos um algoritmo N que computa da seguinte maneira: dada uma entrada $w \in \{0,1\}^*$

```
1 simula M com entrada w até no máximo 3p(|w|) passos;
2 se M(w) = 1 então responda 1;
3 senão responda 0.
```

Algoritmo 40: $N(w)$

Se $w \notin L$ então N termina sem aceitar w, portanto $\mathbb{P}[\,\text{erro}\,] = \mathbb{P}[\,N(w) = 1\,] = 0$. Por outro lado, se $w \in L$ então o algoritmo N aceita w (corretamente) ou termina pelo limite dos $3p(|w|)$ passos. No segundo caso $N(w) = 0$ e a resposta está errada. A probabilidade da resposta errada é $\mathbb{P}[\,t_w > 3p(|w|)\,]$ em que, como acima, t_w é a variável aleatória para o tempo de execução do algoritmo M com entrada

w. Pela desigualdade de Markov, equação (3.33) na página 164,

$$\mathbb{P}[\, t_w \geqslant 3p(|w|)+1\,] \leqslant \frac{p(|w|)}{3p(|w|)+1} < \frac{1}{3}$$

portanto, se $w \in \mathsf{L}$ então $\mathbb{P}[\,\text{erro}\,] = \mathbb{P}(\,\mathsf{N}(w)=0\,) < 1/3$ e com isso temos que $\mathsf{L} \in \mathsf{RP}$.

Do maneira análoga, podemos provar que $\mathsf{ZPP} \subset \mathsf{coRP}$; definimos um algoritmo N' que com entrada w

```
1 simula M com entrada w até no máximo 3p(|w|) passos;
2 se M(w) = 0 então responda 0;
3 senão responda 1.
```

Algoritmo 41: N'(w)

Assim, se $w \in \mathsf{L}$ então $\mathbb{P}[\,\text{erro}\,] = \mathbb{P}[\,\mathsf{N}(w)=0\,] = 0$. Se $w \notin \mathsf{L}$ então o algoritmo pode aceitar erroneamente e $\mathbb{P}[\,\text{erro}\,] = \mathbb{P}[\, t_w > 3p(|w|)\,] < 1/3$. Portanto, $\mathsf{ZPP} \subset \mathsf{RP} \cap \mathsf{coRP}$. □

BPP A classe BPP — *Bounded-error Probabilistic Polynomial time* — é a classe das linguagens decididas por algoritmos probabilísticos de tempo polinomial com probabilidade de erro 1/3, ou seja, $\mathsf{L} \in \mathsf{BPP}$ se existe um polinômio p e um algoritmo probabilístico M de tempo polinomial tal que para todo $w \in \{0,1\}^*$

$$\mathop{\mathbb{P}}_{B \in_R \{0,1\}^{p(|w|)}} [\mathsf{M}(w,B) \neq \mathsf{L}(w)] \leqslant 1/3.$$

A probabilidade de erro 1/3 na definição não tem nada de especial, qualquer constante $\varepsilon \in (0,1/2)$ serviria para definir a mesma classe de linguagens, como veremos no lema 4.7 abaixo. Se M é uma máquina probabilística de tempo polinomial que aceita a linguagem L com probabilidade de erro ε, então podemos escrever uma máquina N probabilística e de tempo polinomial que aceita a mesma linguagem L com probabilidade de erro 1/3.

Lema 4.7 *Para toda constante $0 < \varepsilon < 1/2$, todo polinômio p, toda linguagem $\mathsf{L} \subset \{0,1\}^*$ e todo algoritmo probabilístico* M *de tempo polinomial que decide* L *com probabilidade de erro ε, existe um algoritmo probabilístico* N *de tempo polinomial e que decide* L *com probabilidade de erro $2^{-p(n)}$ nas entradas de tamanho n, para todo n.*

Demonstração. Sejam ε, p, L e M como no enunciado. Definimos

$$\delta := 4\varepsilon(1-\varepsilon) \quad \text{e} \quad k(n) := \left\lceil \frac{p(n)}{\log_2(1/\delta)} \right\rceil$$

e notemos que $\delta < 1$ pois $\varepsilon < 1/2$.

Consideremos o algoritmo N que com entrada $w \in \{0,1\}^*$ simula M(w) um número ímpar de vezes

e decide pela resposta dada na maioria das simulações:

```
1 para i de 1 até 2k + 1 faça
2 |   m_i ← M(w);
3 se ∑_i m_i > k então responda 1;
4 senão responda 0.
```

Certamente, N é uma algoritmo probabilístico de tempo polinomial. Seja $S = S(w) \in \{0,1\}^{2k+1}$ uma sequência de respostas dadas na linha 2 em uma execução de N com entrada w. Sejam $c = c(S)$ e $e = e(S)$ o número de respostas certas e respostas erradas, respectivamente, em S, onde resposta certa quer dizer que M decidiu corretamente a pertinência de w em L.

O algoritmo N responde errado se para a sequência S correspondente a uma execução de N ocorre $c < e$. Queremos limitar a probabilidade desse evento. Como M tem probabilidade de erro limitada por $\varepsilon < 1/2$, a probabilidade de uma sequência S que faz N responder errado é no máximo $\varepsilon^e(1-\varepsilon)^c \leqslant \varepsilon^{k+1}(1-\varepsilon)^k$ pois $e \geqslant k+1$ e $\varepsilon < 1-\varepsilon$. Assim, a probabilidade de erro é

$$\mathbb{P}[\,N(w) \neq L(w)\,] \leqslant \sum_S \varepsilon^{e(S)}(1-\varepsilon)^{c(S)} \leqslant 2^{2k+1}\varepsilon^{k+1}(1-\varepsilon)^k = 2\varepsilon(2^2\varepsilon(1-\varepsilon))^k < (4\varepsilon(1-\varepsilon))^k = \delta^k$$

em que a soma é sobre toda sequência S de respostas de M que faz N responder errado. Pela escolha de k e de $\log_2(1/\delta) = 1/\log_\delta(1/2)$ temos $\delta^k \leqslant 2^{-p(|w|)}$. □

Exemplo 4.8 (PIT∈BPP). Seja $p \in \mathbb{F}[x_1,\ldots,x_n]$ um polinômio dado por um circuito aritmético, como definido na página 189. O PIT é o problema da identidade polinomial: decidir se o polinômio p definido pelo circuito $\langle p\rangle$ é identicamente nulo. É um problema importante na teoria sabermos se é possível determinar se conseguimos resolvê-lo em tempo polinomial no tamanho do circuito.

Comentamos na seção 2.2.3 que, a princípio, temos dois problemas computacionais: dado um polinômio p, EZE – *Evaluates to Zero Everywhere* – decidir se, como função, p vale zero em todo elemento do corpo e PIT que é decidir se p na forma canônica tem todos os coeficientes nulos. EZE está em coNP. De fato é coNP-difícil, é possível escrever uma fórmula 3-CNF como um polinômio sobre $\mathbb{F}_2$.

Um circuito aritmético de tamanho m tem profundidade no máximo m, portanto realiza no máximo m multiplicações, logo define um polinômio de grau no máximo 2^m. Nessa situação, o algoritmo 14, página 73, sorteia n valores em $\{1,\ldots,2^{m+2}\}$, avalia o valor de p com esses valores simulando o circuito aritmético em tempo polinomial em m e resolve PIT com probabilidade de erro 1/4. Porém, a sequência sorteada $(x_1,\ldots,x_n)$ tem tamanho $O(nm)$ bits enquanto que x^{2^m} calculado em 2^{m+2} resulta num número com $O(m2^m)$ bits, logo só para escrevê-lo o tempo consumido seria exponencialmente grande, o que não resulta num algoritmo de tempo polinomial para o problema.

Um modo de contornarmos esse problema é calcularmos as operações aritméticas nas portas do circuito módulo um inteiro positivo k apropriado, resultando, no final do cômputo, $p(x_1,\ldots,x_n)$ mod

k. Com isso o tempo de simulação do circuito continua polinomial e os números que ocorrem nas operações têm tamanho controlado, mas aumenta a probabilidade de erro pois podemos ter $p(x_1,\ldots,x_n) \neq 0$ e $p(x_1,\ldots,x_n) \bmod k = 0$, caso k divida $p(x_1,\ldots,x_n)$. Agora, devemos estimar essa probabilidade de erro e fazê-la pequena usando rodadas independentes de escolhas para k.

Tomemos $k \in \{1,2,\ldots,2^{2m}\}$. A quantidade de números primos nesse conjunto é, pelo Teorema dos Números Primos, maior que

$$\frac{2^{2m}}{2m+2}$$

se $m > 2$ (Rosser, 1941).

Assumamos que $p := p(x_1,\ldots,x_n) \neq 0$. A quantidade de fatores primos distintos em p é $(m+2)2^m$, pois se $p = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t}$ então $p \geqslant 2^t$, portanto $t \leqslant \lg p \leqslant \lg(2^{m+2})^{2^m} = (m+2)2^m$. Assim, a quantidade de primos em $\{1,2,\ldots,2^{2m}\}$ que não é um dos t fatores primos de p é maior que

$$\frac{2^{2m}}{2m+2} - (m+2)2^m > \frac{2^{2m}}{8m}$$

para todo $m > 7$, de modo que a probabilidade com que uma escolha aleatória em $\{1,2,\ldots,2^{2m}\}$ resulte num número *bom*, isto é um primo que não divide m, é maior que $1/8m$. Em r sorteios para o valor de k, basta um deles resultar num número *bom* para podermos responder que $p(x_1,\ldots,x_n) \neq 0$. A probabilidade com que nenhum dentre $r := 16m$ sorteios resulte num número *bom* é no máximo

$$\left(1 - \frac{1}{8m}\right)^r = \left(\left(1 - \frac{1}{8m}\right)^{-8m}\right)^{-2} < 0{,}15$$

para todo inteiro $m > 0$, pois $(1 - 1/8m)^{-8m} \geqslant$ e (veja (s.8)). Portanto, se $\langle p \rangle$ é nulo o algoritmo sempre descobre, caso contrário o polinômio não nulo é declarado nulo com probabilidade 0,15, logo PIT $\in$ BPP. $\Diamond$

Da definição das classes deduzimos que

$$\mathsf{RP}, \mathsf{coRP}, \mathsf{ZPP} \subset \mathsf{BPP}.$$

Notemos que $\mathsf{P} \subset \mathsf{BPP}$ pois para todo algoritmo de tempo polinomial M podemos escrever um algoritmo probabilístico de tempo polinomial M' que simula M e ignora os bits aleatórios. Não sabemos se a recíproca vale ou não

Problema 8. $\mathsf{BPP} \subset \mathsf{P}$?

Também não sabemos situar BPP com relação a NP

Problema 9. $\mathsf{BPP} \subset \mathsf{NP}$?

não sabemos se $\mathsf{BPP} \subset \mathsf{NP}$, se $\mathsf{NP} \subset \mathsf{BPP}$ ou se nenhuma dessas duas relações valem. Também, não conhecemos nenhuma linguagem que é completa para BPP. Agora, é um exercício fácil, que deixamos para o leitor, verificar que $\mathsf{BPP} = \mathsf{coBPP}$.

Problema 10. Alguma(s) das inclusões $\mathsf{P} \subset \mathsf{ZPP} \subset \mathsf{RP} \subset \mathsf{BPP}$ são próprias?

Os algoritmos que reconhecem linguagens em BPP são chamados, em algumas referências bibliográficas, de **Atlantic city**, os de RP e coRP são conhecidos como **Monte-Carlo** e os de ZPP são os **Las Vegas**.

Finalizamos essa seção enunciado o seguinte resultado sem prova.

TEOREMA (Impagliazzo e Wigderson, 1999) *Se existe uma linguagem* L *decidida em tempo* $2^{O(n)}$ *que nas palavras de tamanho* n *requer circuito de tamanho* 2^{cn}*, para algum* $c > 0$ *e todo* n*, então* $\mathsf{P} = \mathsf{BPP}$.

Exercício 4.9. Prove que se $\mathsf{L} \in \mathsf{BPP}$, então existe um algoritmo probabilístico que com entrada w decide se $w \in \mathsf{L}$ com probabilidade de erro menor que $1/(3m)$ e em tempo polinomial em $|w|$, em que $m = m(|w|)$ é o número (polinomial) de bits aleatórios usados na computação (dica: ajuste a quantidade de rodadas independentes de simulações no algoritmo acima).

4.2.2 BPP $\subset$ P/POLY

O próximo resultado mostra que toda linguagem decidida por algoritmo probabilístico de tempo polinomial também é decidida por circuitos de tamanho polinomial. Como circuito é um modelo não-uniforme de computação e como há muitas sequências binárias aleatórias que testemunham a favor da decisão correta, podemos escolher uma tal sequência para cada n e todo $w \in \{0,1\}^n$ e projetar os circuitos com as sequências escolhidas. Essa ideia está formalizada no teorema a seguir. A inclusão é própria porque há problemas não decidíveis por algoritmos que são decididos por família de circuitos de tamanho polinomial, como já dissemos acima (exercício 4.25, página 234).

Teorema 4.10 (Adleman, 1978) $\mathsf{BPP} \subsetneq \mathsf{P/poly}$.

Demonstração. Sejam $\mathsf{L} \in \mathsf{BPP}$ e M um algoritmo probabilístico de tempo polinomial que decide L com probabilidade de erro exponencialmente pequena

$$\mathop{\mathbb{P}}_{B \in_R \{0,1\}^{p(n)}} [\mathsf{M}(w, B) \neq \mathsf{L}(w)] < 2^{-n}$$

nas entradas de tamanho n, para algum polinômio p. A existência desse algoritmo é garantido pelo lema 4.7 acima.

Fixado um inteiro positivo n, podemos afirmar que existe uma sequência binária b que é um certificado de pertinência em L para toda entrada w de tamanho n. De fato,

$$\mathop{\mathbb{P}}_{B \in_R \{0,1\}^{p(n)}} \left[\bigcup_{w \in \{0,1\}^n} \mathsf{M}(w, B) \neq \mathsf{L}(w) \right] < \sum_{w \in \{0,1\}^n} 2^{-n} = 1$$

logo, com probabilidade maior que 0 para pelo menos um $B \in \{0,1\}^{p(n)}$ fixo o algoritmo M responde corretamente para todo $w \in \{0,1\}^n$, ou seja, existe (ao menos) uma sequência $b_n \in \{0,1\}^{p(n)}$, com a

qual M não erra nas entradas de tamanho n. O algoritmo (determinístico) M' dado por

$$\mathsf{M}'(w) := \mathsf{M}(w, b_n)$$

não erra em entradas de tamanho n.

A partir de M' construímos, para cada n, um circuito booleano C_n de tamanho polinomial, conforme construção do exercício 4.4 (página 188), tal que $\mathsf{C}_n(w) = \mathsf{M}'(w)$, para todo w de tamanho n. Dessa forma, a família de circuitos $(\mathsf{C}_n)_{n>0}$ decide L, portanto $\mathsf{L} \in \mathsf{P/poly}$. □

4.2.3 BPP ESTÁ NA HIERARQUIA POLINOMIAL

A hierarquia polinomial é uma hierarquia formada por classes de problemas com complexidade polinomial, as *classes do nível i* são denotadas por Σ_i e Π_i, para todo $i \in \mathbb{N}$. Essas classes generalizam P e NP de um certo modo natural. Uma motivação para essa hierarquia pode ser lida no capítulo 17 de Papadimitriou (1994), assim como outros resultados que estão fora do escopo deste texto pois precisam de vários pré-requisitos da Complexidade Computacional. Nesta seção, falaremos brevemente sobre a hierarquia polinomial, sua relação com as classes probabilísticas e daremos alguns outros resultados sem prova para que o leitor possa ter alguma referência sobre a importância de PH e sua relação com outras classes de complexidade.

Por enquanto vamos nos concentrar nos primeiros níveis da hierarquia e em seguida mostrar uma relação com a classe BPP. Começamos declarando que

$$\Sigma_0 = \Pi_0 := \mathsf{P} \text{ e } \Sigma_1 := \mathsf{NP} \text{ e } \Pi_1 := \mathsf{coNP}$$

ou seja, por definição Σ_0 é a classe das linguagens L para as quais existe um algoritmo M de tempo polinomial tal que para todo $w \in \{0,1\}^*$

$$w \in \mathsf{L} \Leftrightarrow \mathsf{M}(w) = 1,$$

Σ_1 é a classe das linguagens L para as quais existe um algoritmo M de tempo polinomial e um polinômio p tais que para todo $w \in \{0,1\}^*$

$$w \in \mathsf{L} \Leftrightarrow \exists u \in \{0,1\}^{p(|w|)},\ \mathsf{M}(w,u) = 1,$$

definimos Σ_2 como a classe das linguagens L para as quais existe um algoritmo M de tempo polinomial e polinômios p e q tais que para todo $w \in \{0,1\}^*$

$$w \in \mathsf{L} \Leftrightarrow \exists u \in \{0,1\}^{p(|w|)}\ \forall v \in \{0,1\}^{q(|w|)},\ \mathsf{M}(w,u,v) = 1. \tag{4.3}$$

Exemplo 4.11 (*conjunto independente máximo*). Em um grafo G, um conjunto $U \subset V(G)$ é dito *independente* (ou *estável*) se $|e \cap U| \leqslant 1$ para toda aresta $e \in E(G)$, isto é, U não contêm os dois vértices de qualquer aresta do grafo. Um conjunto independente é *máximo* se tem cardinalidade

$$\alpha(G) := \max\{|U| : U \subset V(G) \text{ é independente}\}.$$

Decidir se um grafo G tem conjunto independente de cardinalidade (pelo menos) k está em NP: dados G e U é possível verificarmos em tempo polinomial se U é um subconjunto com pelo menos k vértices de G e independente e, se esse é o caso, então uma codificação de U tem tamanho polinomial no tamanho de G. Agora, um tal conjunto independente é máximo em G se o grafo não tem conjunto independente de cardinalidade $k+1$. Decidir se um grafo G não tem conjunto independente de cardinalidade $k+1$ está em coNP (por quê?).

Decidir "$\alpha(G) = k$?" tem um certificado curto (tamanho polinomial) para uma parte do problema e não tem um certificado curto para a outra parte do problema. A linguagem

$$\textsc{max-ind} := \{\langle G, k\rangle : G \text{ é um grafo com } \alpha(G) = k\} \tag{4.4}$$

está em Σ_2. De fato, escrevendo na forma (4.3), um par $\langle G, k\rangle$ está na linguagem se e somente se existe um conjunto independente U de cardinalidade k e todo subconjunto V de cardinalidade (pelo menos) $k+1$ não é independente. Deve ficar claro ao leitor que a codificação dos subconjuntos tem tamanho polinomial no tamanho do grafo. Ainda não sabemos se a linguagem descrita em (4.4) acima está ou não está em $\mathrm{NP} = \Sigma_1$. ♢

O próximo teorema é o principal resultado dessa seção. A demonstração do teorema é de Lautemann (1983). Usaremos a notação $a \oplus u$ para denotar a operação *ou exclusivo* (ou soma módulo 2) coordenada-a-coordenada das sequências binárias $a, u \in \{0,1\}^m$. Também, usaremos o exercício 1.43, página 41, que afirma que se $X \in_R \{0,1\}^m$ e $Y \in_D \{0,1\}^m$ é uma variável aleatória com distribuição arbitrária sobre $\{0,1\}^m$ então $X \oplus Y \in_R \{0,1\}^m$.

Teorema 4.12 (Sipser–Gács–Lautemann, 1983) $\mathrm{BPP} \subset \Sigma_2$.

A ideia da prova é que se $w \in L$ e $L \in \mathrm{BPP}$ então há um algoritmo M para o qual quase toda sequência de um conjunto $\{0,1\}^m$ (m polinomial em $|w|$) é um certificado disso de modo que, dado $r \in \{0,1\}^m$, é certo que alguma translação de r tomada de um número polinomial delas, digamos $r \oplus u_1, \ldots, r \oplus u_m$, é um certificado para w, ou seja, $M(w, r \oplus u_i) = 1$. Por outro lado, se $w \notin L$, então uma fração ínfima dos elementos de $\{0,1\}^m$ certificaria erroneamente a pertinência de x em L de modo que é possível haver $r \in \{0,1\}^m$ tal que nenhuma das translações $r \oplus u_1, \ldots, r \oplus u_m$ é um certificado, ou seja, para todo i temos $M(w, r \oplus u_i) = 0$. Essa afirmação é capturada na sentença (4.5) abaixo que caracteriza L como uma linguagem de Σ_2

$$w \in L \Leftrightarrow \exists u_1, u_2, \ldots, u_m \in \{0,1\}^m \forall r \in \{0,1\}^m, \bigvee_{i=1}^{m} M(w, r \oplus u_i) = 1 \tag{4.5}$$

em que $\bigvee$ denota o operador *ou* lógico dos m termos "$\mathsf{M}(w, r \oplus u_i) = 1$". Além disso, o predicado $\bigvee_{i=1}^{m} \mathsf{M}(w, r \oplus u_i) = 1$ é uma computação de tempo polinomial, pois M é executada em tempo polinomial e um número polinomial de vezes. Feito isso estabeleceremos que $\mathsf{L} \in \Sigma_2$.

Demonstração do teorema 4.12. Sejam $\mathsf{L} \in \mathsf{BPP}$ e M um algoritmo probabilístico de tempo $q(n)$, que nas entradas de tamanho n usa $m = m(n)$ bits aleatórios e erra com probabilidade menor que $1/(3m)$, tal algoritmo existe pelo exercício 4.9, página 199. Vamos provar que vale a equação (4.5). Suponhamos que $w \in \mathsf{L}$. Então

$$\mathop{\mathbb{P}}_{U_1,\ldots,U_m \in_R \{0,1\}^m} [\exists r \in \{0,1\}^m \forall i \in \{1,\ldots,m\},\ \mathsf{M}(w, U_i \oplus r) = 0] \leqslant$$
$$\sum_{r \in \{0,1\}^m} \mathop{\mathbb{P}}_{U_1,\ldots,U_m \in_R \{0,1\}^m} [\forall i \in \{1,\ldots,m\},\ \mathsf{M}(w, U_i \oplus r) = 0] \leqslant 2^m \left(\frac{1}{3m}\right)^m < 1$$

portanto, qualquer que seja $r \in \{0,1\}^m$, existe uma sequência $u_1, \ldots, u_m \in \{0,1\}^m$ tal que vale $\bigvee_{i=1}^{m} \mathsf{M}(w, r \oplus u_i) = 1$.

Agora, suponhamos que $w \notin \mathsf{L}$. Fixada uma sequência $u_1, \ldots, u_m \in \{0,1\}^m$

$$\mathop{\mathbb{P}}_{R \in_R \{0,1\}^m} [\exists i \in \{1,\ldots m\},\ \mathsf{M}(w, R \oplus u_i) = 1] \leqslant$$
$$\sum_{i=1}^{m} \mathop{\mathbb{P}}_{R \in_R \{0,1\}^m} [\mathsf{M}(w, r \oplus u_i) = 1] \leqslant m \frac{1}{3m} = \frac{1}{3}$$

portanto, existe $r \in \{0,1\}^m$ tal que $\mathsf{M}(w, r \oplus u_1) = \mathsf{M}(w, r \oplus u_2) = \cdots = \mathsf{M}(w, r \oplus u_m) = 0$, o que prova a equação (4.5). □

A definição de BPP é simétrica, isto é, $\mathsf{BPP} = \mathsf{coBPP}$ de modo que do teorema acima temos $\mathsf{BPP} \subset \mathsf{co}\Sigma_2$. Definimos $\Pi_2 := \mathsf{co}\Sigma_2$ e, portanto, o teorema 4.12 garante que

$$\mathsf{BPP} \subset \Sigma_2 \cap \Pi_2.$$

UMA VISÃO GERAL DE PH A **hierarquia polinomial** é definida pela sequências de classes de complexidade: $\Sigma_0 = \Pi_0 = \mathsf{P}$ e

- para todo $i \geqslant 1$, Σ_i é a classe das linguagens L para as quais existe um algoritmo M de tempo polinomial e polinômios $p_1, p_2, \ldots, p_i$ tais que para todo $w \in \Sigma^*$

  $$w \in \mathsf{L} \Leftrightarrow \exists u_1,\ \forall u_2,\ \exists u_3,\ \ldots,\ \mathrm{Q} u_i,\ \ \mathsf{M}(w, u_1, u_2, \ldots, u_i) = 1$$

  onde $u_j \in \{0,1\}^{p_j(|w|)}$, para todo j, e Q é um quantificador $\forall$ ou $\exists$ dependendo da paridade do índice i;

- para todo $i \geqslant 1$, Π_i é a classe das linguagens L para as quais existe um algoritmo M de tempo polinomial e polinômios $p_1, p_2, \ldots, p_i$ tais que para todo $w \in \Sigma^*$

$$w \in \mathsf{L} \Leftrightarrow \forall u_1,\ \exists u_2,\ \forall u_3,\ \ldots,\ \mathrm{Q}u_i,\ \ \mathsf{M}(w, u_1, u_2, \ldots, u_i) = 1$$

  onde $u_j \in \{0,1\}^{p_j(|w|)}$, para todo j, e Q é um quantificador $\forall$ ou $\exists$ dependendo da paridade do índice i.

Não é difícil notar que, para cada $i \geqslant 1$, temos que as classes definidas acima são complementares, isto é, $\Pi_i = \mathrm{co}\Sigma_i$. Também vale, e não é difícil provar, que $\Pi_i \cup \Sigma_i \subset \Sigma_{i+1} \cap \Pi_{i+1}$, em particular $\mathsf{NP} \cup \mathrm{co}\mathsf{NP} \subset \Sigma_2 \cap \Pi_2$. Notemos que P, NP e BPP são todos subconjuntos de Σ_2.

Uma generalização do problema 5 ($\mathsf{P} \neq \mathsf{NP}$?) é a conjetura $\Sigma_i \neq \Sigma_{i+1}$? E uma generalização do problema 6 ($\mathsf{NP} \neq \mathrm{co}\mathsf{NP}$?) é a conjectura que $\Pi_i \neq \Sigma_i$ para todo $i \geqslant 1$? É possível mostrar que, para $i \geqslant 1$,

$$\textit{se } \Sigma_i = \Pi_i \textit{ então } \Sigma_i = \Sigma_{i+1}$$

donde deduzimos

$$\textit{se } \Sigma_i = \Pi_i \textit{ então } \Sigma_j = \Sigma_i \textit{ para todo } j \geqslant i$$

e podemos por o problema abaixo.

Problema 11. $\Sigma_i \subsetneqq \Sigma_{i+1}$?

Se, para algum i, $\Sigma_i = \Sigma_{i+1}$ então $\Sigma_j = \Pi_j = \Sigma_i$ para todo $j \geqslant i$. Nesse caso dizemos que a hierarquia **colapsa** no i-ésimo nível.

Definimos a classe PH — *Polynomial Hierarchy* — por

$$\mathsf{PH} := \bigcup_{i \geqslant 0} \Sigma_i = \bigcup_{i \geqslant 0} \Pi_i.$$

O seguinte resultado pode ser estudado no livro de Papadimitriou, 1994.

TEOREMA *Para todo $i \geqslant 1$, se $\Sigma_i = \Pi_i$ então $\Pi_{i+1} = \Sigma_{i+1} = \Sigma_i$ que implica em $\mathsf{PH} = \Sigma_i$. Em particular, se $\mathsf{P} = \mathsf{NP}$ então $\mathsf{PH} = \mathsf{P}$.*

Vimos que $\mathsf{P} \subset \mathsf{P/poly}$, portanto, se $\mathsf{P} = \mathsf{NP}$ então $\mathsf{NP} \subset \mathsf{P/poly}$ (o que não é sabido, problema 7, página 193). Uma questão interessante é saber se $\mathsf{NP} \subset \mathsf{P/poly}$ poderia ocorrer somente se $\mathsf{P} = \mathsf{NP}$. Um resultado nessa direção, num nível acima da hierarquia, é descrito a seguir, mostrando que $\mathsf{NP} \subset \mathsf{P/poly}$ pode ocorrer somente se PH colapsa para o segundo nível.

TEOREMA (Karp e Lipton, 1980) *Se $\mathsf{NP} \subset \mathsf{P/poly}$ então $\Sigma_2 = \Pi_2$.*

Como comentamos acima, se $\mathsf{P} = \mathsf{NP}$ então $\mathsf{P} = \mathsf{PH}$ e como $\mathsf{BPP} \subset \mathsf{PH}$, obtemos $\mathsf{BPP} \subset \mathsf{P}$. Já sabíamos que $\mathsf{P} \subset \mathsf{BPP}$, portanto, se $\mathsf{P} = \mathsf{NP}$ então $\mathsf{P} = \mathsf{BPP}$.

Corolário 4.13 *Se* $\mathsf{P} = \mathsf{NP}$ *então* $\mathsf{BPP} = \mathsf{P}$.

Logo se não há problemas difíceis no sentido de que todo problema NP tem algoritmo eficiente, então as soluções probabilísticas eficientes podem ser *desaleatorizadas*.

4.3 SISTEMAS PROBABILÍSTICOS DE PROVA

Num sistema de prova dois algoritmos interagem comunicando-se através de leitura e escrita num espaço comum. Chamaremos esses algoritmos de *Verificador*, denotado V, e *Provador*, denotado P. Ambos têm acesso a uma memória somente de leitura contendo a entrada w que é o fato a ser provado. O sistema ainda tem mais duas outras memórias compartilhadas por P e por V, numa delas, denotada $\mathsf{M}_{\mathsf{P}\to\mathsf{V}}$, o Provador P pode escrever e V somente fazer leituras, na outra, denotada $\mathsf{M}_{\mathsf{V}\to\mathsf{P}}$, os papéis são trocados, ou seja, V pode escrever e P somente fazer leituras. As memórias compartilhadas estão vazias (há somente $\flat$) no começo da computação. O Verificador *aceita* (responde 1) ou *rejeita* (responde 0) a prova dada pelo Provador e a resposta do Verificador é designada por $(\mathsf{V},\mathsf{P})(w)$. Ademais V é limitado, tem complexidade de tempo polinomial, enquanto que P não tem limitação. O número de rodadas de uma computação do sistema de provas é o número de mensagens $m \in \{0,1\}^*$ trocadas nas duas direções entre os algoritmos.

Uma linguagem L admite um sistema de prova se existe um verificador V de tempo polinomial e um protocolo com k rodadas, com k polinomial no tamanho da entrada, tal que são satisfeitos:

1. *completude*: se $w \in \mathsf{L}$ então em k rodadas $(\mathsf{V},\mathsf{P})(w) = 1$ para algum provador P;
2. *consistência*: se $w \notin \mathsf{L}$ então em k rodadas $(\mathsf{V},\mathsf{P})(w) = 0$ qualquer que seja o provador P.

SISTEMA P DE PROVA A classe P pode ser definida em termos de sistemas de provas da seguinte maneira: uma linguagem L está em P se admite um sistema de provas tal que para todo $w \in \{0,1\}^*$, se $w \in \mathsf{L}$ então quando o algoritmo V termina ele responde 1 sem nunca ter consultado $\mathsf{M}_{\mathsf{P}\to\mathsf{V}}$; se $w \notin \mathsf{L}$ o algoritmo V, no término de sua computação, responde 0 sem nunca ter consultado $\mathsf{M}_{\mathsf{P}\to\mathsf{V}}$.

SISTEMA NP DE PROVA Podemos definir um sistema de provas para qualquer linguagem $\mathsf{L} \in \mathsf{NP}$ pois, se $w \in \mathsf{L}$ então o Provador escreve um certificado c de tamanho polinomial em $\mathsf{M}_{\mathsf{P}\to\mathsf{V}}$ e o Verificador checa em tempo polinomial e com a ajuda da prova c, se $w \in \mathsf{L}$. Tal certificado existe se, e somente se, $w \in \mathsf{L}$.

Exemplo 4.14. Denotamos por 3-COL a linguagem definida pelos grafos que admitem um 3-coloração própria dos seus vértices, ou seja, G está nessa linguagem se e somente se todos os vértices de G podem ser coloridos com três cores distintas de modo que vértices adjacentes têm cores diferentes. A linguagem 3-COL está em NP.

Um sistema de provas para 3-COL consiste de uma instância G conhecida por V e P; o Verificador pergunta a cor de um determinado vértice de G ao Provador que responde com uma de três cores. A cada resposta, o Verificador verifica se a cor dada não conflita com as cores que já foram atribuídas. Se o Provador responde uma cor que viola a propriedade então o Verificador rejeita. Se, ao final, todos os vértices estão coloridos propriamente então o Verificador aceita.

Notemos que o Provador poderia, com seu poder computacional ilimitado, simplesmente computar e responder de uma só vez uma coloração. ◊

Agora, suponhamos que uma linguagem L admite um sistema de prova (V, P) de k (polinomial) rodadas. Para $w \in L$ o conteúdo de $M_{P \to V}$ ao final da computação é uma sequência $(m_1, m_2, \ldots, m_k)$ de tamanho polinomial de mensagens enviada pelo Provador. A sequência de mensagens é um certificado para V (de tempo polinomial) aceitar a entrada w, portanto *a classe das linguagens que admitem tal sistema de provas coincide com* NP.

O Provador só pode antecipar as respostas na estratégia acima se o comportamento do Verificador é determinístico, caso o Verificador use bits aleatórios, o Provador teria que gerar um certificado exponencialmente grande para cobrir todas as possibilidades. Portanto, esse cenário muda quando permitimos um Verificador aleatorizado. A aleatorização é essencial para que a classe das linguagens que admitem um sistema de prova vá além de NP, como ilustra o seguinte exemplo pitoresco e clássico na literatura sobre provas interativas: Alice tem duas bolas de bilhar, uma é vermelha e a outra verde; ambas parecem idênticas ao seu amigo Bob, que é daltônico e cético com respeito à diferença das bolas que lhe parecem iguais. Alice quer convencer Bob que, de fato, as bolas são de cores diferentes e para isso entrega as bolas ao Bob deixando uma em cada mão, sem dizer qual é a cor de cada uma (fato que ela se recordará sempre que for necessário). Em seguida, Bob esconde as mãos para trás e, sem que Alice veja, lança uma moeda e se o resultado for cara ele troca as bolas de mãos, se der coroa ele deixa as bolas como foi entregue por Alice. Feito isso, Bob questiona Alice sobre o seu ato, isto é, se ele trocou ou não as bolas. Se as bolas são diferentes então Alice acerta, sempre. Se as bolas são idênticas Alice acerta com probabilidade 1/2, e essa probabilidade reduz pela metade cada repetição do teste; Bob repete até estar seguro de que Alice está certa ou não, com 10 repetições e bolas idênticas a Alice acerta todas as respostas com probabilidade menor que 0,001.

4.3.1 A CLASSE IP

Um **sistema interativo de prova** é um sistema de prova como acima com a exceção de que o Verificador é um algoritmo *probabilístico* de tempo polinomial e P é um algoritmo probabilístico sem restrições de tempo. Esse conceito foi introduzido por Goldwasser, Micali e Rackoff (1985) e de modo independente por Babai (1985). Uma interação em k rodadas define uma sequência de mensagens $m_1, m_2, \ldots, m_k$ tal que

$$\begin{aligned} m_1 &:= \mathsf{V}(w), \\ m_2 &:= \mathsf{P}(w, m_1), \\ &\vdots \\ m_i &:= \begin{cases} \mathsf{V}(w, m_1, \ldots, m_{i-1}) \text{ se } i \text{ é ímpar menor ou igual a } k \\ \mathsf{P}(w, m_1, \ldots, m_{i-1}) \text{ se } i \text{ é par e menor ou igual a } k \end{cases} \end{aligned}$$

e $(\mathsf{V},\mathsf{P})(w) = \mathsf{V}(w, m_1, \ldots, m_k) \in \{0, 1\}$.

Uma linguagem L admite um sistema interativo de provas se existe um Verificador V probabilístico de tempo polinomial tal que com entrada w

1. *completude*: se $w \in \mathsf{L}$ então *existe* P tal que
$$\mathbb{P}[\,(\mathsf{V},\mathsf{P})(w) = 1\,] \geqslant \frac{2}{3};$$
2. *consistência*: se $w \notin \mathsf{L}$ então *qualquer que seja* o Provador P
$$\mathbb{P}[\,(\mathsf{V},\mathsf{P})(w) = 1\,] \leqslant \frac{1}{3};$$

além disso, os algoritmos trocam no máximo um número polinomial em $|w|$ de mensagens e a *aleatoriedade é privada*, ou seja, P não tem acesso aos bits aleatórios de V e vice-versa.

IP — *Interactive Proof* — é a classe de linguagens que admitem um sistema interativo de prova.

IP(k) é a subclasse de linguagens em IP que admitem um sistema interativo de prova em k rodadas.

A classe IP é invariante com respeito às seguintes modificações na definição: a probabilidade de erro é arbitrária e poderia ser qualquer constante positiva, na completude, assumir erro com probabilidade zero. Não provaremos esses fatos aqui, o leitor pode consultar Arora e Barak (2009, seção 8.3). Ademais, o Provador probabilístico não acrescenta poder ao modelo com respeito ao reconhecimento de linguagens, poderíamos assumir P determinístico e de complexidade de espaço[4] polinomial.

[4]Um máquina de Turing tem complexidade de espaço $\mathsf{S}(n)$ se $\mathsf{S}(n)$ é um limitante superior para a posição mais a direita na fita que é lida ou escrita em qualquer computação com instância de tamanho n.

Convencionamos que um protocolo interativo é escrito como

Entrada: aqui descrevemos a entrada comum as duas partes.

V: aqui estão as computações de V;

V→P: aqui estão as mensagens enviadas de V para P;

P: aqui, são as computações de P;

P→V: aqui, são as mensagens enviadas de P para V;

O exemplo abaixo mostra um problema computacional que está na classe IP mas não se sabe se está na classe NP, o que é um indício de que com a aleatoriedade o modelo interativo vai além de NP.

Exemplo 4.15 (NONISO ∈ IP). Os grafos a seguir são sobre o mesmo conjunto V de vértices, fixamos $V = \{1, 2, \ldots, n\}$ e um isomorfismo é uma permutação σ do conjunto $\mathbb{S}_n$ de todas as permutações de $\{1, 2, \ldots, n\}$ de modo que se G é um grafo então $\sigma(G)$ é o grafo com arestas $\{\{\sigma(u), \sigma(v)\} : \{u, v\} \in E(G)\}$. Definimos a linguagem formada por pares de grafos não isomorfos

$$\text{NONISO} := \{\langle G, H\rangle : G \text{ e } H \text{ não são grafos isomorfos}\}.$$

Não é sabido se NONISO está em NP, de modo que um provador não conhece um certificado efetivo que possa ser enviado a um verificador.

Problema 12. NONISO ∈ NP?

Apesar disso, em 2 rodadas um Provador consegue convencer um Verificador probabilístico desse fato. O Verificador escolhe ao acaso um dos dois grafos e envia ao Provador um cópia isomorfa do grafo escolhido, o Provador tem que descobrir qual foi o grafo escolhido, o que só é possível se os grafos da entrada não forem isomorfos. O protocolo é: com entrada G_0 e G_1; V sorteia $i \in \{0, 1\}$ e sorteia uma permutação π, envia $\pi(G_i)$; o poderoso P testa se $\pi(G_i)$ é isomorfo a G_0 ou a G_1 e envia o que descobriu $j \in \{0, 1\}$; se $i = j$ então V aceita, senão rejeita. Se G_0 não é isomorfo a G_1 (a entrada pertence a linguagem), então existe um Provador que descobre qual isomorfismo o Verificador construiu, por exemplo testando todas as permutações sobre os n vértices, e envia o bit correto para o Verificado, que aceita. Agora se G_0 é isomorfo a G_1 (a entrada não pertence a linguagem) o Provador não consegue descobrir qual gerou $\pi(G_i)$ de modo que o melhor que ele pode fazer é chutar um valor. Nesse caso, o Verificador aceita ou não com probabilidade 1/2. Esse protocolo executado duas vezes ("em paralelo") diminui a probabilidade de erro:

Entrada: os grafos G_0 e G_1.

V: $i \leftarrow_R \{0,1\}$; $j \leftarrow_R \{0,1\}$;
$\sigma \xleftarrow{R} \mathbb{S}_n$;$\pi \xleftarrow{R} \mathbb{S}_n$;
$H \leftarrow \sigma(G_i)$; $J \leftarrow \pi(G_j)$;

V→P: H, J;

P: se G_0 não é isomorfo a H, então $d \leftarrow 1$, senão
se G_1 não é isomorfo a H, então $d \leftarrow 0$, senão
$d \xleftarrow{R} \{0,1\}$;

se G_0 não é isomorfo a J, então $e \leftarrow 1$, senão
se G_1 não é isomorfo a J, então $e \leftarrow 0$, senão
$e \xleftarrow{R} \{0,1\}$;

P→V: d, e;

V: se $(i,j) = (d,e)$ então responda 1, senão responde 0.

Se os grafos G_0 e G_1 não são isomorfos, então o Provador pode sempre distinguir o caso em que H é isomorfo a G_0 do caso em que H é isomorfo a G_1, o mesmo vale para J, e sempre acertar os valores de d e e. Nesse caso o verificador responde 1 (completude *sem erro*).

Se os grafos são isomorfos então mesmo com um grande poder computacional o Provador não sabe distinguir H e J de G_0 ou G_1 de modo que d e e são sorteados pelo Provador. O Verificador responderá errado se $d = i$ e $e = j$, o que ocorre com probabilidade no máximo 1/4.

Notemos que a tarefa do Provador durante a execução é testar isomorfismo entre grafos, o que não sabemos fazer de forma eficiente, entretanto, P não tem restrição de tempo, ele pode testar todas as $n!$ permutações possíveis para descobrir o isomorfismo. ◊

O exemplo acima nos mostra que NONISO $\in$ IP mas, como já dissemos, ainda não sabemos responder se NONISO $\in$ NP, mais que isso, não sabemos se a inclusão NP $\subset$ IP é própria.

Problema 13. NP $\subsetneq$ IP?

A linguagem

$$\text{ISO} := \{\langle G, H\rangle : G \text{ e } H \text{ são grafos isomorfos}\}$$

por sua vez, está em NP pois um isomorfismo é um certificado curto que atesta pertinência na linguagem. Não é sabido se ISO é uma linguagem NP-completa. Se ISO for NP-completa, então NONISO será coNP-completa e chegaríamos a uma resposta afirmativa para o problema

Problema 14. coNP $\subset$ IP(2)?

É sabido (Boppana, Håstad e Zachos, 1987) que se o problema 14 for respondido com sim, $\mathsf{coNP} \subset \mathsf{IP}(2)$, então a Hierarquia Polinomial colapsa no segundo nível.

A classe IP contém a hierarquia polinomial (Lund et al., 1992) e o que foi surpresa para os pesquisadores é o fato de que IP é igual a classe PSPACE — *Polynomial Space* — a classe das linguagens que podem ser decididas por algoritmos com complexidade de espaço polinomial (Shamir, 1992; Shen, 1992). Assim o que sabemos é que

$$\mathsf{P} \subset \mathsf{NP} \subset \mathsf{PH} \subset \mathsf{IP} = \mathsf{PSPACE}.$$

Exemplo 4.16 (*não-resíduo quadrático está em* IP). Seja p um primo. Lembremos que a é um resíduo quadrático modulo p se para algum inteiro x temos $x^2 \equiv a \pmod{p}$. A linguagem definida pelos pares (a,p) tais que p é primo e a é um resíduo quadrático modulo p está em NP pois uma raiz quadrada de a é um certificado que pode ser verificado de modo eficiente, assim como a primalidade de p. Por outro lado, a linguagem definida pelos pares (a,p) tais que p é primo e a *não* é um resíduo quadrático modulo p não se sabe se está em NP, mas tem uma prova interativa como mostra o seguinte protocolo:

Entrada: um par de inteiros (a,p), p primo.

V: $r \leftarrow_{\mathrm{R}} \{1,\ldots,p-1\}$;
$b_1 \leftarrow_{\mathrm{R}} \{0,1\}$; $b_2 \leftarrow_{\mathrm{R}} \{0,1\}$;
Para cada $i \in \{1,2\}$,
se $b_i = 0$, então $w_i \leftarrow r^2 \bmod p$,
senão $w_i \leftarrow ar^2 \bmod p$;

V→P: w_1, w_2;

P: Para cada $i \in \{1,2\}$,
se w_i é resíduo quadrático, então $c_i \leftarrow 0$,
senão $c_i \leftarrow 1$;

P→V: c_1, c_2;

V: se $(c_1,c_2) = (b_1,b_2)$, então responde 1, senão responde 0.

Se a é resíduo quadrático então ar^2 também é um resíduo quadrático então w_1 e w_2 serão resíduos quadráticos, portanto $c_1 = c_2 = 0$. Como o Provador não conhece os bits b_1 e b_2 a probabilidade de que $(c_1,c_2) = (b_1,b_2)$ é 1/4. Agora, se a não é resíduo quadrático então ar^2 também não é, enquanto que r^2 é resíduo quadrático, portanto o Provador consegue distinguir corretamente o bit sorteado pelo Verificador e a resposta, nesse caso, é sempre 1. ◊

4.3.2 SISTEMAS DE PROVA COM BITS ALEATÓRIOS PÚBLICOS

Um fato crucial para o sucesso dos protocolos de comunicação nos dois sistemas de prova dos exemplos acima é que o Provador não conhece os bits sorteados pelo Verificador. Se permitimos que o Verificador envie ao Provador os bits aleatórios, então temos um *sistemas de provas de bits públicos* de k rodadas que dá origem à hierarquia de classes de complexidade $\mathtt{AM}(k)$ — *Arthur–Merlin proofs* — e $\mathtt{MA}(k)$ — *Merlin–Arthur proofs*.

O protocolo AM foi apresentado em Babai (1985) com o objetivo de construir uma versão aleatorizada de NP, e que estivesse "logo acima" de NP, para acomodar certas linguagens. Arthur, um ser humano com suas limitações, refere-se ao Verificador e Merlin, um mago poderoso, ao Provador. Ambas as classes são subclasses de IP obtidas quando restringimos as mensagens que o Verificador envia: todos, e somente os, bits aleatórios que ele usa. Qualquer outra informação que o Verificador precise enviar pode ser computada pelo poderoso Provador.

A diferença entre essas classes AM e MA reside em quem manda a primeira mensagem, o Verificador (Arthur) na classe $\mathtt{AM}(k)$, ou o Provador (Merlin) na classe $\mathtt{MA}(k)$. Em particular, em $\mathtt{MA}(1)$ Merlin envia a mensagem inicial (um certificado) e como não há mais mensagens Arthur toma sua decisão sem sorteio de bits, logo $\mathtt{MA}(1) = \mathtt{NP}$; em $\mathtt{AM}(1)$ Arthur sorteia bits, os envia ao Merlin que não comunica nada, e toma a sua decisão usando esses bits sorteados, logo $\mathtt{AM}(1) = \mathtt{BPP}$; em $\mathtt{MA}(2)$ o Merlin manda uma mensagem inicial, Arthur sorteia seus bits, os manda para Merlin que não tomará outra providência e usa esses bits para tomar sua decisão de aceite ou não.

Seguindo a tradição bibliográfica

$$\mathtt{AM} := \mathtt{AM}(2) \text{ e } \mathtt{MA} := \mathtt{MA}(2)$$

logo AM é a classe das linguagens com prova interativa onde o Verificador manda uma mensagem com bits aleatórios e o Provador responde. Para $p(n)$ uma função limitada polinomialmente, com $p(n) \geqslant 2$, Babai (1985) mostrou que $\mathtt{AM}(p(n)) = \mathtt{AM}(p(n)+1) = \mathtt{MA}(p(n)+1)$em particular, $\mathtt{AM} = \mathtt{AM}(k) = \mathtt{MA}(k+1)$ para qualquer $k \geqslant 2$ fixo. Ainda, Goldwasser e Sipser (1986) mostram a segunda inclusão de $\mathtt{AM}(p(n)) \subset \mathtt{IP}(p(n)) \subset \mathtt{AM}(p(n)+2)$logo, reunindo várias informações que temos até aqui, podemos escrever

$$\mathtt{NP} \cup \mathtt{BPP} \subset \mathtt{MA} \subset \mathtt{AM} \subset \mathtt{AM}(\mathtt{poly}) = \mathtt{IP} = \mathtt{PSPACE}$$

em que

$$\mathtt{AM}(\mathtt{poly}) = \mathtt{MA}(\mathtt{poly}) := \bigcup_{k>0} \mathtt{AM}(n^k).$$

PROTOCOLO COM BITS ALEATÓRIOS PÚBLICOS PARA NONISO Fixamos o conjunto de vértices dos grafos em $V = \{1, 2, \ldots, n\}$. Sejam G_0 e G_1 dois grafos. A quantidade de grafos isomorfos à G_i é

$$|I(G_i)| := \frac{n!}{|\mathrm{Aut}(G_i)|} \tag{4.6}$$

em que $\mathrm{Aut}(G_i)$ é o conjunto de todas as permutações $\pi\colon V \to V$ que definem um isomorfismo de G_i em G_i. Esse conjunto munido da composições de funções define um grupo chamado de grupo dos automorfismos de G_i (o qual é subgrupo do grupo de todas as permutações, logo (4.6) é inteiro, pelo Teorema de Lagrange).

Por exemplo, no caso do grafo $G := (\{1,2,3\},\{12\})$ com três vértices e uma aresta, se $\pi(1) = 2$, $\pi(2) = 1$, $\pi(3) = 3$, então π é um isomorfismo de G em G. Agora se se $\pi'(1) = 3$, $\pi'(3) = 1$, $\pi(2) = 2$, então π' é um isomorfismo de G em $\pi'(G) := (\{1,2,3\},\{13\})$ que é diferente de G. No caso do triângulo, denotado C_3, toda bijeção define um isomorfismo de C_3 em C_3, portanto $|\mathrm{Aut}(C_3)| = 6$. No caso do circuito com 4 vértices, o $C_4 := (\{1,2,3,4\},\{12,23,34,14\})$, das 24 permutações apenas 8 definem automorfismos de C_4, portanto há 3 grafos distintos sobre $\{1,2,3,4\}$ isomorfos ao C_4. Agora, cada grafo em $I(G_i) := \{\pi(G_i)\colon \pi \text{ permutação}\}$ ocorre $|\mathrm{Aut}(G_i)|$ vezes dentre todas $n!$ permutações, logo $|I(G_i)|$ é dado pela equação (4.6).

Seja S o conjunto dos grafos sobre V que são isomorfos a algum G_i, para $i = 0, 1$,

$$S = \{(H,\pi)\colon H \equiv G_0 \text{ ou } H \equiv G_1 \text{ e } \pi \in \mathrm{Aut}(H)\}.$$

A ideia do protocolo é separar (com probabilidade razoavelmente grande) o caso em que G_0 e G_1 não são isomorfos do caso que são isomorfos usando o tamanho de S pois

$$\text{se } G_0 \not\equiv G_1 \text{ então } |S| = 2n! \tag{4.7}$$

$$\text{se } G_0 \equiv G_1 \text{ então } |S| = n! \tag{4.8}$$

a afirmação (4.8) segue imediatamente de (4.6) pois $|\mathrm{Aut}(H)| = |\mathrm{Aut}(G_i)|$. Para a afirmação (4.7), cada $H \equiv G_i$ contribui com $|\mathrm{Aut}(G_i)|$ pares e são $n!/|\mathrm{Aut}(G_i)|$ tais H's, no outro caso, como os grafos G_0 e G_1 não são isomorfos, são $2n!$ pares.

Se o Verificador gerar um grafo H então o Provador pode verificar se esse grafo é isomorfo a G_0 ou a G_1 e quando for o caso enviar o isomorfismo para o Verificador conferir. Repetindo esse procedimento nós observaríamos que no caso (4.7) o Verificador atesta isomorfismo duas vezes mais que no caso (4.8), permitindo uma conclusão a respeito do grafos de entrada. O problema desse protocolo é que para termos uma probabilidade de erro limitada por uma constante o número de repetições tem que ser exponencial, pois o número total de grafos sobre $\{1,2,\ldots,n\}$ é $2^{\binom{n}{2}}$ e $|S|/2^{\binom{n}{2}}$ é exponencialmente pequeno.

Uma solução é usar uma ferramenta introduzida no exemplo 3.49, que estudaremos com mais profundidade na seção 6.2.2. Lembremos do exemplo 3.49, página 164, ao sortear uma função de hash de uma família de funções de hash $\mathcal{H} \subset M^U$ 2-a-2 independentes a probabilidade p com que exista $x \in S \subset U$ tal que $h(x) = 0$ é

$$\frac{|S|}{|S|+m} \leqslant p \leqslant \frac{|S|}{m}.$$

Fixado um inteiro K, tome m a maior potência de 2 menor ou igual a $K/2$, e seja $h\colon U \to \{0,\ldots,m-1\}$ escolhida uniformemente de uma família de funções hash 2-a-2 independente. Então a probabilidade p de existir $x \in S$ tal que $h(x) = 0$ satisfaz $p \geqslant 2/3$, se $|S| \geqslant K$, e $p \leqslant 1/4$ se $|S| \leqslant K/16$.

Com isso, temos o conjunto S cujo tamanho difere por um fator $1/2$ dependendo de se os grafos são isomorfos ou não, equações (4.8) e (4.7) respectivamente, e para usar o resultado estabelecido no parágrafo anterior temos que amplificar este fator para $1/16$. Nós podemos fazer isso usando uma quádrupla $S' = S \times S \times S \times S$ de modo que

$$|S'| = \begin{cases} 16(n!)^4 & \text{se } G_0 \ncong G_1 \\ (n!)^4 & \text{se } G_0 \equiv G_1. \end{cases}$$

Com isso, o Provador afirma que $|S| \geqslant 16(n!)^4$; o Verificador escolhe uma função hash aleatória h de uma família independente aos pares e pede ao Provador um $x \in S$ tal que $h(x) = 0$. O Provador convence com probabilidade pelo menos $2/3$ se $G_0 \ncong G_1$, e no máximo $1/4$ se $G_0 \equiv G_1$.

Vejamos, agora, um protocolo de troca de mensagens com bits aleatórios públicos, devido a Goldwasser e Sipser, em que o objetivo do Provador é convencer o Verificador de que $|S| \geqslant K$ e caso $|S| \leqslant K/16$ o Verificador deve rejeitar com probabilidade de erro positiva. Notemos que se existe $x \in S$ tal que $h(x) = y$ então o Provador consegue determinar x e convencer o Verificador de que sua imagem é y. A única restrição em S é que a pertinência tem que ser determinada de forma eficiente. Essa estratégia do Provador convencer o Verificador de um limitante inferior no tamanho de um conjunto pode ser usada substituir moedas privadas por públicas em qualquer prova interativa. No exercício 6.34, página 329, é apresentada uma família de funções de hash que podem ser usadas nesse protocolo.

Entrada: $S \subset U$ e um natural K.

V: $h \xleftarrow{R} \{f\colon U \to \{0,1\}^{\lceil \log(K) \rceil} : f \text{ satisfaz (3.34)}\}$;
$y \leftarrow_R \{0,1\}^{\lceil \log(K) \rceil}$;

V→P: h, y;

P: determina, se possível, $x \in S$ tal que $h(x) = y$;

P→V: x e um certificado de que $x \in S$;

V: se $h(x) = y$ e o certificado valida a pertinência $x \in S$ então responde 1, senão responde 0.

Como ISO, o problema de isomorfismo de grafos, está em NP, concluímos que ISO $\in$ NP $\cap$ coAM. Acredita-se que AM seja uma classe que inclua propriamente NP e também que NP-completos não

estejam em coAM. Pode-se provar que se NP $\subset$ coAM então a hierarquia polinomial colapsa. Por esta razão, acreditas-se que ısso não seja NP-completo.

4.4 CRIPTOGRAFIA

A Criptografia moderna talvez seja a disciplina que mais se beneficia das técnicas probabilísticas e dos algoritmos aleatorizados e mesmo que venhamos a saber que aleatoriedade não acrescenta poder de computação aos algoritmos a noção atual de segurança criptográfica assegura um papel importante para a aleatoriedade no projeto de sistemas de criptografia seguros.

Vimos na seção 1.2.1 que um sistema de codificação $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ tem sigilo perfeito se as variáveis aleatórias $E_K(P)$ e $E_K(P')$, com sorteio nas chaves $K \in_R \mathcal{K}$, são identicamente distribuídas para quaisquer textos P e P'. Nos requisitos atuais de segurança, dito *segurança semântica*, as distribuições devem ser *indistinguíveis para algoritmos eficientes*, ou seja, a abordagem moderna para a construção de sistemas de criptografia é baseada na *efetividade* computacional de se extrair informação, ao invés de *possibilidade* de se extrair informação. Nesse sentido, um algoritmo determinístico de criptografia não é seguro (Goldwasser e Micali, 1984). Sistemas probabilísticos foram introduzidos por Goldwasser e Micali (1984), que mostraram que, assumindo a hipótese de existir problema computacionalmente difícil, adversário polinomialmente limitado não obtém informação sobre o texto a partir do texto cifrado para qualquer que seja a distribuição sobre o espaço de textos.

A hipótese da existência de problemas computacionalmente difíceis, em NP $\not\subset$ BPP por exemplo, é um princípio importante em Criptografia. Uma linguagem em NP $\not\subset$ BPP significa que temos um problema para o qual não é conhecido um algoritmo eficiente mas, conhecido um fato adicional (certificado), o problema tem um Verificador eficiente. A dificuldade de se resolver computacionalmente tal problema é usado como princípio para garantir a segurança de protocolos criptográficos (no sentido de que quebrar a codificação é computacionalmente equivalente a resolver o problema) e a decodificação é viável somente de posse de um segredo (o certificado). Notemos que NP $\not\subset$ BPP, implica em P $\neq$ NP, entretanto esse último resultado de pior caso (P $\neq$ NP) não é suficiente para os propósitos da Criptografia pois não garante que o problema computacional no qual se baseia um sistema criptográfico seja difícil na maioria das instâncias, mais ainda, o pior caso pode ocorrer em instâncias raras do problema e na maioria da outras instâncias o problema pode ser fácil.

Uma primitiva da Criptografia moderna para implementar o princípio descrito acima é chamada de função unidirecional (*one-way function*, em inglês). Uma função unidirecional pode ser computada por algoritmo eficiente porém quase sempre não pode ser invertida por algoritmo eficiente, isto é, qualquer algoritmo probabilístico de tempo polinomial que dado $f(x)$ busca por y tal que $f(y) = f(x)$ terá sucesso apenas com probabilidade insignificante. Especialmente útil é a classe dessas

funções que podem ser invertidas facilmente quando temos posse de algum segredo, essas funções são conhecidas por *alçapão* (em inglês *trapdoor function*).

Outra primitiva da Criptografia moderna são os geradores pseudoaleatórios. Se o uso de aleatoriedade no projeto de sistemas criptográficos é indispensável, a questão importante que surge é "quanto aleatório" devem ser esses bits. Um gerador pseudoaleatório é um algoritmo que a partir de uma sequência binária genuinamente aleatória produz outra sequência binária, geralmente maior, que tenta imitar uma sequência aleatória. O modo tradicional de apresentar alguma garantia da qualidade de um gerador pseudoaleatório é em função desse ser aceito por testes estatísticos *ad-hoc* (veja e.g., Knuth, 1981, o vol. 2 do TAOCP). O uso de geradores pseudoaleatórios em criptografia levou os pesquisadores a procurarem uma definição robusta para geradores, que garantam que a utilização deles mantenha a segurança dos sistemas de criptografia contra quaisquer *testes computacionalmente viáveis*.

4.4.1 FUNÇÃO UNIDIRECIONAL

Uma função f é chamada unidirecional se entre ela e sua inversa só uma é computacionalmente fácil, isto é, existe um algoritmo eficiente que com entrada x computa $f(x)$, porém, dado $f(x)$, para x uniformemente sorteado, é computacionalmente inviável encontrar, com probabilidade não desprezível, uma pré-imagem de $f(x)$. Formalmente, uma **função unidirecional** é uma função $f\colon \{0,1\}^* \to \{0,1\}^*$ tal que

- $|f(x)|$ e $|x|$ são polinomialmente relacionados[5];
- existe um algoritmo de tempo polinomial que computa $f(x)$ para todo x;
- para todo algoritmo probabilístico de tempo polinomial M, para todo inteiro $d > 0$

$$\mathbb{P}\Big[\mathsf{M}(f(\mathsf{X})) \in f^{-1}(f(\mathsf{X}))\Big] < \frac{1}{n^d}$$

  para todo n suficientemente grande, em que a probabilidade é sobre a escolha aleatória $\mathsf{X} \in_{\mathrm{R}} \{0,1\}^n$ e sobre os bits aleatórios usados por M.

Não sabemos se tais funções existem.

Problema 15. Funções unidirecionais existem?

Uma resposta afirmativa para o problema tem consequências profundas como (*i*) $\mathsf{NP} \not\subset \mathsf{BPP}$ e, consequentemente, $\mathsf{NP} \not\subset \mathsf{P/poly}$ e $\mathsf{P} \neq \mathsf{NP}$, (*ii*) a existência de geradores pseudoaleatórios (seção 4.4.2),

[5] Isso é uma exigência técnica para garantir que inversão não seja inviável por razões triviais, como é o caso de $|x|$ ser exponencialmente maior que $|f(x)|$.

(*iii*) a existência de provas com conhecimento zero (seção 4.4.4) para todo problema em NP, (*iv*) a existência de sistemas criptográficos de chave privada e de assinatura digital seguros, (*v*) a existência de esquemas de empenho de bits (pág. 231), e (*vi*) a não existência de provas naturais para $\mathsf{P} \neq \mathsf{NP}$ (seção 4.5).

Vejamos alguns candidatos a função unidirecional. A função

$$\textsc{mult}(x,y) := \begin{cases} x \cdot y & \text{se } x, y \neq 1 \\ 1 & \text{caso contrário,} \end{cases}$$

podemos computar de modo eficiente e nem sempre é difícil de inverter, embora possa ser em alguns casos. Notemos que $(1, xy) \notin f^{-1}(f(x,y))$. O algoritmo M que computa

$$\mathsf{M}(z) = \begin{cases} (2, z/2) & \text{se } z \text{ é par} \\ (0,0) & \text{caso contrário} \end{cases}$$

tem probabilidade de sucesso é pelo menos 3/4 se $z = xy$ com $x, y \in_{\mathrm{R}} \{0,1\}^n$. Agora, se definimos $\textsc{multP}_n(x,y)$ como o produto de dois primos de tamanho $n/2$ temos uma família unidirecional sob a *hipótese de fatoração* ser um problema difícil.

Observação 4.17 (*hipótese de fatoração*). A hipótese que assumimos é que para qualquer algoritmo probabilístico de tempo polinomial A vale que se p e q são dois primos aleatórios de comprimento n e $\mathsf{N} = pq$ então

$$\mathbb{P}[\,\mathsf{A}(\mathsf{N}) \in \{p,q\}\,] < \frac{1}{n^d}$$

para toda constante $d > 0$, onde a probabilidade é sobre p, q e as escolhas aleatórias do algoritmo A. É possível mostrar que o problema de fatorar o produto de dois primos aleatórios é equivalente a computar *todos* os fatores primos de inteiros aleatórios

> *Problema* 16 (Fatoração). Dado $\mathsf{N} \in \mathbb{N}$, existe algoritmo de tempo polinomial em $\log \mathsf{N}$ que determina todos os divisores primos de N?

e equivalente a determinar uma representação do produto de dois inteiros aleatórios usando dois números de comprimento no máximo $n/2$. Com isso, podemos definir $\textsc{mult}(x)$ como a multiplicação dos números representados pela metade mais significativa dos bits de x pela metade menos significativa dos bits de x, como possível candidata a função unidirecional.

O problema de extrair raízes também é difícil.

> *Problema* 17 (Raiz quadrada módulo n). Dados um inteiro n composto e a um quadrado módulo n, determinar de modo eficiente x tal que $x^2 \equiv a \pmod{n}$.

É possível calcular $f(x,k,\mathsf{N}) = x^k \bmod \mathsf{N}$ de modo eficiente (algoritmo 7, página 59) mas sob certas hipóteses nos parâmetros k e N é difícil inverter f. Por exemplo, é candidata a unidirecional

$$\mathrm{Rabin}_{\mathsf{N}}(x) := x^2 \bmod \mathsf{N}, \quad \text{para todo } x \in \{1, 2, \ldots, \mathsf{N}\}$$

que sabemos que inverter dada a fatoração de N, conhecido os fatores primos podemos usar o Teorema do Resto Chinês para reduzir o problema de obter uma raiz quadrada módulo N para obter raízes quadradas módulo cada fator primo (exercício 2.75, página 111). Se $N = p \cdot q$ com $p, q > 2$ primos $\text{Rabin}_N(x)$ é uma função 4-para-1 no $\mathbb{Z}_N$. Pode-se provar que extrair raiz quadrada módulo N é computacionalmente equivalente a fatorar N (veja o exercício 4.35 no final do capítulo). Assumindo que a fatoração é difícil fica inviável, dados N e $\text{Rabin}_N(x)$ encontrar um pré-imagem. Por outro lado, dada a fatoração de N, é factível encontrar todas as 4 pré-imagens de $\text{Rabin}_N(x)$ (exercício 2.76, página 113). Portanto, supondo que a fatoração seja intratável, o acima produz uma coleção de funções unidirecionais de alçapão. A função Rabin definida acima pode ser generalizada por

$$\text{RSA}_{N,e}(x) := x^e \bmod N, \qquad \text{para todo } x \in \{1, 2, \ldots, N\}$$

e ambas são funções unidirecionais *alçapão* pois podem ser invertidas de modo eficiente se é conhecida a fatoração de N.

Exemplo 4.18 (*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[6] temos[7] 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 $\mathbb{Z}_n^*$ conhecido como raiz primitiva módulo n, como vimos na seção 2.2.4. Por exemplo, a função $\iota\colon \mathbb{Z}_{11}^* \to \mathbb{Z}_{10}$ dada na tabela 4.1 (usando só os representantes das classes dos resíduos) satisfaz $\iota(a \cdot_n b) = \iota(a) +_n \iota(b)$. Todas as raízes primitivas módulo 11 estão descritas no exemplo 2.26, página 76.

$\mathbb{Z}_{11}^*$	$\mathbb{Z}_{10}$
$1 (= 2^0)$	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 4.1: função logaritmo discreto para a raiz primitiva 2.

[6] $(\mathbb{Z}_n^*, \cdot_n)$ 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, cap. 4).

[7] $\varphi(n) := \left|\{a \in \mathbb{Z} \colon 0 \leqslant a < n \text{ e } \text{mdc}(a, n) = 1\}\right| = n \cdot \prod_{i=1}^{k}\left(1 - \frac{1}{p_i}\right)$, onde $n = p_1^{a_1} \cdots p_k^{a_k}$, é a função de Euler (pág. 78).

A função $f(p,\alpha,x) = (p,\alpha,\alpha^x \bmod p)$, com p primo, α uma raiz primitiva 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* 18 (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 $3^{789} \bmod 1999$ porém não sabemos resolver eficientemente a equação $3^x \equiv 1452 \pmod{1999}$. Portanto, é candidata a função unidirecional. ◊

Um papel chave das funções unidirecionais é dado no seguinte resultado cuja demonstração pode ser encontrada em Arora e Barak (2009, teo. 9.6, cap. 9). A definição de *sistema criptográfico computacionalmente seguro* e dada no exercício 4.19 abaixo, na página 218.

TEOREMA *Se existe função unidirecional então para todo $d > 0$ existe um sistema criptográfico computacionalmente seguro que usa chaves de tamanho n para codificar mensagens de tamanho n^d.*

4.4.2 GERADORES PSEUDOALEATÓRIOS SEGUROS

Geradores pseudoaleatórios são algoritmos determinísticos eficientes que tomam uma sequência aleatória curta, usualmente chamada de *semente*, e a *estica*, isto é, devolve um sequência mais longa e que deve se comportar como uma sequência aleatória. Uma formulação genérica de geradores pseudoaleatórios especifica três aspectos fundamentais: (1) a medida de estiramento do gerador; (2) a classe dos algoritmos que não distinguem a sequência longa gerada de uma genuinamente aleatória, isto é, a complexidade dos algoritmos que o gerador supostamente engana; (3) a sua própria complexidade computacional.

Seja p uma função polinomial. As distribuições de $X_m \in_D \{0,1\}^{p(m)}$ e $Y_m \in_E \{0,1\}^{p(m)}$ são ditas **computacionalmente (eficientemente) indistinguíveis** se para toda constante $d > 0$ e todo algoritmo probabilístico A de tempo polinomial em m

$$\left| \mathop{\mathbb{P}}_{X_m \in_D \{0,1\}^{p(m)}} [A(X_m) = 1] - \mathop{\mathbb{P}}_{Y_m \in_E \{0,1\}^{p(m)}} [A(Y_m) = 1] \right| \leqslant \frac{1}{m^d}$$

para todo m suficientemente grande, em que a medida de probabilidade é sobre as distribuições das sequências e as escolhas internas do algoritmo probabilístico A. Como a diferença de probabilidades na resposta 1 é quantidade superpolinomialmente pequena, seria necessário um número superpolinomial de rodadas para distinguir os bits gerados por G dos verdadeiramente aleatórios.

Seja $\ell\colon \mathbb{N} \to \mathbb{N}$ uma função com $\ell(n) > n$ e ℓ computável em tempo polinomial. Um **gerador**

pseudoaleatório seguro contra algoritmos eficientes e medida de estiramento ℓ é um algoritmo (determinístico) de tempo polinomial que computa $G\colon \{0,1\}^* \to \{0,1\}^*$ de modo que

1. $|G(w)| = \ell(|w|)$
2. $G(X_m)$, para $X_m \in_R \{0,1\}^m$, é computacionalmente indistinguível de $X_{\ell(m)} \in_R \{0,1\}^{\ell(m)}$.

Notemos que na definição permitimos que os geradores pseudoaleatórios sejam seguros mesmo contra adversários com poder computacional ligeiramente maior quando comparado aos geradores.

O algoritmo A na definição pode ser um algoritmo que decide a aleatoriedade de uma sequência. Pode, por exemplo, ser um teste estatístico como $A(x) = 1$ se e só se o número de ocorrências do padrão 11 é no máximo 6 desvios padrão da média ou, $A(x) = 1$ se e só se a maior subsequência de 1 consecutivos é no máximo $6\log_2(n)$, ou ambos os testes. Pode ainda usar alguma informação conhecida a respeito do gerador, por exemplo, se G é um gerador tal que para 3/4 das entradas o primeiro bit da saída é 1, então definimos o teste A que com entrada x responde 1 se, e só se, o primeiro bit de x é 1 e

$$\left| \mathbb{P}_{X \in_R \{0,1\}^m}[A(G(X)) = 1] - \mathbb{P}_{Y \in_R \{0,1\}^{\ell(m)}}[A(Y) = 1] \right| = \left| \frac{3}{4} - \frac{1}{2} \right| = \frac{1}{4}.$$

Geradores pseudoaleatórios seguros não podem existir incondicionalmente e uma condição que garante a existência de geradores é a existência de funções unidirecionais. Não provaremos o seguinte resultado (Håstad et al., 1999).

TEOREMA *As seguintes afirmações são equivalentes:*

1. *existe função unidirecional;*
2. *existe gerador pseudoaleatório seguro com* $\ell(n) = n + 1$*;*
3. *para toda constante* $c > 0$*, existe gerador pseudoaleatório seguro com* $\ell(n) = n^c$.

Exercício 4.19. Um sistema criptográfico definido pelas funções de codificação/decodificação (E, D) que usa chaves de n bits para codificar mensagens de comprimento $\ell(n)$ é *computacionalmente seguro* se para quaisquer textos $t_0, t_1 \in \{0,1\}^{\ell(n)}$ vale que $E_k(t_0)$ é computacionalmente indistinguível de $E_k(t_1)$ para as chaves k escolhidas uniformemente em $\{0,1\}^n$.

Prove que se existe gerador pseudoaleatório seguro com estiramento ℓ então existe um sistema criptográfico computacionalmente seguro que usa chaves de tamanho n para codificar mensagens de tamanho $\ell(n)$. (*Dica*: a ideia é imitar o *one-time pad* $E_k(x) = x \oplus G(k)$ e $D_k(y) = y \oplus G(k)$. Prove que $E_{X_n}(x)$, para X_n uniforme, é computacionalmente indistinguível de $X_{\ell(n)}$, também uniforme).

IMPREVISIBILIDADE Se x é uma sequência de bits então denotamos por $x\restriction_i$ a restrição de x aos seus primeiros i bits, isto é, se $x = (x_1, x_2, \ldots, x_n)$ então $x\restriction_i = (x_1, x_2, \ldots, x_i)$. Um vetor aleatório $Z = (Z_1, \ldots, Z_n) \in \{0,1\}^n$ é **imprevisível** para o próximo bit se para qualquer algoritmo A probabilístico de tempo polinomial

$$\mathop{\mathbb{P}}_{i \in_R \{1,2,\ldots,n\}}[A(Z_1, \ldots, Z_{i-1}) = Z_i] \leqslant \frac{1}{2} + \frac{1}{n^d}$$

para todo $d > 0$.

É natural esperarmos de um gerador pseudoaleatório seguro que os bits gerados não possam ser previstos, caso contrário o algoritmo previsor pode ser usado como um teste de aleatoriedade. De fato, se G é um gerador pseudoaleatório e existe um algoritmo probabilístico de tempo polinomial A tal que

$$\mathop{\mathbb{P}}_{\substack{X \in_R \{0,1\}^m \\ i \in_R \{1,\ldots,\ell(m)\}}} [A(G(X)\restriction_{1,\ldots,i-1}) = G(X)_i] > \frac{1}{2} + \frac{1}{m^d}$$

para algum $d > 0$ e todo m suficientemente grande, então podemos projetar o algoritmo B que com entrada x decide se A prevê o próximo ésimo bit, isto é, computa

$$B(x) = \begin{cases} 1, & \text{se } A(x\restriction_{1,\ldots,i-1}) = x_i \\ 0, & \text{caso contrário} \end{cases}$$

e temos que se $Y \in_R \{0,1\}^{\ell(m)}$ então $B(Y) = 1$ com probabilidade $1/2$ e para $X \in_R \{0,1\}^m$ a probabilidade de $B(G(X)) = 1$ é maior que $1/2 + 1/m^d$, logo

$$\left| \mathop{\mathbb{P}}_{Y \in_R \{0,1\}^{\ell(m)}} [B(Y) = 1] - \mathop{\mathbb{P}}_{X \in_R \{0,1\}^m} [B(G(X)) = 1] \right| > \frac{1}{m^d}$$

contrariando o fato de $G(X)$ ser computacionalmente indistinguível de Y.

Yao (1982) mostrou a recíproca desse fato, concluindo que um algoritmo que gera uma sequência de bits é um gerador pseudoaleatório se e somente se a sequência gerada é imprevisível para o próximo bit. Para a recíproca, suponhamos, agora, que A seja um algoritmo probabilístico de tempo polinomial que prova que a sequência $G(X)$ não é pseudoaleatória, ele responde 1 com mais frequência para sequências $G(X)$ do que para sequências X com distribuição uniforme. Vamos provar a previsibilidade. Para $\varepsilon > 0$ pequeno, assumimos que

$$\mathop{\mathbb{P}}_{X \in_R \{0,1\}^m)} [A(G(X)) = 1] - \mathop{\mathbb{P}}_{Y \in_R \{0,1\}^{\ell(m)}} [A(Y) = 1] > \varepsilon. \tag{4.9}$$

Vamos usar A para prever bits de sequências geradas por G da seguinte maneira.

Consideremos o algoritmo C que recebe os bits $y_1, \ldots, y_{i-1}$, sorteia os bits $z_i, \ldots, z_{\ell(m)}$, computa $a := A(y_1, \ldots, y_{i-1}, z_i, \ldots, z_{\ell(m)})$. Se $a = 1$, o algoritmo pressupõe que seu chute para z_i está correto e

então responde z_i, senão responde esse bit invertido

$$C(y_1,\ldots,y_{i-1}) = \begin{cases} z_i, & \text{se } a = 1 \\ 1 - z_i, & \text{se } a = 0. \end{cases}$$

Vamos provar que segue de (4.9) que

$$\mathop{\mathbb{P}}_{\substack{X \in_R \{0,1\}^m \\ i \in_R \{1,\ldots,\ell(m)\} \\ B \in_R \{0,1\}^*}} [C(G(X)\restriction_{1,\ldots,i-1}, B) = G(X)_i] > \frac{1}{2} + \frac{\varepsilon}{\ell(m)},$$

portanto, não vale a imprevisibilidade.

Analisaremos o comportamento do algoritmo C descrito acima a partir das seguintes distribuições sobre $\{0,1\}^{\ell(m)}$. Para cada $j \in \{1,\ldots,\ell(m)\}$, $Z_j \in_R \{0,1\}$, e $X \in_R \{0,1\}^m$ e

$$\begin{aligned} D_0 &:= (Z_1, Z_2, \ldots, Z_{\ell(m)}) \\ D_1 &:= (G(X)_1, Z_2, \ldots, Z_{\ell(m)}) \\ D_2 &:= (G(X)_1, G(X)_2, \ldots, Z_{\ell(m)}) \\ &\vdots \\ D_j &:= (G(X)\restriction_{1,\ldots,j}, Z_{j+1}, \ldots, Z_{\ell(m)}) \\ &\vdots \\ D_{\ell(m)} &:= G(X). \end{aligned}$$

Se $p_i := \mathbb{P}[\, A(D_i) = 1 \,]$, para todo i, então por (4.9) temos

$$\varepsilon < p_m - p_0 = \sum_{j=0}^{\ell(m)-1} (p_{j+1} - p_j).$$

Portanto, existe um índice $i \in \{1,\ldots,m\}$ tal que

$$p_i - p_{i-1} > \frac{\varepsilon}{\ell(m)}.$$

Para estimar $\mathbb{P}[\, C(G(X)\restriction_{1,\ldots,i-1}) = G(X)_i \,]$, que é a probabilidade de C acertar na previsão de $G(X)_i$, analisamos $C(G(X)\restriction_{1,\ldots,i-1})$ e, pela construção do algoritmo C essa probabilidade é

$$\begin{aligned} &\frac{1}{2}\mathbb{P}[\, a = 1 \mid Z_i = G(X)_i \,] + \frac{1}{2}\mathbb{P}[\, a = 0 \mid Z_i = 1 - G(X)\restriction_i \,] = \\ &\frac{1}{2}\mathbb{P}[\, a = 1 \mid Z_i = G(X)_i \,] + \frac{1}{2} - \frac{1}{2}\mathbb{P}[\, a = 1 \mid Z_i = 1 - G(X)_i \,] \end{aligned}$$

entretanto, condicionado em $Z_i = G(X)_i$, C invoca A com a distribuição D_i, logo $\mathbb{P}[\, a = 1 \mid Z_i = G(X)_i \,] = p_i$, e não condicionando invoca A com a distribuição D_{i-1}, logo

$$\frac{1}{2}\mathbb{P}[\, a = 1 \mid Z_i = 1 - G(X)_i \,] + \frac{1}{2}\mathbb{P}[\, a = 1 \mid Z_i = G(X)_i \,] = \mathbb{P}[\, a = 1 \,] = \mathbb{P}[\, A(D_{i-1}) = 1 \,] = p_{i-1}$$

substituindo essas igualdades na dedução anterior

$$\begin{aligned}\mathbb{P}[\,C(G(X)\restriction_{1,\ldots,i-1}) = G(X)_i\,] &= \frac{1}{2}\mathbb{P}[\,a=1 \mid Z_i = G(X)_i\,] + \frac{1}{2} - \frac{1}{2}\mathbb{P}[\,a=1 \mid Z_i = 1 - G(X)_i\,] \\ &= \frac{1}{2}p_i + \frac{1}{2} - p_{i-1} + \frac{1}{2}p_i \\ &\geqslant \frac{1}{2} + p_i - p_{i-1} \\ &> \frac{1}{2} + \frac{\varepsilon}{\ell(m)}\end{aligned}$$

portanto, C prevê o próximo bit com boa probabilidade.

4.4.3 CRIPTOGRAFIA DE CHAVE PÚBLICA

Os sistemas criptográficos se encaixam em duas grandes classes, os *simétricos*, como o *one-time pad* e os *assimétricos*. Criptossistemas simétricos têm as chaves de codificação e decodificação trivialmente relacionadas (podem ser iguais) e supõem-se secretas. Criptossistemas assimétricos têm as chaves de codificação e decodificação diferentes, a chave de codificação e pode ser feita pública mas a chave $d = d(e)$ de decodificação é secreta *e não pode ser computada eficientemente* a partir das informações públicas.

Nessa seção veremos dois sistemas assimétricos conhecidos. A ideia aqui é mostrar como aleatorização faz parte desses sistemas, não abordaremos aspectos de eficiência de modo aprofundado e tampouco abordaremos os aspectos de segurança desses sistemas. Uma boa referência para esses temas e aspectos práticos é Menezes, Oorschot e Vanstone (1997).

TROCA DE CHAVES O protocolo Diffie–Hellman–Merkle (Diffie e Hellman, 1976) permite que duas partes entrem em acordo sobre uma chave secreta usando um canal de comunicação não seguro. Esse protocolo, publicado em 1976, foi um trabalho pioneiro que lançou as bases para a criptografia de chave pública. Suponha que sejam de conhecimento público um primo p e um gerador γ de $\mathbb{Z}_p^*$, o grupo multiplicativo dos resíduos módulo p. Alice e Bob estabelecem um *chave* (secreta) comum da seguinte forma

- Alice sorteia uniformemente $a \in \{2, 3, \ldots, p-2\}$, computa $A = \gamma^a \bmod p$ e manda-o para Bob;
- Bob, por sua vez, sorteia uniformemente $b \in \{2, 3, \ldots, p-2\}$, computa $B = \gamma^b \bmod p$ e manda-o para Alice;
- Alice computa B^a e Bob computa A^b e a chave *secreta* comum entre eles é

$$k = A^b \bmod p = B^a \bmod p = \gamma^{ab} \bmod p.$$

Todas as computações podem ser feitas com algoritmos de tempo polinomial em $\log p$. Se um espião, conhecendo γ e p, intercepta A e B então o problema é determinar a tal que $\gamma^a \bmod p = \mathsf{A}$, determinar b tal que $\gamma^b \bmod p = \mathsf{B}$, e com isso feito basta computar $k = \gamma^{ab} \bmod p$. A parte difícil aqui, que garante a segurança do protocolo, é que determinar a e b é o problema de computar o *logaritmo discreto*, ou seja, como vimos acima o problema é computar a inversa de uma função unidirecional.

Um problema nesse protocolo é que um observador malicioso pode assumir o papel de Alice para Bob e o de Bob para Alice e assim conhecer a chave para decodificar alguma mensagem. Isso pode ser evitado usando um protocolo de *assinatura digital* (descrito abaixo).

O SISTEMA ASSIMÉTRICO ELGAMAL Esse sistema foi proposto pelo egípcio Taher Elgamal (ElGamal, 1985). O protocolo usado para criar as chaves púbica e privada é o seguinte:

- sorteie um primo grande p e determine um gerador α de $\mathbb{Z}_p^*$;
- sorteie uniformemente $d \in \{2, 3, \ldots, p-2\}$ e compute $\mathsf{A} = \alpha^d \bmod p$;
- a *chave pública*, a ser divulgada, é a tripla (p, α, A) e a *chave privada* correspondente, que deve ser mantida em segredo, é d, ou seja, o logaritmo discreto de A com base α.

Para simplificar, vamos assumir que o universo dos textos é o $\mathbb{Z}_p$ e o espaço das cifras (textos codificados) é $\mathbb{Z}_p \times \mathbb{Z}_p$. A codificação (ou cifragem) é feita da seguinte maneira

- dado o texto P e chave pública (p, α, A);
- sorteie uniformemente $b \in \{1, \ldots, p-2\}$;
- compute $\mathsf{E}_{(p,\alpha,\mathsf{A})}(b, \mathsf{P}) := \big(\alpha^b \bmod p, \mathsf{A}^b \cdot \mathsf{P} \bmod p\big)$.

Esse é um protocolo probabilístico por causa do parâmetro b na função de codificação, o que acredita-se dificulta muito a criptoanálise. A decodificação que inverte essa codificação é feita do seguinte modo

- dado o texto cifrado (B, C) e considerando a chave privada d;
- compute $\mathsf{D}_d(\mathsf{B}, \mathsf{C}) = \mathsf{B}^{p-1-d} \cdot \mathsf{C} \bmod p$.

Dessa forma recupera-se o texto original

$$\begin{aligned}
\mathsf{D}_d\Big(\mathsf{E}_{(p,\alpha,\mathsf{A})}(b, \mathsf{P})\Big) &= \mathsf{D}_d\Big(\alpha^b \bmod p, \mathsf{A}^b \cdot \mathsf{P} \bmod p\Big) \\
&= (\alpha^b \bmod p)^{p-1-d} \cdot (\alpha^d \bmod p)^b \cdot \mathsf{P} \bmod p \\
&= \alpha^{(p-1)b} \cdot \mathsf{P} \bmod p = \mathsf{P}.
\end{aligned}$$

Essas funções são computadas eficientemente.

Por exemplo, seja $(23, 7, 4)$ a chave pública, e 6 a chave privada. Para codificar o texto 7, sorteamos $b = 3$ e enviamos $(21, 11)$. O receptor computa $21^{23-1-6} \cdot 11 \bmod 23 = 7$.

A segurança do Elgamal é baseada na dificuldade computacional do *Problema de Diffie–Hellman*:

> *Problema* 19 (Diffie–Hellman). Existe um algoritmo eficiente que, dados p um primo, α uma raiz primitiva módulo p e os elementos $\alpha^a \bmod p$ e $\alpha^b \bmod p$, compute $\alpha^{ab} \bmod p$?

Conhecidos um texto cifrado $(\alpha^b \bmod p, (\alpha^d)^b \cdot \mathrm{P} \bmod p)$ e a chave pública (p, α, α^d) um algoritmo eficiente que resolve o problema acima pode ser usado para calcular $\alpha^{-db} \bmod p$ e resolver $(\alpha^d)^b \cdot \mathrm{P} \bmod p$ para P.

> *Problema* 20. Um algoritmo eficiente para o Problema de Diffie–Hellman implica num algoritmo eficiente para o problema do logaritmo discreto?

ASSINATURA DIGITAL ELGAMAL Alice quer enviar um documento P com uma assinatura $\mathrm{ass}(\mathrm{P}) = (r, s)$ para Bob. Sejam (p, α, A) e d as chaves pública e privada, respectivamente, geradas pelo protocolo do sistema Elgamal de Alice. A Alice

- sorteia uniformemente $k \in \{2, 3, \ldots, p-2\}$,
- computa $r = \alpha^k \bmod p$ e $s = (\mathrm{P} - dr)(k^{-1} \bmod p-1) \bmod p-1$ usando sua chave privada,

e envia $(\mathrm{P}, (r, s))$ para Bob. Para a verificação da assinatura o Bob, de posse da chave pública de Alice, verifica se $\alpha^{\mathrm{P}} = \mathrm{A}^r r^s \bmod p$.

O CRIPTOSSISTEMA ASSIMÉTRICO RSA Esse é provavelmente o mais famoso sistema criptográfico assimétrico, deve o seu nome aos três professores do MIT responsáveis pela sua criação: Ron Rivest, Adi Shamir e Len Adleman (Rivest, Shamir e Adleman, 1978), os quais receberam o prestigiado prêmio Alan Turing em 2002 por "sua engenhosa contribuição que fez a criptografia de chaves públicas úteis na prática".

Para simplificar, vamos assumir que o universo dos textos é o $\mathbb{Z}_n$, onde n é um parâmetro do protocolo. A determinação das chaves é da seguinte maneira

- sorteie dois primo p e q grandes, aproximadamente do mesmo tamanho;
- compute $n = pq$ e a função de Euler $\varphi(n) = (p-1)(q-1)$;
- sorteie uniformemente $e \in \{2, 3, \ldots, \varphi(n)-1\}$ com $\mathrm{mdc}(e, \varphi(n)) = 1$;
- compute o único $d \in \{2, 3, \ldots, \varphi(n)-1\}$ tal que $ed \equiv 1 \pmod{\varphi(n)}$;

- a *chave pública*, a ser divulgada, é (e, n) e a *chave privada* correspondente, que deve ser mantida em segredo, é d, o inverso multiplicativo de e módulo $\varphi(n)$.

Dado um texto P, a codificação com a chave (e, n) é dada por $E_e(P) := P^e \bmod n$. A chave de decodificação correspondente é d, o inverso multiplicativo de e módulo $\varphi(n)$. Dado um texto cifrado C, a decodificação é realizada por $D_d(C) := C^d \bmod n$. Essas funções podem ser computadas eficientemente.

Por exemplo, tome os primos $p = 61$ e $q = 53$, de modo que $n = pq = 3233$. A função de Euler é $\varphi(n) = (p-1)(q-1) = 3120$. Escolhemos $e = 17$ (chave pública) e obtemos $d = 2753$ (chave privada). Se o texto é $m = 123$ então o texto cifrado é $c = m^{17} \bmod 3233 = 855$, que é decifrado por $c^{2753} \bmod 3233 = 855^{2753} \bmod 3233 = 123$.

Agora, vamos verificar que se $P \in \mathbb{Z}_n$ então $D_d(E_e(P)) = P$. Temos que $D_d(E_e(P))$ é $P^{ed} \bmod n$ com $ed \equiv 1 \pmod{\varphi(n)}$, ou seja, $ed = 1 + r\varphi(n)$ para algum $r \in \mathbb{Z}$. Assim,

$$P^{ed} = P^{1+r\varphi(n)} = P(P^{p-1})^{(q-1)r}$$

e pelo Pequeno Teorema de Fermat $P^{p-1} \equiv 1 \pmod{p}$, se p não divide P, logo

$$P\left(P^{p-1}\right)^{(q-1)r} \equiv P \pmod{p}. \tag{4.10}$$

A equação (4.10) também vale quando p divide P. Portanto, $P^{ed} \equiv P \pmod{p}$. Analogamente, vale que $P^{ed} \equiv P \pmod{q}$ e dessas duas congruências temos que $pq | P^{ed} - P$, ou seja, $P^{ed} \equiv P \pmod{n}$, assim

$$D_d\left(E_e(P)\right) \equiv P^{ed} \equiv P \pmod{n}.$$

Um algoritmo eficiente para FATORAÇÃO (problema 16 na página 215) implica que a partir de (e, n) é possível computar $\varphi(n)$, pelo algoritmo de Euclides estendido, e depois computar a chave privada d, quebrando a codificação. Então fatoração eficiente é suficiente para a quebra do RSA, mas não se sabe se é necessária. Pode haver, por exemplo, um algoritmo para computar $\varphi(n)$ sem que se recorra à fatoração de n e que seja eficiente. Porém, a existência desse algoritmo implicaria num algoritmo eficiente que determina p e q.

De fato, a segurança do RSA é baseada na dificuldade computacional do *Problema RSA*, ou seja, inverter uma função candidata a unidirecional.

> *Problema* 21 (PROBLEMA RSA). Existe um algoritmo eficiente que, dada uma chave pública (e, n) e um inteiro C, determine um inteiro P tal que $P^e = C \bmod n$?

Acredita-se que esse problema seja equivalente ao problema da fatoração.

CONJECTURA 4.20 (RIVEST, SHAMIR E ADLEMAN) *Um algoritmo eficiente para o problema RSA implica num algoritmo eficiente para o* problema da Fatoração.

Sabemos que a partir de (e, n) e d é possível determinar p e q em tempo polinomial (Coron e May, 2007).

ASSINATURA DIGITAL RSA Alice quer enviar um documento P com uma assinatura ass(P) para Bob. Sejam (e, n) e $d = d(e)$ um par de chaves RSA, pública e privada respectivamente, de Alice. Alice usa sua chave privada e computa $s = P^d \bmod n$ e envia (P, s) para Bob. Para a verificação da assinatura Bob obtém a chave pública de Alice e verifica se $P = s^e \bmod n$.

4.4.4 PROVAS COM CONHECIMENTO ZERO

Conhecimento zero em um sistema de prova é uma característica de certos Provadores de que somente a validade da prova é transmitida ao Verificador no processo de interação, ou seja, o Verificador não ganha informação no processo e mesmo assim convence-se da validade da prova no final do processo (Goldwasser, Micali e Rackoff, 1985). Por exemplo, no caso das bolas de bilhar, página 205, Bob fica convencido de que as bolas têm cor diferente mas não sabe a cor de cada bola individualmente. Essa propriedade é bastante interessante para projetos de protocolos em Criptografia pois ela permite que a prova de um fato sem revelar segredos.

O fato do Verificador não ganhar informação significa que num sistema de prova (V, P) com entrada w tudo aquilo que o Verificador V puder computar ao interagir com P a partir da entrada w é igual a tudo aquilo que um algoritmo da classe do Verificador (com o mesmo poder de computação) consegue computar com a entrada w por ele mesmo, isto é, não interagindo com um Provador P. Por exemplo, no sistema para 3-COL, exemplo 4.14 na página 205, o Verificador recebe do Provador uma 3-coloração, informação que um algoritmo de tempo polinomial não consegue computar. Nesse exemplo o Verificador ganha informação a partir da interação.

Um sistema de prova tem conhecimento zero se para todo Verificador V existe um algoritmo eficiente M, chamado **simulador**, tal que a distribuição da resposta de V quando interage com P, que denotamos por $(V, P)(w)$, e a distribuição $M(w)$ são idênticas. Formalmente, relaxamos um pouco a exigência sobre o tempo do simulador, permitindo que seja polinomial em média. Dizemos que um Provador de um sistema de prova interativo para a linguagem L tem **conhecimento-zero perfeito** se para todo Verificador V existe um algoritmo probabilístico M de tempo *esperado* polinomial, tal que para todo $w \in L$, as variáveis aleatórias $(V, P)(w)$ e $M(w)$ são identicamente distribuídas.

Para propósitos práticos podemos exigir menos da relação entre as distribuições $(V, P)(x)$ e $M(x)$, por exemplo, que sejam indistinguíveis estatisticamente ou por algoritmos eficientes. De acordo com alguma definição de distância entre essas distribuições, temos a seguinte hierarquia de conhecimento zero

perfeito – as distribuições são idênticas, como foi discutido acima;

estatístico – as distribuições são estatisticamente próximas[8];

computacional – as distribuições são computacionalmente indistinguíveis por algoritmos eficientes,

claramente, os requisitos de conhecimento zero perfeito são mais exigentes que o estatístico que, por sua vez, são mais exigentes que o computacional, que é o mais prático. Essas definições dão origem as classes de linguagens com conhecimento-zero computacional CSK — *Computational Zero-knowledge* – e conhecimento-zero estatístico SZK – *Statistical Zero-knowledge*. É sabido que

$$\mathsf{BPP} \subset \mathsf{PZK} \subset \mathsf{SZK} \subset \mathsf{CZK} \subset \mathsf{IP}.$$

Com a hipótese de existirem funções unidirecionais temos que CZK = IP (Ben-Or et al., 1990) e que NP ⊂ CZK (Goldreich, Micali e Wigderson, 1991). Por outro lado, é improvável que NP ⊂ SZK (Fortnow, 1987); apesar disso vários problemas difíceis estão em SZK, como resíduo e não-resíduo quadrático, isomorfismo e não-isomorfismo de grafos, logaritmo discreto. Fortnow (1987) provou que a existência de um problema NP-completo em SZK implica no colapso de PH no nível 2. Sob a hipótese de existir função unidirecional e de não haver colapso em PH temos CZK ≠ SZK.

Conjectura 4.21 $\mathsf{BPP} \subsetneqq \mathsf{PZK}$ *e* $\mathsf{SZK} \subsetneqq \mathsf{CZK}$ *e* $\mathsf{CZK} = \mathsf{IP}$.

LOGARITMO DISCRETO ∈ PZK Um provador deseja convencer qualquer verificador de que ele conhece k, o logaritmo discreto de β com relação à base (raiz primitiva) α e um primo p grande. Os parâmetros α, β e p são públicos.

Entrada: p primo, α raiz primitiva módulo p e β um resíduo módulo p.

P: $r \overset{R}{\leftarrow} \{0,\ldots,p-1\}$;
$h \leftarrow \alpha^r \bmod p$;

P→V: h;

V: $b \leftarrow_R \{0,1\}$;

V→P: b;

P: $a \leftarrow r + bk \bmod p$

P→V: a;

V: se $\alpha^a \equiv h\beta^b \pmod p$ então responde 1,
senão responde 0.

[8] segundo a métrica $\max_{S\subset\Omega} \left|\mathbb{P}[X \in S] - \mathbb{P}[Y \in S]\right|$.

Notemos que V executa em tempo polinomial. Ademais, dado que P conhece k, o logaritmo discreto de β, o verificador fica convencido (sem erro) desse fato pois $\alpha^a = \alpha^r \alpha^{kb} = h\beta^b$, logo, o sistema interativo é completo.

Para provar a consistência suponhamos que não haja provador que conheça k. Fixemos um deles. No protocolo P deve enviar um h na primeira mensagem, suponhamos que P sorteia r e envia h. Se V sorteia 0, então o provador pode enviar $a = r$ e V aceita, senão V sorteia 1 e o provador precisa descobrir o logaritmo discreto de $h\beta$, que é o logaritmo discreto de h mais o de β, ou seja, $r + k$. O melhor que ele pode fazer é chutar e acertar com probabilidade $1/(p-1)$. A probabilidade de um aceite nesse caso é

$$\frac{1}{2} + \frac{1}{2}\frac{1}{p-1}.$$

Agora, vamos considerar que P envia h sem conhecer r. Se V sorteia 0, o provador precisa descobrir o logaritmo discreto de h, então ele chuta. Se V sorteia 1, então um provador esperto poderia ter enviado $h = \alpha^r \beta^{-1}$ pois, com isso, agora basta enviar $a = r$ que V aceita pois $\alpha^a = \alpha^r = (\alpha^r \beta^{-1})\beta = h\beta$. Agora, a probabilidade de aceite é no máximo

$$\frac{1}{2}\frac{1}{p-1} + \frac{1}{2}.$$

Se p é primo ímpar então $\frac{1}{2}\frac{1}{p-1} + \frac{1}{2} < 1$ e a repetição do protocolo faz a probabilidade de erro pequena suficiente para valer a consistência.

Para verificar o conhecimento-zero do protocolo nós precisamos provar que nenhum verificador recebe informação adicional àquelas transmitidas nas mensagens do protocolo. Nesse protocolo, podemos ver a computação do verificador em duas etapas: (1) $\mathsf{V}_1(h)$ que recebe a informação h do Provador e responde com um bit b e (2) $\mathsf{V}_2(h, a)$ que é a computação com as informações h e a recebidas do Provador. Essa duas etapas são de tempo polinomial. Fixado um verificador V que interage com P dado acima, escrevemos o seguinte simulador que denominamos M.

Entrada: p primo, α raiz primitiva módulo p e β um resíduo módulo p.

1. sorteia $b' \in_{\mathrm{R}} \{0, 1\}$ e $r' \in_{\mathrm{R}} \{0, \ldots, p-1\}$;
2. $[\mathsf{V}_1(h)]$ simula V com $h := \alpha^{r'} \beta^{b'}$ na memória $\mathsf{F}_{\mathsf{P}\to\mathsf{V}}$ para obter $b \in \{0, 1\}$ após um número polinomial de passos;
3. se $b = b'$ então $[\mathsf{V}_2(h, a)]$ simula V com o envio de $a := r'$ por P e devolve a resposta da simulação; caso contrário, recomeça no passo 1.

Para provar que a resposta do simulador M tem a mesma distribuição da resposta de V na interação basta provar que nas respostas parciais a distribuição é a mesma.

Primeiro, *a distribuição de h computado por* M *é idêntica a de h escolhida pelo provador*: em ambos os

casos h é uma potência de α com expoente aleatório com distribuição uniforme. Quando o simulador atribui $h := \alpha^{r'}\beta^{b'}$ temos $h = \alpha^{r'-b'k}$ e o expoente tem distribuição uniforme.

Segundo, $b = b'$ *com probabilidade pelo menos* $1/2$: segue do fato no parágrafo anterior que as distribuições dos h's são idênticas, portanto, o comportamento do verificador não muda, de modo que $b = b'$ com probabilidade $1/2$. Portanto, o número esperado de execuções do simulador é no máximo 2, o que resulta num tempo esperado de execução que é polinomial no tamanho da entrada.

Finalmente, para concluir que as respostas têm a mesma distribuição, *condicionado a* $b' = b$ *e ao valor de* h, *o valor* a *computado por* M *é o mesmo que* P *envia quando a primeira mensagem foi* h *e a resposta de* V *foi* b: o valor enviado de a por P é

$$a = \begin{cases} \text{log discreto de } h & \text{se } b = 0 \\ \text{log discreto de } h\beta & \text{se } b = 1 \end{cases}$$

e no caso de M, como $\alpha^{r'} = h\beta^{b'}$ e $a = r'$

$$a = \begin{cases} \text{log discreto de } h & \text{se } b' = 0 \\ \text{log discreto de } h\beta & \text{se } b' = 1 \end{cases}.$$

Nessa condições a resposta de V_2 é a mesma resposta que V daria. Assim toda escrita de V na fita ocorre com as mesma distribuições de V_1 e V_2 donde concluímos que M e o sistema (V,P) respondem com a mesma distribuição.

$\mathsf{ISO} \in \mathsf{PZK}$ Vejamos um sistema de prova interativa com conhecimento zero para a linguagem ISO a qual não sabemos se pertence a BPP.

Sejam G_0 e G_1 grafos isomorfos sobre o mesmo conjunto de vértices, sem perda de generalidade $V := \{1, 2, \ldots, n\}$. Consideremos uma permutação dos vértices $\pi \in_R \mathbb{S}_n$ e $\pi(G_X)$ o grafo isomorfo a G_X pela permutação π. Então as variáveis aleatórias $X \in_R \{0,1\}$ e $\pi(G_X)$ são independentes se X e π são independentes.

De fato, sejam G_0 e G_1 grafos isomorfos e $X \in_R \{0,1\}$ e $\pi \in_R \mathbb{S}_n$ variáveis aleatórias independentes. Os conjuntos de permutações $\Pi_0 = \{\sigma : \sigma(G_0) = H\}$ e $\Pi_1 = \{\sigma : \sigma(G_1) = H\}$ têm a mesma cardinalidade, portanto, $\mathbb{P}[\,\pi \in \Pi_1\,] = \mathbb{P}[\,\pi \in \Pi_0\,]$. Como $\mathbb{P}[\,\pi(G_X) = H \mid X = 1\,] = \mathbb{P}[\pi(G_1) = H] = \mathbb{P}[\,Y \in \Pi_1\,]$ e, analogamente, $\mathbb{P}[\,\pi \in \Pi_0\,] = \mathbb{P}[\,\pi(G_X) = H \mid X = 0\,]$ temos, pelo Teorema de Bayes, que

$$\begin{aligned}
\mathbb{P}[\,X = 1 \mid \pi(G_X) = H\,] &= \frac{\mathbb{P}[\,\pi(G_X) = H \mid X = 1\,]\mathbb{P}[X = 1]}{\mathbb{P}[\,\pi(G_X) = H \mid X = 1\,]\mathbb{P}[X = 1] + \mathbb{P}[\,\pi(G_X) = H \mid X = 0\,]\mathbb{P}[X = 0]} \\
&= \frac{\mathbb{P}[\,\pi(G_X) = H \mid X = 1\,]\mathbb{P}[X = 1]}{\mathbb{P}[\,\pi(G_X) = H \mid X = 1\,]\big(\mathbb{P}[X = 1] + \mathbb{P}[X = 0]\big)} \\
&= \frac{\mathbb{P}[\,\pi(G_X) = H \mid X = 1\,]\mathbb{P}[X = 1]}{\mathbb{P}[\,\pi(G_X) = H \mid X = 1\,]} = \frac{1}{2}
\end{aligned}$$

e $\mathbb{P}[\, X = 0 \mid \pi(G_X) = H \,] = 1/2$ pela mesma razão, assim, para todo grafo H isomorfo a G_0 e G_1

$$\mathbb{P}[\, X = 1 \mid \pi(G_X) = H \,] = \mathbb{P}[\, X = 0 \mid \pi(G_X) = H \,] = \frac{1}{2} = \mathbb{P}[X = 1\,] = \mathbb{P}[X = 0\,]$$

ou seja, conhecer a cópia isomorfa não dá nenhuma pista sobre o bit sorteado.

Suponha agora que um Provador queira provar que dois grafos dados, G_0 e G_1, são isomorfos, e que ele *de fato conhece um isomorfismo*. Vamos apresentar um protocolo para este problema em que tanto o provador quanto o verificador são eficientes.

Entrada: G_0 e G_1 grafos.

P: $\pi \xleftarrow{R} \mathbb{S}_n$;
$H \leftarrow \pi(G_1)$;

P→V: H;

V: $i \leftarrow_R \{0,1\}$;

V→P: i;

P: escolhe uma permutação φ, se possível com $\varphi(G_0) = G_1$;
se $i = 1$ então $\sigma \leftarrow \pi$;
se $i = 0$, então $\sigma \leftarrow \pi \circ \varphi$;

P→V: σ;

V: se $H = \sigma(G_i)$ então responde 1,
senão responde 0.

Se os dois grafos de entrada são isomorfos, P conhece um isomorfismo e o usa em φ e então o Verificador sempre irá responder 1. Suponha que os grafos da entrada não são isomorfos. Não importa como é construído o grafo H que o Provador envia ao Verificador, sempre haverá $i \in \{0,1\}$ tal que H e G_i não são isomorfos, portanto, se o Verificador cumpre seu papel então aceita a entrada somente no caso de sortear o i correto com probabilidade 1/2. Repetindo o protocolo obtemos que a probabilidade de erro no máximo 1/4.

Resta verificarmos que vale o *conhecimento zero* no protocolo. Precisamos mostrar que o Provador não transmite conhecimento qualquer que seja o verificador, mesmo que esse trapaceie com o objetivo de extrair informação do Provador. Seja V um programa fixo e arbitrário de um algoritmo probabilístico de tempo polinomial interagindo com P de acordo com o protocolo. Vamos projetar um algoritmo M de tempo polinomial que gera uma distribuição de probabilidade que é idêntica à distribuição de probabilidade induzida nas memórias de bits aleatório e de comunicação de V durante sua interação. A estratégia de V pode ser vista como duas funções: (1) a computação de V com

H (a primeira mensagem passada por P) que responde com um bit i sorteado por V e (2) a computação de V com H e σ (a duas mensagens passadas por P) é qualquer coisa que V computa depois da resposta de P ao bit i.

Entrada: G_0 e G_1 grafos.

1. escolhe $i' \in_R \{0,1\}$ e escolhe $\pi \in_R \mathbb{S}_n$;
2. simula V com entrada G_0 e G_1 e com $H := \pi(G_{i'})$ na memória $F_{P\to V}$ para obter $i \in \{0,1\}$ após um número polinomial de passos;
3. se $i = i'$ então simula V com o envio de π por P e devolve a mesma resposta; caso contrário, recomeça a simulação.

Reparemos, e isso é um ponto importante, que se G_0 e G_1 são isomorfos então o grafo H gerado não revela a escolha de i', portanto $i = i'$ com probabilidade 1/2; o seguinte resultado, de que $\pi(G_{i'})$ não dá informação sobre i' será provado adiante nessa seção.

O simulador escolhe i' uniformemente na esperança de que, a frente, o Verificador escolha $i = i'$. Agora, para qualquer algoritmo aleatorizado A que com entrada $H = \pi(G_{i'})$ tenta computar i' vale

$$\begin{aligned}
\mathbb{P}[\,A(\pi(G_{i'})) = i'\,] &= \sum_G \mathbb{P}[\,A(G) = i' \mid \pi(G_{i'}) = G\,]\mathbb{P}[\,\pi(G_{i'}) = G\,] \\
&= \sum_G \sum_b \mathbb{P}[\,[A(G) = b] \cap [i' = b] \mid \pi(G_{i'}) = G\,]\mathbb{P}[\,\pi(G_{i'}) = G\,] \\
&= \sum_G \sum_b \mathbb{P}[A(G) = b]\mathbb{P}[\,i' = b \mid \pi(G_{i'}) = G\,]\mathbb{P}[\,\pi(G_{i'}) = G\,] \\
&= \sum_G \sum_b \mathbb{P}[A(G) = b]\frac{1}{2}\mathbb{P}[\,\pi(G_{i'}) = H\,] = \frac{1}{2}.
\end{aligned}$$

Sendo assim, $\mathbb{P}[\,i = i'\,] = 1/2$ e dado que $i = i'$ o Verificador responde com a distribuição correta e, além disso, M reinicia no passo 3 com probabilidade 1/2, portanto o número esperado de rodadas antes de terminar é 2 e como cada rodada é de tempo polinomial, concluímos que o simulador é de tempo esperado polinomial.

Para finalizar, notemos que o Verificador do protocolo dado acima, no inicio desse exemplo, executa em tempo polinomial (probabilístico) e o mesmo vale para o Provador quando um isomorfismo φ é dado como entrada auxiliar. Também, a rigor, precisamos mostrar que vale a propriedade de *conhecimento zero* para qualquer verificador que interage com P, isto é, a análise acima aplica-se no caso em que V segue o protocolo mas esse não é sempre o caso, só o protocolo de P é fixo. Uma prova detalhada dessa análise pode ser vista em Goldreich (2001) ou em Goldreich, Micali e Wigderson (1991).

4.4.5 NP ⊂ CZK SE FUNÇÕES UNIDIRECIONAIS EXISTEM

Lembremos que uma linguagem L é NP-completa se está em NP e para todo $I \in NP$ existe um algoritmo de tempo polinomial R tal que $x \in I$ se e só se $R(x) \in L$. Nessa seção mostramos exemplos de sistema de prova com conhecimento zero computacional para as linguagens NP-completas CH (caixeiro viajante) e 3-COL (3-coloração de grafos); disso temos NP ⊂ CZK. Esse resultado depende da existência de função unidirecional. A discussão nessa seção vai ser informal.

Se G é um grafo sobre o conjunto de vértices $V := \{1, 2, \ldots, n\}$, então dizemos que G é *hamiltoniano* se existe uma permutação π de V tal que $\{\pi(1), \pi(n)\}$ e $\{\pi(i), \pi(i+1)\}$ são arestas para todo $i \in \{1, 2, \ldots, n-1\}$. Em outras palavras, um grafo é hamiltoniano se seus vértices podem ser ordenados de modo que vértices consecutivos com respeito a tal ordem são adjacentes e também são adjacentes o primeiro e último vértices da sequência. Essa sequência de vértices de G é um *circuito hamiltoniano*. A linguagem CH formada pelos grafos hamiltonianos admite um sistema de provas com conhecimento zero computacional caso exista função unidirecional (Blum, 1986). Como essa linguagem é NP-completa temos NP ⊂ CZK.

Vamos começar com um protocolo: o Provador e o Verificador conhecem um grafo G. O Provador afirma conhecer um um circuito hamiltoniano C. **P** escolhe uma permutação $\sigma \in_R \mathbb{S}_n$, criptografa separadamente cada posição da matriz de adjacências do grafo $\sigma(G)$ e envia a matriz para o Verificador; **V** escolhe $b \in_R \{0, 1\}$ e envia o bit para o Provador. **P** testa se $b = 0$, nesse caso decodifica toda a matriz e a revela junto com σ; no caso $b = 1$ revela somente as n entradas da matriz que correspondem às arestas de $\sigma(C)$; finalmente, envia ao Verificador o que foi revelado; **V**, caso $b = 0$, verifica se $G \equiv \sigma(G)$, caso contrário verifica se recebeu um circuito, nesses casos aceita, senão rejeita.

O protocolo acima é baseado no fato de que o Provador entrega ao Verificador um grafo antes do Verificador decidir o que quer, caso $b = 0$ o Verificador quer verificar se a matriz criptografada é legitima e no caso $b = 1$ verificar se o circuito hamiltoniano é legítimo. Como a escolha do Verificador é aleatória o Provador convence o Verificador com probabilidade 1/2, no caso em que ele (o Provador) não conhece um circuito hamiltoniano. O fato do Verificador ser limitado não permite-o conhecer nada a respeito do que foi criptografado pelo Provador. O fato de usar criptografia torna o protocolo de conhecimento zero *computacional*. De fato, nesses casos é usado uma técnica chamada de *esquema de empenho de bits* (Goldreich, Micali e Wigderson, 1991).

Um **esquema de empenho de bits** (*bit commitment*) é um sistema interativo composto por dois algoritmos probabilísticos de tempo polinomial e de duas fases entre as duas partes: um *Remetente* que compromete-se com um valor $b \in \{0, 1\}$ perante um *Receptor*. A primeira fase é chamada a *fase de comprometimento* — o Remetente guarda um valor b numa caixa inviolável — e a segunda é a *fase de revelação* — o Remetente abre a caixa e revela o valor guardado — de modo que são satisfeitos:

1. *sigilo*: no final da primeira fase o Receptor não ganhou nenhum conhecimento do valor guar-

dado, mesmo no caso de um Receptor desonesto;

2. *vínculo*: dada a transcrição da interação na primeira fase, há no máximo um valor que o Receptor possa mais tarde aceitar como um valor legal guardado, mesmo no caso do Remetente ser desonesto.

Se ambas as partes seguem o protocolo, então no final da segunda fase o Receptor obtém o valor empenhado pelo Remetente. Exigimos que a fase de comprometimento não produza conhecimento para o Receptor do valor empenhado, enquanto que a fase de revelação vincula ao Remetente um valor único, no sentido de que na fase de revelação o Receptor só pode aceitar esse valor.

Exercício 4.22 (predicado hard-core). (Goldreich e Levin, 1989) Um predicado $b\colon \{0,1\}^* \to \{0,1\}$ computável em tempo polinomial é dito **hard-core** da função $f\colon \{0,1\}^* \to \{0,1\}^*$ se para todo algoritmo probabilístico de tempo polinomial A e todo inteiro $d > 0$

$$\mathop{\mathbb{P}}_{X \in_R \{0,1\}^n}[A(f(X)) = b(X)] < \frac{1}{2} + \frac{1}{n^d}$$

para todo n suficientemente grande, ou seja, Um algoritmo probabilístico de tempo polinomial A com entrada $f(x)$ consegue prever $b(x)$ com probabilidade no máximo próxima a $1/2$, um pouco melhor do que responder jogando uma moeda justa. Suponha f unidirecional e defina g por $g(x,r) := (f(x), r)$ com $|x| = |r|$ e tome $b(x,r) := \sum_i x_i r_i \pmod 2$. Prove que g é unidirecional e que b é um predicado *hard-core* de g.

Para um exemplo de empenho, tomemos $f\colon \{0,1\}^* \to \{0,1\}^*$ uma função injetiva unidirecional e $b\colon \{0,1\}^* \to \{0,1\}$ um predicado *hard-core* de f (definido no exercício 4.22 acima).

1. *Fase de comprometimento*: para empenhar $v \in \{0,1\}$ o Remetente escolhe $x \in_R \{0,1\}^n$ (n é o parâmetro de segurança) e envia $(f(x), b(x) \oplus v)$.

   Após essa fase, por causa do predicado *hard-core*, temos que a probabilidade de um Receptor limitado qualquer prever o valor empenhado v é no máximo $(1/2) + (1/p(n))$.

2. *Fase da revelação*: o Remetente revela v e x e o Receptor verifica se a mensagem recebida (α, β) corresponde a $\alpha = f(x)$ e $\beta = b(x) \oplus v$.

É possível provar que se f é unidirecional e b um predicado *hard-core* de f então o protocolo acima é um esquema de empenho. O sigilo segue da definição de *hard-core* e o vínculo segue da injetividade de f. Para cada mensagem (α, σ) para o Receptor existe um único s tal que $f(s) = \alpha$ e então um único $v \in \{0,1\}$ tal que $b(s) \oplus v = \sigma$.

Exemplo 4.23 (3-COL $\in$ CZK). o problema de determinar se há uma 3-coloração própria dos vértices de um grafo também é NP-completo. Suponhamos existir uma função unidirecional ou, mais especificamente, de um esquema de empenho de bits. Uma repetição de $n|E|$ vezes o protocolo abaixo

resulta numa prova de conhecimento-zero para 3-COL com probabilidade de erro exponencialmente pequena em n para a consistência.

Entrada: Um grafo $G = (V, E)$;

P: $\pi \leftarrow_R \mathbb{S}_3$;
$\phi \leftarrow \pi \circ \psi$, com $\psi\colon V \to \{1,2,3\}$, uma 3-coloração própria de G;

P→V: $\{c_v\}_{v \in V}$ onde c_v é um empenho do valor $\phi(v)$.

V: $\{u, v\} \leftarrow_R E$;

V→P: $\{u, v\}$;

P→V: c_u e c_v;

V: Se os valores revelados $\phi(u)$ e $\phi(v)$ são distintos, responde 1,
senão responde 0.

Se o Provador conhece uma 3-coloração do grafo então, com probabilidade 1, ele convence o Verificador desse fato; se o grafo não admite uma 3-coloração então o Verificador rejeita com probabilidade pelo menos $1/|E|$. A probabilidade de não haver erro nas $n|E|$ rodadas é $(1 - |E|^{-1})^{n|E|} \approx \exp(-n)$ (veja (s.8)). O Verificador executa em tempo polinomial e também o Provador assumindo que ele receba ψ como entrada auxiliar.

O conhecimento-zero é mais difícil de provar. Intuitivamente, as únicas informações reveladas para o Verificador numa interação são $\pi(\psi(u))$ e $\pi(\psi(v))$, ou seja, um par de cores escolhido aleatoriamente em $\{1,2,3\}$ a partir de π; qualquer simulador que devolva um par de cores $(i, j) \in_R \{1,2,3\}^2$, $i \neq j$, é computacionalmente indistinguível da informação equivalente que surge na interação do protocolo.

Um simulador começa com uma 3-coloração aleatória $\psi \in_R \{1,2,3\}^n$ e simula o verificador com um empenho dessas cores. Notemos que essa entrada para a simulação do Verificador tem distribuição diferente da entrada na execução do protocolo interativo, entretanto, pela segurança envolvida no esquema de empenho, essas distribuições são indistinguíveis. Nesse momento, se o Verificador pede para examinar se uma aresta sorteada está propriamente colorida o simulador revela as cores empenhadas dos vértices.

1. escolhe aleatoriamente $\{i', j'\} \in E$ e $c_i, c_j \in \{1,2,3\}$, $c_i \neq c_j$; todos os outros vértices recebem cor arbitrária; empenha essas cores;

2. simula V e se o Verificador pede para ver as cores de $\{i', j'\}$, então revela as cores e responde 1; caso contrária reinicia a simulação. Após $n|E|$ repetições responde 0.

Uma demonstração detalhada pode ser lida em Goldreich, Micali e Wigderson (1991). ◊

4.5 PROVAS NATURAIS

$\Gamma = \Lambda = \mathsf{P/POLY}$:

4.5.1 GERADORES DE FUNÇÕES PSEUDOALEATÓRIAS

4.5.2 UMA PROVA NATURAL

4.6 EXERCÍCIOS

Exercício 4.24 (Arora e Barak, 2009). Prove que toda função $f\colon \{0,1\}^* \to \{0,1\}$ computável por uma Máquina de Turing com alfabeto Γ em tempo $t(n)$ também é computável por uma Máquina de Turing com alfabeto $\{0,1,\text{ƀ}\}$ em tempo $O(\log(|\Gamma|)t(n))$.

Exercício 4.25. Seja $I \subset \{0,1\}^*$ uma linguagem indecidível por máquina de Turing (por exemplo, a linguagem definida pelo Problema da Parada, definido na página 183). Defina

$$L = \{1^n\colon n \text{ em base 2 pertence a } I\}.$$

Prove que L é indecidível. Exiba uma família de circuitos $(C_n\colon n \in \mathbb{N})$ com número de vértices polinomial em n e que decide pertinência em L.

Exercício 4.26 (Problema de busca NP). Um problema de busca definido pela relação R é um problema de busca NP se dados x e y, decidir $(x,y) \in R$ pode ser feito em tempo polinomial e existe um polinômio p tal que se $(x,y) \in R$ então $|y| \leqslant p(|x|)$. Prove que para todo problema de busca NP, existe um problema de decisão NP tal que se o problema de decisão tem solução em tempo $T(n)$, então o problema de busca tem solução em tempo $O(n^{c_1} \cdot T(n^{c_2}))$, para constantes positivas c_1 e c_2. Conclua que $\mathsf{P} = \mathsf{NP}$ se, e só se, todo problema de busca NP tem solução em tempo polinomial.

Exercício 4.27. (Adleman, 1978) Defina *circuito booleano aleatorizado* como um circuito booleano que além das n portas da entrada contém portas que recebem um bit aleatório uniformemente; esse circuito computa $f\colon \{0,1\}^n \to \{0,1\}$ se quando $f(x_1,\ldots,x_n) = 0$ o circuito com $(x_1,\ldots,x_n)$ responde 0 e quando $f(x_1,\ldots,x_n) = 1$ o circuito com $(x_1,\ldots,x_n)$ responde 1 com probabilidade pelo menos 1/2. Prove que se $f\colon \{0,1\}^* \to \{0,1\}$ é computada por uma família de circuitos booleanos aleatorizados de tamanho polinomial então f é computada por uma família de circuitos booleanos de tamanho polinomial (que não usa bits aleatórios).

Exercício 4.28. Prove que

1. $\mathsf{BPP} = \mathsf{coBPP}$.

2. $\mathsf{ZPP} = \mathsf{coZPP}$.

Exercício 4.29. Prove que se $\mathsf{NP} \neq \mathsf{coNP}$ então $\mathsf{P} \neq \mathsf{NP}$.

Exercício 4.30. Prove que se $\mathsf{NP} \not\subset \mathsf{P/poly}$ então $\mathsf{P} \neq \mathsf{NP}$.

Exercício 4.31. Prove que se $\mathsf{NP} \subset \mathsf{BPP}$ então $\mathsf{NP} = \mathsf{RP}$.

Exercício 4.32. Prove que se $\mathsf{RP} \subset \mathsf{coPP}$ então $\mathsf{ZPP} = \mathsf{RP}$ e $\mathsf{RP} \subset \mathsf{NP} \cap \mathsf{coNP}$. (Atualmente, não se sabe $\mathsf{RP} = \mathsf{coRP}$, se $\mathsf{RP} \subset \mathsf{NP} \cap \mathsf{coNP}$ e se $\mathsf{NP} = \mathsf{coNP}$.)

Exercício 4.33. Sejam M uma máquina de Turing probabilística e L uma linguagem tais que para constantes $0 < \varepsilon_1 < \varepsilon_2 < 1$ e todo $w \in \{0,1\}^*$

1. se $w \in \mathsf{L}$, então $\mathbb{P}[\,\mathsf{M}(w) = 1\,] \geqslant \varepsilon_2$, e
2. se $w \notin \mathsf{L}$, então $\mathbb{P}[\,\mathsf{M}(w) = 1\,] < \varepsilon_1$;

prove que L está em BPP.

Exercício 4.34. A classe PP — *Probabilistic Polynomial time* — é a classe das linguagens L para as quais existe um algoritmo probabilístico M de tempo polinomial tal que

$$\mathbb{P}[\,\mathsf{M}(w) = \mathsf{L}(w)\,] \geqslant 1/2, \text{ para todo } w \in \{0,1\}^*.$$

Prove as seguintes inclusões

1. $\mathsf{P} \subset \mathsf{ZPP} \subset \mathsf{RP} \subset \mathsf{NP} \subset \mathsf{PP}$.
2. $\mathsf{RP} \subset \mathsf{BPP} \subset \mathsf{PP}$.
3. $\mathsf{PP} = \mathsf{coPP}$.

Exercício 4.35. Suponha que exista um algoritmo determinístico eficiente A que, dado inteiros a e $\mathsf{N} = pq > 1$, para p e q primos distintos, determina de modo eficiente uma solução de $x^2 \equiv a \pmod{\mathsf{N}}$. Sorteie $b \in \mathbb{Z}_{\mathsf{N}}^*$ e compute $b^2 \bmod \mathsf{N}$, em seguida use o algoritmo A para determinar uma solução x de $\mathsf{X}^2 \equiv b^2 \pmod{\mathsf{N}}$. Verifique que com probabilidade $1/2$ temos $x \not\equiv \pm b \pmod{\mathsf{N}}$. Prove que nesse caso $\mathrm{mdc}(b - x, \mathsf{N}) \in \{p, q\}$. Conclua que se é possível determinar raiz quadrada módulo N de maneira eficiente então é possível fatorar N de maneira eficiente.

Exercício 4.36. Prove que a linguagem do exemplo 4.11 está em Π_2.

Exercício 4.37. Mostre que se o i-ésimo bit de $p \in (0,1)$ pode ser determinado em tempo polinomial em i então a distribuição sobre $\{0,1\}$ com $\mathbb{P}(1) = p$ pode ser simulada por uma máquina de Turing probabilística de tempo esperado $O(1)$.

Exercício 4.38. Mostre que uma moeda com $\mathbb{P}[\text{cara}] = 1/2$ pode ser simulada por uma máquina de Turing probabilística que lança moeda com $\mathbb{P}[\text{cara}] = p$ com tempo esperado $O(1/(p-p^2))$.

Exercício 4.39. Mostre que se trocarmos $1/2$ por $1/p(|w|)$ ou por $1-(1/2)^{p(|w|)}$, para qualquer polinômio positivo p nas definições de RP e de coRP, obteremos as mesmas classes de linguagens.

Exercício 4.40. Prove que se trocarmos a constante na definição de sistema interativo por $(1/2)^{n^c}$, para qualquer constante positiva c então a classe de linguagens reconhecidas não muda, ou seja, é IP.

Exercício 4.41. (Goldwasser, Micali e Rackoff, 1989) Sejam a e b inteiros, $0 < b < a$, tais que $\text{mdc}(a,b) = 1$. Prove que a linguagem $\text{QR} := \{\langle a,b\rangle : b \text{ é um resíduo quadrático módulo } a\}$ admite o seguinte sistema de prova com conhecimento zero: com entrada a, b (**P** conhece supostamente conhece s tal que $b = s^2 \pmod a$). **P** sorteia s' e envia $b' = bs'^2 \pmod a$. **V** sorteia um bit i e envia. Se $b = 0$ então **P** revela ss' (uma raiz para x'); se $b = 1$ then **P** revela s' (uma raiz para $x'x^{-1}$). **V** verifica se o valor revelado é uma raiz de $x'x^{-b}$, se for aceita.

Exercício 4.42. Como consequência de $\text{AM}(p(n)) = \text{AM}(p(n)+1) = \text{MA}(p(n)+1)$ temos: se $\text{coNP} \subset \text{AM}$ então a hierarquia polinomial colapsa, $\Sigma_2 = \Pi_2 = \text{AM}$ (veja o Problema 14, página 208, e o parágrafo seguinte). Para provar esse corolário

1. Prove que $\text{AM} \subset \Pi_2$ e que $\text{MA} \subset \Sigma_2 \cap \Pi_2$ (generalize a prova de que $\text{BPP} \subset \text{PH}$, seção 4.2.3).
2. Prove que se $L \in \Sigma_2$ então existe $L_1 \in \text{coNP}$ tal que

$$w \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|w|)},\ (w,u) \in L_1$$

   para algum polinômio p (veja (4.3), página 200). Conclua que $L \in \text{AM}$.
3. Prove que $\Sigma_2 = \Pi_2 = \text{AM}$.

Exercício 4.43. (Arora e Barak, 2009, Lema 9.2) Assuma que $\text{P} = \text{NP}$. Suponha um sistema de codificação tal que $D_k(E_k(p)) = p$, com $|p| < |k|$, com a funções D_k e E_k computáveis em tempo polinomial. Mostre que existe um algoritmo de tempo polinomial tal que para todo w, existem $x_0, x_1 \in \{0,1\}^{|w|}$ com

$$\mathop{\mathbb{P}}_{\substack{b \in_R \{0,1\} \\ k \in_R \{0,1\}^n}} [A(E_k(x_b)) = b] \geqslant \frac{3}{4}$$

em que $n < |w|$.

Exercício 4.44. Uma variável aleatória X_n que assume valores em $\{0,1\}^{\ell(n)}$, em que ℓ é polinomial em n, é eficientemente *amostrável* se existe um algoritmo probabilístico de tempo polinomial M tal

que, se 1^n denota a palavra com n ocorrências de 1, então $\mathsf{M}(1^n)$ e X_n têm mesma distribuição para todo natural n. Assuma que temos um sistema criptográfico em que as funções de codificação em $\mathcal{E}$ usam chaves de tamanho n para mensagens de tamanho $\ell(n)$. Tal sistema é dito **semanticamente seguro** se para qualquer família $\{\mathsf{X}_n : n \in \mathbb{N}\}$ de variáveis aleatórias eficientemente amostráveis, qualquer $f : \{0,1\}^* \to \{0,1\}$ eficientemente computável e qualquer algoritmo probabilístico A de tempo polinomial, existe um algoritmo probabilístico B de tempo polinomial tal que

$$\left|\mathbb{P}[\,\mathsf{A}(\mathsf{E}_k(\mathsf{X}_n)) = f(\mathsf{X}_n)\,]\mathbb{P}[\,\mathsf{B}(1^n) = f(\mathsf{X}_n)\,]\right| < \frac{1}{n^d} \tag{4.11}$$

para todo $d > 0$ e $k \in_{\mathrm{R}} \{0,1\}^n$. Em (4.11) X_n é uma distribuição qualquer sobre os textos de comprimento n, $f(x)$ modela uma informação qualquer obtida de x. A equação nos diz que A não consegue computar $f(x)$ a partir de uma codificação de x com probabilidade melhor que chutando a partir do conhecimento da distribuição X_n. Considere $\mathsf{G} : \{0,1\}^n \to \{0,1\}^\ell$ um gerador pseudoaleatório. Prove que um sistema criptográfico com $\mathsf{E}_k(x) = x \oplus \mathsf{G}(k)$ e $\mathsf{D}_k(y) = y \oplus \mathsf{G}(k)$ é semanticamente seguro.

Exercício 4.45. Prove que $\mathsf{BPP} \subset \mathsf{PZK}$.

Exercício 4.46. (Vadhan e Goldwasser, 1999, pág. 21) Prove que $\textsc{noniso} \in \mathsf{PZK}$ exibindo um simulador para o protocolo da página 208.

Exercício 4.47. O objetivo desse exercício é mostrar que se um adversário consegue decifrar 1% das mensagens codificadas com o RSA, então ele consegue decifrar todas. Suponha que $\mathsf{E}^{-1}_{(e,n)}$ pode ser computada para $\varepsilon\varphi(n)$ elementos do $\mathbb{Z}^*_n$ em tempo t. Prove os enunciados a seguir e conclua que existe um algoritmo aleatorizado que computa $\mathsf{E}_{(e,n)}(y)$, para todo y, em tempo esperado t/ε.

- Mostre que se $z \in_{\mathrm{R}} \mathbb{Z}^*_n$ então z, z^e e $z^e y$ são identicamente distribuídos para todo $y \in \mathbb{Z}^*_n$.
- Suponha A um algoritmo de tempo t para o qual existe $\mathsf{Y} \subseteq \mathbb{Z}^*_n$ com $\varepsilon\varphi(n)$ elementos tal que $\mathsf{A}(y)^e = y$ para todo $y \in \mathsf{Y}$. Prove que

$$\mathop{\mathbb{P}}_{z \in_{\mathrm{R}} \mathbb{Z}^*_n}[\, v = a(z) \mid v^e = z\,] \geqslant \varepsilon$$

5 | PASSEIOS ALEATÓRIOS

Notas de aula por J. Donadelli (cmcc–ufabc)

Por todo este capítulo adotamos a seguinte notação

$$[X_1 \leqslant k_1,\, X_2 \leqslant k_2, \ldots,\, X_j \leqslant k_j] := [X_1 \leqslant k_1] \cap [X_2 \leqslant k_2] \cap \cdots \cap [X_j \leqslant k_j]$$

para simplificar a escrita.

5.1 UMA INTRODUÇÃO AOS PASSEIOS ALEATÓRIOS

Consideremos um jogo em que um jogador desafia uma banca com apostas de R$1,00 contra a probabilidade de sair coroa ao lançar uma moeda em rodadas que são independentes. O jogador possui um capital inicial $t > 0$ e em cada rodada ganha uma unidade monetária caso o lançamento resulte cara ou perde uma unidade monetária caso o lançamento resulte coroa. O jogo termina quando o jogador alcança um capital desejado $m > t$ ou quando atinge a *ruína* com capital 0.

Sejam $p \in (0,1)$ a probabilidade do evento C dado por "o lançamento da moeda resulta em cara", $\mathbb{P}_t \in [0,1]$ a medida de probabilidade condicionada ao capital inicial $t \in \{1,\dots,m-1\}$, e F o evento "o jogador alcança a fortuna m".

Denotamos por P_t a probabilidade do jogador ganhar dado que iniciou com t reais, com $\mathsf{P}_0 = 0$ e $\mathsf{P}_m = 1$, e

$$\mathsf{P}_t = \mathbb{P}_t(\mathsf{F}) = \mathbb{P}_t(\mathsf{F} \mid \mathsf{C})\,\mathbb{P}_t(\mathsf{C}) + \mathbb{P}_t(\mathsf{F} \mid \overline{\mathsf{C}})\,\mathbb{P}_t(\overline{\mathsf{C}})$$

pela lei de probabilidade total. O evento C é independente do capital do jogador e do instante t do processo, logo

$$\mathbb{P}_t(\mathsf{C}) = p \quad \text{e} \quad \mathbb{P}_t(\mathsf{F} \mid \mathsf{C}) = \mathbb{P}_{t+1}(\mathsf{F}),$$

para todo $0 < t < m$ inteiro, ou seja

$$\mathsf{P}_t = \mathsf{P}_{t+1} \cdot p + \mathsf{P}_{t-1} \cdot (1-p) \text{ para todo } t \in \{1,\dots,m-1\}, \quad \mathsf{P}_0 = 0, \quad \mathsf{P}_m = 1.$$

Rearranjando a equação de recorrência

$$\mathsf{P}_{t+1} - \mathsf{P}_t = \frac{p-1}{p}(\mathsf{P}_t - \mathsf{P}_{t-1})$$

fazendo $\mathsf{K} := (1-p)/p$ e $\mathsf{Q}_{t+1} := \mathsf{P}_{t+1} - \mathsf{P}_t$ obtemos $\mathsf{Q}_{t+1} = \mathsf{K}\mathsf{Q}_t$ com $\mathsf{Q}_1 = \mathsf{P}_1$, portanto, $\mathsf{Q}_{t+1} = \mathsf{K}^t\mathsf{P}_1$, ou seja, $\mathsf{P}_{t+1} = \mathsf{P}_t + \mathsf{K}^t\mathsf{P}_1$. Iterando essa recorrência

$$\mathsf{P}_{t+1} = (1 + \mathsf{K} + \mathsf{K}^2 + \mathsf{K}^3 + \mathsf{K}^4 + \cdots + \mathsf{K}^{t-1} + \mathsf{K}^t) \cdot \mathsf{P}_1 \tag{5.2}$$

e tratamos essa somatória em dois casos: (1) se $\mathsf{K} = 1$, isto é $p = 1 - p = 1/2$, então a somatória é $t+1$ e (2) senão $p \neq 1-p$ e a somatória é $(1-\mathsf{K}^{t+1})/(1-\mathsf{K})$ (soma de uma progressão geométrica).

Fazendo $t = m-1$ e de $\mathsf{P}_m = 1$ podemos concluir que $\mathsf{P}_1 = 1/m$, se $p = 1/2$, e $\mathsf{P}_1 = (1-\mathsf{K})/(1-\mathsf{K}^m)$, se $p \neq 1/2$. De volta para (5.2), com essas informações nós deduzimos que para $1 \leqslant t \leqslant m$

$$\mathsf{P}_t = \begin{cases} \frac{t}{m} & \text{se } p = 1/2 \\ \frac{1-\mathsf{K}^t}{1-\mathsf{K}^m} & \text{se } p \neq 1/2 \end{cases}, \quad \text{com } \mathsf{K} = \frac{1-p}{p}.$$

Para $K \neq 1$, se a chance de ganho em cada aposta é ligeiramente maior do que a de perda, digamos que $K = 0{,}9$, e se o jogador começa com t reais e almeja dobrar esse capital, ganhar mais t reais, antes de perder os t reais, temos da equação acima que a probabilidade é

$$\frac{(1-0{,}9^t)}{(1-0{,}9^{2t})} = \frac{1}{1+0{,}9^t}$$

para $t = 2$ a probabilidade é aproximadamente 0,55, para $t = 5$ a probabilidade é aproximadamente 0,65 e para $t = 50$ a probabilidade é aproximadamente 0,99. Agora, se $K = 0{,}49$, caso em que o ganho é bem mais favorável que a perda, então para $t = 2$ a probabilidade é aproximadamente 0,8, para $t = 5$ a probabilidade é aproximadamente 0,97 e para $t = 50$ a probabilidade é muito próxima de 1.

Se $1/2 < p < 1$ então $K \in (0,1)$ logo $K^m \to 0$ quando $m \to \infty$, portanto, $\lim_{m\to\infty} P_m = 1 - K^t \in (0,1)$, ou seja, há a possibilidade de que o jogador nunca quebre e se torne infinitamente rico.

Por outro lado, se $K = 10/9$, o ganho é ligeiramente desvantajoso, então a probabilidade de dobrar a fortuna antes de perder tudo é aproximadamente 0,22 para $t = 2$, aproximadamente 0,37 para $t = 5$ e aproximadamente 0,005 para $t = 50$.

No caso $p \leqslant 1/2$ temos $K > 1$ logo $\lim_{m\to\infty} P_m = 0$, então com probabilidade 1 o jogador perde todo seu dinheiro se almeja aumentar indefinidamente sua fortuna.

O JOGO ACABA? Notemos que há infinitas possibilidades de realizar o jogo sem que ele acabe, entretanto a probabilidade desse evento ocorrer é 0. De fato, se Q_t agora denota a probabilidade do evento "o jogo não acaba dado que começa no estado t" então

$$Q_t = pQ_{t+1} + (1-p)Q_{t-1} \ (0 < t < m), \quad Q_0 = Q_m = 0.$$

Como no recorrência para P_t temos $Q_t = Q_1(1 + K + \cdots + K^{t-1})$ para todo $0 < t < m$, e como $Q_m = 0$ deduzimos que $Q_1 = 0$ e, portanto, $Q_t = 0$ para todo t, ou seja, o jogo acaba com probabilidade 1.

Em quantas rodadas jogo acaba? Denotemos por E_t o valor esperado da variável aleatória Y que conta o número de rodadas até o jogo terminar dado que o capital inicial é t. Pela lei da esperança total

$$\begin{aligned} E_t &= \mathbb{E}[Y \mid C]p + \mathbb{E}[Y \mid \overline{C}](1-p) \\ &= (1 + E_{t+1})p + (1 + E_{t-1})(1-p) \\ &= 1 + E_{t+1}p + E_{t-1}(1-p) \end{aligned}$$

com $E_0 = E_m = 0$ as condições de contorno. Reescrevendo essa igualdade e fazendo $q = 1 - p$ obtemos a seguinte equação de recorrência de segunda ordem e não-homogênea

$$pE_{t+1} - E_t + qE_{t-1} = -1. \tag{5.3}$$

A parte homogênea da equação de recorrência, $pE_{t+1} - E_t + qE_{t-1} = 0$, tem solução da forma $AK^t + B$ quando $p \neq q$ e $K = q/p$, com A e B dependendo das condições de contorno, como vimos acima. Toda solução de (5.3) é da forma $(AK^t + B) + f(t)$ com $f(t)$ solução de (5.3) da forma $\alpha t + \beta$; substituindo em (5.3) e resolvendo para α e β, resulta em uma solução $f(t) = t/(1-2p)$. Assim, $E_t = AK^t + B + t/(1-2p)$ e das condições de contorno obtemos

$$E_t = \begin{cases} \frac{t}{1-2p} - \frac{m}{1-2p} \cdot \frac{1-K^t}{1-K^m} & , \text{ se } p \neq 1/2 \\ t(m-t) & , \text{ se } p = 1/2. \end{cases}$$

No caso $p < 1/2$ temos $E_t \leqslant t/(1-2p)$ e $E_t \to t/(1-2p)$ quando $m \to \infty$. Curiosamente, o ganho esperado em cada rodada é $1 \cdot p + (-1) \cdot (1-p) = 2p - 1 < 0$ de modo que se o jogador perde essa quantia por rodada então o jogo dura $t/(1-2p)$ rodadas.

Esse jogo é um problema clássico da probabilidade conhecido como **ruína do jogador**.

UMA VERSÃO SEM RUÍNA Considere o caso em que o jogo termina somente quando o jogador alcança a fortuna m, toda vez que ele chega na ruína ele ganha R$ 1,00 e continua a jogar. No restante, as regras e a evolução do jogo são como no jogo tradicional. Vamos calcular o valor esperado para o número de passos até a fortuna a partir do capital inicial $t \leqslant m$. O número esperado de rodadas condicionado ao valor inicial, denotado também por E_t, é modelado como na caso anterior

$$pE_{t+1} - E_t + qE_{t-1} = -1$$

para $0 < t < m$ e $q = 1-p$, porém com as condições de contorno agora são $E_0 = E_1 + 1$ e $E_m = 0$. Como $p+q = 1$, podemos escrever $pE_{t+1} - (p+q)E_t + qE_{t-1} = -1$ donde deduzimos $p(E_t - E_{t+1}) = q(E_{t-1} - E_t) + 1$ e fazendo $U_t := E_{t-1} - E_t$, logo $U_1 = 1$, e $K = q/p$, obtemos

$$U_{t+1} = K \cdot U_t + \frac{1}{p} \tag{5.4}$$

logo, para $K \neq 1$ (ou $p \neq 1/2$)

$$U_{t+1} = K^t \cdot U_1 + \frac{1}{p}\left(k^{t-1} + \cdots + K + 1\right) = K^t + \frac{1}{p}\left(\frac{1-K^t}{1-K}\right)$$

e retomando o valor esperado

$$E_t = E_{t+1} + K^t + \frac{1}{p}\left(\frac{1-K^t}{1-K}\right)$$

que, com a condição de parada $E_m = 0$, iteramos para obter

$$\begin{aligned} E_t =& E_m + \left(K^{m-1} + K^{m-2} + \cdots + K^t\right) + \frac{1}{p}\left(\frac{1-K^{m-1}}{1-K} + \frac{1-K^{m-2}}{1-K} + \cdots + \frac{1-K^t}{1-K}\right) \\ =& K^t\left(\frac{1-K^{m-t}}{1-K}\right) + \frac{1}{p}\left(\frac{(m-t)}{(1-K)} - \frac{K^t}{(1-K)}\left(\frac{1-K^{m-t}}{1-K}\right)\right) \\ =& \frac{1}{p}\frac{(m-t)}{(1-K)} + K^t\left(\frac{1-K^{m-t}}{1-K}\right)\left(1 - \frac{1}{p(1-K)}\right) = \frac{m-t}{2p-1} + \frac{K^t - K^m}{1-K} \cdot \frac{2p-2}{2p-1} \end{aligned}$$

Caso $p = 1/2$, da equação (5.4) obtemos $U_{t+1} = U_t + 2$, donde deduzimos $U_{t+1} = 2t + 1$, logo $E_t = E_{t+1} + 2t + 1$ com $E_m = 0$, portanto $E_t = E_m + \sum_{j=t}^{m-1} 2j + 1 = m^2 - t^2$. Assim,

$$E_t = \begin{cases} \frac{m-t}{2p-1} + \frac{K^t - K^m}{1-K} \cdot \frac{2p-2}{2p-1} & \text{, se } p \neq \frac{1}{2} \\ m^2 - t^2 & \text{, se } p = \frac{1}{2}. \end{cases}$$

5.1.1 PASSEIOS ALEATÓRIOS

Passeio aleatório é um modelo matemático que formaliza a ideia de se mover em direções aleatórias em algum ambiente, como um grafo por exemplo. Em Ciência da Computação passeios aleatórios em grafos modelam, por exemplo, o tráfego de informação por nós de uma rede sem fio e foi o método que originou o mecanismo de pesquisa do Google para procurar e classificar as páginas *web* da Internet.

A evolução de um processo aleatório pode ser modelada por uma sequência $(X_t : t \in T)$ de variáveis aleatórias que assumem valores num conjunto S de *estados* e são indexadas pelo parâmetro t que chamamos, genericamente, de *tempo*. Usualmente, no caso discreto, $T = \mathbb{Z}^+$ e S enumerável. Neste capítulo, estudaremos passeios aleatórios que são instância de um processo estocástico markoviano o que, intuitivamente, significa que a qualquer momento o próximo estado do sistema depende de seu estado atual, mas não de seu passado. Além disso, também consideramos apenas o caso em que a probabilidade de transição entre os estados depende dos estados mas não depende do tempo.

Exemplo 5.1 (*passeio aleatório simples nos inteiros*). As variáveis aleatórias X_t assumem valores em $\mathbb{Z}$, para todo inteiro $t \geqslant 0$, a começar do *estado inicial* $X_0 = a$. Com isso, queremos dizer que $\mathbb{P}[X_0 = a] = 1$ e a partir do estado X_t no instante t, em cada passo, o estado do passeio se move de $+1$ ou de -1 com mesma probabilidade, independentemente do passado e de t.

Por exemplo, para $a = 0$, quando $t = 1$ temos o evento $[X_t = -1]$ com probabilidade $1/2$ ou o evento $[X_t = +1]$ com probabilidade $1/2$. No instante seguinte, $t = 2$, temos $[X_t = -2]$ com probabilidade $1/4$ ou $[X_t = +2]$ com probabilidade $1/4$ ou $[X_t = 0]$ com probabilidade $1/2$. No terceiro passo temos $[X_t = -3]$ com probabilidade $1/8$ ou $[X_t = +3]$ com probabilidade $1/8$ ou $[X_t = -1]$ com probabilidade $3/8$ ou $[X_t = +1]$ com probabilidade $3/8$.

As transições entre estados têm as probabilidades

$$\mathbb{P}[X_{t+1} = i + 1 \mid X_t = i] = \mathbb{P}[X_{t+1} = i - 1 \mid X_t = i] = \frac{1}{2}$$

para todo inteiro i e todo inteiro positivo t. O estado X_t do passeio aleatório no instante $t \geqslant 1$ é dado por

$$X_t = X_0 + \sum_{n=1}^{t} Z_n = a + \sum_{n=1}^{t} Z_n, \text{ com } Z_n \sim \text{Bernoulli}(1/2) \text{ independentes}$$

e, para todo n, $\{Z_n, X_0, X_1, \ldots, X_{n-1}\}$ também independentes. Assim, para qualquer inteiro b

$$\mathbb{P}[\, X_t = j \mid X_0 = a \,] = \mathbb{P}\left[\sum_{n=1}^{t} Z_n = j - a\right] = \mathbb{P}[\, X_t = j + b \mid X_0 = a + b \,]$$

ou seja, o passeio é homogêneo no espaço. Também, para qualquer inteiro positivo s

$$\mathbb{P}[\, X_t = j \mid X_0 = a \,] = \mathbb{P}\left[\sum_{n=1}^{t} Z_n = j - a\right] = \mathbb{P}\left[\sum_{n=1+s}^{t+s} Z_n = j - a\right] = \mathbb{P}[\, X_{t+s} = j \mid X_s = a \,]$$

ou seja, o passeio é homogêneo no tempo.

Nesse processo, o estado futuro X_{t+1} depende somente do estado presente X_t e não depende dos estados passados $X_0, X_1, \ldots, X_{t-1}$, ou seja, conhecido X_t, a distribuição de X_{t+1} depende de Z_{t+1} somente, não depende de X_0, $X_1, \ldots, X_{t-1}$,

$$\begin{aligned}
&\mathbb{P}[\, X_{t+1} = j \mid X_t = i,\, X_{t-1} = s_{t-1}, \cdots,\, X_1 = s_1,\, X_0 = a \,] \\
&= \mathbb{P}[\, Z_t = j - i \mid X_t = i,\, X_{t-1} = s_{t-1}, \cdots,\, X_1 = s_1,\, X_0 = a \,] \\
&= \mathbb{P}[\, Z_t = j - i \,] = \mathbb{P}[\, X_{t+1} = j \mid X_t = i \,]
\end{aligned}$$

para quaisquer $i, s_{t-1}, \cdots, s_1, a$ inteiros

$$\mathbb{P}[\, X_{t+1} = j \mid X_t = i,\, X_{t-1} = s_{t-1}, \cdots,\, X_1 = s_1,\, X_0 = a \,] = \mathbb{P}[\, X_{t+1} = j \mid X_t = i \,]. \tag{5.5}$$

Algumas perguntas interessantes nesse caso são: em média, quão longe da origem está passeio após n passos? Com que probabilidade em um determinado momento o passeio estará de volta na origem? O passeio aleatório continua voltando à origem ou, eventualmente, o passeio a deixa para sempre?

Esse passeio aleatório pelos inteiros pode ser facilmente generalizado para várias dimensões tomando os estados em $\mathbb{Z}^d$ e cada passo é escolhido com igual probabilidade dentre 2^d direções possíveis (figura 5.1). Curiosamente, o retorno para o estado inicial do passeio ocorre com probabilidade 1 nos casos $d = 1$ e $d = 2$, para $d \geqslant 3$ o passeio eventualmente o deixa e nunca mais retorna (exercício 5.53). Esse resultado foi provado por George Pólya em 1921 e motivou a anedota "um homem bêbado vai encontrar o caminho de casa, mas um pássaro bêbado pode se perder para sempre". ◊

Exemplo 5.2 (*modelo de difusão de Ehrenfest*). Um modelo idealizado de difusão foi proposto pelos físicos Tatiana e Paul Ehrenfest em 1907 para descrever a troca de calor entre dois sistemas em diferentes temperaturas. No modelo há dois compartimentos que chamaremos de A e B separados por uma membrana (figura 5.2). No início N moléculas de um gás estão no compartimento A e o compartimento B está vazio. Em cada instante uma única molécula troca de compartimento e todas as moléculas de um mesmo compartimento tem a mesma probabilidade de passar pela membrana.

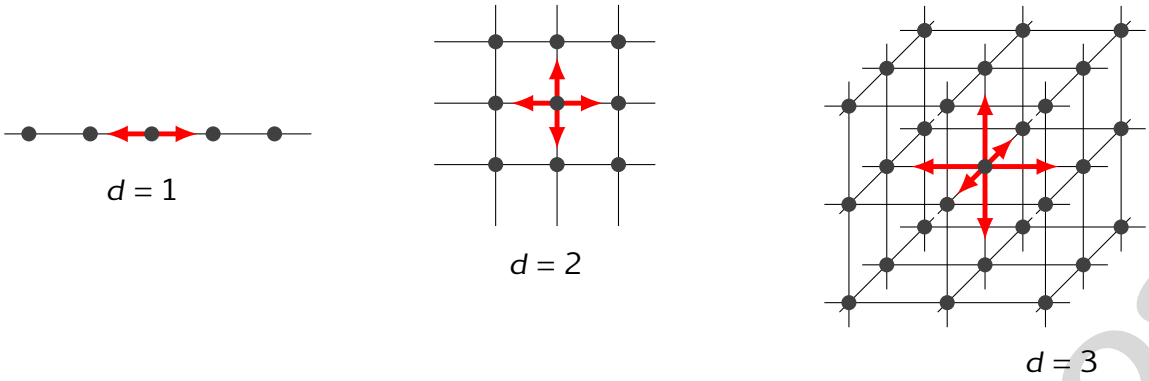


Figura 5.1: os 2^d possíveis passos no $\mathbb{Z}^d$ a partir de um estado, para $d = 1, 2, 3$.

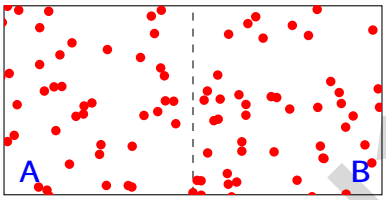


Figura 5.2: esquema do modelo de Ehrenfest.

O experimento idealizado por Ehrenfest é modelado por um passeio aleatório em $S := \{0, 1, \ldots, N\}$. Digamos que no instante $t \geqslant 0$ a quantidade de moléculas em B seja X_t, inicialmente $X_0 = 0$. Conhecido X_t, no próximo momento temos $X_{t+1} = X_t - 1$, caso uma molécula tenha passado de B para A, ou $X_{t+1} = X_t + 1$ caso uma molécula tenha passado de A para B. Essas variáveis aleatórias definem um processo estocástico sobre o conjunto de estados S e as transições com probabilidade positiva são $\mathbb{P}[\, X_1 = 1 \mid X_0 = 0 \,] = 1$ e para todo $t > 0$

$$\begin{aligned} \mathbb{P}[\, X_{t+1} = N - 1 \mid X_t = N \,] &= 1 \\ \mathbb{P}[\, X_{t+1} = k + 1 \mid X_t = k \,] &= \frac{N - k}{N} \\ \mathbb{P}[\, X_{t+1} = k - 1 \mid X_t = k \,] &= \frac{k}{N}; \end{aligned}$$

em qualquer outro caso $\mathbb{P}[\, X_{t+1} = j \mid X_t = i \,] = 0$. Esse processo satisfaz a equação (5.5)? ◊

Exemplo 5.3 (*Ruína do jogador*). Na ruína do jogador, o nosso primeiro exemplo, o capital do jogador no instante t é uma variável aleatória, digamos X_t, e $(X_t \colon t \geqslant 0)$ é um passeio aleatório pelos estados $S =:= \{0, 1, \ldots, m\}$ com

$$\begin{aligned} \mathbb{P}[\, X_{t+1} = 0 \mid X_t = 0 \,] &= 1, & \mathbb{P}[\, X_{t+1} = i + 1 \mid X_t = i \,] &= p, \\ \mathbb{P}[\, X_{t+1} = m \mid X_t = m \,] &= 1, & \mathbb{P}[\, X_{t+1} = i - 1 \mid X_t = i \,] &= q \end{aligned}$$

para todo $1 \leqslant i < N$ e todo $t \geqslant 0$. Esse processo satisfaz a equação (5.5)? ◊

5.1.2 PROPRIEDADE DE MARKOV

Os três exemplos apresentados acima são exemplos de processos estocásticos com a propriedade de Markov. Um processo estocástico $(X_t\colon t \in \mathbb{Z}^+)$ possui a **propriedade de Markov** se o comportamento probabilístico do processo no futuro (X_{t+1}), dado o presente (X_t), é condicionalmente independente do passado $(X_0, X_1, \ldots, X_{t-1})$, o que é expresso pela condição dada na equação (5.5) acima.

Neste capítulo todos os processos considerados são desse tipo, a menos que dito o contrário. Também, os processo que estudaremos são homogêneos, isto é, a probabilidade de transição não depende de t, apenas dos estados

$$p_{i,j} := \mathbb{P}[\, X_1 = j \mid X_0 = i \,] = \mathbb{P}[\, X_{t+1} = j \mid X_t = i \,].$$

Um processo $(X_t\colon t \geqslant 0)$ é *markoviano* se, e só se, para qualquer sequência de estados $(i_0, i_1, \ldots, i_n)$ temos que a lei conjunta $\mathbb{P}[X_0 = i_0, X_1 = i_1, \ldots, X_n = i_n]$ é dada por

$$\begin{aligned}
&\mathbb{P}[X_0 = i_0, X_1 = i_1, \ldots, X_n = i_n] \\
&= \mathbb{P}[X_0 = i_0] \cdot \mathbb{P}[\, X_1 = i_1 \mid X_0 = i_0 \,] \cdot \mathbb{P}[\, X_2 = i_2 \mid X_1 = i_1, X_0 = i_0 \,] \cdots \mathbb{P}[\, X_n = i_n \mid X_{n-1} = i_{n-1}, \ldots, X_0 = i_0 \,] \\
&= \mathbb{P}[X_0 = i_0] p_{i_0,i_1} p_{i_1,i_2} \cdots p_{i_{n-1},i_n}.
\end{aligned}$$

É bastante frequente termos a distribuição inicial concentrada num estado k, isto é $\mathbb{P}[X_0 = k] = 1$. Nesse caso, denotamos a distribuição por δ_k

$$\delta_k(x) = \begin{cases} 1, & \text{se } x = k \\ 0, & \text{se } x \neq k. \end{cases}$$

Não é difícil verificar que a condição de Markov dada na equação (5.5) é equivalente à condição

$$\mathbb{P}[\, X_{t+n} = j \mid X_t = i,\ X_{t-1} = i_{t-1}, \cdots,\ X_0 = i_0 \,] = \mathbb{P}[\, X_{t+n} = j \mid X_t = i \,]$$

para quaisquer estados $i, s_{t-1}, \cdots, s_1, a$, para todo $t \geqslant 0$ e todo $n \geqslant 1$.

Exercício 5.4 (propriedade de Markov). Prove que se $(X_t\colon t \geqslant 0)$ com a distribuição inicial ν e probabilidades de transição $p_{i,j}$ é markoviano então $(X_{t+m}\colon t \geqslant 0)$ condicionado a $X_m = i$, com distribuição inicial δ_i e probabilidades de transição $p_{i,j}$ é markoviano e independente de $X_0, \ldots, X_m$.

A PROPRIEDADE FORTE DE MARKOV No experimento de lançar um dado até que ocorra o resultado 3 ocorra, o instante em que tal evento ocorre é uma variável aleatória T e o evento $[T = n]$ é completamente determinado pelos valores $X_1, \ldots, X_n$ dos resultados observados no lançamento até o instante n, $[T = n] = [X_1 \neq 3, X_2 \neq 3, \ldots, X_n = 3]$.

No caso em que $T\colon \Omega \to \{0,1,\ldots\} \cup \{\infty\}$ é uma variável aleatória, definida no mesmo espaço amostral que as variáveis aleatórias de um processo estocástico $(X_t\colon t \in \mathbb{Z}^+)$, e $[T = n]$ é completamente determinado[1] pelos valores $X_0, X_1, \ldots, X_n$ é chamada **tempo de parada**.

No caso da ruína do jogador o tempo em que atinge um dos estados 0 ou m, aquele que ocorrer primeiro, é um tempo de parada. "Intuitivamente, assistindo o processo sabemos quando T ocorre no instante que ocorre. Se temos que parar em T, sabemos quando parar" (Norris, 1997).

Quando vale a propriedade de Markov dizemos que o comportamento futuro do processo depende do presente e não depende do passado. Na propriedade forte é análogo, exceto que o presente é um instante aleatório, como no caso do lançamento de um dado, ou seja, a propriedade de Markov vale para todo tempo de parada. O ponto importante a ser observado é que se E é um evento determinado por $X_0, \ldots, X_T$, então $E \cap [T = n]$ é determinado por $X_0, \ldots, X_n$, para todo n. Nesse caso, pela propriedade de Markov

$$\begin{aligned} &\mathbb{P}([X_T = i_0,\ X_{T+1} = i_1, \cdots,\ X_{T+n} = i_n] \cap B \cap [T = n] \cap [X_T = i]) = \\ &\mathbb{P}(\,[X_0 = i_0,\ X_1 = i_1, \cdots,\ X_n = i_n] \mid [X_0 = i]\,)\mathbb{P}(B \cap [T = n] \cap [X_T = i]) \end{aligned} \tag{5.6}$$

A propriedade forte de Markov é a seguinte: para todo $n \geqslant 1$ e todos $j, i, i_{t-1}, \cdots, i_0$

$$\mathbb{P}[\,X_{T+n} = j \mid X_T = i,\ X_{T-1} = i_{T-1}, \cdots,\ X_0 = i_0\,] = \mathbb{P}[\,X_{T+n} = j \mid X_T = i\,]$$

o que vale sempre que T é um tempo de parada. Assim, condicionado a $T < \infty$ e $X_T = i$ temos que $(X_{T+n}\colon n \in \mathbb{Z}^+)$ é um processo markoviano com estado inicial i, independente de $X_0, X_1, \ldots, X_T$ e com as mesmas probabilidades de transição. A propriedade de Markov forte diz, em termos gerais, que uma cadeia de Markov "regenera-se" ou recomeça em um momento de parada. Se T é um tempo de

[1] Alternativamente, podemos dizer que a função $\mathbb{1}_{[T=n]}$ é dada pelas variáveis $X_0, \ldots, X_n$.

parada de $(X_0, X_1, \ldots)$ tal que $\mathbb{P}[\,T < \infty\,] = 1$. Então

$$\begin{aligned}
\mathbb{P}[\,X_{T+n} = j \mid X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0\,] &= \sum_{m\geqslant 0} \mathbb{P}[\,X_{T+n} = j, T = m \mid X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0\,] \\
&= \sum_{m\geqslant 0} \mathbb{P}[\,X_{T+n} = j \mid X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0, T = m\,] \cdot \mathbb{P}[\,T = m \mid X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0\,] \\
&= \sum_{m\geqslant 0} \frac{\mathbb{P}[\,X_{T+n} = j, X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0, T = m\,]}{\mathbb{P}[\,T = m, X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0\,]} \cdot \mathbb{P}[\,T = m \mid X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0\,] \\
&= \sum_{m\geqslant 0} \frac{\mathbb{P}[\,X_{m+n} = j, X_m = i,\, X_{m-1} = i_{m-1}, \cdots,\, X_0 = i_0\,]}{\mathbb{P}[\,X_m = i,\, X_{m-1} = i_{m-1}, \cdots,\, X_0 = i_0\,]} \cdot \mathbb{P}[\,T = m \mid X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0\,] \\
&= \sum_{m\geqslant 0} \mathbb{P}[\,X_{m+n} = j \mid X_m = i,\, X_{m-1} = i_{m-1}, \cdots,\, X_0 = i_0\,] \cdot \mathbb{P}[\,T = m \mid X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0\,] \\
&= \sum_{m\geqslant 0} \mathbb{P}[\,X_{m+n} = j \mid X_m = i\,] \cdot \mathbb{P}[\,T = m \mid X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0\,] \\
&= \mathbb{P}[\,X_{m+n} = j \mid X_m = i\,] \cdot \sum_{m\geqslant 0} \mathbb{P}[\,T = m \mid X_T = i,\, X_{T-1} = i_{T-1}, \cdots,\, X_0 = i_0\,] \\
&= \mathbb{P}[\,X_{m+n} = j \mid X_m = i\,]
\end{aligned}$$

logo (5.6) fica verificada.

Concluímos que todo processo $(X_t\colon t \in \mathbb{Z}^+)$ markoviano satisfaz a condição forte de Markov.

EQUAÇÃO DE WALD Sejam $(X_n\colon n \geqslant 1)$ variáveis aleatórias independentes e com a mesma distribuição de valor esperado $\mathbb{E}\,X$ tal que $\mathbb{E}\,|X| < \infty$. Suponha que N seja um tempo de parada com respeito a tal sequência com $\mathbb{E}\,N < \infty$. Então

$$\sum_{n\geqslant 1}^{N} X_n = \sum_{n\geqslant 1} X_n \mathbb{1}_{[N>n-1]},$$

tomando a esperança

$$\mathbb{E}\left[\sum_{n\geqslant 1}^{N} X_n\right] = \mathbb{E}\left[\sum_{n\geqslant 1} X_n \mathbb{1}_{[N>n-1]}\right] = \sum_{n\geqslant 1} \mathbb{E}\Big[X_n \mathbb{1}_{[N>n-1]}\Big]$$

pois as esperanças são finitas, $\mathbb{E}\,|X| < \infty$ e $\mathbb{E}\,N < \infty$.

O evento $[N > n-1]$ pode depender (no máximo) de $X_1, \ldots, X_{n-1}$. A variável X_n é independente de $X_1, \ldots, X_{n-1}$, portanto independente de $[N > n-1]$, assim $\mathbb{E}[X_n \mathbb{1}_{[N>n-1]}] = \mathbb{E}[X]\mathbb{P}[N > n-1]$. Assim, ficamos com

$$\mathbb{E}\left[\sum_{n\geqslant 1}^{N} X_n\right] = \mathbb{E}[X] \sum_{n\geqslant 1} \mathbb{P}[N > n-1] = \mathbb{E}[X] \sum_{n\geqslant 0} \mathbb{P}[N > n] = \mathbb{E}[X]\mathbb{E}[N].$$

5.1.3 2SAT

O 2SAT é o problema computacional de decidir satisfazibilidade de uma fórmula booleana 2-CNF, digamos $\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, sobre um conjunto de variáveis V. Esse problema tem solução de tempo polinomial (exercício 3.73, página 174). Abaixo descrevemos um algoritmo aleatorizado bem simples para o 2SAT, de tempo polinomial e que responde corretamente com alta probabilidade.

A partir de uma valoração qualquer das variáveis, se a fórmula não está satisfeita, escolhemos uma cláusula falsa e inverta o valor de uma das duas variáveis escolhida ao acaso. A cláusula escolhida ficará satisfeita, embora algumas outras podem se tornarem falsa. Esse processo lembra o do jogo da ruína sem-ruína, o "capital" do algoritmo é a quantidade de clausulas satisfeitas e a meta atingir m. Entretanto, em cada etapa não temos controle de quanto o capital muda.

Um outra ideia que contorna o problema apresentado no parágrafo acima é a seguinte. Suponha que haja uma valoração ν^* que torne Φ verdadeira. O algoritmo começa com uma valoração ν que é alterada durante a execução. O capital do algoritmo num dado instante é a quantidade de variáveis do conjunto V que recebem o mesmo valor lógico pelas funções ν e ν^*.

Enquanto houver cláusula falsa na fórmula de entrada, o algoritmo toma uma dessas cláusulas, sorteia um literal dessa cláusula e inverte o valor da variável dela. Notemos que se os dois literais da cláusula escolhida pelo algoritmo discordam de ν^*, então a fortuna do algoritmo aumenta de 1, certamente, senão apenas um literal discorda da valoração ν^* de modo que com probabilidade $1/2$ a fortuna aumenta de 1 ou diminui de $1/2$.

Assumindo que a fortuna do algoritmo pode ser modelada como a do jogador (do jogo sem ruína), temos a probabilidade $p \geqslant 1/2$ da fortuna incrementar de 1 e $q < 1/2$ de cair. Portanto, no pior caso, $p = q = 1/2$, o algoritmo termina, isto é encontra a fortuna n, em $O(n^2)$ passos em média pela nossa análise prévia. Eventualmente, o algoritmo pode terminar antes encontrando uma outra valoração que satisfaça a fórmula.

No caso da fórmula Φ não ser satisfazível precisamos de um mecanismo de interromper o algoritmo que não estrague muito a possibilidade dele encontrar a valoração no caso de fórmulas satisfazíveis. Usando a desigualdade de Markov, equação (3.33) na página 164, temos que a probabilidade de, no caso Φ satisfazível, o número de passos superar n^3 é menos que $1/n$, portanto, com alta probabilidade o algoritmo encontra ν^* ou outra valoração que satisfaça a fórmula.

Resumindo, o algoritmo sorteia uma valoração; enquanto houver cláusula falsa, toma uma cláusula, sorteia uma literal e inverte o valor, até que encontre uma valoração que satisfaça a fórmula

(que pode ser diferente de ν^*), ou que exceda um limitante pré-determinado de rodadas.

Instância: uma fórmula 2-CNF $\Phi = \{C_1, \ldots, C_m\}$ sobre as variáveis $\{v_1, \ldots, v_n\}$ e o número de tentativas k.

Resposta: *sim* se encontrou uma valoração que satisfaz Φ, caso contrário *não* com probabilidade de erro menor que $1/2^k$.

```
1 para i de 1 até n faça v_i ←_R {0,1};
2 repita
3     se Φ(v_1,...,v_n) = 1 então responda sim.
4     sorteie um literal ℓ de uma cláusula não satisfeita;
5     ℓ ← ℓ + 1 mod 2;
6 até que complete 2kn^2 rodadas;
7 responda não.
```

Algoritmo 42: 2SAT.

Se não existe valoração que satisfaça todas as cláusulas, o algoritmo termina após $2tn^2$ rodadas e responde corretamente. Vamos assumir que Φ é satisfazível. Seja ν uma valoração que satisfaz Φ e ν_i a valoração construída pelo algoritmo após a i-ésima rodada do laço interno, da linha 2, com ν_0 a valoração inicial construída na linha 1.

Denotemos por X_i o número de valores em comum nas valorações ν e ν_i, isto é, a quantidade de variáveis em V que *têm o mesmo valor* nessas duas valorações. Então $(X_i : i \geqslant 0)$ é uma coleção de variáveis aleatórias que assumem valor no conjunto de estados $S = \{0, 1, \ldots, n\}$. O algoritmo termina com $X_i = n$ ou quando ele encontra alguma valoração diferente que ν que satisfaça a fórmula booleana Φ.

Por exemplo, consideremos $\Phi = (v_1 \vee v_2) \wedge (v_2 \vee v_3) \wedge (\neg v_1 \vee \neg v_3)$ e ν dada por $(v_1, v_2, v_3) = (1, 1, 0)$. Se ν_i é $(v_1, v_2, v_3) = (0, 0, 1)$ então $X_i = 0$ de modo que $X_{i+1} = 1$ com probabilidade 1. Agora, se ν_i é $(v_1, v_2, v_3) = (0, 0, 0)$, então $X_i = 1$ (as valorações coincidem em v_3), a cláusula $v_1 \vee v_2$ é falsa, assim como $v_2 \vee v_3$; se a primeira cláusula é escolhida então

$$X_{i+1} = \begin{cases} 2, \text{ com probabilidade } 1 \\ 0, \text{ com probabilidade } 0 \end{cases}$$

senão a segunda será escolhida e

$$X_{i+1} = \begin{cases} 2, \text{ com probabilidade } 1/2 \\ 0, \text{ com probabilidade } 1/2. \end{cases}$$

Se ν_i é $(v_1, v_2, v_3) = (1, 0, 0)$, então $X_i = 2$ e a cláusula $v_2 \vee v_3$ é (a única) falsa de modo que $X_{i+1} = 1$ com probabilidade $1/2$ e $X_{i+1} = 3$ com probabilidade $1/2$.

Notemos que o valor da probabilidade condicional de X_{i+1} dado ν_i não depende apenas de X_i mas depende de quais variáveis estão de acordo com ν, o que depende do histórico de como chegamos a ν_i. Esse processo não é um passeio aleatório markoviano, isto é, viola a condição dada na equação (5.5). Se $C_k = \{\ell_{k_1}, \ell_{k_2}\}$ não está satisfeita por ν_i, então ambos os literais são falsos por ν_i, mas pelo menos um deles é satisfeito por ν, portanto, $\mathbb{P}[\, X_{i+1} = j+1 \mid X_i = j \,] \geqslant 1/2$. Por outro lado, $\mathbb{P}[\, X_{i+1} = j-1 \mid X_i = j \,] \leqslant 1/2$. Também, vale $\mathbb{P}[\, X_{i+1} = 1 \mid X_i = 0 \,] = 1$. Uma condição suficiente para o algoritmo responder corretamente é que em algum momento valha $X_i = n$, nesse caso, vamos assumir que $X_j = n$ para todo $j > i$.

Agora, definimos um passeio aleatório $(Z_i : i \geqslant 0)$ no mesmo conjunto de estados e pondo $Z_0 := X_0$ e, para $i \geqslant 1$, $Z_i := X_i - E_i$ com $E_0 := 0$ e

1. se $E_i = X_i$ e $X_i \geqslant 1$, então $E_{i+1} := X_{i+1} - 1$;
2. senão, se $E_i \neq X_i$ e
   (a) $X_i = n$, então $E_{i+1} := E_i + 1$ com probabilidade 1/2 e $E_{i+1} := E_i - 1$ com probabilidade 1/2;
   (b) $n \neq X_i \geqslant 1$ e ambos os literais da cláusula escolhida na iteração $i+1$ do laço têm valores diferentes em ν_i e ν (portanto $X_{i+1} = X_i + 1$), então $E_{i+1} := E_i$ com probabilidade 1/2 e $E_{i+1} := E_i + 2$ com probabilidade 1/2;
3. caso contrário $E_{i+1} := E_i$.

Se $Z_i = 0$, então $E_i = X_i$. Se $X_i = 0$ então $X_{i+1} = 1$, ademais $E_{i+1} = E_i = 0$ pelo item 3 e por hipótese. Logo $Z_{i+1} = X_{i+1} - E_{i+1} = 1 - 0 = 1$. Se $X_i \geqslant 1$, então $E_{i+1} = X_{i+1} - 1$, logo $Z_{i+1} = X_{i+1} - E_{i+1} = 1$. Portanto

$$\mathbb{P}[\, Z_{i+1} = 1 \mid Z_i = 0 \,] = 1.$$

Assumamos que $Z_i = j \geqslant 1$. Se $X_i = n$ então, com probabilidade 1/2, temos $E_{i+1} = E_i + 1 = X_i - Z_i + 1 = n - j + 1$ ou $E_{i+1}E_i - 1 = n - j - 1$, logo $Z_{i+1} = X_{i+1} - E_{i+1} = n - (n - j \pm 1)$, ou seja, $Z_{i+1} = j + 1$ com probabilidade 1/2 e $Z_{i+1} = j - 1$ com probabilidade 1/2. Se $X_i \geqslant 1$ e ambos os literais da cláusula escolhida na iteração $i+1$ do laço têm valores diferentes em ν_i e ν, então $Z_{i+1} = X_{i+1} - E_{i+1} = X_i + 1 - E_i = Z_i + 1 = j + 1$ com probabilidade 1/2 e $Z_{i+1} = X_{i+1} - E_{i+1} = X_i + 1 - E_i - 2 = Z_i - 1 = j - 1$ com probabilidade 1/2. Caso contrário no máximo um literal da cláusula escolhida na iteração $i+1$ do laço têm valores diferentes em ν_i e ν, então $X_{i+1} = X_i \pm 1$ e $E_{i+1} := E_i$, portanto, $Z_{i+1} = X_i \pm 1 - E_i = j \pm 1$, cada um com probabilidade 1/2. Assim,

$$\mathbb{P}[\, Z_{t+1} = i+1 \mid Z_t = i \,] = \mathbb{P}[\, Z_{t+1} = i-1 \mid Z_t = i \,] = 1/2.$$

Nesse passeio podemos usar os resultados do XXXX para garantir que o número esperado de passos até o passeio (Z_t) atingir o estado n é no máximo n^2. Por definição $Z_t \leqslant X_t$ para todo $t \geqslant 0$, portanto, o número esperado de passos até o passeio (X_t) atingir o estado n é no máximo n^2.

Com que probabilidade o laço interno do algoritmo 42 não acha a valoração v? Se R é o número de rodadas de trocas de bits até acha uma valoração específica, então $\mathbb{E}\,\mathsf{R} \leqslant n^2$ e pela desigualdade de Markov, equação (6.1),

$$\mathbb{P}[\,\mathsf{R} \geqslant 2n^2\,] \leqslant \frac{1}{2}.$$

O algoritmo falha em cada uma das tentativas do laço externo com probabilidade no máximo $(1/2)^t$.

3SAT O que acontece quando aplicamos a mesma estratégia para o problema k-SAT? No caso $k = 3$ a generalização imediata do algoritmo e da análise acima nos leva ao passeio (X_t) com

$$\mathbb{P}[\,\mathsf{X}_{t+1} = j+1 \mid \mathsf{X}_t = j\,] \geqslant 1/3$$
$$\mathbb{P}[\,\mathsf{X}_{t+1} = j-1 \mid \mathsf{X}_t = j\,] \leqslant 2/3$$

que domina estocasticamente o passeio (Z_t) $(\mathsf{X}_t \geqslant \mathsf{Z}_t)$ tal que

$$\mathbb{P}[\,\mathsf{Z}_{t+1} = 1 \mid \mathsf{Z}_t = 0\,] = \mathbb{P}[\,\mathsf{Z}_{t+1} = n \mid \mathsf{Z}_t = n\,] = 1$$
$$\mathbb{P}[\,\mathsf{Z}_{t+1} = j+1 \mid \mathsf{Z}_t = j\,] = 1/3$$
$$\mathbb{P}[\,\mathsf{Z}_{t+1} = j-1 \mid \mathsf{Z}_t = j\,] = 2/3$$

que, infelizmente, é mais provável decrescer do que crescer. Isso faz com que o número esperado de rodadas seja exponencial, $O(2^n)$.

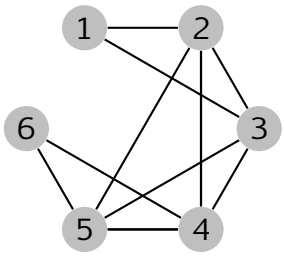
Exercício 5.5. Prove que o passeio aleatório (Z_t) definido acima tem tempo esperado $O(2^n)$ para atingir o estado n.

5.2 PASSEIOS ALEATÓRIOS EM GRAFOS

Seja $G = (V, E)$ um grafo com número finito de vértices. Para os nossos propósitos podemos assumir, sem perda de generalidade, que os vértices são dados por $V = \{1, 2, \ldots, n\}$, com $n \geqslant 2$. Um grafo pode ser representado por sua **matriz de adjacências** $A_G = (a_{i,j})$ dada por

$$a_{i,j} := \begin{cases} 1 & \text{se } \{i,j\} \in E \\ 0 & \text{caso contrário.} \end{cases}$$

Para o grafo $(\{1,2,3,4,5,6\}, \{\{1,2\},\{1,3\},\{2,3\},\{2,4\},\{2,5\},\{3,4\},\{3,5\},\{4,5\},\{4,6\},\{5,6\}\})$, por exemplo, a matriz de adjacências $A = A_G$ que o representa, juntamente com um diagrama, são dados na figura 5.3 abaixo.



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

Figura 5.3: um grafo e sua matriz de adjacências.

Um **passeio** em um grafo é, simplesmente, uma sequência $(v_0, v_1, \ldots, v_k)$ de vértices do grafo com vértices consecutivos *adjacentes*, isto é, $\{v_i, v_{i+1}\}$ é aresta para todo $0 \leqslant i \leqslant k-1$.

Um **passeio aleatório simples** pelos vértices do grafo G é um processo estocástico $(X_t : t \geqslant 0)$, com $X_t \in V$ para todo t, tal que

$$p_{i,j} = \mathbb{P}[\, X_{t+1} = j \mid X_t = i \,] := \begin{cases} 1/d_G(i), & \text{se } \{i,j\} \in E(G) \\ 0, & \text{caso contrário} \end{cases}$$

onde $d_G(i)$, o grau do vértice i, é a quantidade de arestas que incidem em i, ou seja $d_G(i) = \sum_j a_{i,j}$. Esse passeio satisfaz a condição de Markov, equação (5.5).

A matriz $P = (p_{i,j})$ é a **matriz de transição** do passeio aleatório no grafo G, que também é dada por

$$P = D^{-1} \cdot A_G \tag{5.7}$$

para a matriz diagonal $D = (d_{i,j})$ dada por $d_{i,i} := d_G(i)$ e 0 nas outras entradas.

No caso do exemplo acima a matriz de transição no grafo é

$$\begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 1/4 & 0 \\ 1/4 & 1/4 & 0 & 1/4 & 1/4 & 0 \\ 0 & 1/4 & 1/4 & 0 & 1/4 & 1/4 \\ 0 & 1/4 & 1/4 & 1/4 & 0 & 1/4 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} = \begin{pmatrix} 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Exercício 5.6 (matriz estocástica). Uma **matriz estocástica** tem todas as entradas não negativas e cada linha soma 1. Prove que para todo grafo G com n vértices e todo inteiro $t \geqslant 1$, para P a matriz de transição como foi definida acima, a matriz P^t é estocástica.

Com essas definições temos uma representação elegante para a dinâmica do processo aleatório. Seja $\nu = (\nu_i)$ um **vetor estocástico**, isto é, com entradas não negativas e que somam 1. Tal vetor é uma distribuição de probabilidade sobre o conjunto V dos vértices de G. No caso em que ν é a distribuição de X_0, dada por $\nu_i = \mathbb{P}[\,\mathsf{X}_0 = i\,]$, chamamos ν de **distribuição inicial**. Então, $\nu\mathsf{P}$ na posição j é

$$\sum_{i=1}^{n} \nu_i p_{i,j} = \sum_{i=1}^{n} \mathbb{P}[\,\mathsf{X}_1 = j \mid \mathsf{X}_0 = i\,]\mathbb{P}[\,\mathsf{X}_0 = i\,] = \mathbb{P}[\,\mathsf{X}_1 = j\,]$$

de modo que $\nu\mathsf{P}$ é a distribuição de X_1 e, de modo geral, a distribuição de X_t é dada por $\nu\mathsf{P}^t$ para todo $t \geqslant 0$. A matriz $\mathsf{P}^t = (p_{i,j}^{(t)})$ é chamada de **matriz de transição em t passos** dada por

$$p_{i,j}^{(t)} := \mathbb{P}[\,\mathsf{X}_t = j \mid \mathsf{X}_0 = i\,].$$

Se $\nu^{(0)} = \nu$ e para todo $t > 0$ pomos $\nu^{(t)} = \nu^{(t-1)}\mathsf{P} = \nu\mathsf{P}^t$ então para todo $t \geqslant 0$ o vetor $\nu^{(t)}$ denota a distribuição de X_t.

No caso em que um vetor estocástico π satisfaz

$$\pi = \pi\mathsf{P}$$

dizemos que π é uma **distribuição invariante**, ou **distribuição estacionária**, do passeio aleatório, já que a lei de X_t é a mesma em todo instante t a partir dessa distribuição. Se $(\mathsf{X}_t\colon t \geqslant 0)$ é um passeio aleatório com matriz de transição P e com distribuição inicial π, então $(\mathsf{X}_{m+t}\colon t \geqslant 0)$ é um passeio aleatório com matriz de transição P e com distribuição inicial π pois $\pi\mathsf{P}^m = \pi_i$.

No caso de passeio aleatório em grafos sempre há uma distribuição invariante, que será única se o grafo for conexo. Se o grafo for desconexo há uma distribuição invariante diferente para cada componente conexa do grafo. Se $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ é um grafo, então a soma dos graus dos vértices é $2|\mathsf{E}|$ e, para qualquer vértice j. Se $\mathsf{d} = (d_i)$ é o vetor dos graus dos vértices,

$$d_j = d_{\mathsf{G}}(j) = \sum_{i\colon \{i,j\}\in \mathsf{E}} d_i \frac{1}{d_i} = \sum_{i\in \mathsf{V}} d_i p_{i,j}$$

ou seja, $\mathsf{d} = \mathsf{dP}$, normalizando o vetor por $2|\mathsf{E}|$ obtemos o vetor estocástico

$$\pi := \frac{\mathsf{d}}{2|\mathsf{E}|} = \pi\mathsf{P}, \text{ com } \pi_i = \frac{d_{\mathsf{G}}(i)}{2|\mathsf{E}|}.$$

Sob certas hipóteses, caso em que o passeio é chamado de **ergódico**, um passeio aleatório num grafo G tem uma única distribuição invariante e ela é sempre atingida no limite, quando $t \to \infty$, independentemente da distribuição inicial. Essa **convergência ao equilíbrio** é outra propriedade notável de certos processos aleatórios. Para o exemplo da figura 5.3 acima algumas potências da matriz P são

$$\mathsf{P}^2 = \begin{pmatrix} 0{,}25 & 0{,}125 & 0{,}125 & 0{,}25 & 0{,}25 & 0{,}0 \\ 0{,}0625 & 0{,}3125 & 0{,}25 & 0{,}125 & 0{,}125 & 0{,}125 \\ 0{,}0625 & 0{,}25 & 0{,}3125 & 0{,}125 & 0{,}125 & 0{,}125 \\ 0{,}125 & 0{,}125 & 0{,}125 & 0{,}3125 & 0{,}25 & 0{,}0625 \\ 0{,}125 & 0{,}125 & 0{,}125 & 0{,}25 & 0{,}3125 & 0{,}0625 \\ 0{,}0 & 0{,}25 & 0{,}25 & 0{,}125 & 0{,}125 & 0{,}25 \end{pmatrix},$$

$$\mathsf{P}^{20} = \begin{pmatrix} 0{,}100025480 & 0{,}199967369 & 0{,}199967369 & 0{,}200032630 & 0{,}200032630 & 0{,}099974519 \\ 0{,}099983684 & 0{,}200020897 & 0{,}200020897 & 0{,}199979102 & 0{,}199979102 & 0{,}100016315 \\ 0{,}099983684 & 0{,}200020897 & 0{,}200020897 & 0{,}199979102 & 0{,}199979102 & 0{,}100016315 \\ 0{,}100016315 & 0{,}199979102 & 0{,}199979102 & 0{,}200020897 & 0{,}200020897 & 0{,}099983684 \\ 0{,}100016315 & 0{,}199979102 & 0{,}199979102 & 0{,}200020897 & 0{,}200020897 & 0{,}099983684 \\ 0{,}099974519 & 0{,}200032630 & 0{,}200032630 & 0{,}199967369 & 0{,}199967369 & 0{,}100025480 \end{pmatrix},$$

$$\mathsf{P}^{80} = \begin{pmatrix} 0{,}10000001 & 0{,}2 & 0{,}2 & 0{,}20000001 & 0{,}20000001 & 0{,}099999999 \\ 0{,}1 & 0{,}20000001 & 0{,}20000001 & 0{,}2 & 0{,}2 & 0{,}1 \\ 0{,}1 & 0{,}20000001 & 0{,}20000001 & 0{,}2 & 0{,}2 & 0{,}1 \\ 0{,}1 & 0{,}19999999 & 0{,}19999999 & 0{,}2 & 0{,}2 & 0{,}09999999 \\ 0{,}1 & 0{,}19999999 & 0{,}199999999 & 0{,}2 & 0{,}2 & 0{,}09999999 \\ 0{,}09999995 & 0{,}20000001 & 0{,}20000001 & 0{,}19999999 & 0{,}19999999 & 0{,}1 \end{pmatrix},$$

de fato,

$$\mathsf{P}^t \longrightarrow \begin{pmatrix} 0{,}1 & 0{,}2 & 0{,}2 & 0{,}2 & 0{,}2 & 0{,}1 \\ 0{,}1 & 0{,}2 & 0{,}2 & 0{,}2 & 0{,}2 & 0{,}1 \\ 0{,}1 & 0{,}2 & 0{,}2 & 0{,}2 & 0{,}2 & 0{,}1 \\ 0{,}1 & 0{,}2 & 0{,}2 & 0{,}2 & 0{,}2 & 0{,}1 \end{pmatrix}.$$

É um tanto surpreendente que para passeios em grafos finitos a ergodicidade é muito fácil de descrever e depende apenas de propriedades estruturais simples do grafo: é suficiente que o grafo seja conexo e não bipartido (veja a definição de grafo bipartido no exercício 2.64, página 109).

Quando o grafo é bipartido, se tomarmos uma distribuição ν cujas entradas positivas (o suporte) estão apenas em uma das partes da bipartição, então em cada passo do passeio aleatório o suporte move-se para a outra parte e assim P^t será sempre diferente para valores de t com paridade diferente.

Não é difícil verificar que o problema acima com grafos bipartidos não está mais presente quando consideramos a variante *preguiçosa* de passeios aleatórios em grafos. Nessa modificação o passeio joga uma moeda antes de cada passo e escolhe com igual probabilidade entre ficar no vértice atual ou ir para um vizinho. A matriz de transição desse passeio é

$$\hat{P} = \frac{1}{2}(\mathrm{Id} + P)$$

em que Id é a matriz identidade de dimensão igual a de P (veja o exercício 5.58).

Proposição 5.7 *Se para algum $i \in \{1, 2, \ldots, n\}$ existem os limites*

$$\lim_{t\to\infty} p_{i,j}^{(t)} = \pi_j \in [0, 1]$$

para todo $j \in \{1, 2, \ldots, n\}$ então π é invariante.

Demonstração. O limite é uma distribuição

$$\sum_{j=1}^{n} \pi_j = \sum_{j=1}^{n} \lim_{t\to\infty} p_{i,j}^{(t)} = \lim_{t\to\infty} \sum_{j=1}^{n} p_{i,j}^{(t)} = 1$$

ainda

$$\pi_j = \lim_{t\to\infty} p_{i,j}^{(t+1)} = \lim_{t\to\infty} \sum_{k=1}^{n} p_{i,k}^{(t)} p_{k,j} = \sum_{k=1}^{n} \lim_{t\to\infty} p_{i,k}^{(t)} p_{k,j} = \sum_{k=1}^{n} \pi_k p_{k,j},$$

portanto π é invariante. □

A exigência de que o grafo seja conexo equivale a pedir que a matriz de adjacências seja r irredutível, isto é, não existe permutação que aplicada nas linhas e nas colunas resulta numa matriz triangular superior, ou seja, da forma

$$\begin{pmatrix} B & C \\ 0 & D \end{pmatrix}$$

e exigência de que o grafo seja não-bipartido equivale a não existir uma permutação que aplicada nas linhas e nas colunas resulta numa matriz da forma

$$\begin{pmatrix} \mathbf{0} & B \\ B^T & \mathbf{0} \end{pmatrix} \tag{5.8}$$

onde $\mathbf{0}$ é uma matriz quadrada.

Num contexto mais geral de processos markovianos a exigência de que o grafo seja conexo e não bipartido se traduz em, respectivamente, irredutibilidade e aperiodicidade do processo e é desse ponto de vista que discutiremos a seguir tais propriedades.

Observação 5.8 (*sobre a existência de distribuição invariante*). Há vários modos de provar a a existência de uma distribuição invariante, há argumentos probabilístico, algébrico e analítico para esse fato. Veremos mais adiante, na página 265, um argumento elementar. O algébrico é derivado do teorema de Perron–Frobenius (apêndice A.5.1) e o analítico, possivelmente a dedução mais simples, segue do Teorema do Ponto Fixo de Brower: *Para toda função contínua* $f\colon \mathsf{T} \to \mathsf{T}$, *onde* $\mathsf{T} \subset \mathbb{R}^n$ *é compacto e convexo, existe* $x \in \mathsf{T}$ *tal que* $f(x) = x$. Consideremos o conjunto (compacto e convexo)

$$\mathsf{T} := \left\{ \nu = (v_1, \ldots, v_n) \in \mathbb{R}^n \colon v_i \geqslant 0 \ (\forall\, i) \text{ e } \sum_{i=1}^{n} |v_i| = 1 \right\}$$

a função linear de T em T dada por $\mathsf{x} \mapsto \mathsf{xP}$ é contínua, portanto, pelo Teorema da Ponto Fixo de Brower deduzimos a existência de um vetor invariante. Maltese (1986) apresenta uma demostração simples desse fato envolvendo conceitos básicos de topologia. Também é conhecida uma prova combinatória que só envolve conceitos bastante elementares (Prasolov, 2006, seção 2.3). ◊

IRREDUTIBILIDADE Se $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ é um grafo conexo, então para todos $i, j \in \mathsf{V}$ distintos existe um caminho $i = v_0, v_1, \ldots, v_k = j$ ($\{v_i, v_{i+1}\} \in \mathsf{E}$ para todo $i = 0, \ldots, k-1$), portanto $p_{i,j}^{(t)} > 0$ para algum $t = t(i,j) > 0$, nesse caso o passeio é dito **irredutível**. A propriedade boa dos passeios irredutíveis é dada pelo seguinte resultado.

Lema 5.9 *Se* π *é uma distribuição invariante para um passeio aleatório irredutível, então* $\pi_i > 0$ *para todo vértice* i. *Ademais,* π *é única.*

Demonstração. Seja $(\mathsf{X}_t \colon t \geqslant 0)$ um passeio aleatório com matriz de transição P e distribuição invariante π.

Tome v um vértice com $\pi_v > 0$, que existe pois π é uma distribuição de probabilidade no conjunto (finito) dos vértices. Para cada vértice i existe um natural $t = t(i,v)$ tal que $p_{v,i}^{(t)} > 0$, então da invariância de π

$$\pi_i = \sum_{j=1}^{n} \pi_j p_{j,i}^{(t)} \geqslant \pi_v p_{v,i}^{(t)} > 0$$

para todo vértice i.

Se π e σ são distribuições invariantes e $x \in \mathbb{R}$ então para a combinação linear $(1+x)\pi + x\sigma$ vale

$$\sum_j ((1+x)\pi - x\sigma)_j p_{j,i} = \sum_j \Big((1+x)\pi_j - x\sigma_j\Big) p_{j,i} = \sum_j (1+x)\pi_j p_{j,i} - \sum_j x\sigma_j p_{j,i} = (1+x)\pi_i - x\sigma_i$$

ou seja, o vetor estocástico $(1+x)\pi - x\sigma$ é invariante para P. Assim, se $\pi \neq \sigma$ então $\pi_j \neq \sigma_j$ para algum vértice j de modo que para

$$x = \frac{\pi_j}{\sigma_j - \pi_j}$$

temos $((1+x)\pi - x\sigma)_j = 0$ contrariando o fato demonstrado acima. □

PERIODICIDADE Para cada vértice $i \in V$ olhamos o número[2]

$$\operatorname{mdc}\{t > 1 \colon p_{i,i}^{(t)} > 0\}$$

que é o período do passeio no vértice i. Se esse número é o inteiro T então $p_{i,i}^{(t)} = 0$ a menos que t seja múltiplo de T. Se $\operatorname{mdc}\{t \colon p_{i,i}^{(t)} > 0\} = 1$ então o vértice i é um estado **aperiódico** do passeio aleatório.

Se G é um grafo conexo e bipartido com bipartição $\{A, B\}$, então todos os passeios de i para j (quaisquer) têm a mesma paridade no número de arestas, que é par nos casos $i, j \in A$ ou $i, j \in B$ e ímpar caso $i \in A$ e $j \in B$ ou $i \in A$ e $j \in B$. Logo

1. se $i, j \in A$ ou $i, j \in B$ então $p_{i,j}^{(k)} > 0$ exceto quando k é ímpar,
2. se $i \in A$ e $j \in B$, ou $i \in A$ e $j \in B$, então $p_{i,j}^{(k)} > 0$ exceto quando k é par.

Portanto o passeio tem período 2.

Agora, se G não é bipartido, então contém um circuito ímpar C. Tomemos i e j dois vértices quaisquer e seja k o número de arestas no menor caminho com extremos i e j em G, ou seja, k é a *distância* entre os vértices.

Um passeio de i para j com $k + 2r$ arestas existe para todo $r \in \mathbb{N}$, basta repetir alguma aresta r vezes num passeio de comprimento k. Como G é conexo existe um passeio de i até algum vértice de C e de todo vértice de C até j, portanto podemos usar as arestas de C para obter passeios de j para i que têm a paridade oposta a de $k + 2r$. Logo, $p_{i,i}^{(t)} > 0$ para todo t suficientemente grande, ou seja, o passeio é aperiódico.

A partir dessa dedução, se G não é bipartido e é conexo, então todo vértice é aperiódico e nesse caso o chamamos o passeio aleatório $(X_t \colon t \geqslant 0)$ de **passeio aperiódico**.

A propriedade boa dos passeios aperiódicos é o seguinte.

Lema 5.10 *Seja* P *a matriz de transição de uma passeio aleatório aperiódico. Existe um natural* t_0 *tal que para todo* i, *vale* $p_{i,i}^{(t)} > 0$ *para todo* $t \geqslant t_0$.

Demonstração. Para cada vértice i definimos $A_i := \{t \in \mathbb{N} \colon p_{i,i}^{(t)} > 0\}$. Esse conjunto é fechado para a adição pois se $a, b \in A_i$ então, por definição, $p_{i,i}^{(a)} > 0$ e $p_{i,i}^{(b)} > 0$ logo

$$p_{i,i}^{(a+b)} = \sum_j p_{i,j}^{(a)} p_{j,i}^{(b)} \geqslant p_{i,i}^{(a)} p_{i,i}^{(b)} > 0,$$

portanto $a + b \in A_i$.

Sejam $m_1, m_2, \ldots, m_k \in A_i$ tais que $\operatorname{mdc}(m_1, m_2, \ldots, m_k) = 1$, os quais existem pela hipótese de aperiodicidade e definição de mdc. Pelo teorema de Bézout (apêndice A.3.1) existem inteiros $x_1, \ldots, x_k$

[2] Se $(a_n \colon n \geqslant 1)$ é uma sequência de inteiros positivos então $(d_k \colon k \geqslant 1)$ definida por $d_k := \operatorname{mdc}(a_1, \ldots, a_k)$ é monótona decrescente e limitada inferiormente por 1, portanto tem limite $d \geqslant 1$. $d := \operatorname{mdc}(a_n \colon n \geqslant 1)$.

tais que

$$1 = m_1 x_1 + m_2 x_2 + \cdots + m_k x_k$$

que reescrevemos como $1 = M - N$ onde M é a parte da soma que corresponde aos x_i positivos e N a parte que corresponde aos x_i negativos. Pelo fechamento para a soma temos $M, N \in A_i$. Fixe $m \in \mathbb{N}$ com $m \geqslant N(N-1)$; pelo teorema da divisão podemos escrever $m = qN + r$ com $r \in \{0, 1, \ldots, N-1\}$. Necessariamente $q \geqslant N-1$ pois, caso contrário, se $q \leqslant N-2$ então $m = qN + r < N(N-1)$. De $1 = M - N$ deduzimos que $m = qN + r(M-N) = (q-r)N + rM$. Porém $q - r \geqslant 0$, portanto, $m \in A_i$. Com isso concluímos que todo $m \in \mathbb{N}$ suficientemente grande pertence a A_i.

Podemos, então, garantir que para todo vértice i, existe $m(i) \in \mathbb{N}$ tal que $m \in A_i$ para todo $m \geqslant m(i)$. Para fechar a prova do lema basta tomar $t_0 = \max_i \{m(i)\}$. □

Se além de aperiódico o passeio for irredutível, a propriedade boa é o seguinte resultado.

Corolário 5.11 *Seja P a matriz de transição de uma passeio aleatório irredutível e aperiódico. Existe um natural t_0 tal que para todos i, j vale $p_{i,j}^{(t)} > 0$ para todo $t \geqslant t_0$.*

Demonstração. Do lema acima, podemos garantir que para todo vértice i, existe $m(i) \in \mathbb{N}$ tal que $m \in A_i$ para todo $m \geqslant m(i)$. Sejam $j \neq i$ um vértice e $m(i,j) \in \mathbb{N}$ tal que $p_{i,j}^{(m(i,j))} > 0$, que existe pela irredutibilidade. Tomando $M = M(i,j) := m(i) + m(i,j)$ temos para todo $t \geqslant M$

$$p_{i,j}^{(t)} = \sum_k p_{i,k}^{(t-m(i,j))} p_{k,j}^{(m(i,j))} \geqslant p_{i,i}^{(t-m(i,j))} p_{i,j}^{(m(i,j))} > 0$$

pois $t - m(i,j) \geqslant m(i)$. Finalmente, basta tomar $t_0 = \max_i \{m(i) + \max_j M(i,j)\}$. □

A partir desse resultado temos que para os passeios aleatórios em um grafo conexo e não bipartido há uma chance não nula de, a partir de um determinado momento, visitar qualquer vértice a partir de qualquer vértice inicial. Uma versão desse resultado para passeios periódicos é dada no exercício 5.56 no final do capítulo.

5.2.1 CONVERGÊNCIA PARA A DISTRIBUIÇÃO ESTACIONÁRIA

Um vetor não nulo $v \in \mathbb{R}^n$ é uma autovetor da matriz A, real e $n \times n$, se existe $\lambda \in \mathbb{R}$ tal que $Av^\intercal = \lambda v^\intercal$, em que $v^\intercal$ é o transposto do vetor v, ou seja, o vetor na forma de matriz coluna. Uma distribuição estacionária π é uma autovetor a esquerda da matriz de transições P associado ao autovalor 1. Se P for simétrica (isto é, $P = P^\intercal$), então $\pi^\intercal$ é autovetor de P associado ao autovalor 1, pois $\pi^\intercal = (\pi P)^\intercal = P^\intercal \pi^\intercal = P\pi^\intercal$.

O Teorema Espectral para matrizes reais diz que se A é uma matriz $n \times n$ simétrica então existe uma base do $\mathbb{R}^n$ formada por autovetores ortonormais[3] $v_1, v_2, \ldots, v_n$ de A com os autovalores as-

[3] $v_i v_i^\intercal = 1$ e $v_j v_i^\intercal = 0$ para $i \neq j$.

sociados $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$; como consequência podemos derivar do teorema a seguinte decomposição espectral da matriz A

$$A = \lambda_1 v_1^\intercal v_1 + \lambda_2 v_2^\intercal v_2 + \cdots + \lambda_n v_n^\intercal v_n.$$

Pelo Teorema de Perron–Frobenius, se A é simétrica, não negativa e irredutível com autovalores reais $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$ então $\lambda_1 > 0$, $\lambda_1 > \lambda_2$ e $\lambda_1 \geqslant |\lambda_i|$, para todo i. Além disso, se a matriz não é da forma (5.8) e as linhas somam 1 então

$$1 = \lambda_1 > \lambda_2 \geqslant \cdots \geqslant \lambda_n > -1. \tag{5.9}$$

De $\lambda_1 > \lambda_2$ temos que o subespaço vetorial gerado pelos autovetores associados a λ_1 acrescidos do vetor nulo tem dimensão 1, ou seja, todo autovetor associado ao λ_1 é múltiplo escalar de v_1.

Esses teoremas e suas demonstrações podem ser lidos em Horn e Johnson (1990).

Um **grafo d-regular** é uma grafo em que todos os seus vértices têm grau d, para algum inteiro $d \geqslant 0$. Nesse caso a matriz de transição de um passeio em G é $P = (1/d)A$, *simétrica* e **duplamente estocástica** (as linhas e colunas somam 1). Um passeio aleatório irredutível e aperiódico num grafo conexo regular converge para a distribuição uniforme.

Seja G um grafo d-regular, conexo e não bipartido com $n \geqslant 2$ vértices. Seja P a matriz das transições nesse grafo com autovalores reais como em (5.9) acima e de modo que pela decomposição espectral temos $P = \lambda_1 v_1^\intercal v_1 + \lambda_2 v_2^\intercal v_2 + \cdots + \lambda_n v_n^\intercal v_n$ para uma base $v_1, \ldots, v_n$ do $\mathbb{R}^n$ de autovetores ortonormais de P.

Da ortonormalidade segue que $(\lambda_i v_i^\intercal v_i)(\lambda_j v_j^\intercal v_j) = \lambda_i \lambda_j v_i^\intercal v_i v_j^\intercal v_j$ e essa expressão vale $\lambda_i^2 v_i^\intercal v_i$ caso $i = j$ e vale 0 caso $i \neq j$, portanto, $P^2 = \sum_{i=1}^n \lambda_i^2 v_i^\intercal v_i$. Por indução,

$$P^t = \lambda_1^t v_1^\intercal v_1 + \lambda_i^t v_2^\intercal v_2 + \cdots + \lambda_i^t v_n^\intercal v_n,$$

logo, também da ortonormalidade $v_i P^t = \lambda_1^t v_i$.

Sejam

$$\lambda := \max\{|\lambda_i| \colon i \in \{2, 3, \ldots, n\}\} = \max\{|\lambda_2|, |\lambda_n|\}$$

e $\nu^{(t)}$ a distribuição de X_t para todo $t \geqslant 0$. Vamos mostrar que $\nu^{(t)}$ converge para a distribuição uniforme $\pi = \frac{1}{n}(1, \ldots, 1)$ com velocidade controlada por λ.

Como 1 é autovalor de P (associado à distribuição uniforme) temos $\lambda_1 \geqslant 1$ por (5.9). Por outro lado, se k é a coordenada de maior valor em $v_1 > 0$, isto é, $v_{1k} = \max_j v_{1j}$ então

$$\lambda_1 v_{1k} = \sum_{i=1}^n v_{1j} p_{j,k} \leqslant v_{1k} \sum_{j=1}^n p_{j,k} = v_{1k}$$

ou seja, $\lambda_1 \leqslant 1$, portanto $\lambda_1 = 1$ é o maior autovalor e o autovetor unitário associado é

$$v_1 = \frac{1}{\sqrt{n}}(1, \ldots, 1).$$

Se o vetor estocástico que define a distribuição inicial escrito como combinação linear dos vetores da base ortonormal de autovetores da matriz P é $\nu^{(0)} = \alpha_1 \mathsf{v}_1 + \cdots + \alpha_n \mathsf{v}_n$ então para qualquer $t > 0$

$$\nu^{(t)} = \nu^{(0)}\mathsf{P}^t = \alpha_1 \mathsf{v}_1 \mathsf{P}^t + \cdots + \alpha_n \mathsf{v}_n \mathsf{P}^t = \alpha_1 \lambda_1^t \mathsf{v}_1 + \cdots + \alpha_n \lambda_n^t \mathsf{v}_n$$

e, portanto,

$$\|\nu^{(t)} - \alpha_1 \mathsf{v}_1\|_2^2 = \left(\alpha_1 \lambda_1^t - \alpha_1\right)^2 + \sum_{i=2}^{n} (\alpha_i \lambda_i^t)^2 = \sum_{i=2}^{n} (\alpha_i \lambda_i^t)^2 \leqslant \lambda^{2t} \|\nu^{(0)}\|_2^2 = \lambda^{2t}$$

donde segue que $\|\nu^{(t)} - \alpha_1 \mathsf{v}_1\|_2 \leqslant \lambda^t$ e como $\lambda < 1$ temos $\nu^{(t)}$ converge exponencialmente rápido para $\alpha_1 \mathsf{v}_1$ quando $t \to \infty$. Finalmente, notemos que $\alpha_1 \mathsf{v}_1 = \pi$ pois

$$\alpha_1 = \nu^{(0)} \mathsf{v}_1^{\mathsf{T}} = \frac{1}{\sqrt{n}} \sum_i \nu^{(0)}{}_i = \frac{1}{\sqrt{n}}.$$

Lema 5.12 *Seja P a matriz de transição de um passeio aleatório em um grafo G com n vértices, conexo, d-regular e não bipartido, e seja $\lambda = \max\{|\alpha| : \alpha \text{ é autovalor de } \mathsf{P} \text{ e } \alpha < 1\}$. Então para todo vetor de probabilidades ν e todo $t \in \mathbb{N}$*

$$\|\nu \mathsf{P}^t - \pi\|_2 \leqslant \lambda^t$$

onde $\pi = (1/n, \ldots, 1/n)$ é a distribuição estacionária do passeio aleatório. □

Quanto menor o λ mais rápido é a convergência, no entanto

$$\lambda \geqslant \sqrt{\frac{n-d}{d(n-1)}}$$

como provaremos agora. O traço de uma matriz é a soma das suas entradas na diagonal, tem a propriedade de que traço(AB) = traço(BA) donde concluímos que matrizes semelhantes têm o mesmo traço, logo se for diagonalizável, então o traço e a soma dos autovalores. Mais geral é o fato de que o traço de P^t é a soma das t-ésimas potências dos autovalores de P. Dessa propriedade e da definição de traço tiramos

$$\text{traço}(\mathsf{P}^2) = \frac{n}{d} = \sum_{i=1}^{n} \lambda_i^2 \leqslant 1 + (n-1)\lambda^2$$

de onde segue o limitante inferior para λ.

Observação 5.13 (*expansão espectral × expansão combinatória*). No caso de grafos, o segundo maior autovalor λ da matriz de adjacências ser pequeno significa que o grafo é um bom "expansor", ou seja, todos os subconjuntos de vértices S têm pelo menos uma fração $\alpha = \alpha(\lambda)$ fixa de $|\mathsf{S}|$ vértices na vizinhança. E vale a recíproca, bons expansores têm λ pequeno. ◊

Usando que $\log(1+x) \approx x$ temos que $\log\lambda \approx \lambda - 1$, portanto tomando $t = \log_\lambda(1/n) = O\left(\frac{1}{1-\lambda}\log n\right)$ temos

$$\|\nu P^t - \pi\|_2 < \frac{1}{n}.$$

O fator $1-\lambda$ é usualmente referido como **distância espectral**[4] de P.

As vezes é conveniente expressar a velocidade de convergência em outras métricas. As mais usuais são as dadas pelas normas

$$\|\nu\|_1 = \sum_{i=1}^{n} |\nu_i|$$

$$\|\nu\|_\infty = \max_{i=1,\dots,n} |\nu_i|$$

para as quais valem $\|\nu\|_\infty \leqslant \|\nu\|_2 \leqslant \sqrt{n}\|\nu\|_\infty$ e $\|\nu\|_\infty \leqslant \|\nu\|_1 \leqslant n\|\nu\|_\infty$ e ainda usamos a **distância de variação total** entre duas distribuições

$$\|\pi - \nu\|_{VT} = \frac{1}{2}\|\pi - \nu\|_1.$$

Teorema 5.14 *Seja (X_t) um passeio aleatório num grafo G com n vértices, conexo, d-regular, não bipartido e com segundo maior autovalor λ. Então, para todo $\delta > 0$ existe $t_0 = t_0(\lambda, \delta)$ tal que para todo vértice v e todo $t > t_0$*

$$\left| \mathbb{P}[X_t = v] - \frac{1}{n} \right| < \delta.$$

Demonstração. Sejam G como no enunciado e $\nu = \nu^{(0)}$ uma distribuição inicial qualquer, então a distribuição de X_t é dada pelo vetor $\nu^{(t)} = \nu P^t$. Pelo lema 5.12 e a relação entre as normas dadas acima

$$\max_v \left| \mathbb{P}(X_t = v) - \frac{1}{n} \right| = \|\nu^{(t)} - \pi\|_\infty \leqslant \|\nu^{(t)} - \pi\|_2 \leqslant \lambda^t$$

portanto, se $t > \log_\lambda(\delta/\lambda)$ então $\|\nu^{(t)} - \pi\|_\infty < \delta$. □

Exemplo 5.15 (*passeio aleatório num circuito ímpar*). Por exemplo, tomemos $n \geqslant 3$ ímpar e o circuito de n vértices definido por $V = \{0, 1, \dots, n-1\}$ e $\{i, j\} \in E$ se, e somente se, $i \equiv j \pm 1 \pmod n$ (figura 5.4). Esse é um grafo 2-regular, conexo e não bipartido. Os autovetores da matriz P são, para todo $a \in V$, dados por

$$v_a := \left(e^{\frac{2\pi i}{n}0},\ e^{\frac{2\pi i}{n}a},\ e^{\frac{2\pi i}{n}2a},\ \cdots,\ e^{\frac{2\pi i}{n}(n-1)a}\right)$$

com os respectivos autovalores

$$\lambda_a = \cos\left(\frac{2\pi a}{n}\right)$$

[4] *spectral gap* em inglês

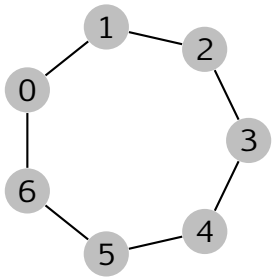


Figura 5.4: um circuito de comprimento 7.

logo $\lambda_0 = 1$ é o maior autovalor e

$$\lambda = \left|\cos\frac{2\pi(n+1)/2}{n}\right| = \left|\cos(\pi + \frac{\pi}{n})\right| = \left|\cos(\frac{\pi}{n})\right| \leqslant \exp\left(\frac{-\pi^2}{2n^2}\right)$$

pois $\cos(x) \leqslant \exp(x^2/2)$, portanto

$$\left\|\nu^{(t)} - \pi\right\|_2 \leqslant \exp\left(\frac{-\pi^2 t}{2n^2}\right).$$

Para t suficientemente grande podemos dizer que o estado no tempo t é, na prática, qualquer um dos n vértices com probabilidade uniforme. Para $t = 2n^2$ temos $\|\nu^{(t)} - \pi\|_2 \leqslant 0{,}0000518$. Para $t = 2n^2 \log n$ temos $\|\nu^{(t)} - \pi\|_2 \leqslant n^{-\pi^2}$. Nesse caso temos um alternativa para gerar números aleatórios (seção 1.4.2) a partir de bits aleatórios, $b \in_R \{0,1\}$; usamos o passeio no circuito fazendo $X_{t+1} = X_t + (2b-1)$. Para t suficientemente grande X_t é qualquer número em $\{0,1,\ldots,n-1\}$ com probabilidade uniforme, praticamente. Um problema interessante para investigação é determinar grafos regulares de grau constante com λ pequeno para que a convergência se dê em tempo muito menor que $O(\log n)$, isso implicaria em usar muito menos que os $\Theta(\log n)$ bits aleatórios usado para gerar um número sorteando todos os bits sua representação binária. ◊

Exercício 5.16 (autovalores do passeio preguiçoso). Sejam G um grafo com n vértices, conexo, d-regular e $\mu_1 \geqslant \mu_2 \geqslant \cdots \geqslant \mu_n$ os autovalores de A, a matriz de adjacências de G. Considere as matrizes estocásticas $P = (1/d)A$ e $\hat{P} = (P + Id)/2$ cujos autovalores são, respectivamente, $\lambda_1 \geqslant \lambda_2 \geqslant \cdots, \lambda_n$ e $\kappa_1 \geqslant \kappa_2 \geqslant \cdots \geqslant \kappa_n$. Prove que para todo inteiro i

$$\lambda_i = \frac{\mu_i}{d} = 2\kappa_i - 1.$$

A hipótese que pede que o grafo seja não bipartido nos resultados acima é necessária porque, de outro modo, $\lambda = 1$ e não há convergência. Determine se valem os resultados desta seção para G bipartido, conexo e d-regular com o passeio aleatório preguiçoso dado pela matriz de transição $\hat{P}$.

Exemplo 5.17 (*o modelo de Ehrenfest e o cubo discreto*). A evolução no modelo de Ehrenfest pode ser modelada em um grafo. Fixamos uma enumeração $1, 2, \ldots, N$ das moléculas e cada vértice v representa a função indicadora da molécula estar no compartimento B, isto é, $v_i = 1$ se, e só se, a molécula i

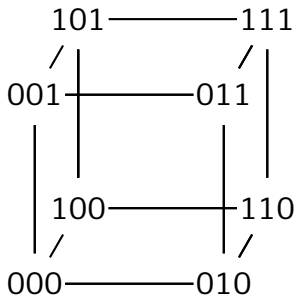


Figura 5.5: um diagrama do 3-cubo.

está em B. Esse grafo é o N-**cubo discreto**, denotado por Q_N e dado pelos vértices $(v_1,\dots,v_N)\in\{0,1\}^N$ com dois deles distintos, digamos $(v_1,\dots,v_N)$ e $(u_1,\dots,u_N)$, adjacentes se, e só se, $v_i = u_i$ para todo i exceto para um único i (figura 5.5). O N-cubo é um grafo N-regular, conexo e bipartido, portanto, o passeio aleatório do modelo de Ehrenfest é irredutível, porém, é de período 2

$$p_{i,j}^{(2t+1)} = 0 \text{ e } p_{i,j}^{(2t)} \to \binom{N}{j}\left(\frac{1}{2}\right)^{N-1} \text{ se } i \text{ e } j \text{ têm a mesma paridade}$$

$$p_{i,j}^{(2t)} = 0 \text{ e } p_{i,j}^{(2t+1)} \to \binom{N}{j}\left(\frac{1}{2}\right)^{N-1} \text{ se } i \text{ e } j \text{ não têm a mesma paridade.}$$

Os autovalores da matriz de adjacências são os inteiros $N - 2\sum_i u_i$ para cada $u = (u_i) \in V$. Nesse modelo não temos convergência quando $t \to \infty$.

Se consideramos o passeio aleatório preguiçoso nesse grafo, os autovalores da matriz de transição $\hat{P}$ são, usando o exercício 5.16, os inteiros $1 - \frac{\sum_i u_i}{N}$. Uma interpretação dessa modificação do experimento é: em cada instante, independentemente do passado, um compartimento é escolhido aleatoriamente, em seguida uma molécula é escolhida aleatoriamente; a molécula escolhida vai para o compartimento escolhido.

No passeio aleatório preguiçoso correspondente ao modelo modificado do processo de Eherenfest a distribuição $\nu^{(t)}$ converge para a distribuição uniforme, $1/2^N$ (verifique). Assim a probabilidade do evento "j moléculas em B" ocorre com probabilidade $\binom{N}{j}1/2^N$. Se $\hat{P} = (q_{i,j})$ é a matriz das transições no processo modificado dado acima, então

$$q_{i,j}^{(t)} \to \binom{N}{j}\left(\frac{1}{2}\right)^N$$

quando $t \to \infty$. A distribuição invariante é, para todo j,

$$\pi_j := \frac{1}{2^N}\binom{N}{j}$$

e pelo lema acima

$$\left\|\nu^{(t)} - \pi\right\|_2 \leqslant \left(1 - \frac{1}{N}\right)^t$$

para qualquer distribuição inicial $\nu^{(0)}$. $\Diamond$

GRAFOS NÃO REGULARES Em geral a matriz de transições $\mathsf{P} = \mathsf{D}^{-1} \cdot \mathsf{A}$ de um passeio aleatório em um grafo $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ (equação (5.7)) não é simétrica, porém

$$\mathsf{P} = \mathsf{D}^{-1/2}\mathsf{D}^{-1/2}\mathsf{A}\mathsf{D}^{-1/2}\mathsf{D}^{1/2} = \mathsf{D}^{-1/2}\mathsf{Q}\mathsf{D}^{1/2}$$

e $\mathsf{Q} := \mathsf{D}^{-1/2}\mathsf{A}\mathsf{D}^{-1/2}$ é simétrica, se $\mathsf{Q} = (q_{i,j})$ então

$$q_{i,j} = \frac{a_{i,j}}{\sqrt{d_{\mathsf{G}}(i)d_{\mathsf{G}}(j)}} = q_{j,i}.$$

portanto, P é semelhante a Q, logo tem os mesmos autovalores reais $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$.

Pela decomposição espectral $\mathsf{Q} = \lambda_1 \mathsf{v}_1^\intercal \mathsf{v}_1 + \lambda_2 \mathsf{v}_2^\intercal \mathsf{v}_2 + \cdots + \lambda_n \mathsf{v}_n^\intercal \mathsf{v}_n$ para uma base $\mathsf{v}_1, \ldots, \mathsf{v}_n$ do $\mathbb{R}^n$ de autovetores ortonormais de Q. Da ortonormalidade, como fizemos acima no caso regular, segue que $\mathsf{Q}^t = \lambda_1^t \mathsf{v}_1^\intercal \mathsf{v}_1 + \lambda_i^t \mathsf{v}_2^\intercal \mathsf{v}_2 + \cdots + \lambda_i^t \mathsf{v}_n^\intercal \mathsf{v}_n$, para todo inteiro $t \geqslant 1$, portanto,

$$\mathsf{P}^t = \lambda_1^t \mathsf{D}^{-1/2}\mathsf{v}_1^\intercal \mathsf{v}_1 \mathsf{D}^{1/2} + \lambda_2^t \mathsf{D}^{-1/2}\mathsf{v}_2^\intercal \mathsf{v}_2 \mathsf{D}^{1/2} + \cdots + \lambda_n^t \mathsf{D}^{-1/2}\mathsf{v}_n^\intercal \mathsf{v}_n \mathsf{D}^{1/2} \tag{5.10}$$

e é fácil verificar, direto da definição, que o vetor $\left(\sqrt{d_{\mathsf{G}}(1)}, \ldots, \sqrt{d_{\mathsf{G}}(n)}\right)$ é autovetor de Q associado ao autovalor $\lambda_1 = 1$, logo

$$\mathsf{v}_1 = \frac{1}{\sqrt{2|\mathsf{E}|}}\left(\sqrt{d_{\mathsf{G}}(1)}, \ldots, \sqrt{d_{\mathsf{G}}(n)}\right)$$

e o primeiro termo de (5.10) é a matriz Π cujas linhas são dadas pelo vetor estacionário

$$\lambda_1^t \mathsf{D}^{-1/2}\mathsf{v}_1^\intercal \mathsf{v}_1 \mathsf{D}^{1/2} = \begin{pmatrix} \frac{d_{\mathsf{G}}(1)}{2|\mathsf{E}|} & \frac{d_{\mathsf{G}}(2)}{2|\mathsf{E}|} & \cdots & \frac{d_{\mathsf{G}}(n)}{2|\mathsf{E}|} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{d_{\mathsf{G}}(1)}{2|\mathsf{E}|} & \frac{d_{\mathsf{G}}(2)}{2|\mathsf{E}|} & \cdots & \frac{d_{\mathsf{G}}(n)}{2|\mathsf{E}|}. \end{pmatrix}$$

logo $\mathsf{P}^t = \Pi + \sum_{\ell=2}^n \lambda_\ell^t \mathsf{D}^{1/2}\mathsf{v}_\ell \mathsf{v}_\ell^\intercal \mathsf{D}^{-1/2}$ e como $-1 < \lambda_\ell < 1$ para todo $2 \leqslant \ell \leqslant n$, temos $\lambda_\ell \to 0$ quando $t \to \infty$

$$\lim_{t\to\infty} \mathsf{P}^t = \Pi = \begin{pmatrix} \pi_1 & \pi_2 & \cdots & \pi_n \\ \pi_1 & \pi_2 & \cdots & \pi_n \\ \vdots & \ddots & \vdots & \\ \pi_1 & \pi_2 & \cdots & \pi_n \end{pmatrix} = \begin{pmatrix} \frac{d_{\mathsf{G}}(1)}{2|\mathsf{E}|} & \cdots & \frac{d_{\mathsf{G}}(n)}{2|\mathsf{E}|} \\ \frac{d_{\mathsf{G}}(1)}{2|\mathsf{E}|} & \cdots & \frac{d_{\mathsf{G}}(n)}{2|\mathsf{E}|} \\ \vdots & \ddots & \vdots \\ \frac{d_{\mathsf{G}}(1)}{2|\mathsf{E}|} & \cdots & \frac{d_{\mathsf{G}}(n)}{2|\mathsf{E}|} \end{pmatrix}.$$

como queríamos demonstrar.

TAXA DE CONVERGÊNCIA Da equação (5.10) podemos escrever os elementos $p_{i,j}^{(t)}$ de P^t como

$$p_{i,j}^{(t)} = \pi_j + \sum_{\ell=2}^n \lambda_\ell^t \frac{1}{\sqrt{d_{\mathsf{G}}(i)}} (\mathsf{v}_\ell^\intercal \mathsf{v}_\ell)_{i,j} \sqrt{d_{\mathsf{G}}(j)} \leqslant \sum_{\ell=2}^n \lambda_\ell^t \frac{\sqrt{d_{\mathsf{G}}(j)}}{\sqrt{d_{\mathsf{G}}(i)}}.$$

Fazendo $\lambda := \max\{|\lambda_2|, |\lambda_n|\}$

$$\left| p_{i,j}^{(t)} - \pi_j \right| \leqslant \lambda^t \sqrt{\frac{\pi_j}{\pi_i}}$$

e, claramente, $\lim_{t\to\infty} p_{i,j}^{(t)} = \pi_j$ para todo i, isto é $\lim_{t\to\infty} \mathbb{P}[\,X_t = j\,] = \pi_j$.

UMA DEMONSTRAÇÃO ELEMENTAR DA CONVERGÊNCIA AO EQUILÍBRIO Vamos provar de modo elementar que para P matriz de transição de uma passeio aleatório irredutível e aperiódico

$$\lim_{k\to\infty} \mathsf{P}^k = \mathsf{Q}$$

com Q uma matriz estocástica com as colunas constantes e que uma linha qualquer de Q define π invariante. Por causa do corolário 5.11 podemos considerar o caso em que P é uma matriz $n \times n$, estocástica e positiva. Na demonstração usaremos o seguinte fato.

Exercício 5.18. Sejam $\mathsf{P} = (p_{i,j})$ uma matriz estocástica com todas as entradas positivas, $\varepsilon = \min_{i,j}\{p_{i,j}\}$ e $\mathsf{c} = (c_i)$ um vetor *coluna* qualquer e $m_0 = \min \mathsf{c}$ e $\mathsf{M}_0 = \max \mathsf{c}$. Prove que se $m_1 = \min \mathsf{Pc}$ e $\mathsf{M}_1 = \max \mathsf{Pc}$, então $m_1 \geqslant m_0$, $\mathsf{M}_1 \leqslant \mathsf{M}_0$ e $\mathsf{M}_1 - m_1 \leqslant (1 - 2\varepsilon)(\mathsf{M}_0 - m_0)$ (dica: tome c' o vetor obtido a partir de c trocando todas as coordenadas por m_0 menos a coordenada de M_0; tome c'' o vetor obtido a partir de c trocando todas as coordenadas por M_0 menos a coordenada de m_0. Verifique e use que $\mathsf{Pc}' \leqslant \mathsf{Pc} \leqslant \mathsf{Pc}''$.)

Denotemos por e_i o vetor *coluna* $(\delta_{i,j})_j$ com todas as entradas nulas a menos da posição i que é $\delta_{i,i} = 1$.

Seja $\mathsf{P} = (p_{i,j})$ uma matriz estocástica $n \times n$ com todas entradas positivas. Definimos

$$\varepsilon := \min \mathsf{P} = \min_{i,j}\{p_{i,j}\} > 0,$$

fixamos uma coluna $j \in \{1, 2, \ldots, n\}$ de P e definimos

$$a_k := \min \mathsf{P}^k \mathsf{e}_j$$
$$b_k := \max \mathsf{P}^k \mathsf{e}_j$$

o mínimo e o máximo, respectivamente, da j-ésima coluna de P^k, para todo $k \geqslant 1$ e tomamos as condições iniciais $a_0 := 0$ e $b_0 := 1$.

Da definição e do exercício 5.18 temos para todo inteiro $k \geqslant 0$ que

$$a_{k+1} = \min \mathsf{P}(\mathsf{P}^k \mathsf{e}_j) \geqslant a_k = \min \mathsf{P}^k \mathsf{e}_j$$
$$b_{k+1} = \max \mathsf{P}(\mathsf{P}^k \mathsf{e}_j) \leqslant b_k = \max \mathsf{P}^k \mathsf{e}_j.$$

As sequências de máximos e mínimos da coluna j de P^k são monótonas e limitadas

$$1 = b_0 > b_1 \geqslant \cdots \geqslant b_k \geqslant \cdots \geqslant a_k \geqslant a_{k-1} \geqslant \cdots \geqslant a_1 > a_0 = 0$$

logo têm limite. Mais que isso, têm o mesmo limite pois

$$b_k - a_k \leqslant (1-2\varepsilon)(b_{k-1} - a_{k-1}) \leqslant (1-2\varepsilon)^k$$

logo $\lim_{k\to\infty} b_k - a_k = 0$. Definimos para todo j

$$\pi_j := \lim_{k\to\infty} b_k = \lim_{k\to\infty} a_k \in (0,1).$$

Com isso provamos que todas as entradas da j-ésima coluna de P^k convergem para um valor constante π_j, isto é,

$$\lim_{k\to\infty} p_{i,j}^{(k)} = \pi_j \quad \text{para todo } i.$$

Definimos o vetor $\pi := (\pi_i)$ e se $\mathbf{1} = (1_i)$ é o vetor constante 1, então $\mathsf{Q} = \mathbf{1}^{\mathsf{T}}\pi$ é uma matriz $n \times n$ com cada linha igual a π, logo $\lim_{k\to\infty} \mathsf{P}^k = \mathsf{Q}$. Notemos que $\sum_j \pi_j = 1$.

O vetor π é invariante como já demonstramos na proposição 5.7.

Para finalizar essa seção, provamos que se o estado inicial é algum elemento de $\{1,2,\ldots,n\}$ com probabilidade dada pela distribuição $\nu = \nu^{(0)}$ então $\nu\mathsf{P}^k \to \pi$ quando $k \to \infty$. De fato, com a notação

$$\nu^{(t)} = \begin{pmatrix} q_1^{(t)} & \cdots & q_n^{(t)} \end{pmatrix}$$

para todo t, na posição j de $\nu\mathsf{P}^k = \nu^{(k)}\mathsf{P}$ temos

$$\lim_{k\to\infty} \left(\nu\mathsf{P}^k\right)_j = \lim_{k\to\infty} \sum_i q_i^{(0)} p_{i,j}^{(k)} = \sum_i q_i^{(0)} \lim_{k\to\infty} p_{i,j}^{(k)} = \sum_i q_i^{(0)} \pi_j = \pi_j$$

pois ν é um vetor estocástico logo a soma das coordenadas é 1, portanto, $\nu\mathsf{Q} = \pi$ como queríamos provar.

5.2.2 RECORRÊNCIA

Quanto tempo, em média, demora para um passeio aleatório num grafo conexo (re)visitar um vértice? Por exemplo, no caso em que G é o grafo completo sobre o conjunto de vértice $\{1,2,\ldots,n\}$ e consideramos um passeio aleatório simples em G que parte do vértice u, a probabilidade de atingir o vértice $v \neq u$ em exatamente 1 passo é $1/(n-1)$, a probabilidade de atingir v em 2 passos é $(n-2)/(n-1)^2$, em 3 passos $(n-2)^2/(n-1)^3$, e assim por diante. O número esperado de passos para sair de u e chegar em v pela primeira vez é

$$\sum_{k\geqslant 1} k \frac{1}{n-1}\left(\frac{n-2}{n-1}\right)^{k-1} = \frac{1}{n-1}\frac{1}{\left(1-\frac{n-2}{n-1}\right)^2} = n-1.$$

O número esperado de passos para visitar todos os vértices do grafo pode ser estimado da seguinte forma. Seja t_i o instante em que pela primeira vez temos exatamente i vértices visitados, portanto,

$t_{i+1} - t_i$ é uma variável aleatória geométrica que conta o número de passos enquanto espera-se para conhecer um novo vértice, evento que ocorre com probabilidade $(n-i)/(n-1)$, logo $\mathbb{E}[\, t_{i+1} - t_i \,] = (n-1)/(n-i)$ e t_n é o número de passos até visitar todos os vértices

$$\mathbb{E}\, t_n = \sum_{i=1}^{n-1} \mathbb{E}[\, t_{i+1} - t_i \,] = \sum_{i=1}^{n-1} \frac{n-1}{n-i} = (n-1) \sum_{i=1}^{n-1} \frac{1}{i} = (n-1)\mathsf{H}_{n-1}$$

em que H_n denota o n-ésimo número harmônico, $\mathsf{H}_n = \Theta(\log n)$ (veja equação (2.16), página 100).

Vamos definir as seguintes variáveis aleatórias dos tempos de passagem do passeio aleatório pelos vértices de um grafo G

$$\mathsf{T}_{i,j} := \min\{\, t \geqslant 0 \colon \mathsf{X}_t = j \text{ dado que } \mathsf{X}_0 = i \,\}$$
$$\mathsf{T}_i := \min\{\, t \geqslant 1 \colon \mathsf{X}_t = i \text{ dado que } \mathsf{X}_0 = i \,\}$$

em particular $\mathsf{T}_{i,i} = 0$ e convencionamos que $\min \varnothing = \infty$. Condicionando ao estado inicial temos, pelo teorema da esperança total (página 154) os respectivos valores esperados para essas variáveis definidas acima

$$\mu_{i,j} = 1 + \sum_{k=1}^{n} p_{i,k}\mu_{k,j}$$
$$\mu_i = 1 + \sum_{k=1}^{n} p_{i,k}\mu_{k,i}$$

os quais podem ser escritos numa única equação como

$$\mu_{i,j} + \delta_i(j)\mu_j = 1 + \sum_k p_{i,k}\mu_{k,i} \tag{5.11}$$

com $\delta_i(j) = 1$ se $i = j$ e $\delta_i(j) = 0$ nos outros casos.

Dada a distribuição invariante $\pi = (\pi_i)$ de um passeio ergódico, multiplicamos a equação acima por π_i e somamos para todo i; obtemos

$$\sum_{i=1}^{n} \pi_i\mu_{i,j} + \sum_{i=1}^{n} \pi_i\delta_i(j)\mu_j = \sum_{i=1}^{n} \pi_i 1 + \sum_{i=1}^{n} \pi_i \sum_{k=1}^{n} p_{i,k}\mu_{k,i}$$

usando a definição de $\delta_i(j)$ deduzimos

$$\sum_{i=1}^{n} \pi_i\mu_{i,j} + \pi_j\mu_j = 1 + \sum_{k=1}^{n} \pi_k\mu_{k,i}$$

portanto $\pi_j\mu_j = 1$ e como G é conexo a irredutibilidade implica que $\pi_j > 0$ para todo j, logo o tempo médio de retorno ao vértice j é

$$\mu_j = \frac{1}{\pi_j}. \tag{5.12}$$

Lema 5.19 *Para passeios irredutíveis e aperiódicos em um grafo finito valem* $\mathbb{P}[\,\mathsf{T}_{i,j} < \infty\,] = 1$ *e* $\mu_{i,j} < \infty$.

Demonstração. Sejam P a matriz de um passeio irredutível e aperiódico em um grafo finito e t_0 um inteiro positivo tal que $p_{i,j}^{(t)} > 0$ para todos i, j e $t \geqslant t_0$ cuja existência é garantida pelo corolário 5.11.

Dados os vértices i e j, se $\varepsilon := 1 - \min\{p_{i,j}^{(t_0)}\}$, então

$$\mathbb{P}[\,\mathsf{T}_{i,j} > t_0\,] \leqslant \mathbb{P}\Big[\mathsf{X}_{t_0} \neq j\Big] \leqslant \varepsilon.$$

Ainda,

$$\mathbb{P}[\,\mathsf{T}_{i,j} > 2t_0\,] = \mathbb{P}[\,\mathsf{T}_{i,j} > 2t_0 \mid \mathsf{T}_{i,j} > t_0\,] \cdot \mathbb{P}[\,\mathsf{T}_{i,j} > t_0\,] \leqslant \mathbb{P}[\,\mathsf{X}_{2t_0} \neq j \mid \mathsf{T}_{i,j} > t_0\,] \cdot \mathbb{P}[\,\mathsf{T}_{i,j} > t_0\,] \leqslant \varepsilon^2$$

e, por indução, prova-se que $\mathbb{P}[\,\mathsf{T}_{i,j} > \ell t_0\,] \leqslant \varepsilon^\ell$. Como $\varepsilon < 1$, $\mathbb{P}[\,\mathsf{T}_{i,j} > k t_0\,] \to 0$ quando $k \to \infty$ de modo que $\mathbb{P}[\,\mathsf{T}_{i,j} = \infty\,] = 0$.

Usando a proposição 3.22, página 135

$$\begin{aligned}\mu_{i,j} = \mathbb{E}\,\mathsf{T}_{i,j} &= \sum_{k\geqslant 1} \mathbb{P}[\,\mathsf{T}_{i,j} \geqslant k\,] = \sum_{k\geqslant 0} \mathbb{P}[\,\mathsf{T}_{i,j} > k\,] \\ &= \sum_{\ell\geqslant 0} \sum_{k=\ell t_0}^{(\ell+1)t_0-1} \mathbb{P}[\,\mathsf{T}_{i,j} > k\,] \leqslant \sum_{\ell\geqslant 0} \sum_{k=\ell t_0}^{(\ell+1)t_0-1} \mathbb{P}[\,\mathsf{T}_{i,j} > \ell t_0\,] \leqslant t_0 \sum_{\ell\geqslant 0} \varepsilon^\ell = \frac{t_0}{1-\varepsilon}\end{aligned}$$

portanto $\mu_{i,j}$ é finito. □

Exercício 5.20. Seja $(\mathsf{X}_t\colon t \geqslant 0)$ um passeio aleatório irredutível e aperiódico num grafo finito, com $\mathsf{X}_0 = i$. Defina para cada vértice $j \in \{1, 2, \ldots, n\}$

$$\rho_j := \sum_{k\geqslant 0} \mathbb{P}[\,\mathsf{X}_k = j, \mathsf{T}_i > k\,]$$

que é número esperado de visitas ao vértice j no intervalo de tempo $[0, 1, \ldots, \mathsf{T}_i - 1]$. Verifique que $\rho_j < \mu_j$. Prove que o vetor

$$\pi := \Big(\tfrac{\rho_1}{\mu_1},\ \tfrac{\rho_2}{\mu_2},\ \cdots,\ \tfrac{\rho_n}{\mu_n},\ \Big)$$

define a distribuição estacionária do passeio aleatório.

Exemplo 5.21 (*o modelo de Ehrenfest*). O propósito do modelo de difusão de Ehrenfest é entender o fenômeno da irreversibilidade termodinâmica, segundo a qual a entropia de um sistema fechado só pode aumentar. Entretanto, o famoso teorema de recorrência de Poincaré implica que, na situação em que no tempo 0 todas as moléculas estão em A, haverá um tempo subsequente t, com $t > n$ para qualquer n dado, no qual todas as moléculas irão novamente estar em A. Esse fenômeno previsto pelo teorema nunca é observado na vida cotidiana. No modelo de Ehrenfest o papel do teorema da recorrência de Poincaré é desempenhado pela recorrência do passeio aleatório que garante que o

passeio visita qualquer estado fixo infinitamente. A tendência do sistema ao equilíbrio, com aproximadamente metade das moléculas em cada compartimento, é caracterizado pela convergência da distribuição do passeio para a distribuição invariante

$$\pi_j = \frac{1}{2^N}\binom{N}{j}$$

e

$$\left\|\nu^{(t)} - \pi\right\|_2 \leqslant \left(1 - \frac{1}{N}\right)^t$$

para qualquer distribuição inicial $\nu^{(0)}$. Se $t = cN\log N$ então $\left\|\nu^{(t)} - \pi\right\|_2 \leqslant N^{-c}$.

Não é difícil verificar que essa distribuição é simétrica em torno de $j = N/2$, logo o sistema tem maior probabilidade de estar num estado com aproximadamente metade das moléculas em cada compartimento. No longo prazo, a fração do tempo que o processo passa no estado $N/2$ é

$$\pi_{N/2} = \frac{1}{2^N}\binom{N}{N/2} \approx \frac{1}{2^N}\frac{2^N}{\sqrt{\pi N/2}} = \frac{1}{\sqrt{\pi N/2}}$$

onde usamos no coeficiente binomial uma aproximação via fórmula de Stirling. Já a fração do tempo que o processo passa no estado 0 é $\pi_0 = 2^{-N}$, assim o passeio está no entorno do estado de equilíbrio na maior parte do tempo.

Ademais, o tempo médio de recorrência do estado j é $\mu_j = 1/\pi_j = 2^N/\binom{N}{j}$, em particular o retorno à origem, com toda as moléculas no lado B, se dá a cada 2^N passos em média. Assim, se $N = 1.000$ então $\mu_0 = 2^{1.000}$ e se medimos o tempo em segundos então $2^{1.000}$ segundos $\approx 10^{993}$ anos. ◊

5.2.3 S – T CONEXIDADE

O problema $s - t$ CONEXIDADE EM GRAFOS, conhecido pela sigla USTCON, é o de decidir se dois vértices são ligados por um caminho em um grafo.

Problema computacional da $s - t$ conectividade (USTCON):

Instância: um grafo $G = (V, E)$ e dois vértices s, t em G.

Resposta: *sim* se em G tem um caminho de s para t em G, *não* caso contrário.

Esse problema pode ser resolvido pelos algoritmos clássicos de busca em tempo linear no tamanho do grafo, $|V| + |E|$, e usando espaço extra $\Omega(|V|)$. A pergunta que interessa nesse caso é se o problema pode ser resolvido usando espaço logarítmico. Essa pergunta foi respondida afirmativamente por Reingold (2008, veja o segundo parágrafo da seção 7.2.2). Até então, o único progresso significativo havia sido o algoritmo aleatorizado devido a Aleliunas et al. (1979) que apresentaremos nessa seção. A ideia é deixar um passeio aleatório percorrendo o grafo a partir do vértice s por tempo suficiente para que a probabilidade de encontrar t, desde que possível, seja grande.

O tempo esperado para um passeio aleatório visitar todos os vértices do grafo a partir do vértice v é denotado por C_v e o tempo esperado máximo para visitar todos os vértices do grafo, onde o máximo é tomado sobre todos os vértices como o inicial

$$\mathcal{C}_{\mathsf{G}} = \max_{v \in \mathsf{V}(\mathsf{G})} \mathsf{C}_v$$

é chamado de **tempo de cobertura** de G. Se o grafo é conexo, a probabilidade de um passeio aleatório não visitar todos os vértices após $2\mathcal{C}_{\mathsf{G}}$ passos é no máximo $1/2$, pela desigualdade de Markov (equação (3.33), página 164).

Podemos estimar o tempo de cobertura pelo tempo esperado para percorrer um passeio específico W em G. Para definir W tomamos $\mathsf{H} \subseteq \mathsf{G}$ uma árvore geradora de G, isto é, o grafo $\mathsf{H} = (\mathsf{V}, \mathsf{A})$ com $\mathsf{A} \subset \mathsf{E}$ é subgrafo conexo e sem circuitos (veja um exemplo na figura 5.6). Sempre conseguimos percorrer

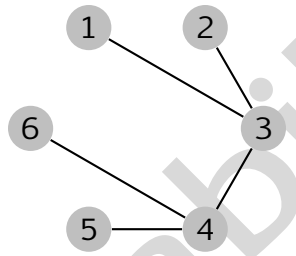


Figura 5.6: uma árvore geradora do grafo da figura 5.3.

uma árvore com n vértices usando uma sequência $(v_0, v_1, \ldots, v_{2n-2} = v_0)$ dos vértices em V tal que cada vértice é visitado pelo menos uma vez e cada aresta é percorrida duas vezes, uma vez em cada sentido. No exemplo acima, $\mathsf{W} = (3, 2, 3, 1, 3, 4, 5, 4, 6, 4, 3)$.

Lema 5.22 *O tempo de cobertura de um grafo conexo* $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ *é no máximo* $4|\mathsf{V}||\mathsf{E}|$.

Demonstração. Sejam $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ um grafo conexo com n vértices e m arestas e $\mathsf{H} \subseteq \mathsf{G}$ uma árvore geradora de G e seja $\mathsf{W} = (v_0, v_1, \ldots, v_{2n-2} = v_0)$ uma sequência de vértices que determina um passeio por todas as $n-1$ arestas dessa árvore de modo que cada aresta seja percorrida exatamente duas vezes, uma vez em cada sentido. A equação (5.12) e a distribuição invariante nos dão o tempo médio de retorno

$$\mu_i = \frac{2m}{d_{\mathsf{G}}(i)}$$

e condicionando no primeiro passo temos, como na equação (5.11), que $\mu_i = 1 + \sum_k p_{i,k}\mu_{k,i}$, portanto

$$\frac{2m}{d_{\mathsf{G}}(i)} = \frac{1}{d_{\mathsf{G}}(i)} \sum_{k \in \mathsf{N}(i)} (1 + \mu_{k,i})$$

em que o conjunto $\mathsf{N}(i)$ é o conjunto de todos os vértice adjacentes a i em G, portanto, $2m = \sum_{k \in \mathsf{N}(i)}(1 + \mu_{k,i})$ donde tiramos que $\mu_{j,i} < 2m$ para toda aresta $\{i, j\}$. Com essa estimativa, o tempo esperado para

um passeio percorrer a sequência W é

$$\sum_{i=0}^{2n-3} \mu_{v_i,v_{i+1}} < (2n-2)(2m) < 4nm$$

o qual é um limitante para o tempo de cobertura de G. □

Com esse resultado conseguimos estimar a probabilidade de erro do algoritmo 44 abaixo para o problema USTCON. O algoritmo começa no vértice s e, sempre que está num vértice, o próximo passo é escolher aleatoriamente um vizinho do vértice atual. Isso é repetido até encontrar t ou até que desista de procurar por ter dado mais que n^4 passos.

Quando o algoritmo devolve *sim* o grafo contém um passeio entre os vértices s e t. Por outro lado, se o algoritmo responde *não* então ou não há passeio, ou o algoritmo não foi capaz de encontrá-lo no tempo disponível. O segundo caso caracteriza uma resposta errada e usando a desigualdade de Markov, equação (3.33), a probabilidade disso acontecer é

$$\mathbb{P}[\,\text{erro}\,] = \mathbb{P}[\,T_{s,t} \geqslant n^4\,] \leqslant \frac{\mu_{s,t}}{n^4} \leqslant \frac{C(G)}{n^4} \leqslant \frac{4n|E|}{n^4} < \frac{4}{n}.$$

Instância: grafo G com $n \geqslant 2$ vértices e dois vértices s e t.
Resposta : *sim*, se há $s-t$-caminho em G, caso contrário *não* com probabilidade de erro $4/n$.

$v \leftarrow s$;
repita
  se $v = t$ **então responda** *sim*.
  $v \leftarrow_R N(v)$;
até que *complete* n^4 *rodadas;*
responda *não.*

Algoritmo 44: $s-t$-conexidade.

Notemos que o espaço extra gasto pelo algoritmo é só o suficiente para manter o vértice atual e o contador de rodadas para o laço, portanto, o espaço utilizado é $O(\log n)$ para um grafo com n vértices.

5.2.4 PASSEIO ALEATÓRIO EM GRAFOS DIRIGIDOS

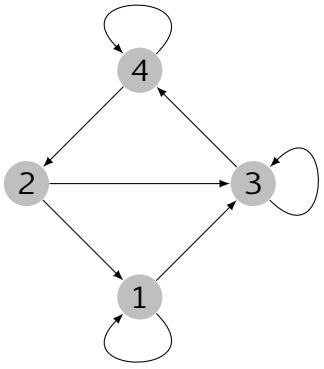
Não é raro modelarmos problemas usando um grafo com a restrição de que algum fluxo que se dá entre os vértices adjacentes só é possível em um sentido, de um dos vértices para o outro. Nesse caso as arestas são orientadas, ou dirigidas. Um **grafo dirigido** $G = (V, E)$, ou simplesmente **digrafo**, com vértices $V = \{1, 2, \ldots, n\}$ tem, como arestas, pares ordenados de vértices, isto é, $E \subset V \times V$. Um

digrafo pode ser representado por uma matriz $A_G = (a_{i,j})$ pondo

$$a_{i,j} := \begin{cases} 1 & \text{se } (i,j) \in E \\ 0 & \text{caso contrário.} \end{cases}$$

Neste capítulo vamos permitir digrafos com laços, ou seja, arestas da forma (v,v), quando não for o caso deixaremos explicito.

Por exemplo, a matriz juntamente com um diagrama para o grafo dirigido com vértices $\{1,2,3,4\}$ e arestas $\{(1,1),(1,3),(2,1),(2,3),(3,3),(3,4),(4,4),(4,2)\}$ são dados na figura 5.7 abaixo.



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

Figura 5.7: diagrama e matriz de um grafo dirigido.

Em um digrafo definimos $d^-(i)$ como o **grau de entrada** do vértice i, para qualquer $i \in V$, e é a quantidade de arestas da forma (x,i) em E. Por outro lado, o **grau de saída** do vértice i, denotado $d^+(i)$, é a quantidade de arestas da forma (i,x) em E. No exemplo da figura 5.7 temos $d^+(1) = d^-(1) = 2$ e $d^+(2) = 2$ e $d^-(2) = 1$.

Numa generalização natural dos passeios em grafos, em digrafos consideramos que um passeio respeita a direção das arestas. A **matriz de transição** $P = (p_{i,j})$ do **passeio aleatório simples** é definida por

$$p_{i,j} = \frac{1}{d^+(i)}$$

e quando $d^+(i) = 0$, caso em que chamamos o vértice i de **sorvedouro** (é o caso do vértice 4 na figura 5.8) adotamos a convenção de que, se não for dito nada, o passeio permanece para sempre nesse vértice, o que reflete no diagrama como a adição de um laço com probabilidade 1 (figura 5.12).

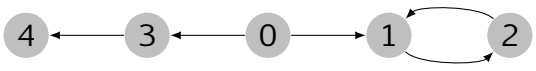
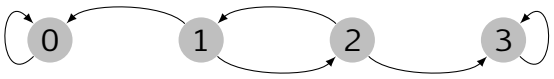


Figura 5.8: um digrafo com sorvedouro.

Por exemplo, o problema da ruína do jogador, página 244, com $p = q$ e $m = 3$ pode ser representado como no diagrama da figura 5.9.



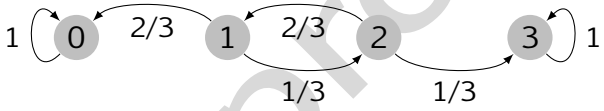
$$\mathsf{P}_{\mathsf{G}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figura 5.9: diagrama de um caso simétrico da ruína do jogador.

Esse exemplo da ruína do jogador sugere ainda outra generalização: um passeio aleatório num **grafo dirigido com pesos** $w\colon \mathsf{E} \to (0,1]$ nas arestas. A **matriz de transição** $\mathsf{P} = (p_{i,j})$ de um passeio nesse grafo é definida por

$$p_{i,j} = \begin{cases} \frac{w(i,j)}{\sum_j w(i,j)} & \text{se } (i,j) \in \mathsf{E} \\ 0 & \text{se } (i,j) \notin \mathsf{E} \end{cases}$$

e ainda vale, como no caso de grafos, que $\mathsf{P} = \mathsf{D}^{-1}\mathsf{A}_{\mathsf{G}}$ se tomamos $d(i) := \sum_j w(i,j)$ e $a_{i,j} = w(i,j)$. Por exemplo, o problema da ruína do jogador, página 244, com $p = 1/3$ e $m = 3$ pode ser representado como no diagrama da figura 5.10.



$$\mathsf{P}_{\mathsf{G}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2/3 & 0 & 1/3 & 0 \\ 0 & 2/3 & 0 & 1/3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figura 5.10: diagrama de um caso assimétrico da ruína do jogador.

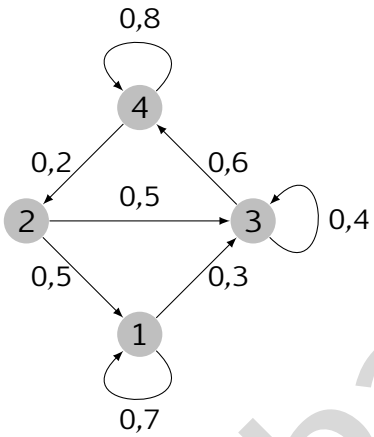
Para uma distribuição inicial $\nu^{(0)} := \nu$ nos vértices de G, se denotarmos por $\nu^{(t)}$ a distribuição de X_t, então $\nu^{(t+1)} = \nu^{(t)}\mathsf{P}$ é a distribuição de X_{t+1}. Para qualquer $t \in \mathbb{N}$, como antes, temos $\mathsf{P}^t = (p_{i,j}^{(t)})$ para

$$p_{i,j}^{(t)} := \mathbb{P}[\, \mathsf{X}_{t+n} = j \mid \mathsf{X}_t = i \,] = \mathbb{P}[\, \mathsf{X}_t = j \mid \mathsf{X}_0 = i \,]$$

a transição do estado i para o estado j em t passos.

Exemplo 5.23 (Ross 2010, exercício 9.9 da seção 9.2). Suponha que a ocorrência de chuva num dia é determinada pela condição do clima nos dois dias anteriores do seguinte modo: amanhã chove com probabilidade 0,7 se chove nos dois dias anteriores; amanhã chove com probabilidade 0,5 se chove hoje mas não choveu ontem; amanhã chove com probabilidade 0,4 se não chove hoje mas choveu

ontem; amanhã chove com probabilidade 0,2 se não chove nos dois dias anteriores. Identificamos cada uma das quatro situações acima com seguintes estados: **1** se choveu hoje e ontem; **2** se choveu hoje mas não choveu ontem; **3** se não choveu hoje mas choveu ontem; e **4** se não choveu nem hoje e nem ontem. O estado X_{n+1} depende da condição nos dias anteriores; assim a transição de $X_n = 1$ para $X_{n+1} = 3$, por exemplo, ocorre quando não chove amanhã mas choveu hoje, dado que choveu hoje e ontem, nessa configuração a probabilidade de não-chuva amanhã se choveu hoje e ontem é $1-0,7 = 0,3$. A evolução desse processo aleatório pode ser visto como um passeio aleatório num grafo onde as arestas são orientadas e têm pesos não negativos associados às probabilidades de transição entre os estados. O diagrama da figura 5.11 ilustra este exemplo. Por exemplo, a transição do estado



$$P = \begin{pmatrix} 0,7 & 0 & 0,3 & 0 \\ 0,5 & 0 & 0,5 & 0 \\ 0 & 0,4 & 0 & 0,6 \\ 0 & 0,2 & 0 & 0,8 \end{pmatrix}$$

Figura 5.11: diagrama de um digrafo com pesos nas arestas e a matriz de transição correspondente.

2 para o estado 1 tem probabilidade 0,5 e do estado 1 pra ele mesmo 0,7, e assim por diante. Dado que choveu na segunda-feira e na terça-feira, qual é a probabilidade de chover na quinta? A matriz de transição em 2 passos é

$$P^2 = \begin{pmatrix} 0,49 & 0,12 & 0,21 & 0,18 \\ 0,35 & 0,20 & 0,15 & 0,30 \\ 0,20 & 0,12 & 0,20 & 0,48 \\ 0,10 & 0,16 & 0,10 & 0,64 \end{pmatrix}.$$

Chover na quinta equivale ao processo estar no estado 1 ou no estado 2, logo a probabilidade é $p^{(2)}_{0,0} + p^{(2)}_{0,1} = 0,61$. Se levarmos um um pouco mais adiante as potências da matriz de transição acima obtemos

$$P^{10} = \begin{pmatrix} 0,2553440 & 0,1491648 & 0,1519040 & 0,4435872 \\ 0,2531733 & 0,1495093 & 0,1511255 & 0,4461919 \\ 0,2486079 & 0,1502124 & 0,1495093 & 0,4516704 \\ 0,2464373 & 0,1505568 & 0,1487306 & 0,4542752 \end{pmatrix},$$

$$P^{20} = \begin{pmatrix} 0{,}2500461 & 0{,}1499928 & 0{,}1500164 & 0{,}4499447 \\ 0{,}2500274 & 0{,}1499957 & 0{,}1500098 & 0{,}4499671 \\ 0{,}2499880 & 0{,}1500019 & 0{,}1499957 & 0{,}4500144 \\ 0{,}2499693 & 0{,}1500048 & 0{,}1499890 & 0{,}4500369 \end{pmatrix},$$

$$P^{30} = \begin{pmatrix} 0{,}2500004 & 0{,}1499999 & 0{,}1500001 & 0{,}4499995 \\ 0{,}2500002 & 0{,}1500000 & 0{,}1500001 & 0{,}4499997 \\ 0{,}2499999 & 0{,}1500000 & 0{,}1500000 & 0{,}4500001 \\ 0{,}2499997 & 0{,}1500000 & 0{,}1499999 & 0{,}4500003 \end{pmatrix},$$

$$P^{34} = \begin{pmatrix} 0{,}2500001 & 0{,}15 & 0{,}15 & 0{,}4499999 \\ 0{,}2500000 & 0{,}15 & 0{,}15 & 0{,}4500000 \\ 0{,}2500000 & 0{,}15 & 0{,}15 & 0{,}4500000 \\ 0{,}2500000 & 0{,}15 & 0{,}15 & 0{,}4500000 \end{pmatrix},$$

$$P^{35} = P^{36} = \cdots P^{40} = \cdots = \begin{pmatrix} 0{,}25 & 0{,}15 & 0{,}15 & 0{,}45 \\ 0{,}25 & 0{,}15 & 0{,}15 & 0{,}45 \\ 0{,}25 & 0{,}15 & 0{,}15 & 0{,}45 \\ 0{,}25 & 0{,}15 & 0{,}15 & 0{,}45 \end{pmatrix}.$$

Portanto, $\nu^{(t)}$ para todo t suficientemente *não depende da distribuição inicial* $\nu^{(0)}$, isto é, se t for suficientemente grande, então a probabilidade do passeio estar em um dos quatro estados é dada pelo vetor $\pi = (0{,}25, 0{,}15, 0{,}15, 0{,}45)$ independentemente do estado inicial. Para π e P acima valem $\pi P = \pi$ e $\pi_j = \lim_{t\to\infty} p_{i,j}^{(t)}$, ou seja, como no passeio em grafo, temos uma *distribuição invariante* e temos *convergência ao equilíbrio*. ◊

Todos os resultados da seção 5.2 valem para digrafos quando generalizamos de maneira adequada as definições de irredutibilidade e periodicidade. Lembremos que, a menos que dito outra coisa, consideramos somente o caso de passeios que satisfazem a condição de Markov, equação (5.5).

IRREDUTIBILIDADE E PERIODICIDADE Nesta seção revemos as condições que garantem *distribuição invariante* e *convergência ao equilíbrio* para um passeio aleatório num grafo dirigido com pesos nas arestas. A existência da distribuição invariante segue da observação 5.8, a mesma que garante a distribuição invariante para passeios em grafos.

Seja P a matriz de transição de um passeio aleatório num digrafo (V, E) com pesos nas arestas. A condição $p_{i,j}^{(t)} > 0$ para algum t significa que a partir do vértice i um passeio chega no vértice j em t passos, portanto esse evento ocorre com probabilidade positiva; nesse caso dizemos que j é um vértice **acessível** a partir de i e escrevemos $i \to j$. Quando j não é acessível a partir de i a probabilidade do passeio chegar em j saindo de i é nula.

Escrevemos $i \leftrightarrow j$ se $i \to j$ e $j \to i$ e dizemos que os vértices i e j se **comunicam**. Deixamos para o leitor a verificação de que $\leftrightarrow$ define uma relação de equivalência sobre V, portanto, particiona V em classes de equivalência chamadas de **componentes fortemente conexas** do digrafo. Nos textos que tratam de Cadeias de Markov as componentes fortemente conexas são chamados de **classes de comunicação**.

Um conjunto de vértices $C \subset V$ contido numa componente fortemente conexa é dito **irredutível**, ou seja, C é irredutível se vale $i \leftrightarrow j$ para todos $i, j \in C$. Quando um conjunto não é irredutível, dizemos **redutível**. Quando há uma única componente fortemente conexa dizemos que o passeio aleatório é um passeio aleatório **irredutível**.

O **período** de um vértice i é, como no caso não dirigido, o $\mathrm{mdc}(t \in \mathbb{N} : p_{i,i}^{(t)} > 0)$. Caso o mdc seja maior que 1 dizemos que i é **periódico** e nesse caso $p_{i,i}^{(t)} = 0$ a menos que t seja múltiplo do período (e o período é maximal com respeito a essa propriedade). Quando o período é 1 nós dizemos que i é **aperiódico**.

Num conjunto irredutível de vértices todos os vértices têm o mesmo período, logo podemos classificá-lo (ou mesmo o próprio passeio irredutível) como: *conjunto/passeio aperiódico* se todos os seus vértices o forem; *conjunto/passeio periódico* se todos os seus vértices o forem.

Por exemplo, um passeio sobre os vértices $\{1,2\}$ dado pela matriz de transições

$$P = \begin{pmatrix} 1 & 0 \\ 1/2 & 1/2 \end{pmatrix}$$

não corresponde a um passeio aleatório irredutível pois para todo inteiro $k > 0$

$$P^k = \begin{pmatrix} 1 & 0 \\ \frac{2^k-1}{2^k} & \frac{1}{2^k} \end{pmatrix}$$

portanto $1 \not\to 2$.

Os vértices 0 e m do exemplo *ruína do jogador*, exemplo na página 244, quando são alcançados pelo passeio aleatório o estado não muda mais de vértice; são sorvedouros. Nesse exemplo (ruína do jogador) o passeio é redutível por causa dos sorvedouros. Nos textos que tratam de Cadeias de Markov são chamados de estados **absorventes**.

Todo passeio aleatório sobre um digrafo com pelo menos dois vértices onde pelo menos um deles é sorvedouro é redutível. Por exemplo, o caso do passeio no grafo dirigido do diagrama da figura 5.12, o passeio é redutível, a classe $\{1,2\}$ é periódica de período 2 e o estado 4 é absorvente.

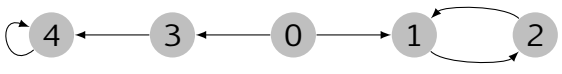


Figura 5.12: um passeio aleatório simples nesse digrafo é redutível.

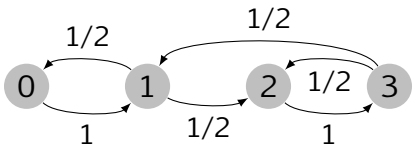


Figura 5.13: diagrama de passeio aleatório irredutível e aperiódico.

No exemplo da figura 5.13, o passeio é aperiódico e irredutível.

Como no caso não dirigido, a propriedade boa é a seguinte, cuja demonstração é análoga ao do caso não dirigido.

Lema 5.24 *Se um passeio aleatório num digrafo finito é irredutível, então ha uma única distribuição invariante e positiva. Se o passeio também é aperiódico, então existe $t_0 \in \mathbb{N}$ tal que para todos i, j vértices, se $t \geqslant t_0$ então $p_{i,j}^{(t)} > 0$.* □

Como no caso não dirigido temos, novamente, que a mesma prova dada na página 265 vale agora, para P matriz de transição de uma passeio aleatório irredutível e aperiódico num digrafo com pesos nas arestas:

$$\lim_{k\to\infty} P^k = Q$$

com Q uma matriz estocástica com as colunas constantes e que uma linha qualquer de Q define π invariante. De fato, se é irredutível então tem uma única distribuição invariante $\pi = (\pi_j : j \in V)$ positiva

$$\pi_j = \lim_{t\to\infty} p_{i,j}^{(t)}$$

para todo i.

Observação 5.25 (*distribuição invariante e convergência em passeios redutíveis ou periódicos*). Num passeio redutível a distribuição invariante pode não ser única. Por exemplo, um passeio com matriz de transição

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

tem distribuição invariante $\pi = (p, 1-p)$ para qualquer $p \in (0,1)$ e, por exemplo, para $p = 1/2$ temos que $p_{1,1}^{(n)} \not\to 1/2$ quando $n \to \infty$. O mesmo acontece com a matriz de transição

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

e o vetor $\pi = (1/2, 1/2)$, não há convergência ao equilíbrio porque o passeio é periódico. Num passeio

por quatro estados com matriz de transições

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

$(1,0,1,0)$ e $(0,1,0,1)$ são vetores invariantes. ◊

PASSEIO ALEATÓRIO NA *WEB*: O *GOOGLE PAGERANK*

> *"To test the utility of PageRank for search, we built a web search engine called Google"* (Page et al., 1998).

O *grafo web* é um grafo dirigido definido pelas páginas *web* e as ligações (*links* ou *hyperlinks*) entre as páginas. Conhecer a estrutura desse grafo é importante para, por exemplo, o desenvolvimento de algoritmos eficientes para a *web*. Um marco no projeto e desenvolvimento desses algoritmos é o algoritmo *PageRank* (Page et al., 1998) desenvolvido como parte da ferramenta de busca na *web* batizada *Google* pelos fundadores da empresa.

O *PageRank* é um algoritmo para classificação de páginas na *web*. A ideia que o motivou é modelar a importância relativa de uma página de modo que uma busca devolva primeiro resultados com maior relevância. A própria empresa explica a ideia da seguinte maneira (Google, 2011):

> "O coração do nosso software é o PageRank(TM), um sistema para dar notas para páginas na web, desenvolvido pelos nossos fundadores Larry Page e Sergey Brin na Universidade de Stanford. E enquanto nós temos dúzias de engenheiros trabalhando para melhorar todos os aspectos do Google no dia a dia, PageRank continua a ser a base para todas nossas ferramentas de busca na web.
>
> Explicações sobre o PageRank
>
> A classificação das páginas (PageRank) confia na natureza excepcionalmente democrática da Web, usando sua vasta estrutura de links como um indicador do valor de uma página individual. Essencialmente, o Google interpreta um link da página A para a página B como um voto da página A para a página B. Mas o Google olha além do volume de votos, ou links, que uma página recebe; analisa também a página que dá o voto. Os votos dados por páginas "importantes" pesam mais e ajudam a tornar outras páginas "importantes"."

Assim, uma página tem uma classificação alta se é referenciada por páginas com classificação alta. O modelo adotado no *PageRank* pode ser interpretado como um passeio aleatório no grafo *web*.

O modelo simplificado do *PageRank* é descrito da seguinte maneira. Seja $G = (V, E)$ o digrafo da *web*, ou seja, V é o conjuntos formado pelas páginas *web*, as quais serão consideradas sem perda de generalidade $\{1, 2, \ldots, n\}$, e $(a, b) \in E$ se na página a há um *link* que leva para a página b. Denotamos por $N^+(a)$ o conjunto dos vértices b tais que (a, b) é uma aresta de G, ou seja, é o conjunto das páginas *web* que que têm um *link* na página a e cuja cardinalidade é $d^+(a)$. Denotamos por $N^-(a)$ o conjunto dos vértices b tais que (b, a) é uma aresta de G, ou seja, é o conjunto das páginas *web* que têm um *link* apontando para a página a e cuja cardinalidade é $d^-(a)$. Por exemplo, no grafo *web* da figura 5.14 $d^+(3) = |N^+(3)| = 2$ e $d^-(3) = |N^-(3)| = 3$.

A classificação dos vértices é dada por um vetor $r = (r_a)_{a \in V}$ onde r_a é a classificação do vértice a e satisfaz

$$r_a = \sum_{b \in N^-(a)} \frac{r_b}{d^+(b)} \tag{5.13}$$

ou seja, se b aponta para a então b contribui com $1/d^+(b)$ de sua relevância para a relevância de a. Seja P a matriz de transições do passeio aleatório simples, dada pelos elementos

$$p_{a,b} := \begin{cases} \frac{1}{d^+(a)} & \text{se } (a, b) \text{ é aresta} \\ 0 & \text{caso contrário,} \end{cases}$$

então de (5.13)

$$r = rP$$

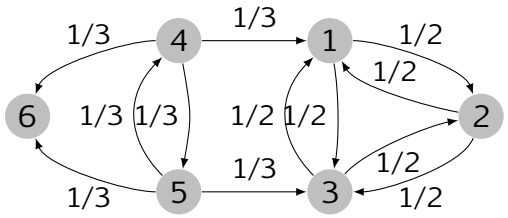
ou seja, r é um vetor invariante de P.

Se P for uma matriz estocástica então o vetor r pode ser obtido escolhendo uma distribuição inicial $\nu^{(0)}$ e fazendo $\nu^{(k+1)} = \nu^{(k)}P$ pois como vimos, sob certas hipóteses $\nu^{(k)}$ converge para r. Esse método, conhecido como método das potências, é usado há muito tempo para calcular autovetores associado ao maior autovalor. Entretanto, na atual situação não sabemos se o vetor converge pois não temos garantia que a matriz P seja irredutível (garante $\lambda_1 > \lambda_2$) ou mesmo estocástica (garante o autovalor $\lambda_1 = 1$). Por exemplo, para

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

e $\nu^{(0)} = (0, 1)$ temos $\nu^{(2)} = r = (0, 0)$, No caso de um circuito dirigido, um vetor inicial pode não convergir.

No exemplo da figura 5.14 abaixo a matriz P não é estocástica. Os vértices sem arestas que saem, ou seja os vértices v tais que $d^+(v) = 0$ são chamados, no trabalho original, de pendentes (*dangling*) e por causa deles a matriz não é estocástica. No exemplo da figura 5.14 o vértice 6 é pendente. Na *web* de fato há vários vértices pendentes, por exemplo, os documentos em pdf disponíveis em páginas *web* são, normalmente, vértices pendentes do grafo.



$$P = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1/3 & 1/3 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

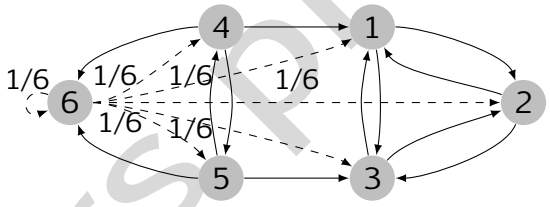
Figura 5.14: exemplo de grafo *web*.

Definimos uma matriz auxiliar B pondo para cada vértice pendente v, o que corresponde a uma página sem *link*s, a linha v com entradas $1/n$ e tomamos

$$Q = P + B$$

isso significa que um passeio aleatório que chega numa página sem saída continua em qualquer outra página com igual probabilidade. A matriz Q é estocástica.

A matriz Q referente a matriz P da figura 5.14 é dada a seguir na figura 5.15 ao lado do grafo *web* que corresponde à modificação no grafo que reflete a modificação na matriz P, as arestas tracejadas são as arestas incluídas e correspondem à possibilidade de navegação do internauta que chega a uma página sem saída e recomeça a navegação de qualquer lugar uniformemente.



$$Q = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1/3 & 1/3 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{pmatrix}$$

Figura 5.15: modelo de grafo *web* modificado, sem vértices pendentes. As arestas tracejadas são as arestas incluídas artificialmente.

No exemplo dado na figura 5.15, o maior autovalor da matriz Q é 1 com multiplicidade 1 e autovetor associado $(1,1,1,0,0,0)$. Notemos que os vértices 4, 5, 6 tem classificação 0. Isso decorre do fato de não haver passeios que saem de $\{1,2,3\}$ e chegam em qualquer outro vértice diferente desses. Esse conjunto é chamado de sorvedouro (*rank sink*).

Para lidar com esses sorvedouros basta garantir que a matriz seja irredutível pois se $p_{a,b}^{(k)} > 0$ para algum k, para todo $a,\ b \in V$ então não há sorvedouros no grafo. Para garantir uma matriz irredutível tomamos $p \in (0,1)$ e consideramos um passeio aleatório que segue as transições de Q com probabilidade p ou que com probabilidade $1-p$ vai pra qualquer outra página *web*, ou seja,

$$R = pQ + (1-p)\frac{1}{n}\mathbf{1}$$

onde $\mathbf{1}$ é a matriz com todas as entradas iguais a 1. A matriz obtida de Q do exemplo da figura 5.15

$$R = \begin{pmatrix} \frac{1-p}{6} & \frac{p}{2}+\frac{1-p}{6} & \frac{p}{2}+\frac{1-p}{6} & \frac{1-p}{6} & \frac{1-p}{6} & \frac{1-p}{6} \\ \frac{p}{2}+\frac{1-p}{6} & \frac{1-p}{6} & \frac{p}{2}+\frac{1-p}{6} & \frac{1-p}{6} & \frac{1-p}{6} & \frac{1-p}{6} \\ \frac{p}{2}+\frac{1-p}{6} & \frac{p}{2}+\frac{1-p}{6} & \frac{1-p}{6} & \frac{1-p}{6} & \frac{1-p}{6} & \frac{1-p}{6} \\ \frac{1-p}{6} & \frac{1-p}{6} & \frac{p}{3}+\frac{1-p}{6} & \frac{1-p}{6} & \frac{p}{3}+\frac{1-p}{6} & \frac{p}{3}+\frac{1-p}{6} \\ \frac{1-p}{6} & \frac{p}{3}+\frac{1-p}{6} & \frac{1-p}{6} & \frac{p}{3}+\frac{1-p}{6} & \frac{1-p}{6} & \frac{p}{3}+\frac{1-p}{6} \\ \frac{p}{6}+\frac{1-p}{6} & \frac{p}{6}+\frac{1-p}{6} & \frac{p}{6}+\frac{1-p}{6} & \frac{p}{6}+\frac{1-p}{6} & \frac{p}{6}+\frac{1-p}{6} & \frac{p}{6}+\frac{1-p}{6} \end{pmatrix}$$

É sabido que o método iterativo para computar o vetor estacionário descrito, o *método das potências*, converge com velocidade $|\lambda_2/\lambda_1|$ (Golub e Van Loan, 1989), onde $\lambda_1 > \lambda_2$ são os dois maiores autovalores da matriz R. Ainda, é sabido que $\lambda_1 = 1$ e que $|\lambda_2| \leqslant p$ (Haveliwala e Kamvar, 2003).

O parâmetro p é conhecido como fator de amortecimento (*damping factor*). No trabalho que originou o algoritmo os autores do *PageRank* estabeleceram o valor $p = 0,85$ (Page et al., 1998) após testes e, em seguida, a empresa não divulgou mais o valor desse fator. No mesmo trabalho (Page et al., 1998), os autores reportam de 50 a 100 iterações do método das potências até a condição de parada do método das potências. O critério tradicional de parada é da forma $\|\nu^{(t+1)} - \nu^{(t)}\| < \varepsilon$ para uma tolerância $\varepsilon > 0$ pequena; notemos que não é necessário conhecer as grandezas do vetor estacionário, só é preciso determinar a ordem das coordenadas, o que pode ser usado para diminuir o número de iterações. Para $p = 0,85$ foi reportado que com 29 iterações $\|\nu^{(t+1)} - \nu^{(t)}\| < 10^{-2}$, e no caso de 50 a 100 iterações a tolerância é de 10^{-3} a 10^{-7}.

Para concluirmos esta seção vão mostrar uma alternativa para escrever essa matriz, uma vez que atordoantes 1 trilhão de páginas indexadas foi reportado pela Google em julho de 2008 (*We knew the web was big...* 2008). A matriz P é esparsa pela natureza da *web*: páginas com poucos *link*s, 52 em média (Wills, 2006), e muitos vértices pendentes, a matriz Q é mais densa e a matriz R é positiva. A matriz B pode ser escrita como $u^T a$ onde u é o vetor com todas as entradas iguais a $1/n$. De fato, pode ser qualquer vetor estocástico, o vetor uniforme foi a escolha original, mas atualmente sabe-se que favorece *link spamming* e esse parâmetro também não é divulgado pela empresa Google. O vetor a é o transposto do vetor característico dos vértices pendentes. Com essas definições e $\mathbf{1} := (1,1,\ldots,1)$,

$$\nu^{(t)} = \nu^{(t-1)}R = p\nu^{(t-1)}Q + (1-p)\frac{1}{n}\nu^{(t-1)}\mathbf{1} = p\nu^{(t-1)}Q + (1-p)u$$

pois $\nu^{(t-1)}\mathbf{1}$ é um vetor com todas as entradas iguais a 1, logo

$$\nu^{(t)} = p\nu^{(t-1)}\mathsf{P} + p\nu^{(t-1)}\mathsf{u}^\intercal\mathsf{a} + (1-p)\mathsf{u}$$

portanto só precisamos armazenar a matriz P e os vetores a e u.

Exercício 5.26. Mostre que a matriz R é irredutível.

Exercício 5.27. O grafo dirigido com vértices $\{0, 1, \ldots, n-1\}$ e arestas $(i, i+1 \mod n)$ é um circuito dirigido com n vértices. Analise o comportamento de $\nu^{(k)}$ para $k \in \mathbb{N}$ com $\nu^{(0)} = (1, 0, \ldots, 0)$.

5.2.5 RECORRÊNCIA E UM TEOREMA FUNDAMENTAL

Além da redutibilidade e do período, os vértices de um grafo dirigido são classificados em dois tipos fundamentais com respeito a um passeio nesse grafo: os vértices *recorrentes* e os vértices *transitórios*. Um vértice i é dito **recorrente** se para todo vértice j, $i \to j$ implica $j \to i$. Se um vértice é recorrente, uma vez tendo vértice nele, é certo que o passeio eventualmente volta a ele e, assim, visita-o infinitas vezes. Um vértice que não é recorrente é dito **transitório**. Notemos que essa definição leva a seguinte conclusão imediata: *num passeio em um grafo finito pelo menos um vértice é recorrente*. De fato, consideremos uma sequência de vértices $v_0, v_1, \ldots$ definida recursivamente por: tome $v_0 \in V$ qualquer; para $i \geqslant 1$ se v_{i-1} é recorrente, então pare, senão escolha v_i tal que $v_{i-1} \to v_i$ mas $v_i \not\to v_{i-1}$. Se existirem $i < j$ com $v_i = v_j$ então $v_i \to v_{j-1} \to v_i$, o que é uma contradição. Como V é finito a sequência deve terminar num vértice recorrente.

No exemplo da figura 5.16, temos um passeio irredutível, recorrente e periódico.

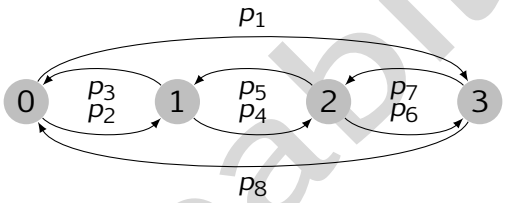


Figura 5.16: exemplo de passeio irredutível, recorrente e periódica.

Agora, considere um passeio sobre $V = \{1,2,3,4\}$ e matriz de transição

$$P = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/4 & 3/4 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

cujas classes de comunicação são $\{1,2\}$ que é recorrente, $\{4\}$ que é absorvente e $\{3\}$ que é tran-

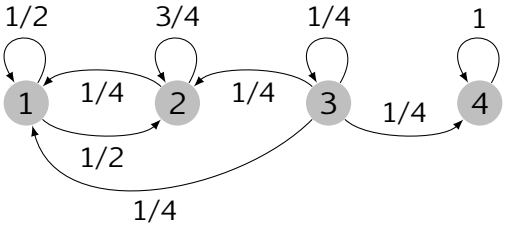


Figura 5.17: probabilidade de transição de classe transitório para recorrente.

sitório. Se $X_0 = 3$, então a probabilidade p com que o passeio entra na classe recorrente $\{1,2\}$ é,

condicionando em X_1, dada por

$$p = \frac{1}{4}1 + \frac{1}{4}1 + \frac{1}{4}p + \frac{1}{4}0 = \frac{1}{2} + \frac{1}{4}p$$

logo $p = 2/3$ e com probabilidade 1/3 o passeio aleatório é absorvido em 4.

No caso da matriz de transição

$$Q = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 2/3 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1/3 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 0 & 1/3 & 1/6 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

as classes de comunicação são $\{1,2\}$, $\{3,4\}$ e $\{5,6\}$, respectivamente, recorrente aperiódica, transitório e recorrente periódica. Se p_i é a probabilidade de entrar na classe recorrente aperiódica descrita acima a partir do estado i, para $i = 3,4$, então, condicionando em X_1,

$$p_3 = \frac{1}{3}1 + 0 \cdot 1 + 0p_3 + \frac{1}{3}p_4 + \frac{1}{6}0 + \frac{1}{6}0$$
$$p_4 = \frac{1}{6}1 + \frac{1}{6}1 + \frac{1}{6}p_3 + 0p_4 + \frac{1}{3}0 + \frac{1}{6}0$$

cuja solução é $p_3 = 8/17$ e $p_4 = 7/17$.

No caso de um passeio sobre $V = \{1,2,3,4,5,6\}$ com as probabilidades das transições não nulas descritas no diagrama da figura 5.18, os vértices $1,2,5,6$ são recorrentes enquanto que $3,4$ são tran-

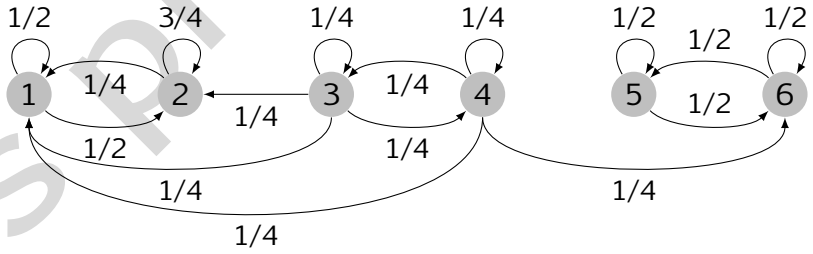


Figura 5.18: diagrama para um passeio aleatório.

sitórios. Ainda, a probabilidade de retorno ao vértice 1, dado que $X_0 = 1$, em t passos é 1/2 para $t = 1$ e para $t \geqslant 2$ é $(1/2)(3/4)^{t-2}(1/4)$, portanto, o tempo médio de retorno para o vértice 1 é (usando s.6c)

$$\mu_1 = \frac{1}{2} + \frac{1}{8}\sum_{t\geqslant 2} t\left(\frac{3}{4}\right)^{t-2} = \frac{1}{2} + \frac{2}{9}\sum_{t\geqslant 2} t\left(\frac{3}{4}\right)^{t} = \frac{1}{2} + \frac{5}{2} = 3$$

Lema 5.28 *Recorrência, transiência e período são propriedades de classe de comunicação, isto é, dois vértices que se comunicam tem a mesma classificação.*

Demonstração. Suponha que o vértice i é transitório, ou seja, para algum j, $i \to j$ porém $j \not\to i$ e suponha que i e m são vértices que se comunicam. Então $m \to i$ e $i \to j$, portanto, então $m \to j$. Agora, se $j \to m$, então $j \to i$, uma contradição. Logo m é transitório.

Sejam i e j vértices que se comunicam e sejam n e m inteiros positivos tais que $p_{i,j}^{(n)} > 0$ e $p_{j,i}^{(m)} > 0$. Para todo t

$$p_{i,i}^{(t+n+m)} \geqslant p_{i,j}^{(n)} p_{j,j}^{(t)} p_{j,i}^{(m)}$$

portanto, $p_{i,i}^{(t+n+m)} > 0$ sempre que $p_{j,j}^{(t)} > 0$.

Seja $\ell(i)$ o período do vértice i. Das hipóteses temos $p_{i,i}^{(n+m)} \geqslant p_{i,j}^{(n)} p_{j,i}^{(m)} > 0$ portanto $\ell(i)$ divide $n+m$. Se $p_{j,j}^{(t)} > 0$ então $p_{i,i}^{(t+n+m)} > 0$, logo $\ell(i)$ divide $t+n+m$. Como $\ell(i)$ divide $n+m$, concluímos que $\ell(i)$ divide t para todo t tal que $p_{j,j}^{(t)} > 0$, logo $\ell(i) \leqslant \ell(j)$.

Trocando os papéis de i e j concluímos que $\ell(j) \leqslant \ell(j)$. Portanto, os períodos são iguais. □

Observação 5.29. No caso P irredutível e periódico de período ℓ temos uma partição $V_0, V_1, \ldots, V_{\ell-1}$ de V com cada parte fechada por acessibilidade ($j \in C$ sempre que $i \to j$ e $i \in C$) e irredutível com respeito a matriz de transição P^ℓ:

assuma que todos os vértices tenham período $\ell > 1$ e que para cada par (i,j) existe um $t_0 = t_0(i,j)$ para o qual $p_{i,j}^{(t_0)} > 0$. Seja v_0 um vértice qualquer e V_0 sua classe de comunicação. Da irredutibilidade, para todo vértice k devem existir instantes t_1 e t_2 tais que

$$p_{v_0,k}^{(t_1)},\ p_{k,v_0}^{(t_2)} > 0$$

e ponha $k \in V_r$ se, e só se, $r \in \{0,1,\ldots,\ell-1\}$ é único tal que $t_1 = \ell q + r$. Desse modo *se* $k \in V_r$ *então* $p_{v_0,k}^{(t)} = 0$ *a menos que* $t = r + \ell a$ *para algum inteiro positivo* a. Agora, consideramos $V_0, \ldots, V_{\ell-1}, V_0$ ciclicamente nessa ordem; cada passo do passeio sai de um vértice em V_i e chega num vértice de V_{i+1} (a soma no índice é módulo ℓ) e a cada ℓ passos o passeio está de volta à parte de saída.

Assim, se $k \in V_r$ então $p_{v_0 0,k}^{(t)} = 0$ a menos que $t \equiv r \pmod{\ell}$, para todo $r \in \{0,1,\ldots,\ell-1\}$. A partir desse fato temos que para um vértice $k \in V$ recorrente vale que o tempo de recorrência médio com respeito a P^ℓ é μ_k/ℓ, e (com respeito a essa matriz) cada parte é irredutível e recorrente, assim para cada $j \in V_r$

$$\lim_{t\to\infty} p_{j,k}^{(t\ell)} = \frac{\ell}{\mu_k}$$

(e tal limite é zero quando $k \notin V_r$ ou k é transitório). Ademais, $(1/\mu_k : k \in V)$ é uma distribuição invariante (veja Feller, 1968, seção XV.9).

TEOREMA FUNDAMENTAL O número de visitas ao vértice i é a variável aleatória

$$\mathsf{V}_i := \sum_{n\geqslant 1} \mathbb{1}_{[\mathsf{X}_n = i]}$$

em que $\mathbb{1}_{[\mathsf{X}_n=i]}$ é a variável aleatória indicadora do evento $[\mathsf{X}_n = i]$, que é infinito se e só se n é transiente. De fato, pela proposição 3.22, página 135,

$$\mathbb{E}[\,\mathsf{V}_i \mid \mathsf{X}_0 = i\,] = \sum_{k\geqslant 1} \mathbb{P}[\,\mathsf{V}_i \geqslant k \mid \mathsf{X}_0 = i\,]$$

e, saindo de i, visitar j pelo menos k vezes equivale a visitar j e em seguida visitar j pelo menos $k-1$ vezes, o que ocorre com probabilidade $f_{i,j}(f_{j,j})^{k-1}$ logo, fazendo $j = i$ nessa dedução ficamos com

$$\mathbb{E}[\,\mathsf{V}_i \mid \mathsf{X}_0 = i\,] = \sum_{k\geqslant 1} (f_{i,i})^k = \begin{cases} \frac{f_{i,i}}{1-f_{i,i}} & \text{se } f_{i,i} < 1, \text{ ou seja, se } i \text{ é transiente} \\ \infty & \text{se } f_{i,i} = 1, \text{ ou seja, se } i \text{ é recorrente.} \end{cases}$$

Ainda, pela linearidade da esperança,

$$\mathbb{E}[\,\mathsf{V}_i \mid \mathsf{X}_0 = i\,] = \sum_{n\geqslant 0} \mathbb{E}\Big[\,\mathbb{1}_{[\mathsf{X}_n=i]} \,\Big|\, \mathsf{X}_0 = i\Big] = \sum_{n\geqslant 0} \mathbb{P}[\,\mathsf{X}_n = i \mid \mathsf{X}_0 = i\,] = \sum_{n\geqslant 0} p_{i,i}^{(n)}$$

e a classificação de vértices pode ser caracterizada pelas probabilidades de transição da maneira descrita a seguir.

Lema 5.30 *O vértice i é*

- *transiente se e só se* $\sum_{n\geqslant 0} p_{i,i}^{(n)} < \infty$.
- *recorrente se e só se* $\sum_{n\geqslant 0} p_{i,i}^{(n)} = \infty$.

□

Observação 5.31 (*Teorema Ergódico*). Se uma sequência converge $\lim_{n\to\infty} a_n = a$, então a sequência das médias parciais também converge

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} a_i = a$$

portanto de (5.18) temos

$$\lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} p_{i,j}^{(t)} = \frac{1}{\mu_j}$$

no caso de uma cadeia ergódica. Agora

$$\frac{1}{n} \sum_{t=0}^{n-1} p_{i,j}^{(t)} = \frac{1}{n} \sum_{t=0}^{n-1} \mathbb{E}\Big[\,\mathbb{1}_{[\mathsf{X}_t=j]} \,\Big|\, \mathsf{X}_0 = i\Big]$$

ou seja, se $V_j(n)$ é o número de visitas a j no intervalo de tempo $0, \ldots, n-1$

$$\sum_{t=0}^{n-1} p_{i,j}^{(t)} = \mathbb{E}\Big[V_j(n) \,\Big|\, X_0 = i \Big]$$

logo

$$\lim_{n\to\infty} \frac{1}{n} \mathbb{E}\Big[V_j(n) \,\Big|\, X_0 = i \Big] = \frac{1}{\mu_j}.$$

Em palavras, π_j é a fração média de visitas ao estado j por unidade de tempo. Aqui, não provaremos o seguinte resultado, conhecido como **Teorema ergódico** para cadeias de Markov finitas: *Para qualquer cadeia de Markov irredutível*

$$\mathbb{P}\left(\lim_{n\to\infty} \frac{V_j(n)}{n} = \frac{1}{\mu_j} \right) = 1.$$

Resumindo alguns dos resultados provados enunciamos o seguinte teorema.

Teorema 5.32 (teorema fundamental dos passeios aleatórios em grafos finitos) *Em um passeio aleatório irredutível e aperiódico em um grafo (dirigido) temos que*

1. *existe uma única distribuição invariante e positiva π;*
2. *para qualquer distribuição inicial ν, $\lim_{t\to\infty} \nu P^t = \pi$, ou ainda, $\lim_{t\to\infty} p_{i,j}^{(t)} = \pi_j$ e esse limite é independente de i;*
3. *o tempo médio de retorno ao estado j é $\mu_j = \frac{1}{\pi_j}$.*
4. *Se $V_i(t)$ é o número de visitas ao vértice i no intervalo de tempo $[0, t-1]$, então para qualquer distribuição inicial*

$$\lim_{t\to\infty} \frac{V_i(t)}{t} = \pi_i$$

com probabilidade que tende a 1.

5.3 CADEIAS DE MARKOV HOMOGÊNEAS

Cadeias de Markov discretas são um caso particular de processo estocástico discreto que satisfazem a **condição de Markov** de que o próximo estado depende somente do estado atual e é independente dos estados passados

$$\mathbb{P}[\, X_{t+1} = j \mid X_t = i,\, X_{t-1} = s_{t-1}, \cdots,\, X_0 = s_0 \,] = \mathbb{P}[\, X_{t+1} = j \mid X_t = i \,] \tag{5.14}$$

e ela é dita **homogênea** se essa probabilidade não depende do tempo t. Como todas as cadeias desse capítulo são discretas, homogêneas e em tempo discreto nós omitiremos esses adjetivos daqui em

diante. Chamamos X_0 de **estado inicial**. Uma coleção $\rho = \{\rho_i : i \in S\}$ com $\rho_i \geqslant 0$, de modo que $\sum_{i\in S} \rho_i = 1$ é chamada **distribuição** e, particularmente, é chamada **distribuição inicial** no caso em que $\rho_i = \mathbb{P}(X_0 = i)$. Uma **cadeia de Markov** é caracterizada por

1. um conjunto de estados S,
2. uma distribuição inicial $\{\rho_i : i \in S\}$,
3. probabilidades de transição $p(i,j) \geqslant 0$ do estado i para o estado j, para todo $(i,j) \in S^2$ com $\sum_{j\in S} p(i,j) = 1$ para todo $i \in S$

de modo que o processo estocástico $\{X_t : t \geqslant 0\}$ com valores em S é uma *cadeia de Markov homogênea em tempo discreto, com distribuição inicial* ρ *e probabilidade transição* p se a distribuição conjunta de $X_0, \ldots, X_n$, para todo $n \in \mathbb{Z}^+$, é

$$\mathbb{P}[X_0 = s_0, X_1 = s_1, \cdots, X_n = s_n] = \rho_{s_0} \prod_{i=0}^{n-1} p(s_i, s_{i+1}) \tag{5.15}$$

para toda escolha de $s_0, s_1, \ldots, s_n \in S$. Dessa distribuição conjunta podemos derivar (verifique) a condição de Markov dada na equação (5.14) acima. Ademais, fixado um natural n, temos a partir dos itens 2 e 3 acima que

$$\sum_{s_0\in S}\sum_{s_1\in S}\cdots\sum_{s_{n-1}\in S}\sum_{s_n\in S} \rho_{s_0}\prod_{i=0}^{n-1} p(s_i,s_{i+1}) = \sum_{s_0\in S}\sum_{s_1\in S}\cdots\sum_{s_{n-1}\in S} \rho_{s_0}\prod_{i=0}^{n-2} p(s_i,s_{i+1})\sum_{s_n\in S} p(s_{n-1},s_n) =$$

$$\sum_{s_0\in S}\sum_{s_1\in S}\cdots\sum_{s_{n-1}\in S} \rho_{s_0}\prod_{i=0}^{n-2} p(s_i,s_{i+1}) = \cdots = \sum_{s_0\in S}\sum_{s_1\in S} \rho_{s_0} p(s_0,s_1) = \sum_{s_0\in S} \rho_{s_0} = 1$$

portanto (5.15) é uma atribuição de probabilidades válida no espaço das realizações das n variáveis aleatórias. A mesma dedução vale para qualquer valor de n portanto a definição da distribuição é consistente. O célebre Teorema da Extensão de Kolmogorov garante a existência do espaço de probabilidade e das variáveis satisfazendo as condições acima.

Convencionamos que quando é dada uma definição de uma cadeia de Markov, isto é, dados o conjunto de estados, a distribuição inicial e as probabilidades de transição, convencionamos que as probabilidades de transição que não são dadas explicitamente são iguais a zero. Ademais, em muitos casos não damos a distribuição inicial porque ou esse parâmetro não é relevante no momento ou é trivial ($\mathbb{P}[X_0 = s] = 1$ para algum s).

Exemplo 5.33 (*embaralhamento*). Numa pilha com n cartas de baralho distintas numeradas de 1 a n, definimos uma cadeia de Markov tomando um estado para cada permutação π, em que $\pi(i)$ é a posição da i-ésima carta; X_0 é a permutação identidade com probabilidade 1. Uma transição entre estados é obtida retirando a carta de uma posição arbitrária escolhida uniformemente entre as cartas

na pilha e colocando-a no topo, logo $p(\pi,\sigma) = 1/n$ para todo instante $t \geqslant 0$ e com a permutação σ obtida a partir da permutação π pelo processo descrito acima (nos casos em que σ não pode ser obtida pelo processo descrito vale $p(\pi,\sigma) = 0$, como convencionamos). Um problema interessante é estimar o valor n, caso exista, para o qual a distribuição de X_n seja aproximadamente uniforme, segundo alguma métrica. ◊

people.eecs.berkeley.edu/~sinclair/cs294/n2.pdf

Exemplo 5.34 (*passeio aleatório em* N). Definimos uma cadeia de Markov se tomamos o conjunto de estados S como sendo os inteiros positivos, o estado inicial 1 e as transições com probabilidades $p(i,1) = 1/i+1$ e $p(i,i+1) = i/i+1$ para todo inteiro $i \geqslant 1$. ◊

Exemplo 5.35 (*passeio aleatório num grafo*). Seja G um grafo localmente finito, isto é, os vértices têm grau finito. Definimos uma cadeia de Markov tomando S como o conjunto de vértices do grafo e as transições são definidas de modo que se $X_t = v$ então X_{t+1} é qualquer vértice adjacente a v com probabilidade uniforme. Esse tipo de cadeia de Markov é conhecida como passeio aleatório em G. ◊

Exemplo 5.36 (*cadeia de Markov não-homogênea*). Consideremos uma caixa com v bolas vermelhas e a bolas azuis e o processo de retirar aleatoriamente uma bola da caixa sem reposição. Seja X_t o número de bolas vermelhas na caixa na t-ésima rodada. A probabilidade de transição não depende só do número de bolas vermelhas (estado), mas também depende do momento t. ◊

*Exercício 5.37 (**propriedade de Markov**).* Prove que se $\{X_t : t \geqslant 0\}$ é uma cadeia de Markov com respeito a distribuição inicial $\{\rho_i : i \in S\}$ e probabilidades de transição $\{p(i,j) : i,j \in S\}$, então condicionado ao evento $[X_m = i]$ o processo $\{X_{t+m} : t \geqslant 0\}$ é uma cadeia de Markov com respeito a distribuição inicial $\{\delta_i(j) : i \in S\}$ e probabilidades de transição $\{p(i,j) : i,j \in S\}$ e independente de $X_0,\ldots,X_m$. Acima, $\delta_i(j) = 1$ se e só se $i = j$.

Nesta seção estudaremos as condições que garantem *distribuição invariante* e *convergência ao equilíbrio* para uma cadeia de Markov.

5.3.1 REPRESENTAÇÃO MATRICIAL

A coleção de números $\{\rho_i : i \in S\}$ com $\rho_i \geqslant 0$ e $\sum_{i\in S} \rho_i = 1$ pode ser vista[5] como um vetor-linha $\rho = (\rho_i : i \in S)$ chamado **vetor de probabilidades** ou **distribuição de** X no caso em que $\rho_i = \mathbb{P}(X = i)$. Do mesmo modo, podemos interpretar as probabilidades de transição como uma matriz $P = (p_{i,j})$ definida por $p_{i,j} = p(i,j)$, chamada **matriz de transição** da cadeia de Markov. Para cada linha i vale que $\sum_j p_{i,j}$. Uma matriz cujas entradas são não-negativas e as linhas somam 1 é uma **matriz estocástica**.

[5] A rigor, ρ é uma função $S \to [0,1]$ que podemos representar por uma sequência implicitamente induzida por uma enumeração $s_0, s_1, s_2, \ldots$ de S.

O espaço de estados pode ser infinito e, portanto, essa matriz não é do tipo estudado em álgebra linear. No entanto, adição e multiplicação são definidas pelas mesmas regras formais. Com isso, se ρ é a distribuição de X_{n-1} para algum $n \geqslant 1$, então o vetor de probabilidades $\pi = \rho P$ dado por

$$\pi_i = \sum_{j \in S} \rho_j p_{j,i} = \sum_{j \in S} \mathbb{P}[\, X_{n-1} = j \,]\mathbb{P}[\, X_n = i \mid X_{n-1} = j \,] = \mathbb{P}[\, X_n = i \,]$$

para todo $i \in S$, é a distribuição de X_n.

Para uma distribuição inicial ρ, se fizermos $\pi^{(0)} := \rho$ e denotarmos por $\pi^{(1)}$ a distribuição de X_1, então $\pi^{(1)} = \pi^{(0)}P$. Analogamente, $\pi^{(2)} = \pi^{(1)}P$ é a distribuição de X_2 e, em geral se $\pi^{(t-1)}$ é a distribuição de X_{t-1} então

$$\pi^{(t)} = \pi^{(t-1)}P = (\pi^{(t-2)}P)P = \cdots = \pi^{(0)}P^t$$

é a distribuição de X_t para todo natural $t \geqslant 1$. Para qualquer $t \in \mathbb{N}$, se

$$p_{i,j}^{(t)} := \mathbb{P}[\, X_{t+n} = j \mid X_t = i \,] = \mathbb{P}[\, X_t = j \mid X_0 = i \,]$$

é a transição do estado i para o estado j em t passos então vale

$$p_{i,j}^{(t)} = \sum_{s \in S} p_{i,s} p_{s,j}^{(t-1)}$$

para todo $t > 1$, com $p_{i,j}^{(1)} = p_{i,j}$. Logo a t-ésima potência da matriz P é a matriz de transição em t passos

$$P^t = (p_{i,j}^{(t)}).$$

Lema 5.38 (Identidade de Chapman–Kolmogorov) *Se $P = (p_{i,j})$ é uma matriz de transição,*

$$p_{i,j}^{(k+t)} = \sum_{s \in S} p_{i,s}^{(k)} p_{s,j}^{(t)} \tag{5.16}$$

para quaisquer i, $j \in S$.

Demonstração. A partir de

$$\mathbb{P}[\, X_{k+t} = j \mid X_0 = i \,] = \sum_{s \in S} \mathbb{P}[\, X_{k+t} = j,\, X_k = s \mid X_0 = i \,]$$

segue que

$$p_{i,j}^{(k+t)} = \sum_{s \in S} \mathbb{P}[\, X_{k+t} = j,\, X_k = s \mid X_0 = i \,]$$

e, a partir da equação (1.12), página 26, podemos deduzir

$$\begin{aligned} p_{i,j}^{(k+t)} &= \sum_{s \in S} \mathbb{P}[\, X_{k+t} = j \mid X_k = s,\, X_0 = i \,]\mathbb{P}[\, X_k = s \mid X_0 = i \,] \\ &= \sum_{s \in S} p_{s,j}^{(t)} p_{i,s}^{(k)} \end{aligned}$$

portanto, vale a equação (5.16). □

PROPRIEDADE FORTE DE MARKOV A propriedade forte de Markov assegura independência a partir do tempo (aleatório) que leva para a cadeia atingir um determinado estado. Um exemplo muito simples é quando jogamos uma moeda até obtermos n caras, $[\mathsf{T} = n]$ é completamente determinado pelos valores da sequência dos lançamentos anteriores e T é aleatório.

Uma variável aleatória T que assume valores em $\{0, 1, 2, \ldots\} \cup \{\infty\}$ é um **tempo de parada** se o evento $[\mathsf{T} = n]$, para $n = 0, 1, 2, \ldots$, depende somente de $\mathsf{X}_0, \mathsf{X}_1, \ldots, \mathsf{X}_n$.

Sejam $\{\mathsf{X}_t : t \geqslant 0\}$ *uma cadeia de Markov com matriz de transição* P*, distribuição inicial* ρ *e* T *tempo de parada. Então, condicionado ao evento* $[\mathsf{T} < \infty]$ *e* $[\mathsf{X}_\mathsf{T} = i]$ *o processo* $\{\mathsf{X}_{t+\mathsf{T}} : t \geqslant 0\}$ *é uma cadeia de Markov com transições* P*, distribuição inicial* δ_i *e independente de* $\mathsf{X}_0, \ldots, \mathsf{X}_\mathsf{T}$

$$\mathbb{P}[\, \mathsf{X}_{\mathsf{T}+m} = j \mid \mathsf{X}_k = x_k \ (0 \leqslant k < \mathsf{T}), \mathsf{X}_\mathsf{T} = i \,] = \mathbb{P}[\, \mathsf{X}_{\mathsf{T}+m} = j \mid \mathsf{X}_\mathsf{T} = i \,].$$

Seja B um evento determinado por $\mathsf{X}_0, \mathsf{X}_1, \ldots, \mathsf{X}_\mathsf{T}$. Então $\mathsf{B} \cap [\mathsf{T} = m]$ é determinado por $\mathsf{X}_0, \mathsf{X}_1, \ldots, \mathsf{X}_m$ e pela propriedade de Markov no tempo m

$$\begin{aligned}\mathbb{P}([\mathsf{X}_\mathsf{T} = x_0, \mathsf{X}_{\mathsf{T}+1} = x_1, \ldots, \mathsf{X}_{\mathsf{T}+n} = x_n] \cap \mathsf{B} \cap [\mathsf{T} = m] \cap [\mathsf{X}_\mathsf{T} = i]) = \\ \mathbb{P}[\, \mathsf{X}_0 = x_0, \mathsf{X}_1 = x_1, \ldots, \mathsf{X}_n = x_n \mid \mathsf{X}_0 = i \,] \mathbb{P}(\mathsf{B} \cap [\mathsf{T} = m] \cap [\mathsf{X}_\mathsf{T} = i])\end{aligned}$$

somando sobre $m = 0, 1, \ldots$ e dividindo por $\mathbb{P}[\, \mathsf{T} < \infty, \mathsf{X}_\mathsf{T} = i \,]$

5.3.2 IRREDUTIBILIDADE, PERIODICIDADE E RECORRÊNCIA

Seja P a matriz de transição de uma cadeia de Markov. A condição $p_{i,j}^{(t)} > 0$ para algum t significa que a partir do estado i a cadeia atinge o estado j em t passos com probabilidade positiva; nesse caso dizemos que j é um estado **acessível** a partir de i e escrevemos $i \to j$. Quando j não é acessível a partir de i a probabilidade da cadeia chegar em j saindo de i é nula

$$\mathbb{P}\left[\bigcup_{n \geqslant 0} [\mathsf{X}_n = j] \;\middle|\; \mathsf{X}_0 = i \right] \leqslant \sum_{n \geqslant 0} \mathbb{P}[\, \mathsf{X}_n = j \mid \mathsf{X}_0 = i \,] = \sum_{n \geqslant 0} p_{i,j}^{(n)} = 0.$$

Escrevemos $i \leftrightarrow j$ se $i \to j$ e $j \to i$ e dizemos que os estados i e j se **comunicam**. Deixamos para o leitor a verificação de que $\leftrightarrow$ define uma relação de equivalência sobre S, portanto, particiona S em classes de equivalência, que chamaremos **classes de comunicação**.

RECORRÊNCIA Os estados de uma cadeia de Markov são classificados em dois tipos fundamentais os *recorrentes* e os *transitórios*. Um estado i de uma cadeia de Markov *finita* é dito *recorrente* se para todo estado j, $i \to j$ implica $j \to i$. Se um estado é recorrente, uma vez tendo estado nele, é certo que a cadeia eventualmente volta a ele e, assim, visita-o infinitas vezes. Um estado que não é recorrente é dito **transitório**. Notemos que essa definição leva a seguinte conclusão imediata: *se* S *é finito então*

pelo menos um estado é recorrente. De fato, consideremos uma sequência de estados $s_0, s_1, \ldots$ definida recursivamente por: tome $s_0 \in S$ qualquer; para $i \geqslant 1$ se s_{i-1} é recorrente então pare senão escolha s_i tal que $s_{i-1} \to s_i$ mas $s_i \not\to s_{i-1}$. Se existirem $i < j$ com $s_i = s_j$ então $s_i \to s_{j-1} \to s_i$, o que é uma contradição. Como S é finito a sequência deve terminar num estado recorrente.

O **tempo da primeira visita** ao estado i é a variável aleatória

$$T_i := \min\{n \geqslant 1 : X_n = i\}$$

com a convenção de que $\min \varnothing = \infty$. Para todo $n \in \mathbb{N}$ definimos

$$f_{i,j}^{(n)} := \mathbb{P}[\, T_j = n \mid X_0 = i \,]$$

que é a probabilidade do evento "primeira visita ao estado j a partir do estado i em n passos", ou seja, a probabilidade do evento $[X_1 \neq j,\ X_2 \neq j,\ \cdots,\ X_{n-1} \neq j,\ X_n = j]$ condicionado a $[X_0 = i]$. Assim $f_{i,i}^{(n)}$ é a **probabilidade do primeiro retorno** a i em n passos e

$$f_{i,i} := \sum_{n>0} f_{i,i}^{(n)} = \mathbb{P}[\, T_i < \infty \mid X_0 = i \,].$$

Com essa definição, podemos estender ao caso enumerável infinito a definição de estado **recorrente**, é todo estado i para o qual $f_{i,i} = 1$, caso contrário i é **transitório**.

Notemos que quando $f_{i,i} = 1$ nós temos uma distribuição de probabilidade $\{f_{i,i}^{(n)} : n > 0\}$ para o tempo de retorno ao estado i cuja média está definida e é chamada de **tempo médio de retorno** para o estado i

$$\mu_i := \sum_{n>0} n f_{i,i}^{(n)} = \mathbb{E}[\, T_i \mid X_0 = i \,].$$

No caso de i transitório convencionamos $\mu_i := \infty$ pois $\mathbb{P}[\, T_i = \infty \mid X_0 = i \,] = 1 - f_{i,i} > 0$.

Dado que o retorno ao estado inicial i é certo, é natural perguntar se o tempo médio de retorno é finito. No caso $\mu_i = \infty$ o estado i é dito **recorrente nulo**, no caso de média finita, $\mu_i < \infty$, o estado i é dito **recorrente positivo**. No caso de $|S|$ finito, todo estado recorrente é positivo.

No caso da cadeia com $S = \{1,2,3,4,5,6\}$ com as probabilidades das transições não nulas descritas no diagrama da figura 5.19, os estados $1,2,5,6$ são recorrentes enquanto que $3,4$ são transitórios. Ainda, a probabilidade de retorno ao estado 1 é

$$f_{1,1}^{(1)} = \frac{1}{2} \quad \text{e} \quad f_{1,1}^{(n)} = \frac{1}{2}\left(\frac{3}{4}\right)^{n-2}\frac{1}{4} \quad \text{para todo } n \geqslant 2$$

portanto, o tempo médio de retorno para o estado 1 é (usando s.6c)

$$\mu_1 = \frac{1}{2} + \frac{1}{8}\sum_{n\geqslant 2} n\left(\frac{3}{4}\right)^{n-2} = \frac{1}{2} + \frac{2}{9}\sum_{n\geqslant 2} n\left(\frac{3}{4}\right)^{n} = \frac{1}{2} + \frac{5}{2} = 3$$

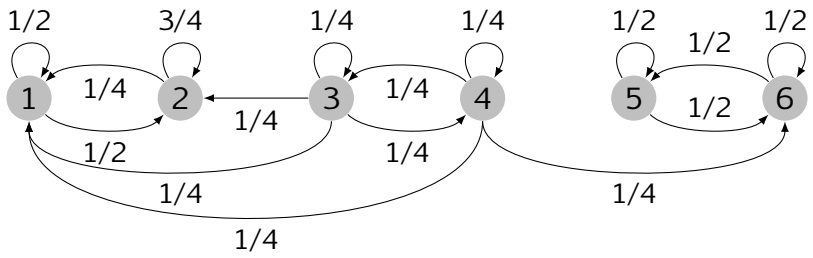


Figura 5.19: diagrama de uma cadeia de Markov.

No exemplo 5.34, página 289, a probabilidade de não retornar ao estado 1 nos $n-1$ primeiros passos é

$$\prod_{j=1}^{n-1} \frac{j}{j+1} = \frac{1}{n}$$

portanto a probabilidade de nunca voltar ao estado 1 é zero e a probabilidade de voltar ao 1 no n-ésimo passo é

$$f_{1,1}^{(n)} = \frac{1}{n}\frac{1}{n+1}$$

e, portanto, o tempo médio de retorno do estado 1 é

$$\mu_1 = \sum_{n>0} n f_{1,1}^{(n)} = \sum_{n>0} \frac{1}{n+1} = \infty$$

de modo que é um estado recorrente nulo.

O número de visitas ao estado i é a variável aleatória

$$\mathsf{V}_i := \sum_{n\geqslant 1} \mathbb{1}_{[\mathsf{X}_n = i]}$$

em que $\mathbb{1}_{[\mathsf{X}_n=i]}$ é a variável aleatória indicadora do evento $[\mathsf{X}_n = i]$, que é finito se e só se n é transiente. De fato, pela proposição 3.22, página 135,

$$\mathbb{E}[\,\mathsf{V}_i \mid \mathsf{X}_0 = i\,] = \sum_{k\geqslant 1} \mathbb{P}[\,\mathsf{V}_i \geqslant k \mid \mathsf{X}_0 = i\,]$$

e, saindo de i, visitar j pelo menos k vezes equivale a visitar j e em seguida visitar j pelo menos $k-1$ vezes, o que ocorre com probabilidade $f_{i,j}(f_{j,j})^{k-1}$ logo, fazendo $j = i$ nessa dedução ficamos com

$$\mathbb{E}[\,\mathsf{V}_i \mid \mathsf{X}_0 = i\,] = \sum_{k\geqslant 1} (f_{i,i})^k = \begin{cases} \frac{f_{i,i}}{1-f_{i,i}} & \text{se } f_{i,i} < 1, \text{ ou seja, se } i \text{ é transiente} \\ \infty & \text{se } f_{i,i} = 1, \text{ ou seja, se } i \text{ é recorrente.} \end{cases}$$

Ainda, pela linearidade da esperança,

$$\mathbb{E}[\,\mathsf{V}_i \mid \mathsf{X}_0 = i\,] = \sum_{n\geqslant 0} \mathbb{E}\Big[\,\mathbb{1}_{[\mathsf{X}_n=i]} \,\Big|\, \mathsf{X}_0 = i\Big] = \sum_{n\geqslant 0} \mathbb{P}[\,\mathsf{X}_n = i \mid \mathsf{X}_0 = i\,] = \sum_{n\geqslant 0} p_{i,i}^{(n)}$$

e a classificação de estados pode ser caracterizada pelas probabilidades de transição da maneira descrita a seguir.

Lema 5.39 *O estado i é*

- *transiente se e só se* $\sum_{n\geqslant 0} p_{i,i}^{(n)} < \infty$.
- *recorrente se e só se* $\sum_{n\geqslant 0} p_{i,i}^{(n)} = \infty$.

□

Exemplo 5.40 (*Passeio aleatório nos inteiros*). Tomemos o conjunto de estados como os inteiros, $S = \mathbb{Z}$, o estado inicial $X_0 = 0$ com probabilidade 1 e as transições de estados têm as probabilidades $p_{i,i-1} = 1 - p$ e $p_{i,i+1} = p$ para todo inteiro i e algum $p \in (0,1)$. Então

$$p_{0,0}^{(2n)} = \binom{2n}{n} p^n q^n$$

para $q := 1 - p$. Usando a aproximação de Striling (d.3) (verifique), para uma constante positiva a

$$p_{0,0}^{(2n)} \to \frac{(4pq)^n}{a\sqrt{n/2}}, \quad \text{quando } n \to \infty.$$

Se $p = 1/2$, então $p_{0,0}^{(2n)} \geqslant (2a\sqrt{n})^{-1}$ para todo n suficientemente grande, logo

$$\sum_{n\geqslant n_0} p_{0,0}^{(2n)} \geqslant \frac{1}{2a} \sum_{n\geqslant n_0} \frac{1}{\sqrt{n}} = \infty.$$

Se $p \neq 1/2$, então $4pq = \varepsilon < 1$ e de modo análogo

$$\sum_{n\geqslant n_0} p_{0,0}^{(2n)} \leqslant \frac{1}{a} \sum_{n\geqslant n_0} \varepsilon^n < \infty.$$

Assim, no caso $p = 1/2$ é o caso de cadeia de Markov com todos os seus estados transitórios. Se $p = 1/2$ então todos os estados são recorrentes positivos. ◊

Os estados 0 e n do exemplo *ruína do jogador*, na página 244, quando são atingidos a cadeia não muda mais de estado; nesses casos chamamos tais estados de **absorvente**, que é um estado recorrente positivo. Por exemplo, na cadeia representada pelo esquema da figura 5.20 abaixo

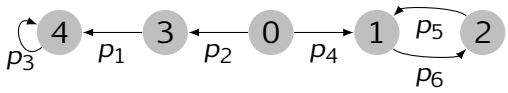


Figura 5.20: uma cadeia com estados transitórios e recorrentes.

o estado 4 é absorvente, os estados 0 e 3 são transiente e os estados 1 e 2 são recorrentes.

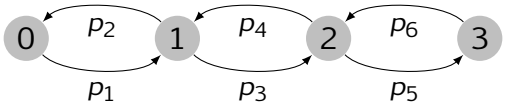


Figura 5.21: esquema de um exemplo de cadeia de Markov com estados periódicos.

PERIODICIDADE No caso de uma cadeia de Markov que pode ser representada como na figura 5.21, se $X_0 = 0$ então $p_{0,0}^{(n)} > 0$ só pode ocorrer na cadeia em instantes n da forma $n = 2k$, $n = 4k$, $n = 6k$. Para o estado 1, as probabilidades de retorno positivas só ocorrem para $n = 2k$, $n = 4k$. Para o estado 2 ocorre o mesmo fenômeno do estado 1 e o caso do estado 3 é semelhante ao caso do estado 0.

O **período** de um estado i numa cadeia de Markov é

$$\tau(i) := \operatorname{mdc}\left\{n\colon p_{i,i}^{(n)} > 0\right\}.$$

Caso $\tau(i) > 1$, dizemos que i é **periódico** e nesse caso $p_{i,i}^{(n)} = 0$ a menos que n seja múltiplo de $\tau(i)$. Quando $\tau(i) = 1$ nós dizemos que i é **aperiódico**. No exemplo acima todos os estados são periódicos de período 2.

Lema 5.41 *Recorrência e transiência são propriedades de classe de comunicação, isto é, dois estados que se comunicam tem a mesma classificação. Ademais, estados que se comunicam têm o mesmo período.*

Demonstração. Suponha que o estado i é transitório, ou seja, para algum j, $i \to j$ porém $j \not\to i$ e suponha que i e m são estados que se comunicam. Então $m \to i$ e $i \to j$, portanto, então $m \to j$. Agora, se $j \to m$, então $j \to i$, uma contradição. Logo m é transitório.

Sejam i e j estados que se comunicam e sejam n e m inteiros positivos tais que $p_{i,j}^{(n)} > 0$ e $p_{j,i}^{(m)} > 0$. Para todo t

$$p_{i,i}^{(t+n+m)} \geqslant p_{i,j}^{(n)} p_{j,j}^{(t)} p_{j,i}^{(m)} \tag{5.17}$$

portanto, $p_{i,i}^{(t+n+m)} > 0$ sempre que $p_{j,j}^{(t)} > 0$. Se fizermos $t = 0$ na equação (5.17), resulta que $p_{i,i}^{(n+m)} > 0$ portanto $\tau(i)$ divide $n+m$. O lado esquerdo da equação (5.17) é nulo a menos nos períodos múltiplos de $\tau(i)$, portanto $p_{j,j}^{(t)} > 0$ só quando t é múltiplo de $\tau(i)$, portanto $\tau(j)$ divide $\tau(i)$. Trocando os papéis de i e j concluiremos que $\tau(i)$ divide $\tau(j)$. Portanto, os períodos são iguais. □

Observação 5.42. Suponha que numa cadeia de Markov todos os estados tenham o mesmo período $\tau > 1$ e que para cada par de estados (i, j) existe um n para o qual $p_{i,j}^{(n)} > 0$. Para todo estado k da cadeia devem existir instantes m e n tais que

$$p_{0,k}^{(n)},\ p_{k,0}^{(m)} > 0$$

e como de Chapman–Kolmogorov, equação (5.16), deduzimos $p_{0,0}^{(n+m)} \geqslant p_{0,k}^{(n)} p_{k,0}^{(m)} > 0$, devemos ter que τ divide $n+m$. Fixando m, concluímos que todo n para o qual vale $p_{0,k}^{(n)} > 0$ é da forma $n = a + v\tau$

para $0 \leqslant a < \tau$ inteiro. Assim, podemos particionar o conjunto de estados S em $S_0, S_1, \ldots, S_{\tau-1}$ (no exemplo acima $S_0 = \{0,2\}$ e $S_1 = \{1,3\}$) de acordo com os valores de a acima de modo que *se* $k \in S_a$ *então* $p_{0,k}^{(n)} = 0$ *a menos que* $n = a + v\tau$. Agora consideramos os estados na ordem $S_0, \ldots, S_{\tau-1}, S_0$ ciclicamente de modo que cada passo da cadeia sai de um estado em S_i e chega num estado em S_{i+1} (a soma no índice é módulo τ) e a cada τ passos a cadeia está de volta à mesma parte.

IRREDUTIBILIDADE Um conjunto de estados $C \subset S$ é **irredutível** se $i \leftrightarrow j$ para todos $i, j \in C$. Quando um conjunto não é irredutível, dizemos *redutível*. Quando há uma única classe de comunicação dizemos que a cadeia é uma **cadeia de Markov irredutível**. Num conjunto irredutível de estados todos os estados são do mesmo tipo, logo podemos classificar um conjunto de estados, ou mesmo uma cadeia de Markov irredutível, como: *aperiódica* se todos os seus estados o forem; *periódica* se todos os seus estados o forem; *transitório* se todos os seus estados o forem e *recorrente* se todos os seus estados o forem. Por exemplo, uma cadeia de Markov com estados $\{1,2\}$ e matriz de transições

$$P = \begin{pmatrix} 1 & 0 \\ 1/2 & 1/2 \end{pmatrix}$$

não corresponde a uma cadeia irredutível pois para todo inteiro $k > 0$

$$P^k = \begin{pmatrix} 1 & 0 \\ \frac{2^k-1}{2^k} & \frac{1}{2^k} \end{pmatrix}$$

portanto $1 \not\to 2$. Os estados 0 e n do exemplo *ruína do jogador*, exemplo na página 244, quando são atingidos a cadeia não muda mais de estado; nesses casos chamamos tais estados de **absorvente**. A cadeia desse exemplo (ruína do jogador) não é irredutível por causa dos estados absorventes. Toda cadeia com pelo menos dois estados e com pelo menos um deles absorvente é uma cadeia redutível, por exemplo, o caso de uma cadeia de Markov representada pelo diagrama da figura 5.22, a cadeia

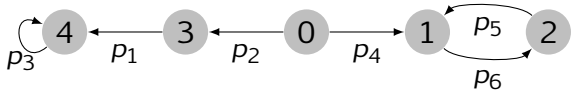


Figura 5.22: exemplo de cadeia de Markov redutível.

é redutível, a classe $\{1,2\}$ é periódica de período 2 e o estado 4 é absorvente. No exemplo da figura 5.23, a cadeia é aperiódica e irredutível.

No exemplo da figura 5.24, temos uma cadeia irredutível, recorrente e periódica.

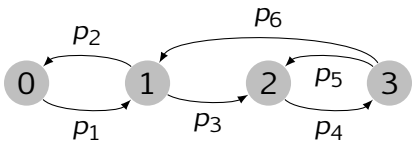


Figura 5.23: exemplo de cadeia de Markov irredutível, recorrente e aperiódica.

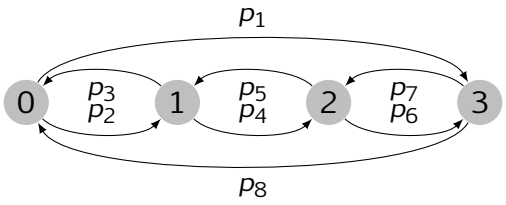


Figura 5.24: exemplo de cadeia de Markov irredutível, recorrente e periódica.

Exemplo 5.43. Considere uma cadeia de Markov com $S = \{1,2,3,4\}$ e matriz de transição

$$P = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/4 & 3/4 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

cujas classes de comunicação são $\{1,2\}$ que é recorrente, $\{4\}$ que é absorvente e $\{3\}$ que é transitório.

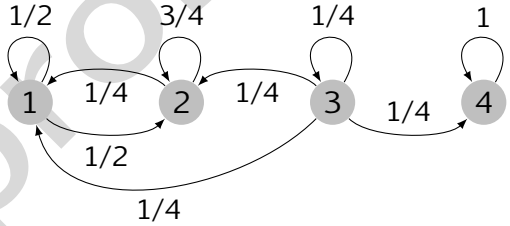


Figura 5.25: probabilidade de transição de classe transitório para recorrente.

Se $X_0 = 3$, então a probabilidade p com que a cadeia entra na classe recorrente $\{1,2\}$ é, condicionando em X_1, dada por

$$p = \frac{1}{4}1 + \frac{1}{4}1 + \frac{1}{4}p + \frac{1}{4}0 = \frac{1}{2} + \frac{1}{4}p$$

logo $p = 2/3$ e com probabilidade $1/3$ a cadeia é absorvida em 4.

No caso da cadeia de Markov com $S = \{1,2,3,4,5,6\}$ e matriz de transição

$$Q = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 2/3 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1/3 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 0 & 1/3 & 1/6 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

as classes de comunicação são $\{1,2\}$, $\{3,4\}$ e $\{5,6\}$, respectivamente, recorrente aperiódica, transitório e recorrente periódica. Se p_i é a probabilidade de entrar na classe recorrente aperiódica descrita acima a partir do estado i, para $i = 3,4$, então, condicionando em X_1,

$$p_3 = \frac{1}{3}1 + 0 \cdot 1 + 0p_3 + \frac{1}{3}p_4 + \frac{1}{6}0 + \frac{1}{6}0$$
$$p_4 = \frac{1}{6}1 + \frac{1}{6}1 + \frac{1}{6}p_3 + 0p_4 + \frac{1}{3}0 + \frac{1}{6}0$$

cuja solução é $p_3 = 8/17$ e $p_4 = 7/17$. ◊

Observação 5.44. Notemos que a Observação 5.42 se aplica a uma classe de comunicação, ou a uma cadeia periódica, pois todos os estados dela têm o mesmo período.

```
A state i is absorbing if i is a closed class.
```

Uma classe de comunicação é dita **fechada** se $j \in C$ sempre que $i \to j$ e $i \in C$. Toda classe de comunicação recorrente é fechada. Observemos que se X_0 está em alguma classe recorrente C_ℓ, então a evolução da cadeia se restringe a essa classe pois assumindo, por hipótese, que $i \in C_\ell$ e $j \notin C_\ell$ são tais que $i \to j$, temos a inclusão de eventos $[X_t = j] \subset \bigcup_{n \geqslant 1}[X_n \neq i]$ donde deduzimos que para algum $t \in \mathbb{N}$

$$\mathbb{P}[\bigcup_{n \geqslant 1}[X_n \neq i] \mid X_0 = i\,] \geqslant p_{i,j}^{(t)} > 0$$

contrariando o fato de i ser recorrente. Se, por outro lado, X_0 é um estado transitório então ou a cadeia fica no conjunto dos estados transitórios para sempre (caso a cadeia não seja finita) ou se move para alguma classe recorrente e não sai mais dessa classe.

Também, notemos que uma cadeia de Markov finita e irredutível é recorrente. Ademais, como veremos, se for aperiódica então existe um natural n_0 tal que para todo $n \geqslant n_0$ a potência P^n da matriz de transição P é (estritamente) positiva.

Lema 5.45 *Se uma cadeia de Markov* finita *é irredutível e aperiódica então existe* $K_0 \in \mathbb{N}$ *tal que para todos* $i, j \in S$, *se* $k \geqslant K_0$ *então* $p_{i,j}^{(k)} > 0$.

Demonstração. Da cadeia ser irredutível temos que para cada $i, j \in S$ existe $k_0(i,j)$ tal que $p_{i,j}^{(k_0(i,j))} > 0$. Da cadeia ser aperiódica temos que para cada $i \in S$ existe $k_1(i)$ tal que $p_{i,i}^{(k_1)} > 0$ para todo $k \geqslant k_1(i)$. Com essas constantes, se $t \geqslant 0$ então $p_{i,i}^{(k_1(i)+t)} > 0$ e $p_{i,j}^{(k_0(i,j))} > 0$ portanto, por Chapman–Kolmogorov, equação (5.16) na página 290, para todo $t \in \mathbb{N}$ temos

$$p_{i,j}^{(k_0(i,j)+k_1(i)+t)} = \sum_{\ell \in S} p_{i,\ell}^{((k_1(i)+t))} p_{\ell,j}^{(k_0(i,j))} \geqslant p_{i,i}^{((k_1(i)+t))} p_{i,j}^{(k_0(i,j))} > 0.$$

Assim, basta tomarmos $K_0 := \max\{k_0(i,j) + k_1(i) \colon i, j \in S\}$. □

*Exercício 5.46 (*Passeio aleatório em $\mathbb{Z}$*).* Consideremos o caso do passeio aleatório do exemplo 5.40, página 294, onde $p_{i,i+1} = p = 1 - p_{i,i-1}$. Claramente, a cadeia é irredutível (justifique) logo todos os estados são ou recorrentes ou transitórios.

(i) Prove que $p_{0,0}^{(2n-1)} = 0$, para $n \geqslant 1$. Prove que $p_{0,0}^{(2n)} = \binom{2n}{n} p^n (1-p)^n$.

(ii) Use a aproximação de Stirling $n! \sim n^{n+1/2} e^{-n} \sqrt{2\pi}$ e conclua que

$$p_{0,0}^{(2n)} \sim \frac{(4p(1-p))^n}{\sqrt{\pi n}}.$$

Verifique que $4p(1-p) \leqslant 1$, e vale a igualdade se e somente se $p = 1/2$.

(iii) Conclua que a cadeia é recorrente caso $p = 1/2$ e, se $p \neq 1/2$, a cadeia é transitório.

5.3.3 DISTRIBUIÇÃO INVARIANTE E CONVERGÊNCIA AO EQUILÍBRIO

CADEIAS FINITAS: Nesta seção provaremos o importante resultado que estabelece que toda cadeia de Markov finita converge para a distribuição estacionária. Observamos que ele também vale no caso enumerável infinito com a hipótese adicional de que $\mu_i < +\infty$ para todo estado i.

No caso finito *toda matriz estocástica tem distribuição invariante* e esse fato é uma consequência simples do conhecido Teorema do Ponto Fixo de Brower: *Para toda função contínua* $f \colon T \to T$, *onde* $T \subset \mathbb{R}^n$ *é compacto e convexo, existe* $x \in T$ *tal que* $f(x) = x$. Consideremos o conjunto (compacto e convexo)

$$T = \left\{ (v_1, \ldots, v_n) \in \mathbb{R}^n \colon \forall\, i,\ v_i \geqslant 0 \text{ e } \sum_{i=1}^{n} |v_i| = 1 \right\}$$

e a função linear de T em T dada por $x \mapsto xP$ em que P é uma matriz estocástica. Por ser linear a função é contínua, portanto, pelo Teorema da Ponto Fixo de Brower deduzimos a existência de um vetor invariante. Ainda, é possível provar mostrar de modo elementar que $\lim_{n\to\infty} P^n$ existe e as linhas são dadas pelo vetor invariante.

Teorema 5.47 *Uma cadeia de Markov irredutível tem uma única distribuição invariante* $\pi = (\pi_j \colon j \in S)$ *positiva dada por*

$$\pi_j = \frac{1}{\mu_j}$$

para todo estado j. Ainda, se a cadeia for aperiódica então

$$\pi_j = \lim_{n\to\infty} p_{i,j}^{(n)}. \tag{5.18}$$

A prova desse teorema será dada mais a frente. Antes, faremos algumas considerações.

Observação 5.48 (*existência da distribuição invariante*). Numa cadeia redutível a distribuição invariante pode não ser única. Por exemplo, uma cadeia redutível com matriz de transição

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

tem distribuição invariante $\pi = \begin{pmatrix} p & 1-p \end{pmatrix}$ para qualquer $p \in (0,1)$ e, por exemplo, para $p = 1/2$ temos que $p_{1,1}^{(n)} \not\to 1/2$ quando $n \to \infty$. O mesmo acontece com a matriz de transição

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

e o vetor $\pi = \begin{pmatrix} 1/2 & 1/2 \end{pmatrix}$, não há convergência ao equilíbrio porque a cadeia é periódica. Numa cadeia de Markov com quatro estados e matriz de transições

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

$(1,0,1,0)$ e $(0,1,0,1)$ são vetores invariantes.

Observação 5.49 (*convergência em cadeias periódicas*). No caso da cadeia irredutível com matriz de transição P ser periódica de período τ temos uma partição $S_0, S_1, \ldots, S_{\tau-1}$ de S, como foi feito na observação 5.42, página 295, cada parte fechada e irredutível com respeito a matriz de transição P^τ. Lembremos que se $k \in S_a$ então $p_{0,k}^{(n)} = 0$ a menos que $n \equiv a \pmod{\tau}$, para todo $a \in \{0,1,\ldots,\tau-1\}$. A partir desse fato temos que para um estado $k \in S$ recorrente vale que o tempo de recorrência médio com respeito a P^τ é μ_k/τ, e (com respeito a essa matriz) cada parte é irredutível e recorrente, assim para cada $j \in S_a$

$$\lim_{n\to\infty} p_{j,k}^{(n\tau)} = \frac{\tau}{\mu_k}$$

(e tal limite é zero quando $k \notin S_a$ ou k é transitório). Ademais, $(1/\mu_k : k \in S)$ é uma distribuição invariante (veja Feller, 1968, seção XV.9).

DEMONSTRAÇÃO DO TEOREMA **??**:

REVERSIBILIDADE: seja $\{X_i\}_{i\geqslant 0}$ uma cadeia de Markov irredutível com respeito a matriz estocástica P e a distribuição inicial invariante π. Tomemos a matriz $Q = q_{i,j}$ dada por

$$q_{i,j} = \frac{\pi_j}{\pi_i} p_{j,i}.$$

Essa matriz é estocástica pois, pela invariância de π

$$\sum_{j\in S} q_{i,j} = \frac{1}{\pi_i} \sum_{j\in S} \pi_j p_{j,i} = 1.$$

Ainda,

$$\sum_{j\in S} \pi_i q_{i,j} = \sum_{j\in S} \pi_j p_{j,i} = \pi_i$$

ou seja, π é invariante com relação a Q.

Se tomarmos $Y_n = X_{N-n}$ para $0 \leqslant n \leqslant N$ então

$$\mathbb{P}[Y_0 = i_0, Y_1 = i_1, \ldots, Y_N = i_N] = \mathbb{P}[X_0 = i_N, X_1 = i_{N-1}, \ldots, X_N = i_0] = \pi_{i_N} p_{i_N, i_{N-1}} \cdots p_{i_1, i_0} = \pi_{i_0} q_{i_0, i_1} \cdots q_{i_{N-1}, i_N}$$

logo $\{Y_n\}_{0\leqslant n\leqslant N}$ é uma cadeia de Markov com relação a Q e π. Ademais,

$$q_{i_N, i_{N-1}} \cdots q_{i_1, i_0} = \frac{1}{\pi_{i_0}} \pi_{i_N} p_{i_0, i_1} \cdots p_{i_{N-1}, i_N} > 0$$

portanto, a cadeia é irredutível, chamada *tempo-reverso* da cadeia $\{X_i\}_{0\leqslant i\leqslant N}$.

O seguinte exercício fornece um meio, o mais simples, de verificar se uma dada distribuição é invariante.

Exercício 5.50. Mostre que se P é uma matriz estocástica e λ um vetor não-negativo tal que para todos i, j

$$\lambda_i p_{i,j} = \lambda_j p_{j,i}$$

então $\lambda P = \lambda$.

5.4 ALGORITMO METROPOLIS

5.5 CADEIAS DE MARKOV E CONTAGEM

5.6 EXERCÍCIOS

Exercício 5.51. Considere o jogo da ruína com moeda justa. Vimos que se o jogador começa com t reais e almeja ganhar mais n reais então é esperado que aposte por $E_t = tn$ rodadas. Assim, quando

$n \to \infty$ temos $E_t \to \infty$, ou seja, o jogador que se propor a jogar até quebrar vai jogar pra sempre, em média. Prove que se o jogador se propor a jogar até quebrar, então ele quebra com probabilidade 1, para qualquer quantia inicial.

Exercício 5.52. Neste exercício analisamos uma versão do algoritmo para o 2SAT da seção 5.1.3 para o problema 3SAT. Dadas cláusulas $C_1, C_2, \ldots, C_m$ de uma fórmula booleana Φ na forma 3-CNF sobre as variáveis $x_1, x_2, \ldots, x_n$, o algoritmo responde *sim* se Φ é satisfazível (e, implicitamente, uma valoração que satisfaz Φ) ou responde *não* se a fórmula Φ não é satisfazível ou não achou uma valoração.

```
1  repita
2      c ← 0
3      Atribua aleatoriamente x_i ← {0,1} para todo i
4      enquanto φ é falsa e c ≤ 3n faça
5          Escolha j tal que C_j é falsa
6          Escolha ao acaso um literal ℓ de C_j e inverta o valor lógico da variável de ℓ
7          c ← c + 1
8  até completar m rodadas ou Φ ser verdadeira
9  se φ é verdadeira então responda sim
10 senão responda não.
```

Suponha Φ satisfazível e v^* uma valoração que a satisfaça. Denote por j a quantidade de variáveis que o algoritmo *erra* no passo 3, ou seja, j variáveis não coincidem os valores atribuídos a elas pelo algoritmo e por v^*. Denote por q_j a probabilidade do algoritmo encontrar v^* nas $\leqslant 3n$ rodadas do laço interno.

Prove que

$$q_j \geqslant \binom{3j}{j}\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j} \geq \frac{c}{2^j\sqrt{j}}$$

para alguma constante $c > 0$ (use a aproximação de Stirling para o fatorial para obter a segunda desigualdade).

Agora, prove que a probabilidade com que, para algum j e a partir da valoração inicial aleatória, o algoritmo encontre v^* é

$$q \geqslant \frac{1}{2^n} + \sum_{j=1}^{n}\binom{n}{j}\left(\frac{1}{2}\right)^n \frac{c}{2^j\sqrt{j}} \geqslant \frac{c}{\sqrt{n}}\left(\frac{3}{4}\right)^n.$$

Conclua que o número esperado de rodadas (independentes) é $\leqslant c'\sqrt{n}(4/3)^n$, com $c' \approx 1{,}1866$, e qua para atingir uma probabilidade de erro de, digamos e^{-10}, basta tomar $m \approx 12\sqrt{n}(4/3)^n$. Qual o tempo esperado do algoritmo nesse caso?

Exercício 5.53. Considere o passeio aleatório em $\mathbb{Z}^d$ com cada uma das 2^d direções equiprováveis e denote por $p^{(t)}(0,0)$ a probabilidade de sair da origem, genericamente denotado por 0, e voltar a ela em t passos. Prove que para $d = 1$

$$p^{(2n)}(0,0) = \frac{1}{2^{2n}}\binom{2n}{n} = (1 + o(1))C_1 n^{-1/2}.$$

para alguma constante $C_1 > 0$. Para o caso $d = 2$ considere dois passeios aleatórios unidimensionais independentes. A trajetória conjunta desse passeio visita, em $\mathbb{Z}^2$, apenas o conjunto de pontos $(i,j) \in \mathbb{Z}^2$ tais que $i + j$ é par. Verifique que o grafo com esse conjunto de vértices e com dois vértices adjacentes se e só se diferirem em $+1$ ou -1 em cada coordenada é isomorfo ao $\mathbb{Z}^2$ e que probabilidades são preservadas sob tal isomorfismo. Use esse fato para provar que

$$p^{(2n)}(0,0) = \left(\frac{1}{2^{2n}}\binom{2n}{n}\right)^2 = (1 + o(1))C_2 n^{-1}.$$

para alguma constante $C_2 > 0$. Para o caso $d = 3$, digamos que os passos podem ir nos sentidos frente/trás, esquerda/direita e cima/baixo. Em m passeio com $2n$ passos que começa e termina na origem, n desses passos vão para frente, esquerda e cima; há $\binom{2n}{n}$ modos distintos de cumprir essa regra. Os outros n passos devem ir nos sentidos trás, direita e baixo. Para cada uma dessas escolhas, i passos vão para frente e i para trás, j passos vão para esquerda e j para direita, $n - i - j$ passos vão para cima e $n - i - j$ para baixo. Prove que

$$p^{(2n)}(0,0) = \frac{1}{6^{2n}}\binom{2n}{n}\sum_{i+j\leqslant n}\left(\frac{n!}{i!j!(n-i-j)!}\right)^2;$$

verifique que $i!j!k!$ sujeito a $i + j + k = n$ tem valor mínimo quando $i = j = k = n/3$ e prove que $p^{(2n)}(0,0) = O(n^{-3/2})$.

Use os resultados acima para provar que o número esperado de visitas à origem é infinito para $d = 1, 2$ e finito para $d = 3$.

Exercício 5.54. Neste exercício melhoramos a estimativa de $O(2^n)$ rodadas para o problema do 3-SAT (exercício 5.5, página 251). Considere a seguinte estratégia variante da generalização do algoritmo 42. Dada uma fórmula booleana 3-CNF com n variáveis

Seja ν uma valoração que satisfaz a fórmula.

1. Suponha que numa determinada rodada haja j variáveis booleanas cujos valores diferem de ν. Denote por q_j a probabilidade com que em no máximo $3n$ rodadas a estratégia encontre ν. Prove que

$$q_j \geqslant \binom{3j}{j}\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j}.$$

1. Repita t vezes ou até que a fórmula fique satisfeita:
   (a) tome uma valoração aleatória das variáveis;
   (b) Repita $2n$ vezes ou até que a fórmula fique satisfeita:
      i. escolha uma cláusula não satisfeita;
      ii. escolha aleatoriamente um literal dessa cláusula e inverta o seu valor;
2. se achou uma valoração que satisfaça a fórmula escreva-a, senão escreva "não satisfatível".

2. Use a fórmula de Stirling para provar que para todo $j \geqslant 1$

$$q_j \geqslant \frac{c}{2^j\sqrt{j}}$$

   com $c = (1/8)\sqrt{3/\pi}$. Também, $q_0 = 1$.

3. A probabilidade q com que uma atribuição inicial aleatória leva a v (ou outra atribuição satisfatória) em $3n$ iterações é pelo menos

$$\frac{c}{\sqrt{n}}\left(\frac{3}{4}\right)^n$$

Conclua que o número de valorações iniciais (1(a)) até descobrir uma que satisfaz a fórmula (que assumimos satisfazível) tem distribuição geométrica com parâmetro q, portanto, na média precisamos tentar $1/q$ valorações iniciais e para cada uma iteramos no máximo $3n$ rodadas, portanto, na média realizamos $O(n^{3/2}(4/3)^n)$ passos com essa estratégia.

Exercício 5.55. Mostre que em um passeio aleatório num grafo conexo todos os vértices têm o mesmo período.

Exercício 5.56. Seja P uma matriz de transição de um passeio aleatório irredutível com período d. Mostre que existem n_0 e m tais que $p_{i,j}^{(m+nd)} > 0$ para todos i, j vértices e $n \geqslant n_0$ natural.

Exercício 5.57. Uma **coloração** de um grafo é uma atribuição de cores para cada um de seus vértices. Um grafo é k-**colorível** se existe uma coloração com k cores de modo que vértices adjacentes não recebam a mesma cor. Seja G um grafo 3-colorível.

1. Mostre que é possível colorir o grafo com 2 cores sem que ocorra um triângulo com todos os vértices da mesma cor. Um triângulo é um subgrafo com três vértices adjacentes entre si.

2. Considere a seguinte estratégia para determinar uma coloração de G com duas cores de acordo com o item anterior: (i) tome uma coloração aleatória com 2 cores; (ii) enquanto houver triângulo monocromático, escolha um desses triângulos, sorteie um de seus vértices e troque a cor desse vértice.

   Determine um limitante superior para o número esperado de recolorações até que uma coloração de acordo com o item 1 seja encontrada.

Exercício 5.58. Suponha que P é uma matriz estocástica irredutível. Prove que para todo $0 < \alpha < 1$, a matriz $\alpha P + (1-\alpha)\mathrm{Id}$ é estocástica, irredutível e aperiódica com a mesma distribuição invariante de P.

Exercício 5.59. Considere o experimento de Ehrenfest com modificação do exemplo 5.21. Verifique que tal passeio é irredutível e aperiódico. Compute a distribuição de X_3. Defina Q, a matriz das transições no processo modificado dado acima e verifique que $q_{i,j}^{(t)} \to \binom{N}{j}/2^N$, quando $t \to \infty$. Verifique, também, se essa distribuição binomial é invariante para o processo original.

6 | LEIS DE DESVIO E DE CONCENTRAÇÃO

Exercício 6.1. Prove que $\mathsf{L} \in \mathsf{BPP}$ se, e só se, L é decidida por algoritmo probabilístico de tempo polinomial com probabilidade de erro $(1/2) - (1/p(n))$ nas entradas de tamanho n, para algum polinômio p (dica: como acima, simule um algoritmo que garante pertinência em BPP uma quantidade suficientemente grande de vezes e use a desigualdade de Chernoff para estimar a probabilidade de erro).

6.1 DESIGUALDADE DE MARKOV

A desigualdade de desvio mais simples que apresentamos é a desigualdade de Markov[1]. Uma vantagem dessa desigualdade é que quase não precisa de hipóteses sobre a variável aleatória, o que a torna amplamente aplicável, por outro lado, ela fornece limitantes fracos que muitas vezes são pouco úteis.

Teorema 6.2 (desigualdade de Markov) *Se X é uma variável aleatória somável*

$$\mathbb{P}[|X| \geqslant a] \leqslant \frac{\mathbb{E}\,|X|}{a} \tag{6.1}$$

[1] Andrey Markov (1856–1922) foi um matemático russo conhecido principalmente por suas contribuições no que foi posteriormente conhecido como Cadeias de Markov. Seu filho Andrey Markov Jr (1903–1979) ficou conhecido pelas contribuições em Teoria da Computação.

para todo $a > 0$.

Demonstração. Como $|X| \geqslant 0$ vale que $|X| \geqslant a\,\mathbb{1}_{[|X|\geqslant a]}$. De X somável $|X|$ também é somável, logo $\mathbb{E}\,|X| \geqslant a\,\mathbb{E}\,\mathbb{1}_{[|X|\geqslant a]}$ e $\mathbb{E}\,\mathbb{1}_{[|X|\geqslant a]} = a\,\mathbb{P}[\,|X| \geqslant a\,]$. □

Essa desigualdade é justa, como mostra o seguinte exemplo. Seja X uma variável aleatória que assume o valor 0 com probabilidade 99/100 e assume o valor 10 com probabilidade 1/100. Então $\mathbb{E}\,X = 1/10$ e

$$\frac{1}{100} = \mathbb{P}[\,X \geqslant 10\,] \leqslant \frac{1/10}{10} = \frac{1}{100}.$$

Como exemplo de aplicação dessa desigualdade nós vamos ver a seguir uma prova alternativa de que em uma execução do *quicksort* (seção 3.1.4) o número esperado de comparações entre elementos de uma instância com n elementos é $O(n \log n)$ com alta probabilidade. Consideremos um elemento x de uma instância S para o algoritmo e $S_0 = S, S_1, S_2, \ldots, S_M$ as subsequências a que x pertence após cada particionamento durante uma execução. Sorteando um pivô o i-ésimo particionamento é *bom* se $|S_i|/10 \leqslant |S_{i+1}| \leqslant 9|S_i|/10$ o que ocorre com probabilidade $0,75 < p \leqslant 0,8$ dada na equação (3.16). O tamanho esperado das subsequências pode ser estimado por

$$\mathbb{E}[\,|S_{i+1}|\,|\,|S_i| = k\,] \leqslant p\frac{9k}{10} + (1-p)k < 0,8\frac{9k}{10} + 0,25k = 0,97k$$

já que com probabilidade $p \leqslant 0,8$ temos $|S_{i+1}| \leqslant 9|S_i|/10$ e com probabilidade $1 - p < 0,25$ temos, trivialmente, $|S_{i+1}| \leqslant |S_i|$. Agora, podemos usar o teorema 3.42 para escrever

$$\begin{aligned}
\mathbb{E}\,|S_{i+1}| &= \mathbb{E}\,\mathbb{E}[\,|S_{i+1}|\,|\,|S_i|\,] \\
&= \sum_k \mathbb{E}[\,|S_{i+1}|\,|\,|S_i| = k\,]\,\mathbb{P}[\,|S_i| = k\,] \\
&< \sum_k 0,97k\,\mathbb{P}[\,|S_i| = k\,] \\
&= 0,97 \sum_k k\,\mathbb{P}[\,|S_i| = k\,] = 0,97\,\mathbb{E}\,|S_i|
\end{aligned}$$

portanto, iterando essa desigualdade concluímos $\mathbb{E}\,|S_M| < (0,97)^M n$. Para $M = a \log_{100/97} n$ com $a \geqslant 3$ constante, temos $\mathbb{E}\,|S_M| < n^{1-a}$. Pela desigualdade de Markov

$$\mathbb{P}[\,|S_M| > 20\,] \leqslant \frac{1}{20n^{a-1}}$$

de modo que a probabilidade de que exista um $x \in S$ que passe por mais que $a \log_{100/97} n$ particionamentos é

$$< n\frac{1}{20n^{a-1}} = \frac{1}{20n^{a-2}}$$

ou seja, o número de comparações efetuadas pelo *quicksort* é $O(n \log n)$ com probabilidade pelo menos $1 - 1/(20n^{a-2}) \geqslant 1 - 1/(20n)$.

Exemplo 6.3. Uma permutação $\sigma\colon \{1,2,\ldots,n\} \to \{1,2,\ldots,n\}$ tem ponto fixo em i se $\sigma(i) = i$. Para uma permutação escolhida ao acaso o probabilidade de ponto fixo em i é $1/n$, assim, o número esperado de pontos fixos é 1. Pela desigualdade de Markov, a probabilidade de pelo menos t pontos fixos é no máximo $1/t$ para qualquer $t \in \{1,2,\ldots,n\}$.

Exemplo 6.4 (*satisfatibilidade de fórmulas CNF*). Seja $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ uma fórmula booleana CNF e consideremos uma valoração aleatória das variáveis da fórmula. Tomemos por X o número de cláusulas falsas com essa atribuição. Se c_i é o número de literais na cláusula C_i então a probabilidade de C_i ser falsa com a valoração é 2^{-c_i}, portanto $\mathbb{E}\, X = \sum_{i=1}^{m} 2^{-c_i}$. Pela desigualdade de Markov

$$\mathbb{P}[\, X \geqslant 1\,] \leqslant \sum_{i=1}^{m} 2^{-c_i}$$

de modo que se $\sum_{i=1}^{m} 2^{-c_i} < 1$, então $\mathbb{P}[\, X \geqslant 1\,] < 1$ ou seja, o evento $[X = 0]$ tem probabilidade positiva. Em particular, para fórmulas k-CNF (cada cláusula tem k-literais) $\sum_{i=1}^{m} 2^{-c_i} = m/2^k$ logo se o número de cláusulas m é menor que 2^k então a fórmula admite uma valoração que a torna verdadeira. ◊

6.2 MOMENTOS, VARIÂNCIA E A DESIGUALDADE DE CHEBYSHEV

Dado um inteiro positivo k, definimos o momento de ordem k, ou o k-**ésimo momento** da variável aleatória $X\colon \Omega \to \mathbb{R}$ como a esperança de $\mathbb{E}\, X^k$

$$\mathbb{E}\, X^k = \sum_{t \in X(\Omega)} t^k \mathop{\mathbb{P}}_{X}(t)$$

quando está definida. De $|x|^k \leqslant |x|^{k+1} + 1$ para todo $x \in \mathbb{R}$ temos que se o momento de ordem $k+1$ é finito então o de ordem k também é finito. Ainda, usando a desigualdade de Markov podemos provar facilmente que, dado um inteiro positivo k, vale

$$\mathbb{P}[\,|X| \geqslant a\,] \leqslant \mathbb{P}\Big[\,|X|^k \geqslant a^k\,\Big] \leqslant \frac{\mathbb{E}\,|X|^k}{a^k}$$

para toda constante $a > 0$ sempre que $|X|^k$ é somável.

A **função geradora de momentos** da variável X é dada por

$$M_X(s) := \mathbb{E}[\, e^{sX}\,]$$

para todo $s \in \mathbb{R}$. Se em algum intervalo $(-x_0, x_0)$ da reta real a variável aleatória X tem os seus momentos finitos então, usando a série de Taylor, vale que

$$M_X(s) = \sum_{k \geqslant 0} \frac{s^k}{k!}\, \mathbb{E}[\, X^k\,]$$

para todo $s \in (-x_0, x_0)$.

Momentos são muito importante em Probabilidade porém neste texto vamos quase sempre usar apenas os primeiro e segundo momentos.

Se X tem esperança finita, definimos o k-**ésimo momento central** por $\mathbb{E}\,(X - \mathbb{E}\,X)^k$. A **variância** de uma variável aleatória somável como o segundo momento central dessa variável

$$\mathbb{V}[X] := \mathbb{E}\,(X - \mathbb{E}\,X)^2 = \sum_t (t - \mathbb{E}\,X)^2 \mathbb{P}_X(t) \tag{6.2}$$

a qual é finita sempre que X^2 é somável. De fato, dado $\mu \in \mathbb{R}$ temos $(x^2 - \mu)^2 \leqslant 2x^2 + 2\mu^2$ para qualquer real x. Notemos que desenvolvendo o quadrado da diferença em $(X - \mathbb{E}\,X)^2$ e usando a linearidade da esperança chegamos na identidade

$$\mathbb{V}[X] = \mathbb{E}[X^2] - (\mathbb{E}\,X)^2$$

para a variância de X.

Da equação (6.2) temos que a variância depende apenas da distribuição da variável aleatória.

Exemplo 6.5. Se $Z \sim \mathrm{Be}(p)$ então $\mathbb{E}\,Z = p$ e $\mathbb{E}\,Z^2 = 0^2(1-p) + 1^2 p = p$, logo $\mathbb{V}[Z] = p - p^2 = p(1-p) = pq$. Notemos que a variância é maximizada quando $p = 1/2$. Se $Y \sim \mathrm{Geom}(p)$ então $\mathbb{E}\,Y = 1/p$ e $\mathbb{E}\,Y^2 = (2-p)/p^2$, como vimos na equação (3.1.3), logo $\mathbb{V}[Y] = (1-p)/p^2$.

Em muitas aplicações a variável X é uma soma de variáveis aleatórias independentes e com mesma distribuição. Comecemos supondo que $X = \sum_{i=1}^n X_i$, com X_i^2 somável para todo i e, portanto, de variância finita. Com essas hipóteses vale que

$$\mathbb{E}\left(\sum_{i=1}^n X_i\right)^2 - \left(\sum_{i=1}^n \mathbb{E}\,X_i\right)^2 = \sum_{i=1}^n \sum_{k=1}^n \mathbb{E}\,X_i X_k - \sum_{i=1}^n \sum_{k=1}^n \mathbb{E}\,X_i\,\mathbb{E}\,X_k$$

portanto a variância de X é

$$\mathbb{V}[X] = \sum_{i=1}^n \sum_{k=1}^n (\mathbb{E}\,X_i X_k - \mathbb{E}\,X_i\,\mathbb{E}\,X_k), \tag{6.3}$$

e considerando os casos $i = k$ e $i \neq k$ separadamente, o lado direito da equação acima fica reescrito como

$$\sum_{i=1}^n \left(\mathbb{E}[X_i^2] - (\mathbb{E}\,X_i)^2\right) + \sum_{i=1}^n \sum_{i \neq k = 1}^n (\mathbb{E}\,X_i X_k - \mathbb{E}\,X_i\,\mathbb{E}\,X_k)$$

em que o termo $\mathbb{E}[X_i^2] - (\mathbb{E}\,X_i)^2$ é a variância $\mathbb{V}[X_i]$ e o termo $\mathbb{E}\,X_i X_k - \mathbb{E}\,X_i\,\mathbb{E}\,X_k = \mathbb{E}\,(X_i - \mathbb{E}\,X_i)(X_k - \mathbb{E}\,X_k)$ é conhecido como a **covariância** das variáveis aleatórias X_i e X_k, denotada $\mathrm{Cov}(X_i, X_k)$. Reescrevendo a equação (6.3),

$$\mathbb{V}[X] = \sum_{i=1}^n \mathbb{V}[X_i] + 2\sum_{i=1}^n \sum_{k=i+1}^n \mathrm{Cov}(X_i, X_k). \tag{6.4}$$

A covariância $\mathrm{Cov}(X_i, X_k)$ é finita sempre que as variáveis aleatórias tenham quadrado somável e, mais que isso, é $\mathrm{Cov}(X_i, X_k) = 0$ se X_i e X_k são independentes por consequência do teorema 3.30, página 139.

Proposição 6.6 *Se $X_1, \ldots, X_n$ são variáveis aleatórias somáveis e 2-a-2 independentes então*

$$\mathbb{V}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbb{V}[X_i]. \tag{6.5}$$

Demonstração. A afirmação segue imediatamente da equação (6.4) e da observação acima de que independência implica $\mathrm{Cov}(X_i, X_k) = 0$ para todos i e k distintos. □

Uma aplicação imediata do teorema acima é o cômputo da variância de uma variável Binomial. Se $X \sim \mathrm{b}(n, p)$ então a sua variância é $\mathbb{V}[X] = \sum_{i=1}^{n} \mathbb{V}[X_i] = npq$.

Exemplo 6.7. Suponha que os n elementos de S, subconjunto de um universo U, foram distribuídos por h dentre as m listas ligadas de uma tabela de espalhamento N de modo uniforme e 2-a-2 independente. Sabemos do capítulo anterior, página 129, que o número esperado de elementos numa lista é $\mu = n/m$. Denotemos por $\ell_i = \ell_i(h)$ o tamanho da i-ésima lista

$$\ell_i = \sum_{s \in S} \mathbb{1}_{[h(s)=i]}$$

e se fizermos

$$X_s := \mathbb{1}_{[h(s)=i]} - \frac{1}{m}$$

então $\sum_s X_s = \ell_i - \mu$, para todo i. Essa soma é uma variável aleatória cuja esperança é 0. Ademais

$$\mathbb{V}[\ell_i] = \mathbb{E}\,(\ell_i - \mu)^2 = \mathbb{E}\left(\sum_s X_s\right)^2 = \sum_s \mathbb{E}\left[X_s^2\right] + \sum_{s \neq t} \mathbb{E}\, X_s X_t = \sum_s \mathbb{E}\, X_s^2$$

pois caso $s \neq t$ vale $\mathbb{E}\, X_s X_t = \mathbb{E}\, X_s\, \mathbb{E}\, X_t$ pela independência das variáveis e a esperança é zero. Agora

$$\mathbb{E}\, X_s^2 = \mathbb{E}\left[\left(\mathbb{1}_{[h(s)=i]} - \frac{1}{m}\right)^2\right] = \mathbb{E}\, \mathbb{1}^2_{[h(s)=i]} - 2\frac{1}{m}\,\mathbb{E}\, \mathbb{1}_{[h(s)=i]} + \left(\frac{1}{m}\right)^2 = \frac{1}{m} - \frac{1}{m^2}$$

pois $\mathbb{E}\, \mathbb{1}^2_{[h(s)=i]} = \mathbb{E}\, \mathbb{1}_{[h(s)=i]} = \mathbb{P}[h(s) = i] = 1/m$. Concluindo, a variância de ℓ_i é $\mathbb{V}[\ell_i] = n(1/m - 1/m^2) = \mu(1 - 1/m)$. ◊

Exercício 6.8. Mostre que se X^2 é somável e $a, b \in \mathbb{R}$ então

$$\mathbb{V}[aX + b] = a^2\, \mathbb{V}[X].$$

CAUCHY–SCHWARZ O conjunto de todas as variáveis aleatórias reais e somáveis de $(\Omega, \mathbb{P})$ é um espaço vetorial real denotado $L^1(\Omega, \mathbb{P})$ e no qual a esperança é um operador linear. O conjunto $L^2(\Omega, \mathbb{P})$ das variáveis aleatórias X de $(\Omega, \mathbb{P})$ que têm quadrado somável é um subespaço vetorial de $L^1(\Omega, \mathbb{P})$ pelo seguinte corolário to teorema 3.28: *Se X^2 é somável então X é somável.* De fato, notemos que $\mathbb{E}\, X^2$ e $\mathbb{E}\,|X|$ sempre estão definidas por serem séries com termos não negativos e o enunciado segue da desigualdade $0 \leqslant |x| \leqslant x^2 + 1$, que vale para todo real x. Da monotonicidade e da linearidade da esperança $0 \leqslant \mathbb{E}\,|X| \leqslant \mathbb{E}\, X^2 + 1$ e como o lado direito é finito segue que $\mathbb{E}\,|X| < +\infty$.

Em $L^2(\Omega, \mathbb{P})$ vale a **desigualdade de Cauchy–Schwarz**: *Se $X, Y \in L^2(\Omega, \mathbb{P})$ então $X \cdot Y \in L^1(\Omega, \mathbb{P})$* e

$$|\mathbb{E}[XY]| \leqslant \sqrt{\mathbb{E}\, X^2}\sqrt{\mathbb{E}\, Y^2}.$$

Ainda, na segunda desigualdade vale o igual se para alguma constante não negativa c vale $Y = cX$ com probabilidade 1 ou $X = 0$ com probabilidade 1.

Como aplicação da desigualdade e Cauchy–Schwarz derivamos a seguir uma lei de desvio conhecida como **desigualdade de Paley–Zygmund**. Seja $\alpha \in [0, 1]$ um número real e uma variável aleatória X não-negativa e com quadrado somável. Da igualdade $X = X\,\mathbb{1}_{X \leqslant \alpha \mathbb{E}\, X} + X\,\mathbb{1}_{X > \alpha \mathbb{E}\, X}$. Da linearidade da esperança e da desigualdade de Cauchy–Schwarz deduzimos

$$\mathbb{E}\, X = \mathbb{E}[X\,\mathbb{1}_{X \leqslant \alpha \mathbb{E}\, X}] + \mathbb{E}[X\,\mathbb{1}_{X > \alpha \mathbb{E}\, X}] \leqslant \alpha\, \mathbb{E}\, X + \mathbb{E}[X\,\mathbb{1}_{X > \alpha \mathbb{E}\, X}] \leqslant \alpha\, \mathbb{E}\, X + \sqrt{\mathbb{E}[X^2] \cdot \mathbb{E}[\,\mathbb{1}_{X > \alpha \mathbb{E}\, X}]}$$

portanto

$$\mathbb{P}[X > \alpha\, \mathbb{E}\, X] \geqslant (1 - \alpha)^2 \frac{(\mathbb{E}\, X)^2}{\mathbb{E}[X^2]} \quad \text{ou} \quad \mathbb{P}[X > \alpha\, \mathbb{E}\, X] \geqslant (1 - \alpha)^2 \frac{(\mathbb{E}\, X)^2}{\mathbb{V}[X] + (\mathbb{E}\, X)^2}.$$

Para $\alpha = 0$ essa desigualdade foi dada no teorema 3.48.

6.2.1 DESIGUALDADE DE CHEBYSHEV

A próxima desigualdade é conhecida por desigualdade de Chebyshev[2], também conhecida por desigualdade de Bienaymé–Chebyshev[3]. Da equação 6.2 temos a **desigualdade de Chebyshev**

$$\mathbb{P}[|X| \geqslant a] \leqslant \frac{\mathbb{E}[X^2]}{a^2} \tag{6.6}$$

que é mais conhecida na forma enunciada a seguir.

[2]Pafnuty Lvovich Chebyshev (1821 – 1894) foi um matemático russo, dentre seus alunos vários são matemáticos reconhecidos e um deles foi Andrey Markov.

[3]Irénée-Jules Bienaymé (1796–1878) foi um estatístico francês, ele formulou a desigualdade que estudaremos em 1853 porém não a demonstrou.

Teorema 6.9 (desigualdade de Chebyshev) *Para toda constante $a > 0$ e toda variável aleatória X com quadrado somável temos*

$$\mathbb{P}[\,|X - \mathbb{E}\,X| \geqslant a\,] \leqslant \frac{\mathbb{V}[X]}{a^2}. \tag{6.7}$$

Demonstração. Basta usar a desigualdade (6.6) com a variável aleatória $X - \mathbb{E}\,X$. □

Essa desigualdade é justa, como mostra o seguinte exemplo. Seja X uma variável aleatória que assume o valor 0 com probabilidade $q \in (0,1)$ e os valores 1 e -1 com probabilidade $p/2$ cada, com $p = 1 - q$. A esperança de X é 0 e a variância é p. A probabilidade do evento $[|X| \geqslant 1]$ é p de modo que

$$p = \mathbb{P}[\,|X - 0| \geqslant 1\,] \leqslant \frac{\mathbb{V}[X]}{1^2} = p.$$

Como corolário da desigualdade de Chebyshev obtemos a seguinte versão para o *princípio de segundo momento* (teorema 3.48, página 164) para uma variável inteira e não negativa. Então

$$\mathbb{P}[\,X > 0\,] > 1 - \frac{\mathbb{V}[X]}{(\mathbb{E}\,X)^2}. \tag{6.8}$$

Exemplo 6.10 (*versão fraca da* Lei de Grandes Números). Seja $(X_n)_{n\geqslant 0}$ uma sequência de variáveis aleatórias 2-a-2 independentes, identicamente distribuídas e todas com valor esperado μ e variância σ^2. Definimos a média amostral

$$M_n := \frac{X_1 + X_2 + \cdots + X_n}{n}.$$

Do exercício 6.8 temos $\mathbb{V}[M_n] = (1/n)^2\,\mathbb{V}[X_1 + X_2 + \cdots + X_n]$, junto com a aditividade da variância (eq. (6.5)) e a linearidade da esperança, temos

$$\mathbb{E}\,M_n = \mu \quad\text{e}\quad \mathbb{V}[M_n] = \frac{\sigma^2}{n}.$$

Usando a desigualdade de Chebyshev para $\varepsilon > 0$ fixo

$$\mathbb{P}\Big[\,\big|M_n - \mu\big| \geqslant \varepsilon\,\Big] \leqslant \frac{(\sigma/\varepsilon)^2}{n}$$

e notemos que o lado direito tende a zero quando n tende ao infinito, logo

$$\lim_{n\to\infty} \mathbb{P}\Big[\,\big|M_n - \mu\big| < \varepsilon\,\Big] = 1$$

para todo $\varepsilon > 0$, concluímos que a média amostral converge *em probabilidade* ao valor esperado. ◊

Exemplo 6.11 (*tabelas de espalhamento – colisões*). Suponha que os n elementos de S, subconjunto de um universo U, foram distribuídos por h dentre as m listas ligadas de uma tabela de espalhamento N de modo uniforme e 2-a-2 independente. Sabemos do capítulo anterior que o número esperado de

colisões é $\mathbb{E}\,\mathsf{C} = \binom{n}{2}/m$. Para calcular a variância escrevemos C como a soma de variáveis aleatórias 2-a-2 independentes indicadoras de colisão $\mathsf{C} = \sum_{i<j} \mathbb{1}_{[h(i)=h(j)]}$, assim

$$\mathbb{V}[\,\mathsf{C}\,] = \sum_{i<j} \mathbb{V}\Big[\,\mathbb{1}_{[h(i)=h(j)]}\,\Big] = \sum_{i<j}\left(\frac{1}{m} - \frac{1}{m^2}\right) \leqslant \frac{1}{m}\binom{n}{2}.$$

Pela desigualdade de Markov (já fizemos essa conta na página 131)

$$\mathbb{P}[\,\mathsf{C} = 0\,] \geqslant \frac{1}{2}, \text{ se } n \leqslant \sqrt{m}$$

e por Chebyshev (eq. (6.8))

$$\mathbb{P}[\,\mathsf{C} = 0\,] \leqslant \frac{\mathbb{V}[\,\mathsf{C}\,]}{\mathbb{E}[\,\mathsf{C}\,]^2} \leqslant \frac{m}{\binom{n}{2}} \leqslant \frac{1}{2} \text{ se } n \geqslant 4\sqrt{m}.$$

$\Diamond$

Exemplo 6.12 (*tabelas de espalhamento – carga*). No mesmo contexto do exemplo anterior, vamos estimar a probabilidade de haver uma lista longa na tabela. Seja ℓ o tamanho da maior lista em N. Então, o número de colisões C é pelo menos $\binom{\ell}{2}$ e podemos usar essa estimativa do seguinte modo

$$\mathbb{P}\left[\,\frac{(\ell-1)^2}{2} \geqslant \frac{n^2}{m}\,\right] \leqslant \mathbb{P}\left[\,\binom{\ell}{2} \geqslant \frac{n^2}{m}\,\right] \leqslant \mathbb{P}\left[\,\mathsf{C} \geqslant \frac{n^2}{m}\,\right]$$

pois $(\ell-1)^2/2 \leqslant \binom{\ell}{2}$. Se $\mu = n/m$ é a carga da tabela então as desigualdades acima implicam

$$\mathbb{P}[\,\ell \geqslant \sqrt{2n\mu} + 1\,] \leqslant \frac{1}{2}\left(1 - \frac{1}{n}\right).$$

Uma estimativa via desigualdade de Chebyshev segue da seguinte forma. Denotemos por ℓ_i o tamanho da i-ésima lista e vamos estimar a probabilidade de ℓ_i desviar da média μ. Determinamos no exemplo 6.7 que

$$\mathbb{V}[\,\ell_i\,] = n\left(\frac{1}{m} - \frac{1}{m^2}\right).$$

A probabilidade de existir i com $\ell_i \geqslant t$ é no máximo

$$\mathbb{P}\left(\bigcup_{i=1}^{m} [\ell_i \geqslant t]\right) \leqslant \sum_{i=1}^{m} \mathbb{P}[\,\ell_i \geqslant t\,] = m\mathbb{P}[\,\ell_i \geqslant t\,]$$

e usando a desigualdade de Chebyshev com $t = (k-1)\mu$, para algum $k > 1$,

$$\mathbb{P}[\,\ell_i \geqslant k\mu\,] \leqslant \mathbb{P}[\,|\ell_i - \mu| \geqslant (k-1)\mu\,] \leqslant \frac{\mu(1-1/m)}{(k-1)^2\mu^2} = \frac{1}{(k-1)^2\mu}\left(1 - \frac{1}{m}\right).$$

Para compararmos com o limitante estabelecido via desigualdade de Markov dado acima fazemos $k = \sqrt{2n/\mu} + 1$ e $\ell = \max_i \ell_i$ e temos

$$\mathbb{P}[\,\ell \geqslant \sqrt{2n\mu} + 1\,] \leqslant \frac{m}{2n}\left(1 - \frac{1}{m}\right) = \frac{1}{2\mu}\left(1 - \frac{1}{m}\right)$$

que é uma estimativa um pouco melhor sempre que $n > m$. $\Diamond$

Exemplo 6.13. Uma pesquisa de intenção de voto será realizada no Rio de Janeiro para determinar a porcentagem de votos para o Macaco Tião nas próximas eleições para prefeito. Vamos determinar o número de entrevistados para que estejamos pelo menos 95% certos de que o valor tenha calculado tenha sido determinado com erro no máximo de 5%. Assumimos que os votos individuais de cada eleitor são independentes.

Denotamos por n a quantidade de entrevistados e por p ($0 \leqslant p \leqslant 1$) a fração, a ser estimada, dos votantes em Macaco Tião. Seja X_i, para $i = 1, 2, \ldots, n$, uma variável aleatória indicadora do voto da i-ésima no Macaco Tião, $X_i \in_{b_p} \{0, 1\}$. Uma aproximação para p é dada pela variável aleatória

$$\frac{S_n}{n} = \frac{X_1 + \cdots + X_n}{n}$$

cuja esperança é p e variância é $p(1-p)/n \leqslant 1/(4n)$. Queremos que este valor aproximado seja diferente do valor real de p por no máximo 0,05 com probabilidade maior ou igual a 95%, isto é

$$\mathbb{P}\left[\left|\frac{S_n}{n} - p\right| < 0{,}05\right] \geqslant 0{,}95.$$

Pela desigualdade de Chebyshev é suficiente que tenhamos

$$1 - \frac{\mathbb{V}[S_n/n]}{(0{,}05)^2} = 1 - \frac{p(1-p)}{(0{,}05)^2 n} \geqslant 1 - \frac{1}{4(0{,}05)^2 n} \geqslant 0{,}95$$

ou seja, $n \geqslant 2000$ entrevistados. ◊

DESIGUALDADE DE CHEBYSHEV PARA SOMA DE VARIÁVEIS ALEATÓRIAS INDICADORAS E 2-A-2 INDEPENDENTES Lembremos da equação (6.5) na página 310 que expressa que a variância da soma de variáveis aleatórias 2-a-2 independentes $X_1, \ldots, X_n$ é a soma das variâncias dessas variáveis aleatórias. Ainda, caso X_i assuma valores em $\{0, 1\}$, temos

$$\mathbb{V}[X_i] = \mathbb{E}[X_i^2] - (\mathbb{E}\, X_i)^2 = \mathbb{E}\, X_i (1 - \mathbb{E}\, X_i) \leqslant \mathbb{E}\, X_i$$

para $1 \leqslant i \leqslant n$, portanto, $\mathbb{V}[\sum_i X_i] = \sum_i \mathbb{V}[X]_i \leqslant \sum_i \mathbb{E}\, X_i = \mathbb{E} \sum_i X_i$ e segue da desigualdade de Chebyshev, equação (6.7), que *se* X *é soma de variáveis indicadoras 2-a-2 independentes então*

$$\mathbb{P}[|X - \mathbb{E}\, X| \geqslant t] \leqslant \frac{\mathbb{E}\, X}{t^2} \tag{6.9}$$

que é bastante útil. Em particular, se X é binomial então $\mathbb{P}[|X - np| \geqslant \varepsilon np] \leqslant 1/(\varepsilon^2 np)$ ou

$$\mathbb{P}[(1-\varepsilon)np < X < (1+\varepsilon)np] > 1 - \frac{1}{\varepsilon^2 np}$$

6.2.2 *HASHING* 2-UNIVERSAL

Já fizemos várias estimativas sobre probabilidades de alguns eventos relacionados a uma função de *hash* na seção 3.1.2 e em vários exemplos no decorrer deste texto. Nessas estimativas assumimos hipóteses importantes como, por exemplo, que qualquer elementos do domínio assume um valor no contradomínio M com probabilidade $1/|M|$ e que dois elementos distintos colidem com probabilidade $1/|M|$. Certamente, qualquer função que cumpra estatísticas relevantes como, por exemplo, a probabilidade de colisão é $1/|M|$, dará o mesmo número médio de colisões que uma função de *hash* aleatória.

As funções aleatórias não têm uma descrição pequena e ainda há a dificuldade de obtermos uma função que seja genuinamente aleatória. Uma ideia chave para contornar esses problemas foi dada por Carter e Wegman em 1979: a função de *hash* é escolhida de uma família relativamente pequena de funções que são fáceis de computar, precisam de pouco espaço para serem descritas e têm algumas propriedades estatísticas das funções aleatórias (Carter e Wegman, 1979).

Por exemplo, para $U = M = \{0, 1, \ldots, p-1\}$ com p primo, tomemos a família $\{h_a \in M^U : a \in U\}$ das funções dadas por $h_a(x) := ax \bmod p$. Uma escolha aleatória a define unicamente uma função e dados dois elementos x e y distintos no domínio, eles colidem com probabilidade $1/p$ pois, de p primo, existe um único a tal que $ax \bmod p = ay \bmod p$ para quaisquer $x \neq y$.

Para uma família $\mathcal{H} = \{h_\lambda \in M^U : \lambda \in \Lambda\}$ de funções de *hash* pedimos que cada h_λ seja computável eficientemente, tenha descrição curta, que para todo $x \in U$ e $\lambda \in_R \Lambda$ a variável aleatória $h(x)$ dada por

$$h(x)(\lambda) := h_\lambda(x)$$

tenha distribuição uniforme em M, que os eventos

$$[h(x) = i] := \{\lambda \in \Lambda : h_\lambda(x) = i\} \quad \text{e} \quad [h(y) = j] := \{\lambda \in \Lambda : h_\lambda(y) = j\}$$

sejam independentes para $x \neq y$ quaisquer e, finalmente, que $|\Lambda|$ não seja muito grande para que a escolha aleatória de λ também seja eficiente.

Uma família de funções $\mathcal{H} \subset M^U$, com $|U| \geqslant |M|$, é uma família **2-universal de funções de *hash*** se munida da distribuição uniforme $\mathbb{P}_\Lambda$ temos

$$\mathop{\mathbb{P}}_{\lambda \in \Lambda}([h(x) = i] \cap [h(y) = j]) = \frac{1}{|M|^2},$$

para quaisquer $x, y \in U$ distintos e para quaisquer $i, j \in M$.

Notemos que $[h(x) = i]$ e $[h(y) = j]$ são independentes; a distribuição de $h(x)$ é uniforme pois, dados x e $y \in U$ distintos e $i \in M$

$$\mathop{\mathbb{P}}_{h(x)}(i) = \mathop{\mathbb{P}}_{\lambda \in \Lambda}[h(x) = i] = \sum_{j \in M} \mathop{\mathbb{P}}_{\lambda \in \Lambda}([h(x) = i] \cap [h(y) = j]) = \sum_{j \in M} \frac{1}{|M|^2} = \frac{1}{|M|}.$$

Ainda, para $S \subset U$ e qualquer $i \in M$ vale que

$$\mu := \mathbb{E} \sum_{x \in S} \mathbb{1}_{[h(x)=i]} = \sum_{u \in S} \frac{1}{|M|} = \frac{|S|}{|M|}$$

é o número esperado para o tamanho do subconjunto dos elementos de S que são mapeados para um único elemento do conjunto M.

A família de funções sobre $U = M = \{0, 1, \ldots, p-1\}$

$$\left\{ h_{(a,b)} \in M^U \colon a, b \in U,\ a \neq 0 \right\}$$

com p primo, dadas por $h_{(a,b)}(x) := ax+b \bmod p$ é 2-universal. Se $x \neq y$ então $h_{(a,b)}(x) = i$ e $h_{(a,b)}(y) = j$ para um único par $(a, b) \in U^2$ com $a \neq 0$, portanto, para uma escolha aleatória uniforme de (a, b) ocorre $[h_{(a,b)}(x) = i] \cap [h_{(a,b)}(y) = j]$ com probabilidade $1/p^2$ para quaisquer i e j em M.

Exemplo 6.14. Consideremos agora $(\mathbb{F}, +, \cdot)$ um corpo finito com q elementos e a família de funções de $\mathbb{F}^n$ em $\mathbb{F}$

$$\mathcal{H} := \left\{ h_{(a,b)}(x) := \langle a, x \rangle + b \colon a, x \in \mathbb{F}^n,\ b \in \mathbb{F},\ a \neq 0 \right\}$$

onde $\langle u, v \rangle$ denota o produto escalar usual $\langle u, v \rangle = \sum_k u_k v_k$. Uma escolha aleatória de uma função é dada por $(a, b) \in_R \mathbb{F}^n \times \mathbb{F}$ com a medida produto (os sorteios $a_1, \ldots, a_n, b$ em $\mathbb{F}$ são independentes).

Se $x \neq 0$ então para uma escolha aleatória de $a \in \mathbb{F}^n$ a probabilidade do evento $[\langle a, x \rangle = j]$ é $1/q$, pelo princípio da decisão adiada, pois $a_k = (1/x_k)(y - \sum_{l \neq k} a_l x_l)$ para algum k tal que $x_k \neq 0$.

Dados $x, y \in \mathbb{F}^k$ distintos e $i, j \in \mathbb{F}$ o evento

$$[\langle a, x \rangle + b = i] \cap [\langle a, y \rangle + b = j]$$

definido pelos pares $(a, b) \in \mathbb{F}^n \times \mathbb{F}$ é dado também por

$$[\langle a, x \rangle - \langle a, y \rangle = i - j] \cap [b = \langle a, x \rangle - i].$$

Dos dois parágrafos anteriores temos $\mathbb{P}[\langle a, x \rangle - \langle a, y \rangle = i - j] = \mathbb{P}[\langle a, x - y \rangle = i - j] = 1/q$ e $\mathbb{P}[b = \langle a, x \rangle - i] = 1/q$. Logo

$$\mathbb{P}([\langle a, x \rangle + b = i] \cap [\langle a, y \rangle + b = j]) = \frac{1}{q^2}$$

e concluímos que $\mathcal{H}$ é 2-universal. ◊

No exemplo acima com o caso particular de $\mathcal{H}$ ser o conjunto das 2^3 funções de $\{0,1\}^2$ em $\{0,1\}$, cada função é dada por 3 bits, dos quais 2 são do parâmetro a e 1 do parâmetro b. Assim, sorteando três bits temos que $h_{a,b}(00)$, $h_{a,b}(01)$, $h_{a,b}(10)$ e $h_{a,b}(11)$ formam uma sequência de 4 bits com distribuição uniforme e 2-a-2 independentes. No caso geral para o corpo binário, $\mathbb{F} = \{0,1\}$, sorteando $n+1$ bits obtemos uma sequência de 2^n bits com distribuição uniforme e 2-a-2 independentes. Essa sequência pode ser usada para a desaleatorização de algoritmos que usam bits aleatórios com

independência 2-a-2, como é o caso do algoritmo para achar cortes grandes em grafos (algoritmo 36, página 150). Mais detalhes desse caso são dados na página 333.

No caso de uma função aleatória $h \in_R \mathsf{M}^{\mathsf{U}}$, se X é a quantidade de elementos de $\mathsf{S} \subset \mathsf{U}$ que são mapeados em $i \in \mathsf{M}$, então $\mu = |\mathsf{S}|/|\mathsf{M}|$ é o valor esperado de X. A desigualdade de Chebyshev nos diz que

$$\mathbb{P}\Big[\big|\mathsf{X} - \mu\big| < \varepsilon\mu\Big] > 1 - \frac{1}{\varepsilon^2\mu}$$

ou seja, $1 - 1/\varepsilon^2\mu$ das $|\mathsf{M}|^{|\mathsf{U}|}$ funções mapeiam os elementos de S de modo que $(1-\varepsilon)\mu < \mathsf{X} < (1+\varepsilon)\mu$, isto é, a quantidade de elementos de S que são mapeados em i por um escolha aleatória de h é próximo a média. Esse resultado pode ser transferido para famílias 2-universal de funções de *hash*, para todo S suficientemente grande, quase toda função de uma família 2-universal distribui os elementos de S de modo aproximadamente uniforme no contradomínio.

Lema 6.15 *Seja $\mathcal{H} \subset \mathsf{M}^{\mathsf{U}}$ uma família 2-universal de funções* hash. *Para todo $\varepsilon > 0$, para todo $i \in \mathsf{M}$ e todo $\mathsf{S} \subset \mathsf{U}$, se $|\mathsf{S}| > \varepsilon^{-3}|\mathsf{M}|$ então para pelo menos uma fração $1 - \varepsilon$ das funções h em $\mathcal{H}$ vale*

$$(1-\varepsilon)\frac{|\mathsf{S}|}{|\mathsf{M}|} < \Big|\{\, u \in \mathsf{S} \colon h(u) = i \,\}\Big| < (1+\varepsilon)\frac{|\mathsf{S}|}{|\mathsf{M}|}. \tag{6.10}$$

Demonstração. Para demonstrar esse resultado, vamos provar que para uma escolha aleatória uniforme de função em $\mathcal{H} = \{\, h_\lambda : \lambda \in \Lambda \,\}$ o evento correspondente a equação (6.10) tem probabilidade pelo menos $1 - \varepsilon$.

Sejam ε e S como no enunciado. Fixados $u \in \mathsf{U}$ e $i \in \mathsf{M}$, consideremos as variáveis aleatórias indicadoras $\mathbb{1}_{[h(u)=i]}$ do evento "a função sorteada mapeia u em i" e a variável aleatória X dada por

$$\mathsf{X}(\lambda) := \sum_{u \in \mathsf{S}} \mathbb{1}_{[h(u)=i]}(\lambda) = \Big|\{\, u \in \mathsf{S} \colon h_\lambda(u) = i \,\}\Big|$$

cujo valor esperado, como já vimos, é $\mu := |\mathsf{S}|/|\mathsf{M}|$.

Pela desigualdade de Chebyshev, equação (6.9) na página 314

$$\mathbb{P}\Big[\big|\mathsf{X} - \mu\big| \geqslant \varepsilon\mu\Big] \leqslant \frac{\mu}{\varepsilon^2\mu^2} \leqslant \varepsilon$$

a segunda desigualdade decorre de $\mu > \varepsilon^{-3}$. Daí temos que para uma escolha aleatória uniforme em $\mathcal{H}$ vale com probabilidade maior que $1 - \varepsilon$ que $|\mathsf{X} - \mu| < \varepsilon\mu$ ou seja, para uma fração maior que $1 - \varepsilon$ das funções

$$(1-\varepsilon)\mu < \mathsf{X} < (1+\varepsilon)\mu$$

donde segue a afirmação do lema. □

Esse resultado pode ser generalizado de modo que para todo $\mathsf{T} \subset \mathsf{M}$

$$(1-\varepsilon)\frac{|\mathsf{S}|}{|\mathsf{M}|}|\mathsf{T}| < \Big|\{\, u \in \mathsf{S} \colon h(u) \in \mathsf{T} \,\}\Big| < (1+\varepsilon)\frac{|\mathsf{S}|}{|\mathsf{M}|}|\mathsf{T}| \tag{6.11}$$

para quase toda h numa família 2-universal de funções, dado que $|S||T|$ é grande o suficiente. De fato, agora fazemos

$$Y := \sum_{u \in S} \mathbb{1}_{[h(u) \in T]} = \big|\{\, u \in S \colon h(u) \in T \,\}\big| \tag{6.12}$$

a qual tem média $\mathbb{E}\, Y = |S||T|/|M| = \mu|T|$, portanto,

$$\mathbb{P}\Big[\,\big|\, Y - \mu|T| \,\big| \geqslant \varepsilon\mu|T| \,\Big] \leqslant \varepsilon$$

dado que $|S||T| \geqslant \varepsilon^{-3}|M|$. Daí segue (6.11) como na demonstração do lema anterior.

Agora, consideremos o subconjunto das funções $h_\lambda \in \mathcal{H}$ tais que

$$\left| \frac{Y(\lambda)}{|U|} - \frac{|S|}{|U|}\frac{|T|}{|M|} \right| \leqslant \varepsilon \tag{6.13}$$

em que Y é a variável aleatória definida em (6.12). A probabilidade com que uma escolha uniforme em $\mathcal{H}$ não satisfaça a condição da equação (6.13) é

$$\mathbb{P}\left[\left| Y - \frac{|S||T|}{|M|} \right| \geqslant \varepsilon|U| \right] \leqslant \frac{|S||T|/|M|}{\varepsilon^2|U|^2} = \frac{|S||T|/|U||M|}{\varepsilon^2|U|}$$

de acordo com a desigualdade de Chebyshev.

Podemos interpretar a equação (6.13) como a probabilidade de um escolha uniforme u estar em S e ter imagem $h(u)$ em T é aproximadamente a probabilidade de escolhas uniformes e independentes de uma elemento de S e um imagem dele em T

$$\left| \mathop{\mathbb{P}}_{u \in U}([u \in S] \cap [h(u) \in T]) - \mathop{\mathbb{P}}_{U}(S) \mathop{\mathbb{P}}_{M}(T) \right| \leqslant \varepsilon.$$

Nesse caso, dizemos que a função de *hash* $h \colon U \to M$ é *ε-independente* para (S, T).

Lema 6.16 (Hash mixing lemma, Nisan, 1992) *Seja $\mathcal{H} \subset M^U$ uma família 2-universal. Então,*

$$\mathop{\mathbb{P}}_{h \in \mathcal{H}}[h \text{ não é } \varepsilon\text{-independente para } (S,T)] \leqslant \frac{1}{\varepsilon^2} \frac{\frac{|S|}{|U|}\frac{|T|}{|M|}}{|U|}$$

para quaisquer $\varepsilon > 0$, $S \subset U$ e $T \subset M$. □

6.3 DESIGUALDADES DE BERNSTEIN–CHERNOFF–HOEFFDING.

Vamos começar com uma versão bem simples de uma família de desigualdades derivadas de uma mesma técnica que parece ter sido usada pela primeira vez por Bernstein[4] em 1924. Se X é uma variável aleatória e $t \in \mathbb{R}$, pela desigualdade de Markov deduzimos

[4] Sergei Natanovich Bernstein (1880–1968) foi um matemático russo. Em sua tese de doutorado resolveu o décimo-nono problema de Hilbert sobre a solução analítica de equações diferenciais elípticas. Em 1917, sugeriu a primeira base axiomática da teoria de probabilidade.

- $\mathbb{P}[\,X \geqslant t\,] = \mathbb{P}[\,\exp(\lambda X) \geqslant \exp(\lambda t)\,]$ para todo λ positivo, logo

$$\mathbb{P}[\,X \geqslant t\,] = \mathbb{P}[\,e^{\lambda X} \geqslant e^{\lambda t}\,] \leqslant \frac{\mathbb{E}\, e^{\lambda X}}{e^{\lambda t}} \tag{6.14}$$

- $\mathbb{P}[\,X \leqslant t\,] = \mathbb{P}[\,\exp(\lambda X) \geqslant \exp(\lambda t)\,]$ para todo λ negativo, logo

$$\mathbb{P}[\,X \leqslant t\,] = \mathbb{P}[\,e^{\lambda X} \geqslant e^{\lambda t}\,] \leqslant \frac{\mathbb{E}\, e^{\lambda X}}{e^{\lambda t}}. \tag{6.15}$$

Teorema 6.17 (desigualdade de Chernoff) *Se* $X = \sum_{i=1}^{n} X_i$ *com* $X_i \in_R \{-1, 1\}$ *independentes e* $t > 0$ *então*

$$\mathbb{P}[\,|X| \geqslant t\,] < 2\exp\left(\frac{-t^2}{2n}\right). \tag{6.16}$$

Demonstração. Se X é uma variável aleatória como enunciado então pela independência das variáveis aleatórias X_i

$$\mathbb{E}\, e^{\lambda X} = \prod_{i=1}^{n} \mathbb{E}\, e^{\lambda X_i} = \prod_{i=1}^{n} \frac{e^{\lambda} + e^{-\lambda}}{2}$$

cada termo desse produto é a função cosseno hiperbólico cuja série de Taylor é

$$\frac{1}{2}\left(e^{\lambda} + e^{-\lambda}\right) = \sum_{i \geqslant 0} \frac{\lambda^{2i}}{(2i)!} < \sum_{i \geqslant 0} \left(\frac{\lambda^2}{2}\right)^i \frac{1}{i!} = e^{\frac{\lambda^2}{2}}$$

portanto, de (6.14), fazendo $\lambda = t/n$

$$\mathbb{P}[\,X \geqslant t\,] < e^{(\frac{\lambda^2 n}{2} - \lambda t)} = e^{(-\frac{t^2}{2n})} \tag{6.17}$$

de (6.15), fazendo $\lambda = -t/n$

$$\mathbb{P}[\,X \leqslant -t\,] < e^{(\frac{\lambda^2 n}{2} + \lambda t)} = e^{(-\frac{t^2}{2n})}$$

portanto $\mathbb{P}[\,|X| \geqslant t\,] < 2\exp(-t^2/(2n))$. □

Exemplo 6.18 (*Markov, Chebyshev e Chernoff no lançamento de moedas*). Consideremos X_n a quantidade de caras no lançamento de n moedas equilibradas. O número esperado de caras é $n/2$, portanto, pela desigualdade de Markov temos

$$\mathbb{P}[\,X_n \geqslant \frac{3}{4}n\,] \leqslant \frac{n/2}{3n/4} = \frac{2}{3}.$$

Usando a desigualdade de Chebyshev

$$\mathbb{P}[\,\left|X_n - \frac{n}{2}\right| \geqslant \frac{n}{4}\,] \leqslant \frac{n/4}{(n/4)^2} = \frac{4}{n}$$

em particular, $\mathbb{P}[\,X_n \geqslant 3n/4\,] \leqslant 4/n$.

Agora, fazemos $X_n = \sum_{i=1}^n L_i$ para $L_i \in_R \{0,1\}$ que indica se o i-ésimo lançamento foi cara ou não. Definimos $Y_i = 2L_i - 1$ e temos $Y_i \in_R \{-1,1\}$ e por (6.17) temos $\mathbb{P}[\sum_{i=1}^n Y_i \geqslant t] \leqslant \exp(-t^2/2n)$ para todo $t > 0$. Notemos que $\sum_i Y_i = 2X_n - n$ portanto $\mathbb{P}[X_n \geqslant (t+n)/2] \leqslant \exp(-t^2/2n)$; fazendo $t = n/2$

$$\mathbb{P}[X_n \geqslant \frac{3n}{4}] \leqslant e^{-\frac{n}{8}}.$$

Em resumo, temos as seguintes limitantes superiores para a probabilidade de $X_n \geqslant 1{,}5\,\mathbb{E}\,X_n$

	Markov	Chebyshev	Chernoff
$\mathbb{P}[X_n \geqslant 3n/4]$	$\leqslant 2/3$	$\leqslant 4/n$	$\leqslant \exp(-n/8)$

embora a desigualdade de Chernoff seja muito mais poderosa devemos lembrar que essas desigualdades assumem hipóteses diferentes, principalmente, com relação a independência quando temos soma de variáveis aleatórias. ◊

Revendo a transformação de variáveis do exemplo anterior, provamos o seguinte resultado.

Corolário 6.19 *Se X tem distribuição binomial com parâmetros n e $1/2$ e $t > 0$ então*

$$\mathbb{P}[|X - \mathbb{E}\,X| \geqslant t] \leqslant 2\exp\left(\frac{-2t^2}{n}\right).$$

Demonstração. Sejam $X_i \sim \mathrm{Be}(1/2)$ variáveis aleatórias independentes $X = \sum_{i=1}^n X_i$.

Fazendo $Y_i = 2X_i - 1$ temos $Y_i \in \{-1,1\}$, para todo i, e se $Y = \sum_i Y_i$ então $Y = 2X - n$, logo $Y = 2(X - \mathbb{E}\,X)$, portanto,

$$\mathbb{P}[|X - \mathbb{E}\,X| \geqslant t] = \mathbb{P}[Y \geqslant 2t] < 2\exp\left(\frac{2t^2}{n}\right)$$

pela desigualdade de Chernoff, equação (6.16). □

Um caso geral do resultado acima pode ser enunciado da seguinte maneira.

Teorema 6.20 (desigualdades de Chernoff) *Sejam $p_i \in [0,1]$ e $X_i \sim \mathrm{Be}(p_i)$, $1 \leqslant i \leqslant n$, variáveis aleatórias independentes, e seja $t > 0$. Se $X = \sum_{i=1}^n X_i$ então*

$$\begin{aligned} \mathbb{P}[X \geqslant (1+t)\mathbb{E}\,X] &\leqslant \left(\frac{e^t}{(1+t)^{1+t}}\right)^{\mathbb{E}\,X} && \text{para todo } t > 0, \text{ e} \\ \mathbb{P}[X \leqslant (1-t)\mathbb{E}\,X] &\leqslant \left(\frac{e^{-t}}{(1-t)^{1-t}}\right)^{\mathbb{E}\,X} && \text{para todo } t \in (0,1). \end{aligned} \tag{6.18}$$

Demonstração. Consideremos X, X_i e p_i como enunciado. Pela equação (6.14) na página 319, para todo $\lambda > 0$

$$\mathbb{P}[X \geqslant (1+t)\mathbb{E}\,X] \leqslant \frac{\mathbb{E}\,e^{\lambda X}}{e^{\lambda(1+t)\mathbb{E}\,X}} \leqslant \frac{e^{(e^\lambda - 1)\mathbb{E}\,X}}{e^{\lambda(1+t)\mathbb{E}\,X}} \tag{6.19}$$

pois

$$\mathbb{E}\, e^{\lambda X_i} = p_i e^{\lambda} + (1-p_i)e^0 = 1 + p_i(e^{\lambda}-1) \leqslant e^{p_i(e^{\lambda}-1)}.$$

Dado $t > 0$, definimos $\lambda = \ln(1+t) > 0$ e substituindo em (6.19) obtemos

$$\mathbb{P}[\, X \geqslant (1+t)\mathbb{E}\, X\,] \leqslant \left(\frac{e^t}{(1+t)^{1+t}}\right)^{\mathbb{E}\, X}.$$

Analogamente, pela equação (6.15), para todo $\lambda < 0$

$$\mathbb{P}[\, X \leqslant (1-t)\mathbb{E}\, X\,] \leqslant \frac{\mathbb{E}\, e^{\lambda X}}{e^{\lambda(1-t)\mathbb{E}\, X}} \leqslant \frac{e^{(e^{\lambda}-1)\mathbb{E}\, X}}{e^{\lambda(1-t)\mathbb{E}\, X}}.$$

Para $t \in (0,1)$ definimos $\lambda = \ln(1-t) < 0$ e obtemos

$$\mathbb{P}[\, X \leqslant (1-t)\mathbb{E}\, X\,] \leqslant \left(\frac{e^{-t}}{(1-t)^{1-t}}\right)^{\mathbb{E}\, X}$$

o que prova as desigualdades. □

Dessas desigualdades derivamos a seguinte desigualdade que é mais fácil de aplicar.

Corolário 6.21 *Se* $X_i \sim \mathrm{Be}(p_i)$, $1 \leqslant i \leqslant n$, *são independentes com* $p_i \in [0,1]$, $t \in (0,1)$ *e* $X = \sum_{i=1}^{n} X_i$ *então*

$$\mathbb{P}[\, |X - \mathbb{E}\, X| \geqslant t\mathbb{E}\, X\,] \leqslant 2\exp\left(\frac{-t^2\mathbb{E}\, X}{3}\right).$$

Demonstração. Para $t \in (0,1)$ valem as desigualdades

$$\frac{e^t}{(1+t)^{1+t}} \leqslant e^{-t^2/3} \quad \text{e} \quad \frac{e^{-t}}{(1-t)^{1-t}} \leqslant e^{-t^2/3}$$

e a verificação é deixada como exercício. □

Observação 6.22. Na prática podemos ter uma estimativa $\alpha \leqslant \mathbb{E}\, X \leqslant \beta$ e usar as desigualdades para todo $t \in (0,1)$

$$\mathbb{P}[\, X \geqslant (1+t)\beta\,] \leqslant \exp\left(-\frac{t^2\beta}{3}\right) \quad \text{e} \quad \mathbb{P}[\, X \leqslant (1-t)\alpha\,] \leqslant \exp\left(-\frac{t^2\alpha}{3}\right).$$

Exemplo 6.23 (*tabelas de espalhamento*). A desigualdade de Chernoff limita o desvio da média de variáveis que são soma de variáveis independentes. Assumindo $t > 2e-1$ podemos simplificar a expressão a direita da desigualdade em (6.18) e temos

$$\mathbb{P}[\, X \geqslant (1+t)\mathbb{E}\, X\,] \leqslant \left(\frac{e^t}{(2e)^{1+t}}\right)^{\mathbb{E}\, X} < 2^{-t\mathbb{E}\, X}.$$

De volta ao problema do tamanho das listas ligadas numa tabela de espalhamento, se os n elementos são distribuídos uniforme e independentemente pelas m listas, então a desigualdade acima, para todo $i \in \mathsf{M}$, nos dá

$$\mathbb{P}[\,\ell_i \geqslant (1+t)\mu\,] \leqslant 2^{-t\mu}$$

em que ℓ_i é o tamanho da lista i e μ o tamanho esperado. Escolhendo $t = 2(\log_2 n)/\mu$

$$\mathbb{P}[\,\ell_i \geqslant 2\log_2 n + \mu\,] \leqslant 2^{-2\log_2 n} = \frac{1}{n^2}.$$

Dessa forma, a probabilidade de haver uma lista na tabela com mais que $2\log_2 n + \mu$ elementos é

$$\mathbb{P}[\bigcup_{i\in\mathsf{M}}\Big[\ell_i \geqslant 2\log_2 n + \mu\Big]] \leqslant \frac{1}{n}.$$

$\Diamond$

Para finalizar essa seção, vamos mostrar o seguinte resultado.

Teorema 6.24 (desigualdade de Chernoff–Hoeffding, 1963) *Para $1 \leqslant i \leqslant n$, sejam X_i variáveis aleatórias independentes com valor esperado μ_i e tais que $0 \leqslant \mathsf{X}_i \leqslant 1$. Façamos $\overline{\mathsf{X}}$ a média aritmética de $\{\mathsf{X}_i\}_{i=1}^n$ e $\overline{\mu}$ o valor esperado dessa média, isto é*

$$\overline{\mathsf{X}} = \frac{\sum_i \mathsf{X}_i}{n} \;\; e \;\; \overline{\mu} = \mathbb{E}\,\overline{\mathsf{X}}.$$

Então para todo $t > 0$

$$\begin{aligned}\mathbb{P}[\,\overline{\mathsf{X}} - \overline{\mu} > t\,] &\leqslant \left(\left(\frac{\overline{\mu}}{\overline{\mu}+t}\right)^{\overline{\mu}+t}\left(\frac{1-\overline{\mu}}{1-(\overline{\mu}+t)}\right)^{1-(\overline{\mu}+t)}\right)^n \\ &= \exp\left(-\left((\overline{\mu}+t)\ln\frac{\overline{\mu}+t}{\overline{\mu}} + (1-\overline{\mu}-t)\ln\frac{1-\overline{\mu}-t}{1-\overline{\mu}}\right)n\right)\end{aligned} \tag{6.20}$$

Demonstração. Usando (6.14), na página 319,

$$\mathbb{P}[\,\overline{\mathsf{X}} > \overline{\mu} + t\,] = \mathbb{P}[\,\mathsf{X} - \mathbb{E}\,\mathsf{X} \geqslant nt\,] \leqslant \frac{\mathbb{E}\,e^{\lambda \mathsf{X}}}{e^{\lambda(n\overline{\mu}+nt)}} = \frac{\prod_i \mathbb{E}\,e^{\lambda \mathsf{X}_i}}{e^{\lambda n(\overline{\mu}+t)}}$$

para todo $\lambda > 0$. Agora, usamos o fato da função exponencial ser convexa, assim no intervalo $[0,1]$ vale

$$e^{\lambda \mathsf{X}_i} \leqslant \frac{1-\mathsf{X}_i}{1-0}e^{\lambda 0} + \frac{\mathsf{X}_i - 0}{1-0}e^{\lambda 1} = 1 - \mathsf{X}_i + \mathsf{X}_i e^{\lambda}$$

donde deduzimos que

$$\prod_i \mathbb{E}\,e^{\lambda \mathsf{X}_i} \leqslant \prod_i (1 - \mu_i + \mu_i e^{\lambda})$$

e usando a desigualdade entre as médias aritmética e geométrica

$$\left(\prod_i (1-\mu_i+\mu_i e^{\lambda})\right)^{1/n} \leqslant \frac{1}{n}\sum_i (1-\mu_i+\mu_i e^{\lambda})$$

e o lado direito da equação acima é

$$\frac{1}{n}\sum_i (1-\mu_i+\mu_i e^{\lambda}) = 1-\overline{\mu}+\overline{\mu}e^{\lambda}$$

portanto

$$\mathbb{P}[\,\overline{X} > \overline{\mu}+t\,] \leqslant \left(\frac{1-\overline{\mu}+\overline{\mu}e^{\lambda}}{e^{\lambda(\overline{\mu}+t)}}\right)^n$$

e o lado direito assume valor mínimo quando $\lambda = \lambda_0$ tal que

$$e^{\lambda_0} = \frac{(1-\overline{\mu})(\overline{\mu}+t)}{(1-\overline{\mu}-t)\overline{\mu}}$$

que, substituindo na equação acima, resulta em (6.20). □

Exercício 6.25. Derive uma desigualdade para as variáveis $1-X_i$ nas mesmas hipóteses do teorema 6.24 acima. Conclua o exercício estabelecendo uma desigualdade para $\mathbb{P}[\,|\overline{X}-\overline{\mu}| > t\,]$.

Exercício 6.26. Simplifique o lado direito de (6.20) e conclua o resultado abaixo.

Corolário 6.27 (desigualdade de Chernoff–Hoeffding, segunda versão) *Sejam* X_i, $1 \leqslant i \leqslant n$ *tais que* $0 \leqslant X_i \leqslant 1$ *para todo* i. *Então para todo* $t > 0$

$$\mathbb{P}[\,|X-\mathbb{E}\,X| \geqslant nt\,] \leqslant 2e^{-2nt^2}$$

em que $X = \sum_i X_i$. □

6.3.1 *SKIP LIST* REVISITADA

Seja S um conjunto representado por uma *skip list* de níveis $S_0, S_1, \ldots, S_H$ e tamanho $N = \sum_{i=1}^{H} |S_i|$. Provamos que $H = O(\log n)$ com alta probabilidade e agora veremos que $N = O(n)$ com alta probabilidade.

Lembremos que definimos $h(x)$, para todo $x \in S$, como o número de sorteios realizados quando da inserção de x, ou seja, $h(x)$ é a quantidade de ocorrências de x em N portanto, $h(x) \sim \text{Geom}(1/2)$ e $N = \sum_{x\in S} h(x)$.

Primeiro vamos derivar uma desigualdade do tipo Chernoff para soma de variáveis aleatórias com distribuição geométrica. Sejam $X_i \sim \text{Geom}(1/2)$, para $1 \leqslant i \leqslant n$, variáveis aleatórias independentes. Definimos

$$X := \sum_{i=1}^{n} X_i$$

que conta o número de ensaios de Bernoulli até serem obtidos n sucessos. Claramente, $\mathbb{E}\,X = 2n$. Ainda, para todo $\lambda > 0$

$$\mathbb{E}\,e^{\lambda X} = \prod_{i=1}^{n} \mathbb{E}\,e^{\lambda X_i} = \prod_{i=1}^{n}\left(\sum_{k\geqslant 1} e^{\lambda k}\mathbb{P}[X_i = k]\right) = \prod_{i=1}^{n}\left(\sum_{k\geqslant 1}\left(\frac{e^{\lambda}}{2}\right)^{k}\right).$$

Assumindo $\lambda < \ln 2$ a série converge

$$\sum_{k\geqslant 1}\left(\frac{e^{\lambda}}{2}\right)^{k} = \frac{e^{\lambda}}{2-e^{\lambda}}$$

portanto

$$\mathbb{E}\,e^{\lambda X} = \left(\frac{e^{\lambda}}{2-e^{\lambda}}\right)^{n}$$

e, usando a desigualdade de Markov (6.14), página 319, com $t = (2+a)n$ para qualquer $a > 0$

$$\mathbb{P}[X \geqslant (2+a)n] \leqslant e^{-\lambda(2+a)n}\left(\frac{e^{\lambda}}{2-e^{\lambda}}\right)^{n} = \left(\frac{e^{-(1+a)\lambda}}{2-e^{\lambda}}\right)^{n}.$$

Tomando $\lambda = \ln(1 + a/(2+a))$ o valor no lado direito acima é mínimo

$$\mathbb{P}[X \geqslant (2+a)n] \leqslant \left(1+\frac{a}{2}\right)^{n}\left(1-\frac{a}{2+2a}\right)^{(1+a)n},$$

usando que $1 - x \leqslant \exp(-x)$, para todo x,

$$\left(1-\frac{a}{2+2a}\right)^{(1+a)n} \leqslant e^{-an/2}$$

e que $1 + a/2 \leqslant \exp(a/4)$ para $a \geqslant 3$

$$\left(1+\frac{a}{2}\right)^{n} \leqslant e^{an/4},$$

finalmente, para todo $a \geqslant 3$ vale que $\mathbb{P}[X \geqslant (2+a)n] \leqslant \exp(-an/4)$.

Proposição 6.28 *Sejam $X_i \sim \text{Geom}(1/2)$, para $1 \leqslant i \leqslant n$, variáveis aleatórias independentes e $X = \sum_{i=1}^{n} X_i$*

$$\mathbb{P}[X \geqslant (1+a/2)\mathbb{E}\,X] \leqslant \exp(-an/4)$$

para todo $a \geqslant 3$. □

De volta para as *skip lists*, N é uma soma de n variáveis aleatórias Geom(1/2), logo a proposição acima com $a = 3$ nos dá

$$\mathbb{P}[N \geqslant 5n] \leqslant e^{-3n/4}$$

ou seja, com probabilidade maior que $1 - \exp(-3n/4)$ uma *skip list* que representa um conjunto com cardinalidade n tem menos que $5n$ elementos em todos os seus níveis.

Para o número de passos executados em uma busca, consideremos o percurso reverso de uma busca: a partir da posição final de uma busca em S_0, se possível suba de nível (sucesso), senão vá para o nó antecessor no mesmo nível (fracasso). O número de passos Z é o número de ensaios com probabilidade de sucesso 1/2 até completar $H = H(S)$ sucessos

$$Z = \sum_{i=0}^{H-1} Z_i \quad \text{com} \quad Z_i \sim \text{Geom}(1/2)$$

e vimos na seção 3.3.2, proposição 3.43, que $H \leqslant 2\log_2(n)$ com probabilidade pelo menos $1 - 1/n$. Usando a Lei de Probabilidade Total

$$\begin{aligned}
\mathbb{P}[\,Z > m\,] &= \mathbb{P}[\,Z > m \mid H \leqslant 2\log_2 n\,]\mathbb{P}[H \leqslant 2\log_2 n\,] + \mathbb{P}[\,Z > m \mid H > 2\log_2 n\,]\mathbb{P}[H > 2\log_2 n\,] \\
&\leqslant \mathbb{P}[\,Z > m \mid H \leqslant 2\log_2 n\,] + \mathbb{P}[H > 2\log_2 n\,] \\
&= \mathbb{P}[\,Z > m \mid H \leqslant 2\log_2 n\,] + \frac{1}{n}
\end{aligned}$$

Condicionemos a $H = \lceil \log_2 n^2 \rceil$. Sabemos da seção 3.3.2, equação (3.30) na página 163, que $\mathbb{E}\, Z_i \leqslant 2$ pois Z_i é o número de passos no nível i da *skip list*, logo $\mathbb{E}\, Z \leqslant 2H \leqslant 2\lceil \log_2 n^2 \rceil$. Então

$$\mathbb{P}[\,Z \geqslant (2+a)\lceil \log_2 n^2 \rceil\,] \leqslant \mathbb{P}[\,Z \geqslant (1+a/2)\mathbb{E}\, Z\,] \leqslant e^{-a\lceil \log_2 n^2 \rceil/4}.$$

Fazendo $a = 4$, usando que $\lceil \log_2 n^2 \rceil \geqslant \log_2(n^2)$ e que $\log_2(n^2) > \ln(n^2)$ temos

$$\mathbb{P}[\,Z \geqslant (2+a)\lceil \log_2 n^2 \rceil\,] \leqslant e^{-\ln(n^2)} = \frac{1}{n^2}$$

logo com probabilidade $2/n$ nenhuma busca usa mais que $7\log_2(n)$ passos para terminar.

Teorema 6.29 *Com probabilidade* $1 - o(1)$, *numa* skip list *que representa um conjunto de cardinalidade* n *o número de elementos é* $O(n)$ *e qualquer busca termina em* $O(\log(n))$ *passos.* □

6.3.2 *TREAPS*

Como no caso de *skip lists*, denotamos por U o conjunto universo e $S \subset U$ deve ser representado de modo a permitir as operações de dicionário: inserção, busca e remoção. Uma **treap** (Vuillemin, 1980) é árvore binária onde a cada nó v está associado $chave(v)$ e $prioridade(v)$ de modo a árvore se comporta como árvore binária de busca com relação a *chave* e uma *heap* com relação a *prioridade*. Um nó da árvore tem a propriedade *heap* se a sua prioridade é maior que de seus descendentes e tem a propriedade de árvore de busca se sua chave é maior que a chave do filho esquerdo e menor que a chave do filho direito.

Seja S um conjunto de elementos de um universo U representados por uma *treap*, as operações de dicionário são descritas a seguir.

Figura 6.1: Exemplo de *treap* com 6 elementos. As letras são as chaves e os números as prioridades.

busca — dado $x \in U$ queremos saber se $x \in S$. A busca é feita como em árvore binária de busca, a partir da raiz comparamos x com a chave do nó, se x for maior então continuamos na subárvore esquerda, senão continuamos na subárvore direita; o custo dessa operação é proporcional a altura da árvore;

inserção — dado $x \in U$ queremos inserir x em S. Fazemos uma busca por x em S e caso falhe inserimos x na folha resultante da busca na árvore com alguma escolha para a prioridade de x. Essa operação mantém a propriedade de árvore de busca binária mas destrói a propriedade de *heap*, que pode ser refeita por uma série de rotações na árvore. Por exemplo, na figura 6.2 abaixo, se x é filho de y mas tem maior prioridade então uma rotação a direita em y refaz a propriedade de *heap* entre esses nós;

remoção — dado $x \in S$ queremos remover x de S. Após uma busca em S rotacionamos a árvore no filho menos prioritário de x sucessivamente até que x seja uma folha e possa ser removido.

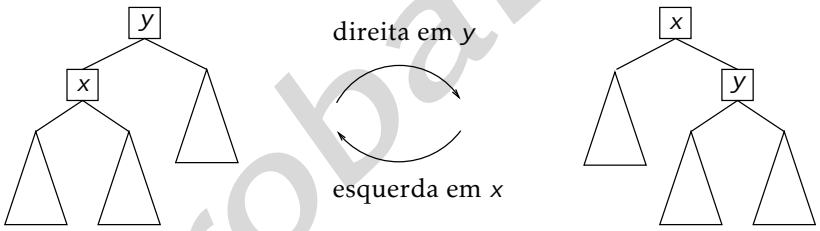


Figura 6.2: Rotações a esquerda e a direita em um nó numa árvore binária.

Essas operações têm custo proporcional a altura da árvore e usamos aleatoriedade para garantir altura logarítmica: vamos assumir que a prioridade atribuída a v quando ele é inserido em S é uma escolha aleatória no intervalo $[0,1]$. A prioridade de um nó não precisa ser um número real, basta garantir que as prioridades sejam distintas.

Seja x_k o k-ésimo menor elemento do conjunto S que está representado por uma *treap*. Vamos estimar a que distância esse nó está da raiz. Seja $\mathbb{1}_{[x_j \prec x_k]}$ variável aleatória indicadora de que x_j é ancestral próprio de x_k na *treap*, fato que denotamos por $x_j \prec x_k$. Então a profundidade de x_k é o seu número de ancestrais próprios e

$$\mathbb{E}\,\text{profundidade}(x_k) = \sum_{j=1}^{n} \mathbb{P}[\,\mathbb{1}_{[x_j \prec x_k]} = 1\,]$$

e precisamos estimar a probabilidade de $x_j \prec x_k$. Para $j < k$ defina $X(j,k) = \{x_j, x_{j+1}, \ldots, x_k\}$ e $X(k,j) = X(j,k)$.

Proposição 6.30 *Seja $j \neq k$. Então, x_j é um ancestral de x_k se e somente se x_j tem prioridade* $\min \mathsf{X}(j,k)$.

Demonstração. Suponha $j \leqslant k$ e que $\mathit{prioridade}(x_j) > \mathit{prioridade}(x_i)$ para todo $j < i \leqslant k$. Pela propriedade *heap* x_j não pode estar numa subárvore de x_k, mais ainda, não pode haver um ancestral x_i de x_k com x_j e x_k em subárvores de x_i distintas (justifique) logo x_j deve ser um ancestral de x_k se $\mathit{prioridade}(x_j) > \mathit{prioridade}(x_i)$ para todo $j < i \leqslant k$. Por outro lado, tomemos um ancestral x_j de x_k e suponhamos que existe existe algum x_i com $j < i \leqslant k$ com prioridade maior do que x_j. Pela propriedade *heap*, x_i não pode estar numa subárvore enraizada em x_j. Mas x_i não pode ser ancestral de x_j: teríamos $x_j < x_i \leqslant x_k$, assim x_j seria armazenado na subárvore esquerda de x_i e x_k não poderia estar armazenado naquela subárvore, contradizendo que x_j é um ancestral de x_k. Resta o caso que existe um ancestral x_ℓ de x_j que tem x_j em uma de suas subárvores e x_i na outra. Mas junto com $x_j < x_i \leqslant x_k$ novamente contradiz que x_j um ancestral de x_k, pois x_k estaria na mesma subárvore de x_ℓ que x_i. Portanto, se x_j é um ancestral de x_k então $\mathit{prioridade}(x_j) > \mathit{prioridade}(x_i)$ para todo $j < i \leqslant k$. □

Agora, qualquer elemento de $\mathsf{X}(j,k)$, $j \neq k$, tem a mesma probabilidade de ser o elemento de maior prioridade, logo

$$\mathbb{P}[\, \mathbb{1}_{[x_j < x_k]} = 1 \,] = \frac{1}{|k-j|+1}$$

e

$$\begin{aligned}
\mathbb{E}\ \text{profundidade}(x_k) &= \sum_{j=1}^{k} \frac{1}{k-j+1} + \sum_{j=k}^{n} \frac{1}{j-k+1} - 1 \\
&= \sum_{i=1}^{k} \frac{1}{i} + \sum_{1}^{n-k+1} \frac{1}{i} - 1 \\
&= \mathsf{H}_k + \mathsf{H}_{n-k+1} - 1
\end{aligned}$$

onde H_n denota o n-ésimo número harmônico, $\mathsf{H}_n = \sum_{i=1}^{n} 1/i = \ln n + \gamma + \Theta(n^{-1})$, onde $\gamma \approx 0{,}577$ é a constante de Euler–Mascheroni (veja Graham, Knuth e Patashnik, 1994). Logo, a profundidade de qualquer nó tem valor esperado

$$\mathbb{E}\ \text{profundidade}(x_k) = \mathsf{O}(\log n).$$

Como aplicação da desigualdade de Chernoff obtemos o seguinte limitante para a altura de uma *treap* e, consequentemente, um limitante para as operações de dicionário na *treap*.

Teorema 6.31 *A altura de uma* treap *com n elementos é* $\mathsf{O}(\log n)$ *com probabilidade* $1 - \mathsf{O}(n^{-5})$.

Demonstração. Podemos provar que a profundidade de todo nó é proporcional a $\log n$ observando o fato de que $\mathbb{1}_{[x_j < x_k]}$ é independente de $\mathbb{1}_{[x_i < x_k]}$ se $i \neq j$ e podemos aplicar (6.18) com $t = \mathrm{e}^2 - 1 > 6$ e

obtermos de $H_k + H_{n-k+1} - 1 \geqslant H_n$ que

$$\mathbb{P}[\,\text{profundidade}(x_k) > (1+t)(H_k + H_{n-k+1} - 1)\,] \leqslant \left(\frac{e^t}{(1+t)^{1+t}}\right)^{H_n} \leqslant \left(\frac{1}{e}\right)^{tH_n} \leqslant \left(\frac{1}{n}\right)^6.$$

Agora, a probabilidade de existir um nó com profundidade grande, maior que $1+t$ vezes o valor esperado, é limitada por

$$\begin{aligned}\mathbb{P}[\bigcup_k [\text{profundidade}(x_k) > (1+t)(H_k + H_{n-k+1} - 1)]\,] \leqslant \\ \sum_{k=1}^{n} \mathbb{P}[\,\text{profundidade}(x_k) > (1+t)(H_k + H_{n-k+1} - 1)\,] \quad \leqslant \left(\tfrac{1}{n}\right)^5\end{aligned}$$

portanto, não existe tal nó com probabilidade pelo menos $1 - n^{-5}$. □

6.4 MARTINGAIS

6.5 EXERCÍCIOS

Exercício 6.32. Calcule a esperança e use o exercício anterior para determinar a variância da variável aleatória $\frac{X - \mathbb{E}\,X}{\sqrt{\mathbb{V}[X]}}$.

Exercício 6.33. Considere a família $\mathcal{H}$ de funções de hashing do exemplo 6.14 e fixe $S \subset \mathbb{F}^n$. Prove que se $|S| \leqslant \sqrt{|\mathbb{F}|}$ então para uma escolha aleatória de $h \in \mathcal{H}$ a probabilidade de não ocorrer colisão para os elementos de S é pelo menos $1/2$.

1. Prove que $\mathbb{V}[X] = 0$ então $\mathbb{P}[X = \mathbb{E}\,X] = 1$.

2. Prove a **desigualdade de Jensen**: se X é uma variável aleatória real com esperança finita e f é uma função real, contínua e convexa[5] então

$$\mathbb{E}\, f(X) \geqslant f(\mathbb{E}\,X).$$

   Conclua que $\mathbb{E}\,|X| \geqslant |\,\mathbb{E}\,X\,|$ e que $\mathbb{E}\,X^2 \geqslant \mathbb{E}\,X^2$.

3. Seja Y uma variável aleatória real com $\mathbb{E}\,X > 0$. Prove que

$$\frac{(\mathbb{E}\,Y)^2}{\mathbb{E}\,(Y^2)} \leqslant \mathbb{P}[Y \neq 0] \leqslant \mathbb{E}\,Y.$$

[5] $f\colon [a,b] \to \mathbb{R}$ é dita convexa se para quaisquer $x, y \in [a,b]$ e para todo $t \in [0,1]$, tem-se que o segmento de reta que une x e y fica acima do gráfico de f entre x e y, isto é, $f(tx + (1-t)y) \leqslant tf(x) + (1-t)f(y)$.

4. Uma família $\mathcal{H} \subset \mathsf{M}^{\mathsf{U}}$ de funções de *hash* é ε-universal para $\mathsf{S} \subset \mathsf{U}$ se $\mathbb{P}[\, h(u) = h(\mathsf{s}) \,] \leqslant \varepsilon$ para todos $u \neq \mathsf{s} \in \mathsf{S}$. Prove que nesse caso deve valer $\varepsilon \geqslant \frac{1}{|\mathsf{M}|} - \frac{1}{|\mathsf{U}|}$.

5. (*Família fracamente 2-universal*) Em muitos casos é suficiente considerarmos uma família de funções $\mathcal{H}$ tais que para quaisquer $x \neq y$

$$\mathbb{P}[\, \mathsf{H}(x) = \mathsf{H}(y) \,] \leqslant \frac{1}{m}.$$

   Dê os detalhes da prova de que a família indexada pelos pares (a, b) dada no exemplo 3.20, página 132 é fracamente 2-universal.

6. (*Desaleatorização de corte grande*) Na seção 3.2.2 apresentamos um algoritmo aleatorizado que determina um corte num grafo com pelo menos metades das arestas desse grafo. Na página 156 apresentamos um versão determinística para esse algoritmo usando uma técnica baseada no cálculo de esperanças condicionais.

   Suponha que é dado um grafo $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, com $\mathsf{V} = \{0, 2, \ldots, n-1\}$. Uma função $h\colon \mathsf{V} \to \{0, 1\}$ determina o subconjunto $\mathsf{A} := \{\, i \in \mathsf{V} \colon h(i) = 1 \,\}$ o qual, por sua vez, determina o corte $\nabla(\mathsf{A})$. Prove que se se sorteamos h em uma família de *hash* fracamente 2-universal então $\mathbb{E}\,|\nabla(\mathsf{A})| \geqslant |\mathsf{E}|/2$. Com isso temos uma outra versão do algoritmo aleatorizado para corte grande da seção 3.2.2. Escreva um algoritmo aleatorizado baseado nessa ideia.

   Escreva um algoritmo (determinístico) que *desaleatoriza* o algoritmo que você escreveu usando a família fracamente 2-universal do exemplo 3.20, página 132.

7. (*Desigualdades de Chernoff*) Prove que se $\mathsf{X} \sim \mathsf{b}(n, 1/2)$ então pondo $\mu = \mathbb{E}\,\mathsf{X} = n/2$ valem

   (a) $\mathbb{P}[\, \mathsf{X} \geqslant \mu + t \,] \leqslant \exp(-2t^2/n)$, para todo $t > 0$;

   (b) $\mathbb{P}[\, \mathsf{X} \geqslant (1+t)\mu \,] \leqslant \exp(-t^2\mu)$, para todo $t > 0$;

   (c) $\mathbb{P}[\, \mathsf{X} \leqslant \mu - t \,] \leqslant \exp(-2t^2/n)$, para todo $t \in (0, \mu)$;

   (d) $\mathbb{P}[\, \mathsf{X} \leqslant (1-t)\mu \,] \leqslant \exp(-t^2\mu)$, para todo $t \in (0, 1)$.

Exercício 6.34. Em particular, para qualquer $n \in \mathbb{N}$ existe uma família 2-universal $\{0,1\}^n \to \{0,1\}^n$. Para $m < n$ podemos definir uma família 2-universal $\{0,1\}^n \to \{0,1\}^m$ fazendo para cada $h_{(a,b)}$ definida acima

$$h'_{(a,b)}(x) := h_{(a,b)}(x) \restriction_m$$

em que $h_{(a,b)}(x) \restriction_m$ denota a restrição de $h_{(a,b)}(x)$ aos primeiros m bits. ◊

8. (*Desigualdades de Chernoff*) Outra versão da desigualdade Chernoff pode ser escrita a da seguinte forma. Sejam $X_1, \ldots, X_n$ variáveis aleatórias independentes com $|X_i| \leqslant 1$ e $\mathbb{E}\, X_i = 0$; se $t > 0$ e $X = \sum_i X_i$ então

$$\mathbb{P}[X \geqslant t] < e^{\left(\frac{-2t^2}{n}\right)} \qquad \text{e} \qquad \mathbb{P}[X \leqslant -t] < e^{\left(\frac{-2t^2}{n}\right)}.$$

Prove essas desigualdades.

9. Refaça a prova de que o *quicksort* aleatorizado ordena fazendo $O(n \log n)$ comparações com alta probabilidade (pelo menos $1 - 1/n$) usando uma desigualdade de Chernoff apropriada, como foi feito na seção 6.3.1 .

10. (*Árvore binária de busca*) Uma árvore binária de busca é uma estrutura de dados para representa um conjunto S totalmente ordenado. Vamos assumir aqui que $S \subset \mathbb{Z}$. Uma árvore binária de busca é uma árvore binária com raiz e tal que a cada nó está associado um inteiro de S (e vice-versa) de modo que se $v \in S$ está em um nó então todos os nós da subárvore esquerda estão associados a elementos de S menores que v e todos os nós da subárvore direita estão associados a elementos de S maiores que v. A figura 6.3 mostra uma árvore binária de busca para $S = \{3, 8, 9, 11, 13, 17, 19\}$. Uma busca por x numa árvore binária de busca começa examinando o

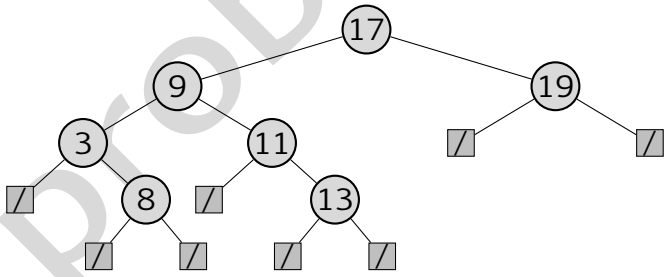


Figura 6.3: exemplo de uma árvore binária de busca.

nó raiz. Se o valor x é igual ao da raiz, a busca termina. Se o valor x é menor do que o da raiz então a busca segue pela subárvore esquerda e se o valor x é maior do que o da raiz, a busca segue pela subárvore direita. Se a (sub)árvore está vazia, o valor procurado não ocorre na árvore. Uma busca por 14 na árvore representada pela figura 6.3, por exemplo, termina na subárvore direita vazia do nó 13. Esse processo é repetido até o valor ser encontrado ou a subárvore ser vazia. O tempo de busca é, no pior caso, proporcional à altura da árvore, isto é, o maior número de arestas percorridas num caminho da raiz até alguma folha. No exemplo da figura 6.3 a altura é 3.

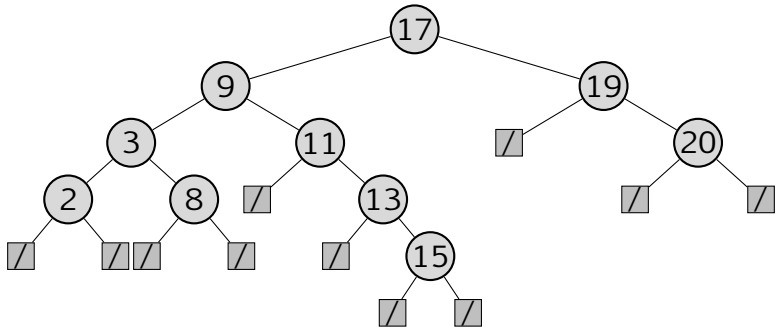


Figura 6.4: árvore de busca binária obtida a partir do exemplo acima após a inserção de 2, 15 e 20.

A inserção de x numa árvore binária de busca começa com uma busca até chegar numa subárvore vazia, é nesse local que o elemento é inserido. Por exemplo, a inserção de 2, 15 e 20 na árvore mostrada em 6.3 resulta na árvore da figura 6.4 abaixo.

A árvore da figura 6.4 pode ser obtida a partir da árvore vazia e inserindo-se os elementos de S, um a um, na seguinte ordem: $17, 9, 3, 2, 8, 11, 13, 15, 19, 20$. Qual seria a árvore resultante se os elementos fossem inseridos na ordem: $2, 3, 8, 9, 11, 13, 15, 17, 19, 20$?

Seja $S \subset \mathbb{Z}$ finito, não vazio e de cardinalidade n. Prove que se uma árvore de busca binária é construída tomando-se uma permutação aleatória dos elementos de S e inserindo-se os elementos na ordem definida pela permutação, então a altura da árvore obtida é $O(\log n)$ com probabilidade $1 - n^{-c}$, para alguma constante $c > 0$. Conclua que com alta probabilidade a árvore é construída em tempo $O(n \log n)$.

11. Dizemos que X é uma (ε, δ)-aproximação para a quantidade V se $\mathbb{P}[\,|X - V| \leqslant \varepsilon V\,] \geqslant 1 - \delta$. Sejam $X_1, \ldots, X_m$ variáveis aleatórias independentes e com mesma distribuição sobre $\{0, 1\}$ e $\mathbb{E}\, X_i = \mu$ para todo i. Denotemos por $\overline{X}_m$ a média amostral, $\overline{X}_m = (1/m)\sum_{i=1}^{m} X_i$. Prove que se $m \geqslant 3\ln(2/\delta)/(\varepsilon^2 \mu)$ então

$$\mathbb{P}[\,\left|\overline{X}_m - \mu\right| \geqslant \varepsilon\mu\,] \leqslant \delta$$

e disso segue que a média amostral é uma (ε, δ)-aproximação para μ.

12. (Karp e Luby, 1983) Suponha que sejam dados conjuntos $S_1, S_2, \ldots, S_n$ com cardinalidades conhecidas. O algoritmo a seguir estima a cardinalidade da união desses conjuntos, em que usa-

mos $\operatorname{cov}(x) := |\{i : x \in S_i\}|$ e $s := \sum_{i=1}^{n} |S_i|$.

Instância: conjuntos $S_1, S_2, \ldots, S_n$.
Resposta: uma estimativa para $S_1 \cup S_2 \cup \cdots \cup S_n$.

escolha $S \in \{S_1, S_2, \ldots, S_n\}$ aleatoriamente de acordo com $\mathbb{P}[S = S_i] = |S_i|/s$
$x \leftarrow_R S$
compute $\operatorname{cov}(x)$
$X \leftarrow \frac{s}{\operatorname{cov}(x)}$
responda X.

Algoritmo 45: Algoritmo de Karp–Luby

Prove que $\mathbb{E}\, X = \left|\bigcup_{i=1}^{n} S_i\right|$.

Sejam $X_1, X_2, \ldots, X_m$ as respostas de m rodadas independentes desse algoritmo com a mesma entrada e $\overline{X}_m$ a média amostral dessas rodadas. Prove que se $m \geqslant (9/2)(n-1)(1/\varepsilon^2)\ln(2/\delta)$ então temos uma (ε, δ)-aproximação para $\left|\bigcup_{i=1}^{n} S_i\right|$.

13. Sejam G^n um grafo d-regular e λ-expansor, e $B \subset V$ com $|B| = \beta n$. Considere $X_1 \in_R V$ e $X_1, \ldots, X_k$ passos de um passeio aleatório em G^n. Prove que para todo $\delta > 0$

$$\mathbb{P}\left[\left|\frac{\sum_{i=1}^{k} \mathbb{1}_{[X_i \in B]}}{k} - \beta\right|\right] < 2e^{-\frac{(1-\lambda)\delta^2 k}{4}}.$$

14.

Exercício 6.35. Sejam X, Y variáveis aleatórias discretas sobre Ω. A **distância estatística** entre elas é

$$\Delta(X, Y) := \max_{S \subset \Omega} \left|\mathbb{P}[X \in S] - \mathbb{P}[Y \in S]\right|.$$

Prove:

(a) $\Delta(X, Y) \geqslant 0$ com igualdade se e só se as variáveis aleatórias são identicamente distribuídas;

(b) $\Delta(X, Y) \leqslant 1$ com igualdade se e só se as variáveis aleatórias têm suporte disjuntos;

(c) $\Delta(Y, X) = \Delta(X, Y)$;

(d) $\Delta(X, Y) \leqslant \Delta(X, Z) + \Delta(Z, Y)$, para qualquer variável aleatória Z sobre Ω;

(e) $\Delta(X, Y) = \frac{1}{2} \sum_x \left|\mathbb{P}[X = x] - \mathbb{P}[Y = x]\right|$;

(f) $\Delta(f(X), f(Y)) \leqslant \Delta(X, Y)$, para qualquer função $f : \Omega \to \Omega'$.

(g) Prove que $\Delta(X, Y) \geqslant \varepsilon$ se, e somente se, existe $f : \Omega \to \{0, 1\}$ tal que $\left|\mathbb{E}\, f(X) - \mathbb{E}\, f(Y)\right| \geqslant \varepsilon$.

7 | DESALEATORIZAÇÃO

DESALEATORIZAÇÃO:

7.1 DESALEATORIZAÇÃO

7.2 CONSTRUÇÕES COMBINATÓRIAS *ADHOC*

A seguir veremos como objetos combinatórios definidos explicitamente são úteis para desaleatorização parcial ou total de algoritmos em alguns casos específicos não assumem hipoteses nao provadas

SEQUÊNCIAS COM DISTRIBUIÇÃO UNIFORME E INDEPENDÊNCIA 2-A-2.

7.2.1 SEQUÊNCIAS COM INDEPENDÊNCIA K-A-K.

DESALEATORIZAÇÃO DE MAX 3-SAT.

DESALEATORIZAÇÃO DE MAX CUT.

7.2.2 CONSTRUÇÃO VIA *HASH* UNIVERSAL

7.2.3 DESALEATORIZAÇÃO COM GRAFOS EXPANSORES

Exemplo 7.1 (Gerador de números primos). Para gerar eficientemente um primo de m bits usando $O(m)$ bits primeiro geramos m^2 números aleatórios independentes 2-a-2, cada um de m bits, usando $O(m)$ bits. Depois testamos cada um deles com o teste de Miller–Rabin (seção 2.3.2); a probabilidade de erro pode ser reduzida para 2^{-m} usando $O(m)$ bits via passeio aleatório em um grafo expansor. Pelo teorema dos números primos uma fração de $1/m$ dos números de m bits são primos. Pela desigualdade de Chebyshev concluímos que com alta probabilidade um dos m^2 números é primo e a probabilidade de erro do teste é no máximo $m^2 2^m$.

7.3 PSEUDOALEATORIEDADE

7.3.1 GERADORES PSEUDOALEATÓRIOS

EXISTÊNCIA DE FUNÇÕES PSEUDOALEATÓRIAS.

7.3.2 EXISTÊNCIA (CONDICIONAL) DE GERADORES PSEUDOALEATÓRIOS

GERADOR PSEUDOALEATÓRIO DE NISAN–WIGDERSON.

EXERCÍCIOS COMPLEMENTARES

1. Mostre que se $\mathsf{NP} \not\subset \mathsf{P/poly}$ então BPP pode ser desaleatorizado em tempo subexponencial. Não é sabido se $\mathsf{NP} \not\subset \mathsf{P/poly}$.

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 $\mathrm{e} = 2{,}71\ldots$ quando $n \to \infty$. Além disso, para todo x real

$$\mathrm{e}^x = \sum_{n\geqslant 0} \frac{x^n}{n!}.$$

A.2 DESIGUALDADES

(d.1) De (s.8), para todo $n \geqslant 0$ vale

$$\left(1-\frac{1}{n}\right)^{-n} \geqslant \mathrm{e} \geqslant \left(1+\frac{1}{n}\right)^n$$

e para qualquer $x > 0$

$$1 - x + \frac{x^2}{2} > \mathrm{e}^{-x} > (1-x).$$

(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)

$$\mathrm{e}^{nx} \geqslant (1+x)^n = \sum_{i=0}^{n} \binom{n}{i} x^i \geqslant \binom{n}{i} x^i$$

para todo $x > 0$ e todo $i \in \{0, 1, \ldots, n\}$. Logo

$$\binom{n}{i} \leqslant \mathrm{e}^{nx} x^{-i}.$$

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

A.3.1 TEOREMA DE BÉZOUT

Definimos

$$a\cdot\mathbb{Z}+b\cdot\mathbb{Z} := \{a\cdot n+b\cdot m\colon n,m\in\mathbb{Z}\}$$

o conjunto de todos os números que são *combinações lineares inteiras* de a e b, ou seja, o conjunto dos inteiros da forma $ax+by$ para algum x e algum y inteiros. Vamos provar que o menor elemento positivo desse conjunto é o mdc de a e b.

Teorema A.1 (teorema de Bézout) *Se $a,b\in\mathbb{Z}$ então existem inteiros x e y tais que*

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

INVERTÍVEIS MÓDULO N.

A.3.2 O ANEL DOS INTEIROS MÓDULO N

A.4 SOLUÇÕES DE EQUAÇÕES MÓDULO N.

Teorema A.2 (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 $n=n_1n_2\cdots n_k$.

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.3 (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;

[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. $\ker(f) = \{x \in X : f(x) = 1_Y\}$ *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.4 (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;*

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

Adleman, Leonard (1978). "Two theorems on random polynomial time". Em: *19th Annual Symposium on Foundations of Computer Science (Ann Arbor, Mich., 1978)*. Long Beach, Calif.: IEEE, pp. 75–83 (ver pp. 199, 234).

Agrawal, Manindra e Somenath Biswas (2003). "Primality and identity testing via Chinese remaindering". Em: *J. ACM* 50.4, 429–443 (electronic) (ver pp. 96, 97).

Agrawal, Manindra, Neeraj Kayal e Nitin Saxena (2004). "PRIMES is in P". Em: *Ann. of Math. (2)* 160.2, pp. 781–793 (ver pp. 85, 190).

Aleliunas, Romas et al. (1979). "Random walks, universal traversal sequences, and the complexity of maze problems". Em: *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*. SFCS '79. Washington, DC, USA: IEEE Computer Society, pp. 218–223. DOI: http://dx.doi.org/10.1109/SFCS.1979.34 (ver p. 269).

Alexander, K. S., K. Baclawski e G. C. Rota (1993). "A stochastic interpretation of the Riemann zeta function". Em: *Proceedings of the National Academy of Sciences of the United States of America* 2.90, 697–699 (ver p. 136).

Arora, Sanjeev e Boaz Barak (2009). *Computational complexity*. A modern approach. Cambridge: Cambridge University Press, pp. xxiv+579 (ver pp. 183, 206, 217, 234, 236).

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. 110).

Babai, László (1985). "Trading Group Theory for Randomness". Em: *STOC*. Ed. por Robert Sedgewick. ACM, pp. 421–429 (ver pp. 206, 210).

Bach, Eric e Jeffrey Shallit (1996). *Algorithmic Number Theory*. Cambridge, MA, USA: MIT Press (ver pp. 76, 78).

Bartle, Robert G. (1976). *The elements of real analysis*. Second. John Wiley & Sons, New York-London-Sydney, pp. xv+480 (ver p. 335).

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. 84).

Ben-Or, Michael et al. (1990). "Everything provable is provable in zero-knowledge". Em: *Advances in cryptology—CRYPTO '88 (Santa Barbara, CA, 1988)*. Vol. 403. Lecture Notes in Comput. Sci. Berlin: Springer, pp. 37–56. DOI: 10.1007/0-387-34799-2_4 (ver p. 226).

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. 98).

Billingsley, P. (1979). *Probability and measure*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley (ver p. 104).

Blum, Manuel (1986). "How to prove a theorem so no one else can claim it". Em: *Proceedings of the International Congress of Mathematicians*. Vol. II. disponível em http://www.mathunion.org/ICM/ICM1986.2/. International Mathematical Union (IMU). Berkeley, pp. 1444–1451 (ver p. 231).

Boppana, Ravi B., Johan Håstad e Stathis Zachos (1987). "Does co-NP have short interactive proofs?" Em: *Inform. Process. Lett.* 25.2, pp. 127–132. DOI: 10.1016/0020-0190(87)90232-8 (ver p. 209).

Carter, J. Lawrence e Mark N. Wegman (1979). "Universal classes of hashing functions". Em: *Journal of Computer and System Sciences* 18, pp. 143–154 (ver p. 315).

Cormen, Thomas H., Charles E. Leiserson e Ronald L. Rivest (1990). *Introduction to algorithms*. The MIT Electrical Engineering and Computer Science Series. Cambridge, MA: MIT Press, pp. xx+1028 (ver pp. 141, 144).

Coron, Jean-Sébastien e Alexander May (2007). "Deterministic polynomial-time equivalence of computing the RSA secret key and factoring". Em: *J. Cryptology* 20.1, pp. 39–50 (ver p. 225).

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. 72).

Diffie, Whitfield e Martin E. Hellman (1976). "New directions in cryptography". Em: *IEEE Trans. Information Theory* IT-22.6, pp. 644–654 (ver pp. 74, 221).

ElGamal, Taher (1985). "A public key cryptosystem and a signature scheme based on discrete logarithms". Em: *Advances in cryptology (Santa Barbara, Calif., 1984)*. Vol. 196. Lecture Notes in Comput. Sci. Berlin: Springer, pp. 10–18 (ver p. 222).

Erdős, P. (1956). "On pseudoprimes and Carmichael numbers". Em: *Publ. Math. Debrecen* 4, pp. 201–206 (ver p. 88).

Feller, William (1968). *An introduction to probability theory and its applications. Vol. I*. Third edition. New York: John Wiley & Sons Inc., pp. xviii+509 (ver pp. 119, 285, 300).

Fortnow, L. (1987). "The complexity of perfect zero-knowledge". Em: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. STOC '87. New York, New York, United States: ACM, pp. 204–209. DOI: 10.1145/28395.28418 (ver p. 226).

Freivalds, Rusins (1977). "Probabilistic Machines Can Use Less Running Time". Em: *IFIP Congress*, pp. 839–842 (ver p. 67).

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. 84).

Garfinkel, Simson (1994). *PGP: Pretty Good Privacy*. O'Reilly (ver p. 88).

Gathen, Joachim von zur e Jürgen Gerhard (2013). *Modern Computer Algebra*. 3ª ed. Cambridge University Press. DOI: 10.1017/CBO9781139856065 (ver p. 79).

Gelbaum, B.R. e J.M.H. Olmsted (1964). *Counterexamples in Analysis*. Dover books on mathematics. Holden-Day (ver p. 12).

Goldreich, O. e L. A. Levin (1989). "A hard-core predicate for all one-way functions". Em: *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*. Seattle, Washington, United States: ACM, pp. 25–32. DOI: http://doi.acm.org/10.1145/73007.73010 (ver p. 232).

Goldreich, Oded (2001). *Foundations of cryptography*. Basic tools. Cambridge: Cambridge University Press, pp. xx+372. DOI: 10.1017/CBO9780511546891 (ver p. 230).

— (2008). *Computational complexity*. A conceptual perspective. Cambridge: Cambridge University Press, pp. xxiv+606 (ver pp. 179, 180).

Goldreich, Oded, Silvio Micali e Avi Wigderson (1991). "Proofs that yield nothing but their validity, or All languages in NP have zero-knowledge proof systems". Em: *J. Assoc. Comput. Mach.* 38.3, pp. 691–729. DOI: 10.1145/116825.116852 (ver pp. 226, 230, 231, 233).

Goldwasser, S, S Micali e C Rackoff (1985). "The knowledge complexity of interactive proof-systems". Em: *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*. Providence, Rhode Island, United States: ACM Press, pp. 291–304. DOI: http://doi.acm.org/10.1145/22145.22178 (ver pp. 206, 225).

Goldwasser, S e M Sipser (1986). "Private coins versus public coins in interactive proof systems". Em: *Proceedings of the eighteenth annual ACM symposium on Theory of computing*. STOC '86. Berkeley, California, United States: ACM, pp. 59–68. DOI: 10.1145/12130.12137 (ver p. 210).

Goldwasser, Shafi e Silvio Micali (1984). "Probabilistic encryption". Em: *Journal of Computer and System Sciences* 28.2, pp. 270 –299. DOI: 10.1016/0022-0000(84)90070-9 (ver p. 213).

Goldwasser, Shafi, Silvio Micali e Charles Rackoff (1989). "The knowledge complexity of interactive proof systems". Em: *SIAM J. Comput.* 18.1, pp. 186–208. DOI: 10.1137/0218012 (ver p. 236).

Golub, Gene H. e Charles F. Van Loan (1989). *Matrix computations*. Second. Vol. 3. Johns Hopkins Series in the Mathematical Sciences. Baltimore, MD: Johns Hopkins University Press, pp. xxii+642 (ver p. 281).

Google, ed. (2011). *Por que usar o Google*. Acesso em 06/04/2022. URL: https://www.google.com/intl/pt-BR/why_use.html (ver p. 278).

Graham, Paul (2002). *A Plan for Spam*. Acesso em 06/04/2009. URL: http://www.paulgraham.com/spam.html (ver p. 31).

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 pp. 100, 327).

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. 88).

Harvey, David e Joris Van Der Hoeven (mar. de 2019). "Integer multiplication in time $O(n \log n)$". working paper or preprint (ver p. 59).

Håstad, Johan (jul. de 2001). "Some Optimal Inapproximability Results". Em: *J. ACM* 48.4, pp. 798–859. DOI: 10.1145/502090.502098 (ver p. 151).

Haveliwala, Taher e Sepandar Kamvar (2003). *The Second Eigenvalue of the Google Matrix*. Rel. técn. 20. Stanford University (ver p. 281).

Hopcroft, John E., Rajeev Motwani e Jeffrey D. Ullman (2000). *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Addison Wesley (ver p. 183).

Horn, Roger A. e Charles R. Johnson (1990). *Matrix analysis*. Corrected reprint of the 1985 original. Cambridge: Cambridge University Press, pp. xiv+561 (ver p. 259).

Håstad, Johan et al. (mar. de 1999). "A Pseudorandom Generator from Any One-way Function". Em: *SIAM J. Comput.* 28.4, pp. 1364–1396. DOI: 10.1137/S0097539793244708 (ver p. 218).

Impagliazzo, Russell e Avi Wigderson (1999). "P = BPP if E requires exponential circuits: derandomizing the XOR lemma". Em: *STOC '97 (El Paso, TX)*. New York: ACM, 220–229 (electronic) (ver p. 199).

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. 216).

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. 65).

Karp, Richard M. e Richard J. Lipton (1980). "Some connections between nonuniform and uniform complexity classes". Em: *STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing*. Los Angeles, California, United States: ACM, pp. 302–309. DOI: http://doi.acm.org/10.1145/800141.804678 (ver p. 203).

Karp, Richard M. e Michael Luby (1983). "Monte-Carlo algorithms for enumeration and reliability problems". Em: *Foundations of Computer Science, IEEE Annual Symposium on* 0, pp. 56–64. DOI: http://doi.ieeecomputersociety.org/10.1109/SFCS.1983.35 (ver p. 331).

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. 69).

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. 110).

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 pp. 67, 214).

Lautemann, Clemens (1983). "BPP and the Polynomial Hierarchy". Em: *Inf. Process. Lett.* 17.4, pp. 215–217 (ver p. 201).

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. 102).

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. 85).

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. 79, 82).

Lund, Carsten et al. (1992). "Algebraic methods for interactive proof systems". Em: *J. Assoc. Comput. Mach.* 39.4, pp. 859–868. DOI: 10.1145/146585.146605 (ver p. 209).

Maltese, George (1986). "A Simple Proof of the Fundamental Theorem of Finite Markov Chains". Em: *The American Mathematical Monthly* 93.8, pp. 629–630 (ver p. 256).

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 pp. 102, 221).

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. 90).

Mitzenmacher, Michael e Eli Upfal (2005). *Probability and computing*. Randomized algorithms and probabilistic analysis. Cambridge: Cambridge University Press, pp. xvi+352 (ver p. 68).

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. 85).

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. 45).

Nisan, Noam (1992). "Pseudorandom generators for space-bounded computation". Em: *Combinatorica* 12.4, pp. 449–461 (ver p. 318).

Norris, J. R. (1997). *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press. DOI: 10.1017/CBO9780511810633 (ver p. 246).

Page, Lawrence et al. (1998). *The PageRank Citation Ranking: Bringing Order to the Web*. Rel. técn. Stanford Digital Library Technologies Project (ver pp. 278, 281).

Papadimitriou, Christos H. (1994). *Computational complexity*. Reading, MA: Addison-Wesley Publishing Company, pp. xvi+523 (ver pp. 186, 190, 200, 203).

Prasolov, V. V. (2006). *Elements of Combinatorial and Differential Topology*. Graduate Studies in Mathematics 74. American Mathematical Society (ver p. 256).

Pugh, William (1989). "Skip lists: a probabilistic alternative to balanced trees". Em: *Algorithms and data structures (Ottawa, ON, 1989)*. Vol. 382. Lecture Notes in Comput. Sci. Berlin: Springer, pp. 437–449 (ver p. 157).

Raab, Martin e Angelika Steger (1998). ""Balls into Bins- A Simple and Tight Analysis". Em: *Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*. RANDOM '98. Berlin, Heidelberg: Springer-Verlag, pp. 159–170 (ver p. 175).

Rabin, Michael O. (1980a). "Probabilistic algorithm for testing primality". Em: *J. Number Theory* 12.1, pp. 128–138 (ver p. 90).

— (1980b). "Probabilistic algorithms in finite fields". Em: *SIAM J. Comput.* 9.2, pp. 273–280 (ver p. 83).

Reingold, Omer (2008). "Undirected connectivity in log-space". Em: *J. ACM* 55.4, Art. 17, 24 (ver p. 269).

Ribenboim, Paulo (1996). *The new book of prime number records*. New York: Springer-Verlag, pp. xxiv+541 (ver p. 101).

Rivest, R. L., A. Shamir e L. Adleman (1978). "A method for obtaining digital signatures and public-key cryptosystems". Em: *Commun. ACM* 21.2, pp. 120–126. DOI: http://doi.acm.org/10.1145/359340.359342 (ver p. 223).

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. 11).

Ross, Sheldon M. (2010). *A first course in Probability*. 8th. New Jersey: Prentice Hall (ver pp. 27, 273).

Rosser, Barkley (1941). "Explicit Bounds for Some Functions of Prime Numbers". Em: *American Journal of Mathematics* 63.1, pp. 211–232 (ver p. 198).

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. 42).

Savage, John E. (1998). *Models of computation - exploring the power of computing*. Addison-Wesley, pp. I–XXIII, 1–672 (ver pp. 178, 186).

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. 72).

Shamir, Adi (1992). "IP = PSPACE". Em: *J. Assoc. Comput. Mach.* 39.4, pp. 869–877. DOI: 10.1145/146585.146609 (ver p. 209).

Shen, A. (out. de 1992). "IP = SPACE: simplified proof". Em: *J. ACM* 39.4, pp. 878–880. DOI: 10.1145/146585.146613 (ver p. 209).

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. 71).

Sipser, Michael (1996). *Introduction to the Theory of Computation*. Course Technology (ver pp. 188, 190, 191).

Vadhan, Salil Pravin e Shafi Goldwasser (1999). "A Study of Statistical Zero-Knowledge Proofs". AAI0801528. Tese de dout. USA: Massachusetts Institute of Technology (ver p. 237).

Vuillemin, Jean (1980). "A unifying look at data structures". Em: *Commun. ACM* 23.4, pp. 229–239. DOI: http://doi.acm.org/10.1145/358841.358852 (ver p. 325).

We knew the web was big... (2008). http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html (ver p. 281).

Wills, Rebecca S. (2006). "Google's PageRank: the math behind the search engine". Em: *Math. Intelligencer* 28.4, pp. 6–11 (ver p. 281).

Yao, Andrew C. (1982). "Theory and applications of trapdoor functions". Em: *23rd annual symposium on foundations of computer science (Chicago, Ill., 1982)*. New York: IEEE, pp. 80–91 (ver p. 219).

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. 72).

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}$
```

$\mathbb{P} — \mathbb{P}[Q] — \mathbb{P}(A) — \mathbb{P}[A] — \mathbb{Q}(A) — \mathbb{Q}(\frac{A}{B}) — \mathbb{P}_1(\omega_1)$

```
 $\Prob A$ --- $\ProbE A$  --- $\Prob[Q]A$ --- $\Prob[Q]{\frac AB}$
```

$\mathbb{P}(A) — \mathbb{P}[A] — \mathbb{Q}(A) — \mathbb{Q}\left(\frac{A}{B}\right)$

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

```
$\probX A{\frac CD}$ --- $\probXE AB$ ---  $\probCX ABC$ --- $\probCXE [Q]AB{\frac CD}$ -
--
 $\probX S {\omega_1}$ ---  $\displaystyle \probX S {\omega_1}$
```

$\mathbb{P}_A(\frac{C}{D}) — \mathbb{P}_A[B] — \mathbb{P}_A(B \mid C) — \mathbb{Q}_A[B \mid \frac{C}{D}] — \mathbb{P}_S(\omega_1) — \underset{S}{\mathbb{P}}(\omega_1)$

```
$\ProbX A{\frac CD}$ --- $\ProbXE AB$ ---  $\ProbCX ABC$ --- $\ProbCXE [Q]AB{\frac CD}$
```

$\mathbb{P}_A\left(\frac{C}{D}\right) — \mathbb{P}_A[B] — \mathbb{P}_A(B \mid C) — \mathbb{Q}_A\left[B \,\middle|\, \frac{C}{D}\right]$

---

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
```

$\mathbb{V}[X]$

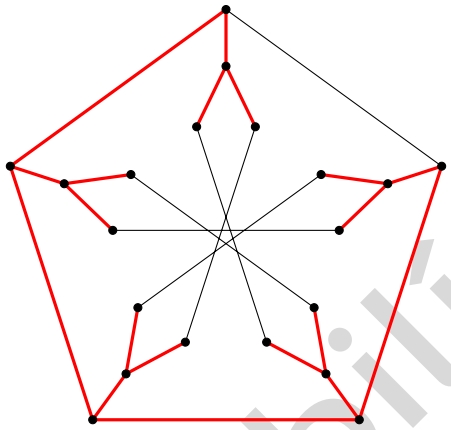


Figura A.1: exemplo de uma árvore geradora. As arestas pretas e vermelhas fazem parte do grafo, só as arestas vermelhas formam , junto com todos os vértices, um árvore geradora do grafo.