# Matemática Discreta

Dirk Hofmann

Departamento de Matemática, Universidade de Aveiro
dirk@ua.pt, http://sweet.ua.pt/dirk/aulas

**Gabinete: 11.3.10**

**Atendimento de dúvidas: Terça, 15:00 – 17:00**

# Árvores e florestas

# Árvores e florestas

## Definição

Um grafo simples $G$ diz-se uma floresta se $G$ não contém circuitos. Uma floresta conexa designa-se por árvore.

# Árvores e florestas

## Definição

Um grafo simples $G$ diz-se uma floresta se $G$ não contém circuitos.
Uma floresta conexa designa-se por árvore.

## Nota

Uma floresta é um grafo cujas componentes conexas são árvore.

---

Mais intuitiva: Uma floresta é uma coleção de árvores.

# Árvores e florestas

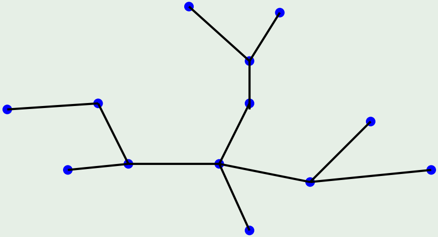
## Definição

Um grafo simples $G$ diz-se uma floresta se $G$ não contém circuitos. Uma floresta conexa designa-se por árvore.

## Nota

Uma floresta é um grafo cujas componentes conexas são árvore.

## Exemplo



# Árvores e florestas

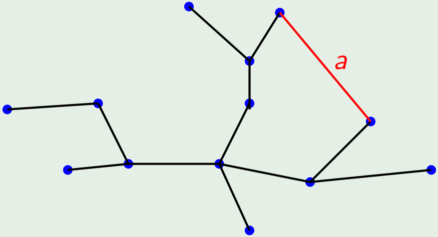
## Definição

Um grafo simples $G$ diz-se uma floresta se $G$ não contém circuitos. Uma floresta conexa designa-se por árvore.

## Nota

Uma floresta é um grafo cujas componentes conexas são árvore.

## Exemplo




Acrescentando a aresta $a$, já não é uma árvore.

**Teorema**

*Para um grafo $G$ com pelo menos um vértice, as seguintes afirmações são equivalentes.*

# Caraterização de árvores e árvores abrangentes

**Teorema**

*Para um grafo $G$ com pelo menos um vértice, as seguintes afirmações são equivalentes.*

(i) *$G$ é uma árvore.*

# Caraterização de árvores e árvores abrangentes

**Teorema**

*Para um grafo $G$ com pelo menos um vértice, as seguintes afirmações são equivalentes.*

*(i) $G$ é uma árvore.*

*(ii) Entre cada par de vértices existe um único caminho.*

# Caraterização de árvores e árvores abrangentes

**Teorema**

*Para um grafo $G$ com pelo menos um vértice, as seguintes afirmações são equivalentes.*

(i) *$G$ é uma árvore.*

(ii) *Entre cada par de vértices existe um único caminho.*

(iii) *$G$ é "minimamente conexo"; ou seja, $G$ é conexo e cada aresta é uma ponte.*

# Caraterização de árvores e árvores abrangentes

**Teorema**

*Para um grafo $G$ com pelo menos um vértice, as seguintes afirmações são equivalentes.*

(i) *$G$ é uma árvore.*

(ii) *Entre cada par de vértices existe um único caminho.*

(iii) *$G$ é "minimamente conexo"; ou seja, $G$ é conexo e cada aresta é uma ponte.*

(iv) *$G$ é "maximamente acíclico", ou seja, $G$ não contém ciclos, mas acrescentando uma aresta obtém-se um ciclo.*

# Caraterização de árvores e árvores abrangentes

### Teorema

*Para um grafo $G$ com pelo menos um vértice, as seguintes afirmações são equivalentes.*

- *(i) $G$ é uma árvore.*
- *(ii) Entre cada par de vértices existe um único caminho.*
- *(iii) $G$ é "minimamente conexo"; ou seja, $G$ é conexo e cada aresta é uma ponte.*
- *(iv) $G$ é "maximamente acíclico", ou seja, $G$ não contém ciclos, mas acrescentando uma aresta obtém-se um ciclo.*

### Definição

Seja $G$ um grafo. Um subgrafo abrangente $T$ de $G$ diz-se árvore abrangente de $G$ quando $T$ é uma árvore.

### Corolário

*Cada grafo finito conexo admite uma árvore abrangente.*

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau* 1 *(chamado folhas).*

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau* $1$ *(chamado folhas).*

**Demonstração.**

(Ver exercício 26 da folha 8.)

Considere, por exemplo, os vértices extremos do caminho mais comprido do grafo. $\square$

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau* 1 *(chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n - 1$ arestas.*

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau* 1 *(chamado* *folhas**).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n - 1$ arestas.*

**Demonstração.**

Indução sobre o número $n$ de vértices da árvore $T$.

□

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n-1$ arestas.*

**Demonstração.**

Indução sobre o número $n$ de vértices da árvore $T$.

- $n = 1$: Claro!!

$\square$

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau* 1 *(chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n-1$ arestas.*

**Demonstração.**

Indução sobre o número $n$ de vértices da árvore $T$.

- $n = 1$: Claro!!
- Seja $n \geq 2$ e suponha que a afirmação é verdadeira para todas as árvores com menos do que $n$ vértices.

$\square$

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n-1$ arestas.*

**Demonstração.**

Indução sobre o número $n$ de vértices da árvore $T$.

- $n = 1$: Claro!!
- Seja $n \geq 2$ e suponha que a afirmação é verdadeira para todas as árvores com menos do que $n$ vértices. Seja $v$ uma folha de $T$. Portanto, $T - v$ é uma árvore;

$\square$

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n-1$ arestas.*

**Demonstração.**

Indução sobre o número $n$ de vértices da árvore $T$.

- $n = 1$: Claro!!
- Seja $n \geq 2$ e suponha que a afirmação é verdadeira para todas as árvores com menos do que $n$ vértices. Seja $v$ uma folha de $T$. Portanto, $T - v$ é uma árvore; por hipótese da indução, $T - v$ tem $n - 2$ arestas.

$\square$

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n - 1$ arestas.*

**Demonstração.**

Indução sobre o número $n$ de vértices da árvore $T$.

- $n = 1$: Claro!!
- Seja $n \geq 2$ e suponha que a afirmação é verdadeira para todas as árvores com menos do que $n$ vértices. Seja $v$ uma folha de $T$. Portanto, $T - v$ é uma árvore; por hipótese da indução, $T - v$ tem $n - 2$ arestas. Logo, $T$ tem $n - 1$ arestas.

$\square$

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau* $1$ *(chamado folhas).*

**Lema**

*Uma árvore com* $n$ *vértices tem precisamente* $n-1$ *arestas.*

**Teorema**

*Um grafo* $G$ *conexo com* $n$ *vértices é uma árvore se e só se* $G$ *tem* $n-1$ *arestas.*

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n - 1$ arestas.*

**Teorema**

*Um grafo $G$ conexo com $n$ vértices é uma árvore se e só se $G$ tem $n - 1$ arestas.*

**Demonstração.**

Suponha que $G$ tem $n - 1$ arestas

$\square$

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n - 1$ arestas.*

**Teorema**

*Um grafo $G$ conexo com $n$ vértices é uma árvore se e só se $G$ tem $n - 1$ arestas.*

**Demonstração.**

Suponha que $G$ tem $n - 1$ arestas e seja $T$ uma árvore abrangente de $G$.

$\square$

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n - 1$ arestas.*

**Teorema**

*Um grafo $G$ conexo com $n$ vértices é uma árvore se e só se $G$ tem $n - 1$ arestas.*

**Demonstração.**

Suponha que $G$ tem $n - 1$ arestas e seja $T$ uma árvore abrangente de $G$. Logo, $T$ tem $n - 1$ arestas,

□

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n - 1$ arestas.*

**Teorema**

*Um grafo $G$ conexo com $n$ vértices é uma árvore se e só se $G$ tem $n - 1$ arestas.*

**Demonstração.**

Suponha que $G$ tem $n - 1$ arestas e seja $T$ uma árvore abrangente de $G$. Logo, $T$ tem $n - 1$ arestas, portanto $G = T$ é uma árvore. $\square$

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n-1$ arestas.*

**Teorema**

*Um grafo $G$ conexo com $n$ vértices é uma árvore se e só se $G$ tem $n-1$ arestas.*

**Teorema**

*Um grafo $G$ sem ciclos com $n \geq 1$ vértices é uma árvore se e só se $G$ tem $n-1$ arestas.*

# Propriedades de árvores

**Lema**

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (chamado folhas).*

**Lema**

*Uma árvore com $n$ vértices tem precisamente $n - 1$ arestas.*

**Teorema**

*Um grafo $G$ conexo com $n$ vértices é uma árvore se e só se $G$ tem $n - 1$ arestas.*

**Teorema**

*Um grafo $G$ sem ciclos com $n \geq 1$ vértices é uma árvore se e só se $G$ tem $n - 1$ arestas.*

**Demonstração.**

TPC (já não há espaço. . . ). $\square$

# Uma caraterização de florestas

**Teorema**

*Um grafo finito $G$ é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \mathrm{cc}(G).$$

# Uma caraterização de florestas

## Teorema

*Um grafo finito $G$ é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \mathrm{cc}(G).$$

## Nota

Se $G$ é uma árvore, obtemos a fórmula já conhecida:

$$\epsilon(G) = \nu(G) - 1.$$

Portanto, num grafo conexo temos

$$\epsilon(G) \geq \epsilon(\text{árvore abrangente}) = \nu(G) - 1.$$

# Uma caraterização de florestas

**Teorema**

*Um grafo finito $G$ é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \mathrm{cc}(G).$$

**Demonstração.**

$\square$

# Uma caraterização de florestas

**Teorema**

*Um grafo finito $G$ é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \mathrm{cc}(G).$$

**Demonstração.**

Suponhamos que $G$ é uma floresta e sejam $G_1, \dots, G_k$ as componentes conexas de $G$.

$\square$

## Uma caraterização de florestas

**Teorema**

*Um grafo finito $G$ é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \mathrm{cc}(G).$$

**Demonstração.**

Suponhamos que $G$ é uma floresta e sejam $G_1, \dots, G_k$ as componentes conexas de $G$. Logo, $\mathrm{cc}(G) = k$ e

$$\epsilon(G) = \epsilon(G_1) + \cdots + \epsilon(G_k) \quad \text{e} \quad \nu(G) = \nu(G_1) + \cdots + \nu(G_k).$$

□

# Uma caraterização de florestas

**Teorema**

*Um grafo finito $G$ é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \mathrm{cc}(G).$$

**Demonstração.**

Suponhamos que $G$ é uma floresta e sejam $G_1, \ldots, G_k$ as componentes conexas de $G$. Logo, $\mathrm{cc}(G) = k$ e

$$\epsilon(G) = \epsilon(G_1) + \cdots + \epsilon(G_k) \quad \text{e} \quad \nu(G) = \nu(G_1) + \cdots + \nu(G_k).$$

Para cada $i = 1, 2, \ldots, k$, $\epsilon(G_i) = \nu(G_i) - 1$ (teorema anterior para árvores),

□

# Uma caraterização de florestas

## Teorema

*Um grafo finito $G$ é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \mathrm{cc}(G).$$

## Demonstração.

Suponhamos que $G$ é uma floresta e sejam $G_1, \dots, G_k$ as componentes conexas de $G$. Logo, $\mathrm{cc}(G) = k$ e

$$\epsilon(G) = \epsilon(G_1) + \cdots + \epsilon(G_k) \quad \text{e} \quad \nu(G) = \nu(G_1) + \cdots + \nu(G_k).$$

Para cada $i = 1, 2, \dots, k$, $\epsilon(G_i) = \nu(G_i) - 1$ (teorema anterior para árvores), portanto,

$$\epsilon(G) = \nu(G) - k.$$

$\square$

# Uma caraterização de florestas

### Teorema

*Um grafo finito $G$ é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \mathrm{cc}(G).$$

### Demonstração.

Suponha agora que $\epsilon(G) - \nu(G) + \mathrm{cc}(G) = 0$ e sejam $G_1, \dots, G_k$ as componentes conexas de $G$.

$\square$

# Uma caraterização de florestas

## Teorema

*Um grafo finito G é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \mathrm{cc}(G).$$

## Demonstração.

Suponha agora que $\epsilon(G) - \nu(G) + \mathrm{cc}(G) = 0$ e sejam $G_1, \dots, G_k$ as componentes conexas de $G$. Logo,

$$0 = \underbrace{(\epsilon(G_1) - \nu(G_1) + 1)}_{\geq 0} + \cdots + \underbrace{(\epsilon(G_k) - \nu(G_k) + 1)}_{\geq 0};$$

$\square$

# Uma caraterização de florestas

## Teorema

*Um grafo finito $G$ é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \mathrm{cc}(G).$$

## Demonstração.

Suponha agora que $\epsilon(G) - \nu(G) + \mathrm{cc}(G) = 0$ e sejam $G_1, \dots, G_k$ as componentes conexas de $G$. Logo,

$$0 = \underbrace{(\epsilon(G_1) - \nu(G_1) + 1)}_{\geq 0} + \cdots + \underbrace{(\epsilon(G_k) - \nu(G_k) + 1)}_{\geq 0};$$

ou seja, $\epsilon(G_i) - \nu(G_i) + 1 = 0$, para cada $i = 1, \dots, k$.

$\square$

# Uma caraterização de florestas

## Teorema

*Um grafo finito $G$ é uma floresta se e só se*

$$\epsilon(G) = \nu(G) - \text{cc}(G).$$

## Demonstração.

Suponha agora que $\epsilon(G) - \nu(G) + \text{cc}(G) = 0$ e sejam $G_1, \dots, G_k$ as componentes conexas de $G$. Logo,

$$0 = \underbrace{(\epsilon(G_1) - \nu(G_1) + 1)}_{\geq 0} + \cdots + \underbrace{(\epsilon(G_k) - \nu(G_k) + 1)}_{\geq 0};$$

ou seja, $\epsilon(G_i) - \nu(G_i) + 1 = 0$, para cada $i = 1, \dots, k$. Pelo teorema anterior (sobre árvores), cada componente conexa é uma árvore. Portanto, $G$ é uma floresta.

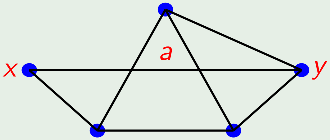
$\square$

# Contrações de arestas

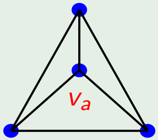
## Contracção de arestas

Seja $G = (V, E)$ um grafo simples e seja $a = xy$ uma aresta de $G$. Denotamos por $G/a$ o grafo obtido a partir de $G$ por contração da aresta $a$ num novo vértice (digamos $v_a$).

## Exemplo




$G$




$G/a$

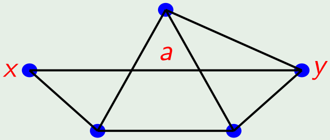
# Contrações de arestas

## Contracção de arestas

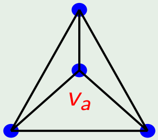
Seja $G = (V, E)$ um grafo simples e seja $a = xy$ uma aresta de $G$. Denotamos por $G/a$ o grafo obtido a partir de $G$ por contração da aresta $a$ num novo vértice (digamos $v_a$). Mais concretamente, $G/a$ é o grafo $(V', E')$ onde $V' = V \setminus \{x, y\} \cup \{v_a\}$ e

$$E' = \{vw \in E \mid \{v, w\} \cap \{x, y\} = \varnothing\} \cup \{v_a w \mid xw \in E \setminus \{a\} \quad \text{ou} \quad yw \in E \setminus \{a\}\}.$$

## Exemplo




$G$



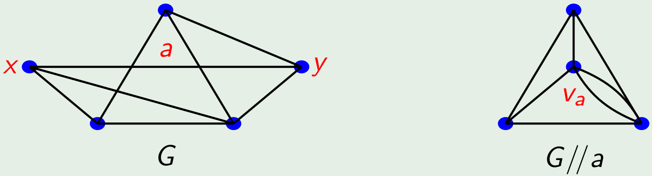

$G/a$

# Fusão de extremos de uma aresta

## Fusão de extremos de uma aresta

Seja $G = (V, E, \psi)$ um grafo e seja $a \in E$ com $\psi(a) = (x, y)$. Denotamos por $G//a$ o grafo obtido a partir de $G$ por fusão de $x$ e $y$.

## Exemplo



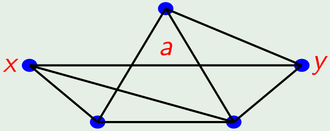
## Fusão de extremos de uma aresta

Seja $G = (V, E, \psi)$ um grafo e seja $a \in E$ com $\psi(a) = (x, y)$. Denotamos por $G/\!/a$ o grafo obtido a partir de $G$ por fusão de $x$ e $y$. Mais concretamente, $G/\!/a = (V', E', \psi')$ onde

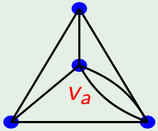
$$V' = V \setminus \{x, y\} \cup \{v_a\}, \quad E' = E \setminus \{a\}$$

e $\psi(e) = \psi'(e)$ para toda a aresta $e \in E$ com $\psi(e) \cap \{x, y\} = \varnothing$, em todos os outros casos $\psi'(e)$ é o par dado por $\psi(e)$ com $v_a$ em lugar de $x$ respetivamente $y$.

## Exemplo




$G$




$G/\!/a$

# Propriedades

**Nota**

Seja $G$ um grafo finito e seja $a$ uma aresta de $G$. Por definição,

$$\epsilon(G//a) = \epsilon(G) - 1.$$

# Propriedades

**Nota**

Seja $G$ um grafo finito e seja $a$ uma aresta de $G$. Por definição,

$$\epsilon(G//a) = \epsilon(G) - 1.$$

**Teorema**

*Seja $G$ um grafo finito e sejam $a, b$ arestas de $G$. Então,*

$$(G//a) - b = (G - b)//a,$$

*ou seja, a operação de fusão de extremos de arestas comuta com a operação de eliminação de arestas.*

# Sobre o número de árvores abrangentes

**Definição**

Para um grafo finito $G$, $\tau(G)$ denota o número de árvores abrangentes de $G$.

# Sobre o número de árvores abrangentes

**Definição**

Para um grafo finito $G$, $\tau(G)$ denota o número de árvores abrangentes de $G$.

**Alguns casos particulares**

# Sobre o número de árvores abrangentes

### Definição

Para um grafo finito $G$, $\tau(G)$ denota o número de árvores abrangentes de $G$.

### Alguns casos particulares

- $\tau(G) = 0 \iff G$ é desconexo.

# Sobre o número de árvores abrangentes

## Definição

Para um grafo finito $G$, $\tau(G)$ denota o número de árvores abrangentes de $G$.

## Alguns casos particulares

- $\tau(G) = 0 \iff G$ é desconexo.
- $\tau(G) = 1 \impliedby G$ é uma árvore.

# Sobre o número de árvores abrangentes

## Definição

Para um grafo finito $G$, $\tau(G)$ denota o número de árvores abrangentes de $G$.

## Alguns casos particulares

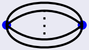
- $\tau(G) = 0 \iff G$ é desconexo.
- $\tau(G) = 1 \impliedby G$ é uma árvore.
- Se $G$ é um ciclo com $k$ arestas, então $\tau(G) = k$ (As árvores abrangentes de $G$ são da forma $G - a$).

# Sobre o número de árvores abrangentes

## Definição

Para um grafo finito $G$, $\tau(G)$ denota o número de árvores abrangentes de $G$.

## Alguns casos particulares

- $\tau(G) = 0 \iff G$ é desconexo.
- $\tau(G) = 1 \Longleftarrow G$ é uma árvore.
- Se $G$ é um ciclo com $k$ arestas, então $\tau(G) = k$
- Se $G =$  ($k$ arestas), então $\tau(G) = k$.

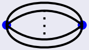
  (as árvores abrangentes de $G$ são precisamente as arestas de $G$).

# Sobre o número de árvores abrangentes

## Definição

Para um grafo finito $G$, $\tau(G)$ denota o número de árvores abrangentes de $G$.

## Alguns casos particulares

- $\tau(G) = 0 \iff G$ é desconexo.
- $\tau(G) = 1 \iff G$ é uma árvore.
- Se $G$ é um ciclo com $k$ arestas, então $\tau(G) = k$
- Se $G =$  ($k$ arestas), então $\tau(G) = k$.
- Se $G$ é um grafo que resulta de dois subgrafos $G_1$ e $G_2$ unidos por uma ponte ou por um único vértice em comum, então $\tau(G) = \tau(G_1) \cdot \tau(G_2)$

  (As árvores abrangentes de $G$ correspondem aos pares $(T_1, T_2)$ onde $T_1$ é uma árvore abrangente de $G_1$ e $T_2$ é uma árvore abrangente de $G_2$).

# Sobre o número de árvores abrangentes

**Teorema (Fórmula recursiva)**

*Seja $G$ um grafo finito e conexo seja $a \in E(G)$ uma aresta de $G$ que não é um lacete. Então,*

$$\tau(G) = \tau(G - a) + \tau(G//a).$$

# Sobre o número de árvores abrangentes

**Teorema (Fórmula recursiva)**

*Seja $G$ um grafo finito e conexo seja $a \in E(G)$ uma aresta de $G$ que não é um lacete. Então,*

$$\tau(G) = \tau(G - a) + \tau(G//a).$$

**Demonstração.**

Temos

$$\begin{aligned} \tau(G) &= |\text{as árvores sem } a\}| + |\{\text{as árvores com } a\}| \\ &= \end{aligned}$$

$\square$

# Sobre o número de árvores abrangentes

**Teorema (Fórmula recursiva)**

*Seja $G$ um grafo finito e conexo seja $a \in E(G)$ uma aresta de $G$ que não é um lacete. Então,*

$$\tau(G) = \tau(G - a) + \tau(G//a).$$

**Demonstração.**

Temos

$$\begin{aligned} \tau(G) &= |\text{as árvores sem } a\}| + |\{\text{as árvores com } a\}| \\ &= \tau(G - a) \end{aligned}$$

$\square$

# Sobre o número de árvores abrangentes

## Teorema (Fórmula recursiva)

*Seja $G$ um grafo finito e conexo seja $a \in E(G)$ uma aresta de $G$ que não é um lacete. Então,*

$$\tau(G) = \tau(G - a) + \tau(G//a).$$

## Demonstração.

Temos

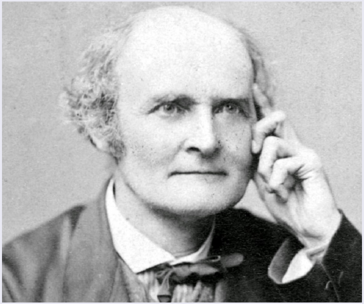
$$\begin{aligned} \tau(G) &= |\text{as árvores sem } a\}| + |\{\text{as árvores com } a\}| \\ &= \tau(G - a) + \tau(G//a). \end{aligned}$$

$\square$

# Contar árvores

**Teorema (Fórmula de Cayley)**

*Para cada $n \geq 1$, o número de árvores com $n$ vértices (etiquetadas) é $n^{n-2}$.*



Arthur Cayley (1889). «A theorem on trees». Em: *The Quarterly Journal of Mathematics* **23**, pp. 376–378.

Arthur Cayley (1821 – 1895), matemático britânico.

# Contar árvores

**Teorema (Fórmula de Cayley)**

*Para cada $n \geq 1$, o número de árvores com $n$ vértices (etiquetadas) é $n^{n-2}$.*

**Existem 1001 provas...**

Provalmente a primeira:

Carl Wilhelm Borchardt (1861). «Über eine Interpolationsformel für eine Art symmetrischer Functionen und über deren Anwendung». Em: *Math. Abh. der Akademie der Wissenschaften zu Berlin* (1–20).

# Contar árvores

**Teorema (Fórmula de Cayley)**

*Para cada $n \geq 1$, o número de árvores com $n$ vértices (etiquetadas) é $n^{n-2}$.*

**Existem 1001 provas...**

André Joyal (1981). «Une théorie combinatoire des séries formelles». Em: *Advances in Mathematics* **42**.(1), pp. 1–82.

Mais acessível: Federico G. Lastaria (2000). «An invitation to combinatorial species». URL: `http://math.unipa.it/~grim/ELastaria221-230.PDF`.

# Contar árvores

**Teorema (Fórmula de Cayley)**

*Para cada $n \geq 1$, o número de árvores com $n$ vértices (etiquetadas) é $n^{n-2}$.*

**Demonstração.**

Utilizando os *códigos de Prüfer* (já a seguir . . . ). $\square$

# Contar árvores

**Teorema (Fórmula de Cayley)**

*Para cada* $n \geq 1$*, o número de árvores com* $n$ *vértices (etiquetadas) é* $n^{n-2}$*.*

**Demonstração.**

Utilizando os *códigos de Prüfer* (já a seguir ...). $\square$

**Corolário**

*Para cada* $n \geq 1$*,* $\tau(K_n) = n^{n-2}$*.*

# Os códigos de Prüfer

## Objetivo

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos (tipicamente $V = \{1, 2, \ldots, n\}$). Estabilizemos uma bijeção entre

*o conjunto de todas as árvores* $T = (V, E)$

e

*o conjunto de todas as sequências* $(a_1, a_2, \ldots, a_{n-2})$ *de comprimento* $n - 2$ *com* $a_i \in V$.

Heinz Prüfer (1918). «Neuer Beweis eines Satzes über Permutationen». Em: *Archiv der Mathematik und Physik* **27**, pp. 742–744.

# Os códigos de Prüfer

## Objetivo

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos (tipicamente $V = \{1, 2, \ldots, n\}$). Estabilizemos uma bijeção entre

*o conjunto de todas as árvores $T = (V, E)$*

e

*o conjunto de todas as sequências $(a_1, a_2, \ldots, a_{n-2})$ de comprimento $n - 2$ com $a_i \in V$.*

A sequência $(a_1, a_2, \ldots, a_{n-2})$ associada à árvore $T$ diz-se código de Prüfer de $T$.

---

Heinz Prüfer (1918). «Neuer Beweis eines Satzes über Permutationen». Em: *Archiv der Mathematik und Physik* **27**, pp. 742–744.

# Os códigos de Prüfer

## Objetivo

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos (tipicamente $V = \{1, 2, \dots, n\}$). Estabilizemos uma bijeção entre

*o conjunto de todas as árvores $T = (V, E)$*

e

*o conjunto de todas as sequências $(a_1, a_2, \dots, a_{n-2})$ de comprimento $n - 2$ com $a_i \in V$.*

A sequência $(a_1, a_2, \dots, a_{n-2})$ associada à árvore $T$ diz-se código de Prüfer de $T$.

Consequentemente, o número de árvores $T = (V, E)$ é $n^{n-2}$.

---

Heinz Prüfer (1918). «Neuer Beweis eines Satzes über Permutationen». Em: *Archiv der Mathematik und Physik* **27**, pp. 742–744.

### A codificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos. Definimos a função

$$C_P\colon \{\text{árvores } T = (V, E)\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

de seguinte maneira.

## A codificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos. Definimos a função

$$C_P\colon \{\text{árvores } T = (V, E)\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

de seguinte maneira. Em primeiro lugar, escolhemos uma ordem total em $V$, e depois aplicamos o seguinte algoritmo:

## A codificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos. Definimos a função

$$C_P\colon \{\text{árvores } T = (V, E)\} \longrightarrow \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\}$$

de seguinte maneira. Em primeiro lugar, escolhemos uma ordem total em $V$, e depois aplicamos o seguinte algoritmo:

(1) $T =$ a árvore em consideração, $i = 1$.

# Os códigos de Prüfer

## A codificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos. Definimos a função

$$C_P\colon \{\text{árvores } T = (V, E)\} \longrightarrow \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\}$$

de seguinte maneira. Em primeiro lugar, escolhemos uma ordem total em $V$, e depois aplicamos o seguinte algoritmo:

(1) $T =$ a árvore em consideração, $i = 1$.

(2) Se $T$ tem dois (ou menos) vértices, **PARAR**.

# Os códigos de Prüfer

## A codificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos. Definimos a função

$$C_P \colon \{\text{árvores } T = (V, E)\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

de seguinte maneira. Em primeiro lugar, escolhemos uma ordem total em $V$, e depois aplicamos o seguinte algoritmo:

(1) $T =$ a árvore em consideração, $i = 1$.

(2) Se $T$ tem dois (ou menos) vértices, **PARAR**.

(3) procurar o menor vértice $v$ com grau 1 (uma folha).

# Os códigos de Prüfer

## A codificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos. Definimos a função

$$C_P\colon \{\text{árvores } T = (V, E)\} \longrightarrow \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\}$$

de seguinte maneira. Em primeiro lugar, escolhemos uma ordem total em $V$, e depois aplicamos o seguinte algoritmo:

(1) $T =$ a árvore em consideração, $i = 1$.

(2) Se $T$ tem dois (ou menos) vértices, **PARAR**.

(3) procurar o menor vértice $v$ com grau 1 (uma folha).

(4) $a_i =$ o único vizinho de $v$.

# Os códigos de Prüfer

## A codificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos. Definimos a função

$$C_P \colon \{\text{árvores } T = (V, E)\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

de seguinte maneira. Em primeiro lugar, escolhemos uma ordem total em $V$, e depois aplicamos o seguinte algoritmo:

(1) $T =$ a árvore em consideração, $i = 1$.

(2) Se $T$ tem dois (ou menos) vértices, **PARAR**.

(3) procurar o menor vértice $v$ com grau 1 (uma folha).

(4) $a_i =$ o único vizinho de $v$.

(5) $T = T - v$ (o que ainda é uma árvore!!) e $i = i + 1$.

## A codificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos. Definimos a função

$$C_P \colon \{\text{árvores } T = (V, E)\} \longrightarrow \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\}$$

de seguinte maneira. Em primeiro lugar, escolhemos uma ordem total em $V$, e depois aplicamos o seguinte algoritmo:

(1) $T =$ a árvore em consideração, $i = 1$.

(2) Se $T$ tem dois (ou menos) vértices, **PARAR**.

(3) procurar o menor vértice $v$ com grau 1 (uma folha).

(4) $a_i =$ o único vizinho de $v$.

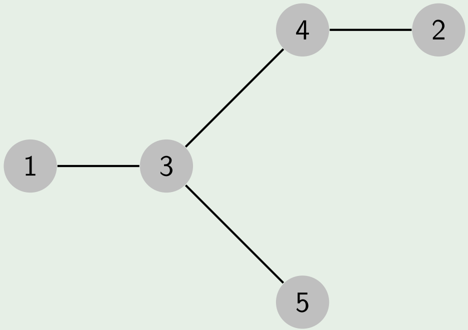
(5) $T = T - v$ (o que ainda é uma árvore!!) e $i = i + 1$.

(6) **Voltar para** (2).

# Os códigos de Prüfer

**Exemplo**

A árvore $T$:

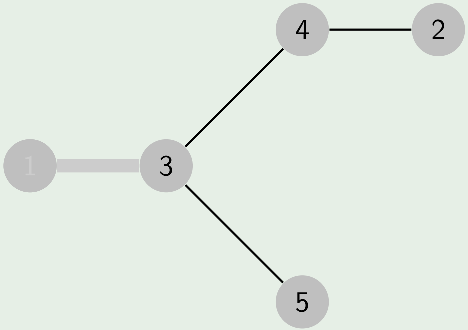



O código de Prüfer de $T$: $P = ( \qquad )$.

# Os códigos de Prüfer

## Exemplo

A árvore $T$:

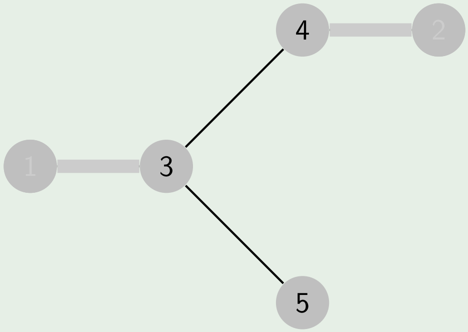



O código de Prüfer de $T$: $P = (3, \quad )$.

# Os códigos de Prüfer

## Exemplo

A árvore $T$:

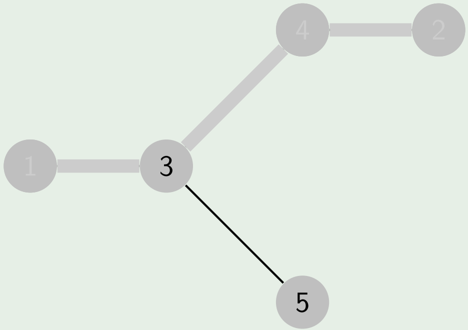



O código de Prüfer de $T$: $P = (3, 4, \;\; )$.

# Os códigos de Prüfer

## Exemplo

A árvore $T$:




O código de Prüfer de $T$: $P = (3, 4, 3)$.

## A decodificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos totalmente ordenado. Definimos a função

$$D_P \colon \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores } T = (V, E)\}$$

de seguinte maneira:

# Os códigos de Prüfer

## A decodificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos totalmente ordenado. Definimos a função

$$D_P\colon \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores } T = (V, E)\}$$

de seguinte maneira:

(1) (Desenhar os $n$ vértices no papel/quadro/areia/....)

## A decodificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos totalmente ordenado. Definimos a função

$$D_P \colon \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores } T = (V, E)\}$$

de seguinte maneira:

(1) (Desenhar os $n$ vértices no papel/quadro/areia/....)

(2) $P =$ a sequência $(a_1, \ldots, a_{n-2})$ dada, $L =$ a lista ordenada dos vértices.

## A decodificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos totalmente ordenado. Definimos a função

$$D_P\colon \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores } T = (V, E)\}$$

de seguinte maneira:

(1) (Desenhar os $n$ vértices no papel/quadro/areia/....)

(2) $P =$ a sequência $(a_1, \ldots, a_{n-2})$ dada, $L =$ a lista ordenada dos vértices.

(3) Se $L$ tem comprimento dois, então ligar os dois vértices correspondentes e **PARAR**.

# Os códigos de Prüfer

## A decodificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos totalmente ordenado. Definimos a função

$$D_P \colon \{(a_1, \dots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores } T = (V, E)\}$$

de seguinte maneira:

(1) (Desenhar os $n$ vértices no papel/quadro/areia/....)

(2) $P =$ a sequência $(a_1, \dots, a_{n-2})$ dada, $L =$ a lista ordenada dos vértices.

(3) Se $L$ tem comprimento dois, então ligar os dois vértices correspondentes e **PARAR**.

(4) Considerar o menor elemento em $L$ que não pertence a $P$, e o primeiro elemento de $P$. Ligar as dois vértices correspondentes e remover estes elementos das respetivas listas.

## A decodificação de Prüfer

Sejam $n \geq 2$ e $V$ um conjunto de $n$ elementos totalmente ordenado. Definimos a função

$$D_P \colon \{(a_1, \dots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores } T = (V, E)\}$$

de seguinte maneira:

(1) (Desenhar os $n$ vértices no papel/quadro/areia/....)

(2) $P =$ a sequência $(a_1, \dots, a_{n-2})$ dada, $L =$ a lista ordenada dos vértices.

(3) Se $L$ tem comprimento dois, então ligar os dois vértices correspondentes e **PARAR**.

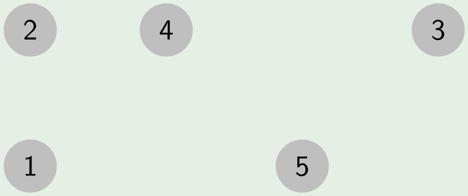
(4) Considerar o menor elemento em $L$ que não pertence a $P$, e o primeiro elemento de $P$. Ligar as dois vértices correspondentes e remover estes elementos das respetivas listas.

(5) **Voltar para** (3).

# Os códigos de Prüfer

## Exemplo

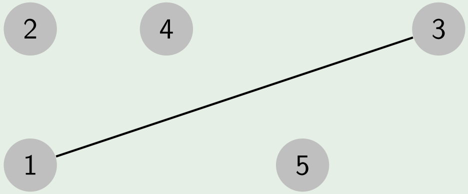
Consideramos $P = (3, 4, 3)$ e $L = (1, 2, 3, 4, 5)$.



# Os códigos de Prüfer

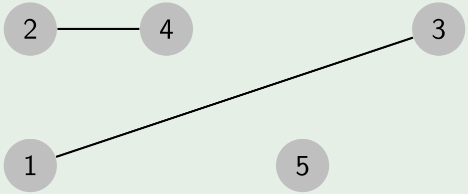
## Exemplo

Consideramos $P = (3, 4, 3)$ e $L = (1, 2, 3, 4, 5)$.



## Exemplo

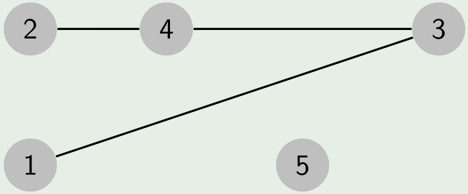
Consideramos $P = (3, 4, 3)$ e $L = (1, 2, 3, 4, 5)$.



# Os códigos de Prüfer

## Exemplo

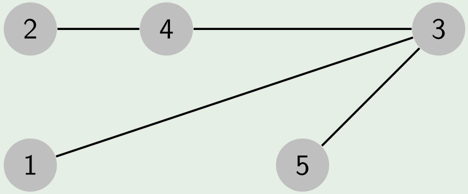
Consideramos $P = (3, 4, 3)$ e $L = (1, 2, 3, 4, 5)$.



# Os códigos de Prüfer

## Exemplo

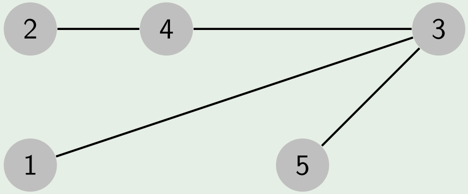
Consideramos $P = (3, 4, 3)$ e $L = (1, 2, 3, 4, 5)$.



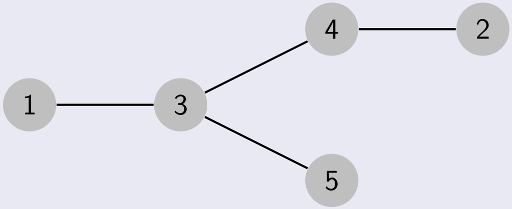
# Os códigos de Prüfer

**Exemplo**

Consideramos $P = (\textcolor{red}{3}, \textcolor{red}{4}, \textcolor{red}{3})$ e $L = (\textcolor{red}{1}, \textcolor{red}{2}, 3, \textcolor{red}{4}, 5)$.



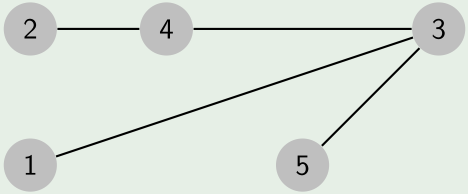

**Para comparar**



# Os códigos de Prüfer

## Exemplo

Consideramos $P = (3, 4, 3)$ e $L = (1, 2, 3, 4, 5)$.




## Nota

Verificam-se as igualdades

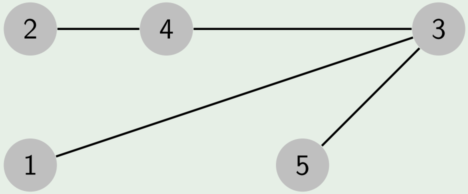
$$D_P \circ C_P = \mathrm{id} \quad \text{e} \quad C_p \circ D_P = \mathrm{id},$$

logo $D_P = C_P^{-1}$

# Os códigos de Prüfer

## Exemplo

Consideramos $P = (3, 4, 3)$ e $L = (1, 2, 3, 4, 5)$.




## Nota

Verificam-se as igualdades

$$D_P \circ C_P = \text{id} \quad \text{e} \quad C_p \circ D_P = \text{id},$$

logo $D_P = C_P^{-1}$ e por isso $C_P$ e $D_p$ são funções bijetivas.

# Árvores abrangentes de custo mínimo

# Árvores abrangentes de custo mínimo

## O contexto

Consideramos grafos finitos $G = (V, E, \psi)$ com uma função

$$W \colon E \longrightarrow [0, \infty]$$

de “custos nas arestas”.

# Árvores abrangentes de custo mínimo

## O contexto

Consideramos grafos finitos $G = (V, E, \psi)$ com uma função

$$W\colon E \longrightarrow [0, \infty]$$

de "custos nas arestas". Dada um subgrafo $H$ de $G$ (com o conjunto de arestas $E' \subseteq E$), definimos o "custo de $H$" como

$$\sum_{e \in E'} W(e).$$

# Árvores abrangentes de custo mínimo

## O contexto

Consideramos grafos finitos $G = (V, E, \psi)$ com uma função

$$W\colon E \longrightarrow [0, \infty]$$

de "custos nas arestas". Dada um subgrafo $H$ de $G$ (com o conjunto de arestas $E' \subseteq E$), definimos o "custo de $H$" como

$$\sum_{e \in E'} W(e).$$

## O objetivo

Para um grafos finito $G = (V, E, \psi)$ com $W\colon E \to [0, \infty]$, encontrar uma árvore abrangente de custo mínimo.

# Árvores abrangentes de custo mínimo

## O contexto

Consideramos grafos finitos $G = (V, E, \psi)$ com uma função

$$W\colon E \longrightarrow [0, \infty]$$

de "custos nas arestas". Dada um subgrafo $H$ de $G$ (com o conjunto de arestas $E' \subseteq E$), definimos o "custo de $H$" como

$$\sum_{e \in E'} W(e).$$

## O objetivo

Para um grafos finito $G = (V, E, \psi)$ com $W\colon E \to [0, \infty]$, encontrar uma árvore abrangente de custo mínimo.

**Convenção:** A partir de agora todos os grafos são finitos.

## Dois algoritmos

- O algoritmo de Kruskal.

  Joseph B. Kruskal (1956). «On the shortest spanning subtree of a graph and the traveling salesman problem». Em: *Proceedings of the American Mathematical Society* **7**.(1), pp. 48–50.

- O algoritmo de Prim.

  Robert C. Prim (1957). «Shortest connection networks and some generalization». Em: *Bell System Technical Journal* **36**.(6), pp. 1389–1401.

# Árvores abrangentes de custo mínimo

## Dois algoritmos

- Ver também:

  Otakar Borůvka (1926). «O jistém problému minimálním [About a minimal problem]». Em: *Práce Moravské Přírodovědecké Společnosti* **3**.(3), pp. 37–58.

  Vojtěch Jarník (1930). «O jistém problému minimálním [About a minimal problem]». Em: *Práce Moravské Přírodovědecké Společnosti* **6**.(4), pp. 57–63.

## Dois algoritmos

- Ver também:

Otakar Borůvka (1926). «O jistém problému minimálním [About a minimal problem]». Em: *Práce Moravské Přírodovědecké Společnosti* **3**.(3), pp. 37–58.

Vojtěch Jarník (1930). «O jistém problému minimálním [About a minimal problem]». Em: *Práce Moravské Přírodovědecké Společnosti* **6**.(4), pp. 57–63.

Martin Mareš (2008). «The saga of minimum spanning trees.». Em: *Computer Science Review* **2**.(3), pp. 165–221.

# A ideia geral

## Alguma notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$ e $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo.

# A ideia geral

## Alguma notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W \colon E \to [0, \infty]$ e $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo. Uma aresta $a \in E$ diz-se "segura para $E'$" quando $E' \cup \{a\}$ faz parte de uma árvore abrangente de $G$ de custo mínimo.

# A ideia geral

## Alguma notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$ e $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo. Uma aresta $a \in E$ diz-se "segura para $E'$" quando $E' \cup \{a\}$ faz parte de uma árvore abrangente de $G$ de custo mínimo.

## Descrição do algoritmo

(1) $E' = \varnothing$.

# A ideia geral

## Alguma notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$ e $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo. Uma aresta $a \in E$ diz-se "segura para $E'$" quando $E' \cup \{a\}$ faz parte de uma árvore abrangente de $G$ de custo mínimo.

## Descrição do algoritmo

(1) $E' = \varnothing$.

(2) **Enquanto** $T = (V, E')$ não é uma árvore abrangente de $G$:

# A ideia geral

## Alguma notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$ e $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo. Uma aresta $a \in E$ diz-se "segura para $E'$" quando $E' \cup \{a\}$ faz parte de uma árvore abrangente de $G$ de custo mínimo.

## Descrição do algoritmo

(1) $E' = \varnothing$.

(2) **Enquanto** $T = (V, E')$ não é uma árvore abrangente de $G$:

- Encontre uma "aresta $a \in E$ segura para $E'$".

# A ideia geral

## Alguma notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$ e $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo. Uma aresta $a \in E$ diz-se "segura para $E'$" quando $E' \cup \{a\}$ faz parte de uma árvore abrangente de $G$ de custo mínimo.

## Descrição do algoritmo

(1) $E' = \varnothing$.

(2) **Enquanto** $T = (V, E')$ não é uma árvore abrangente de $G$:

- Encontre uma "aresta $a \in E$ segura para $E'$".
- $E' = E' \cup \{a\}$.

# A ideia geral

## Alguma notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$ e $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo. Uma aresta $a \in E$ diz-se "segura para $E'$" quando $E' \cup \{a\}$ faz parte de uma árvore abrangente de $G$ de custo mínimo.

## Descrição do algoritmo

(1) $E' = \varnothing$.

(2) **Enquanto** $T = (V, E')$ não é uma árvore abrangente de $G$:

- Encontre uma "aresta $a \in E$ segura para $E'$".
- $E' = E' \cup \{a\}$.
- **Saltar para** o início de (2).

# A ideia geral

## Alguma notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$ e $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo. Uma aresta $a \in E$ diz-se "segura para $E'$" quando $E' \cup \{a\}$ faz parte de uma árvore abrangente de $G$ de custo mínimo.

## Descrição do algoritmo

(1) $E' = \varnothing$.

(2) **Enquanto** $T = (V, E')$ não é uma árvore abrangente de $G$:

- Encontre uma "aresta $a \in E$ segura para $E'$".
- $E' = E' \cup \{a\}$.
- **Saltar para** o início de (2).

(3) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# Garantir arestas seguras

## Mais notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W \colon E \to [0, \infty]$.

# Garantir arestas seguras

## Mais notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W \colon E \to [0, \infty]$.

- Um corte de $G$ é uma partição $(S, V \setminus S)$ de $V$.

# Garantir arestas seguras

## Mais notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$.

- Um corte de $G$ é uma partição $(S, V \setminus S)$ de $V$.
- Uma aresta $a \in E$ "ultrapassa o corte" quando um extremo pertence ao $S$ e o outro ao $V \setminus S$.

# Garantir arestas seguras

## Mais notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$.

- Um corte de $G$ é uma partição $(S, V \setminus S)$ de $V$.
- Uma aresta $a \in E$ "ultrapassa o corte" quando um extremo pertence ao $S$ e o outro ao $V \setminus S$.
- Um corte $(S, V \setminus S)$ "respeita" um conjunto $E' \subseteq E$ de arestas quando nenhuma aresta de $E'$ "ultrapassa o corte".

# Garantir arestas seguras

## Mais notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W \colon E \to [0, \infty]$.

- Um corte de $G$ é uma partição $(S, V \setminus S)$ de $V$.
- Uma aresta $a \in E$ "ultrapassa o corte" quando um extremo pertence ao $S$ e o outro ao $V \setminus S$.
- Um corte $(S, V \setminus S)$ "respeita" um conjunto $E' \subseteq E$ de arestas quando nenhuma aresta de $E'$ "ultrapassa o corte".
- Finalmente, $a \in E$ é uma aresta "leve ultrapassando o corte" quando $a$ ultrapassa o corte e tem custo mínimo entre todas as arestas que ultrapassam o corte.

# Garantir arestas seguras

## Mais notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$.

- Um corte de $G$ é uma partição $(S, V \setminus S)$ de $V$.
- Uma aresta $a \in E$ "ultrapassa o corte" quando um extremo pertence ao $S$ e o outro ao $V \setminus S$.
- Um corte $(S, V \setminus S)$ "respeita" um conjunto $E' \subseteq E$ de arestas quando nenhuma aresta de $E'$ "ultrapassa o corte".
- Finalmente, $a \in E$ é uma aresta "leve ultrapassando o corte" quando $a$ ultrapassa o corte e tem custo mínimo entre todas as arestas que ultrapassam o corte.

## Teorema

*Sejam $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo e $(S, V \setminus S)$ um corte de $V$ que respeita $E'$.*

# Garantir arestas seguras

## Mais notação

Seja $G = (V, E, \psi)$ um grafo conexo com $W : E \to [0, \infty]$.

- Um corte de $G$ é uma partição $(S, V \setminus S)$ de $V$.
- Uma aresta $a \in E$ "ultrapassa o corte" quando um extremo pertence ao $S$ e o outro ao $V \setminus S$.
- Um corte $(S, V \setminus S)$ "respeita" um conjunto $E' \subseteq E$ de arestas quando nenhuma aresta de $E'$ "ultrapassa o corte".
- Finalmente, $a \in E$ é uma aresta "leve ultrapassando o corte" quando $a$ ultrapassa o corte e tem custo mínimo entre todas as arestas que ultrapassam o corte.

## Teorema

*Sejam $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo e $(S, V \setminus S)$ um corte de $V$ que respeita $E'$. Se $a \in E$ é uma aresta "leve ultrapassando o corte" $(S, V \setminus S)$; então, $a$ é "segura para $E'$".*

# Garantir arestas seguras

**Teorema**

*Seja $G = (V, E, \psi)$ um grafo conexo com $W\colon E \to [0, \infty]$. Sejam $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo e $(S, V \setminus S)$ um corte de $V$ que respeita $E'$. Se $a \in E$ é uma aresta "leve ultrapassando o corte" $(S, V \setminus S)$; então, $a$ é "segura para $E'$".*

**Demonstração.**

$\square$

# Garantir arestas seguras

**Teorema**

*Seja $G = (V, E, \psi)$ um grafo conexo com $W : E \to [0, \infty]$. Sejam $E' \subseteq E$ um conjunto de arestas que faz parte de uma árvore abrangente de $G$ de custo mínimo e $(S, V \setminus S)$ um corte de $V$ que respeita $E'$. Se $a \in E$ é uma aresta "leve ultrapassando o corte" $(S, V \setminus S)$; então, $a$ é "segura para $E'$".*

**Demonstração.**

Seja $T$ uma árvore abrangente de $G$ de custo mínimo que inclui $E'$ e $\psi(a) = uv$. Se $a$ pertence à $T$, então "ganhamos".

□

# Garantir arestas seguras

## Teorema

*Seja* $G = (V, E, \psi)$ *um grafo conexo com* $W\colon E \to [0, \infty]$*. Sejam* $E' \subseteq E$ *um conjunto de arestas que faz parte de uma árvore abrangente de* $G$ *de custo mínimo e* $(S, V \setminus S)$ *um corte de* $V$ *que respeita* $E'$*. Se* $a \in E$ *é uma aresta "leve ultrapassando o corte"* $(S, V \setminus S)$*; então,* $a$ *é "segura para* $E'$*".*

## Demonstração.

Seja $T$ uma árvore abrangente de $G$ de custo mínimo que inclui $E'$ e $\psi(a) = uv$. Suponha que $a$ não pertence à $T$.

**Objetivo:** Obter uma árvore abrangente $T'$ de $G$ de custo mínimo que inclui $E' \cup \{a\}$.

□

# Garantir arestas seguras

**Teorema**

*Seja* $G = (V, E, \psi)$ *um grafo conexo com* $W: E \to [0, \infty]$. *Sejam* $E' \subseteq E$ *um conjunto de arestas que faz parte de uma árvore abrangente de* $G$ *de custo mínimo e* $(S, V \setminus S)$ *um corte de* $V$ *que respeita* $E'$. *Se* $a \in E$ *é uma aresta "leve ultrapassando o corte"* $(S, V \setminus S)$*; então,* $a$ *é "segura para* $E'$*".*

**Demonstração.**

Seja $T$ uma árvore abrangente de $G$ de custo mínimo que inclui $E'$ e $\psi(a) = uv$. Suponha que $a$ não pertence à $T$. Juntando $a$ ao caminho entre $u$ e $v$ em $T$ é um ciclo.

$\square$

# Garantir arestas seguras

**Teorema**

*Seja* $G = (V, E, \psi)$ *um grafo conexo com* $W\colon E \to [0, \infty]$. *Sejam* $E' \subseteq E$ *um conjunto de arestas que faz parte de uma árvore abrangente de* $G$ *de custo mínimo e* $(S, V \setminus S)$ *um corte de* $V$ *que respeita* $E'$. *Se* $a \in E$ *é uma aresta "leve ultrapassando o corte"* $(S, V \setminus S)$; *então,* $a$ *é "segura para* $E'$*".*

**Demonstração.**

Seja $T$ uma árvore abrangente de $G$ de custo mínimo que inclui $E'$ e $\psi(a) = uv$. Suponha que $a$ não pertence à $T$. Juntando $a$ ao caminho entre $u$ e $v$ em $T$ é um ciclo. Como a aresta $a$ "ultrapassa o corte $(S, V \setminus S)$", uma aresta do caminho entre $u$ e $v$ em $T$ também "ultrapassa o corte $(S, V \setminus S)$"; digamos $a'$ com $\psi(a') = (x, y)$.

$\square$

# Garantir arestas seguras

**Teorema**

*Seja* $G = (V, E, \psi)$ *um grafo conexo com* $W\colon E \to [0, \infty]$. *Sejam* $E' \subseteq E$ *um conjunto de arestas que faz parte de uma árvore abrangente de* $G$ *de custo mínimo e* $(S, V \setminus S)$ *um corte de* $V$ *que respeita* $E'$. *Se* $a \in E$ *é uma aresta "leve ultrapassando o corte"* $(S, V \setminus S)$; *então,* $a$ *é "segura para* $E'$*".*

**Demonstração.**

Seja $T$ uma árvore abrangente de $G$ de custo mínimo que inclui $E'$ e $\psi(a) = uv$. Suponha que $a$ não pertence à $T$. Juntando $a$ ao caminho entre $u$ e $v$ em $T$ é um ciclo. Como a aresta $a$ "ultrapassa o corte $(S, V \setminus S)$", uma aresta do caminho entre $u$ e $v$ em $T$ também "ultrapassa o corte $(S, V \setminus S)$"; digamos $a'$ com $\psi(a') = (x, y)$. Temos que $a' \notin E'$ porque o corte respeita $E'$. Portanto, $T' = T - a' + a$ é uma árvore abrangente de $G$ que inclui $E' \cup \{a\}$.

□

# Garantir arestas seguras

**Demonstração.**

Seja $T$ uma árvore abrangente de $G$ de custo mínimo que inclui $E'$ e $\psi(a) = uv$. Suponha que $a$ não pertence à $T$. Juntando $a$ ao caminho entre $u$ e $v$ em $T$ é um ciclo. Como a aresta $a$ "ultrapassa o corte $(S, V \setminus S)$", uma aresta do caminho entre $u$ e $v$ em $T$ também "ultrapassa o corte $(S, V \setminus S)$"; digamos $a'$ com $\psi(a') = (x, y)$. Temos que $a' \notin E'$ porque o corte respeita $E'$. Portanto, $T' = T - a' + a$ é uma árvore abrangente de $G$ que inclui $E' \cup \{a\}$.

Falta provar que $T - a' + a$ é de custo mínimo.

$\square$

**Demonstração.**

Seja $T$ uma árvore abrangente de $G$ de custo mínimo que inclui $E'$ e $\psi(a) = uv$. Suponha que $a$ não pertence à $T$. Juntando $a$ ao caminho entre $u$ e $v$ em $T$ é um ciclo. Como a aresta $a$ "ultrapassa o corte $(S, V \setminus S)$", uma aresta do caminho entre $u$ e $v$ em $T$ também "ultrapassa o corte $(S, V \setminus S)$"; digamos $a'$ com $\psi(a') = (x, y)$. Temos que $a' \notin E'$ porque o corte respeita $E'$. Portanto, $T' = T - a' + a$ é uma árvore abrangente de $G$ que inclui $E' \cup \{a\}$.

Falta provar que $T - a' + a$ é de custo mínimo. Como $a$ é uma aresta "leve ultrapassando o corte" e $a'$ também "ultrapassa o corte", $W(a) \leq W(a')$. Portanto,

$$W(T') = W(T) - W(a') + W(a) \leq W(T);$$

$\square$

# Garantir arestas seguras

**Demonstração.**

Seja $T$ uma árvore abrangente de $G$ de custo mínimo que inclui $E'$ e $\psi(a) = uv$. Suponha que $a$ não pertence à $T$. Juntando $a$ ao caminho entre $u$ e $v$ em $T$ é um ciclo. Como a aresta $a$ "ultrapassa o corte $(S, V \setminus S)$", uma aresta do caminho entre $u$ e $v$ em $T$ também "ultrapassa o corte $(S, V \setminus S)$"; digamos $a'$ com $\psi(a') = (x, y)$. Temos que $a' \notin E'$ porque o corte respeita $E'$. Portanto, $T' = T - a' + a$ é uma árvore abrangente de $G$ que inclui $E' \cup \{a\}$.

Falta provar que $T - a' + a$ é de custo mínimo. Como $a$ é uma aresta "leve ultrapassando o corte" e $a'$ também "ultrapassa o corte", $W(a) \leq W(a')$. Portanto,

$$W(T') = W(T) - W(a') + W(a) \leq W(T);$$

mas, como $T$ é de custo mínimo, $W(T) = W(T')$. $\square$

# O algoritmo de Kruskal

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W\colon E \to [0, \infty]$.

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Kruskal

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W \colon E \to [0, \infty]$.

(1) Ordenar as arestas $(a_1, \dots, a_m)$ de $G$ por ordem não decrescente do seu custo; ou seja,

$$W(a_1) \leq W(a_2) \leq \cdots \leq W(a_m).$$

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Kruskal

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W\colon E \to [0, \infty]$.

(1) Ordenar as arestas $(a_1, \dots, a_m)$ de $G$ por ordem não decrescente do seu custo; ou seja,

$$W(a_1) \leq W(a_2) \leq \dots \leq W(a_m).$$

(2) $E' = \varnothing$, $i = 1$.

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Kruskal

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W \colon E \to [0, \infty]$.

(1) Ordenar as arestas $(a_1, \dots, a_m)$ de $G$ por ordem não decrescente do seu custo; ou seja,

$$W(a_1) \leq W(a_2) \leq \cdots \leq W(a_m).$$

(2) $E' = \varnothing$, $i = 1$.

(3) **Enquanto** $T = (V, E')$ não é conexa:

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Kruskal

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W \colon E \to [0, \infty]$.

(1) Ordenar as arestas $(a_1, \dots, a_m)$ de $G$ por ordem não decrescente do seu custo; ou seja,

$$W(a_1) \leq W(a_2) \leq \cdots \leq W(a_m).$$

(2) $E' = \varnothing$, $i = 1$.

(3) **Enquanto** $T = (V, E')$ não é conexa:

- **Se** $(V, E' \cup \{a_i\}$ não tem ciclos, **então** $E' = E' \cup \{a_i\}$.

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Kruskal

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W \colon E \to [0, \infty]$.

(1) Ordenar as arestas $(a_1, \dots, a_m)$ de $G$ por ordem não decrescente do seu custo; ou seja,

$$W(a_1) \leq W(a_2) \leq \cdots \leq W(a_m).$$

(2) $E' = \varnothing$, $i = 1$.

(3) **Enquanto** $T = (V, E')$ não é conexa:

- **Se** $(V, E' \cup \{a_i\}$ não tem ciclos, **então** $E' = E' \cup \{a_i\}$.
- $i = i + 1$.
- **Saltar para** o início de (3).

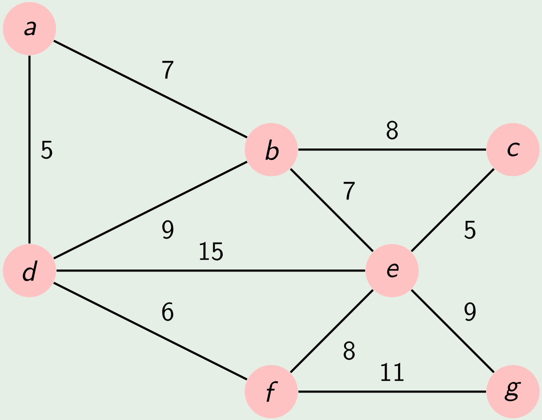
(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Kruskal

## Exemplo

Ordenar as arestas:

ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

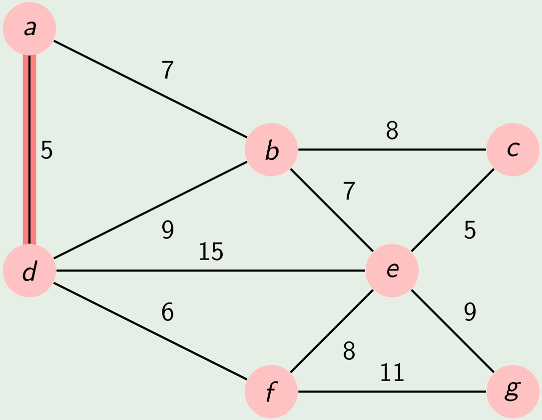


# O algoritmo de Kruskal

## Exemplo

Ordenar as arestas:

ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

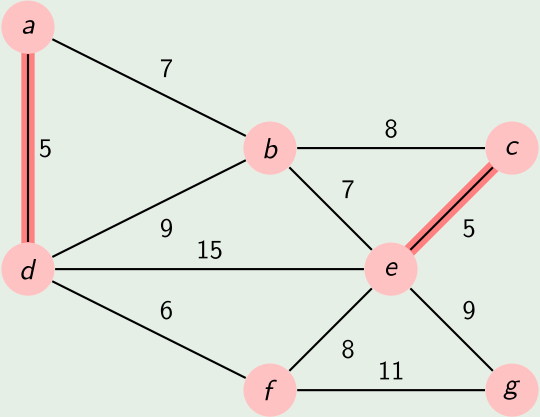


# O algoritmo de Kruskal

## Exemplo

Ordenar as arestas:

ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

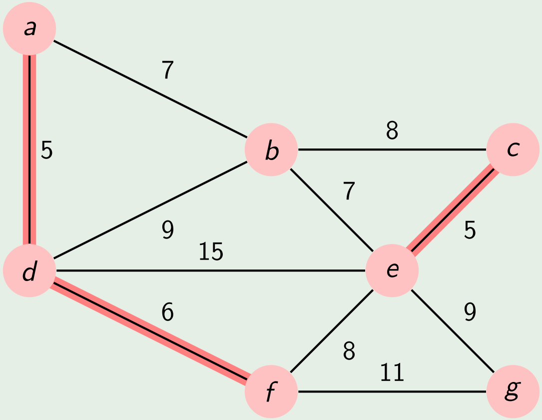


# O algoritmo de Kruskal

## Exemplo

Ordenar as arestas:

ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

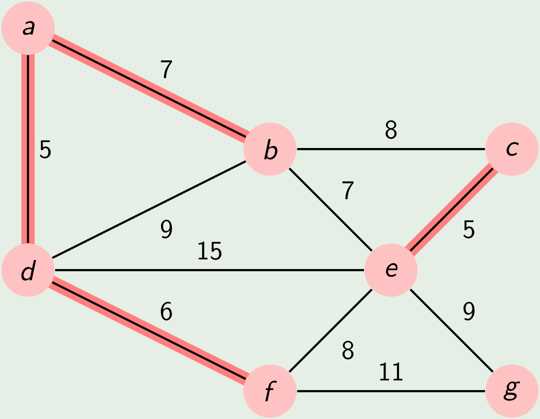


# O algoritmo de Kruskal

## Exemplo

Ordenar as arestas:

ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

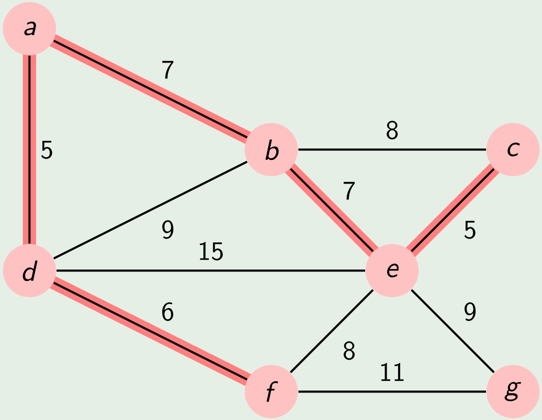


# O algoritmo de Kruskal

## Exemplo

Ordenar as arestas:

ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

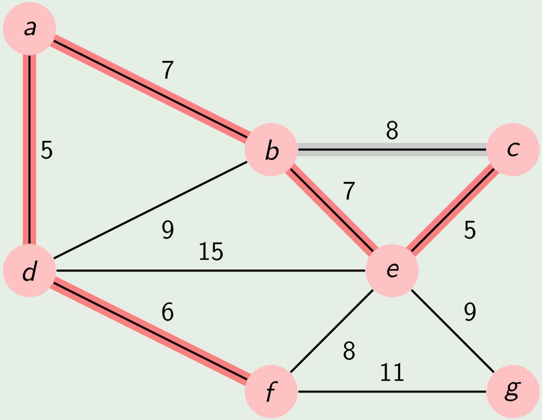


# O algoritmo de Kruskal

## Exemplo

Ordenar as arestas:

ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

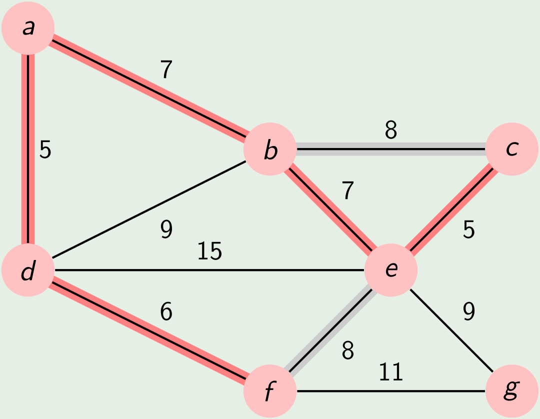


# O algoritmo de Kruskal

## Exemplo

Ordenar as arestas:

ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

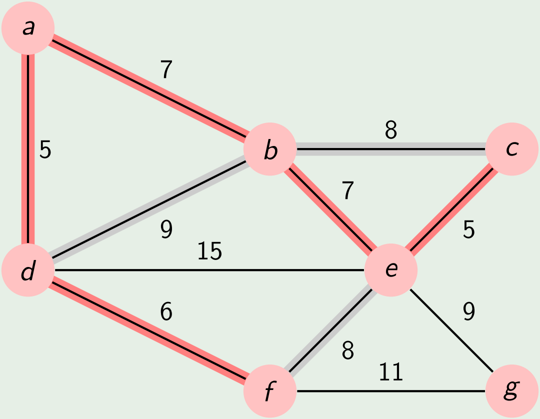


# O algoritmo de Kruskal

## Exemplo

Ordenar as arestas:

ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

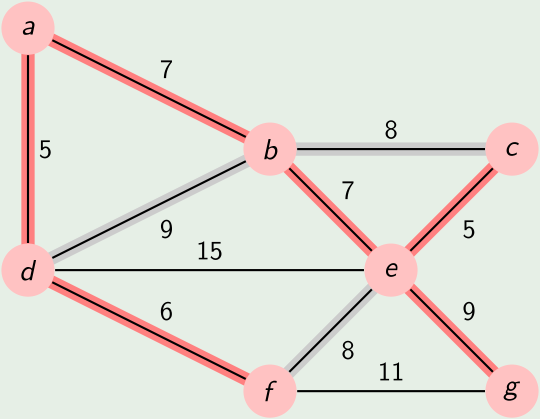


# O algoritmo de Kruskal

## Exemplo

Ordenar as arestas:

ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.



# O algoritmo de Prim

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W \colon E \to [0, \infty]$.

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Prim

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W\colon E \to [0, \infty]$.

(1) Escolher um vértice $u \in V$.

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Prim

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W\colon E \to [0, \infty]$.

(1) Escolher um vértice $u \in V$.

(2) $V' = \{u\}$ e $E' = \varnothing$.

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Prim

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W\colon E \to [0, \infty]$.

(1) Escolher um vértice $u \in V$.

(2) $V' = \{u\}$ e $E' = \varnothing$.

(3) **Enquanto** $V' \subsetneq V$:

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Prim

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W\colon E \to [0, \infty]$.

(1) Escolher um vértice $u \in V$.

(2) $V' = \{u\}$ e $E' = \varnothing$.

(3) **Enquanto** $V' \subsetneq V$:

- Entre todas as arestas $e \in E$ com
$$\psi(e) = vw, \quad v \in V', \quad w \notin V',$$
determinar uma aresta de menor custo: $e^*$ com $\psi(e^*) = v^*w^*$, $v^* \in V'$ e $w^* \notin V'$.

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Prim

## Descrição do algoritmo

Consideramos um grafo conexo $G = (V, E, \psi)$ e $W \colon E \to [0, \infty]$.

(1) Escolher um vértice $u \in V$.

(2) $V' = \{u\}$ e $E' = \varnothing$.

(3) **Enquanto** $V' \subsetneqq V$:

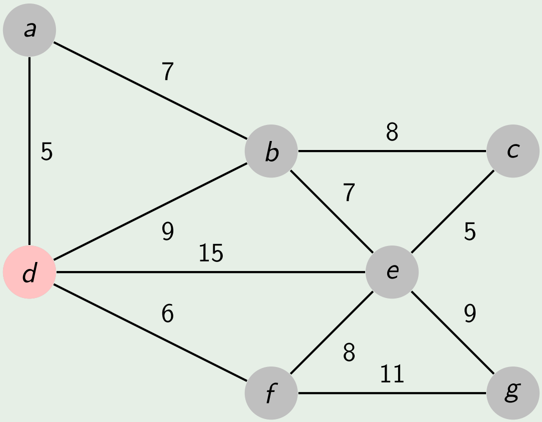
- Entre todas as arestas $e \in E$ com
$$\psi(e) = vw, \quad v \in V', \quad w \notin V',$$
determinar uma aresta de menor custo: $e^*$ com $\psi(e^*) = v^* w^*$, $v^* \in V'$ e $w^* \notin V'$.
- $V' = V' \cup \{w^*\}$, $E' = E' \cup \{e^*\}$.
- **Saltar para** o início de (3).

(4) Devolver a árvore abrangente $(V, E')$ de $G$ de custo mínimo.

# O algoritmo de Prim

## Exemplo

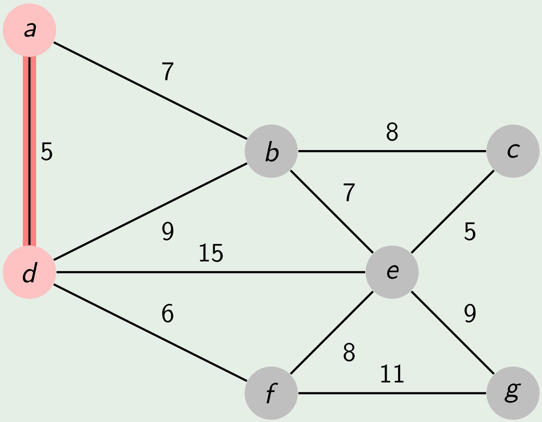
Escolhemos o vértice $d$.



# O algoritmo de Prim

## Exemplo

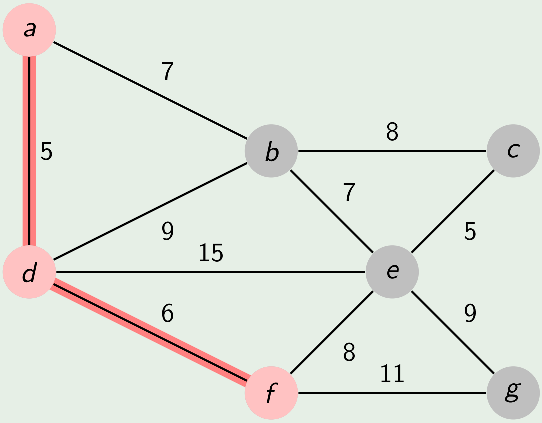
Escolhemos o vértice *d*.



# O algoritmo de Prim

## Exemplo

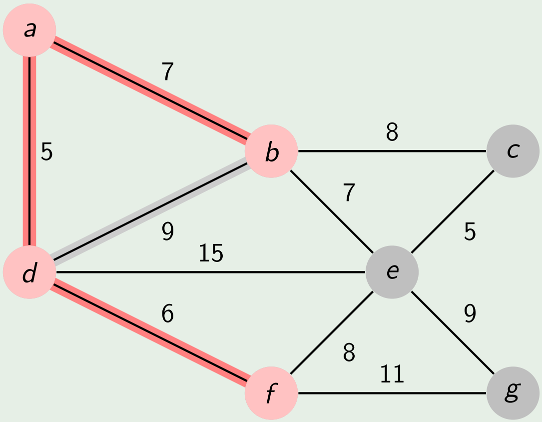
Escolhemos o vértice $d$.



# O algoritmo de Prim

## Exemplo

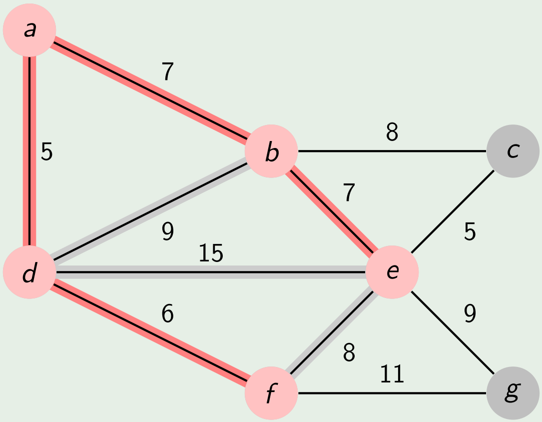
Escolhemos o vértice *d*.



# O algoritmo de Prim

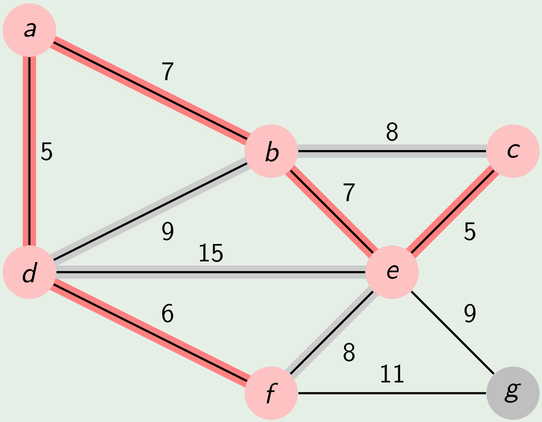
## Exemplo

Escolhemos o vértice $d$.



## Exemplo

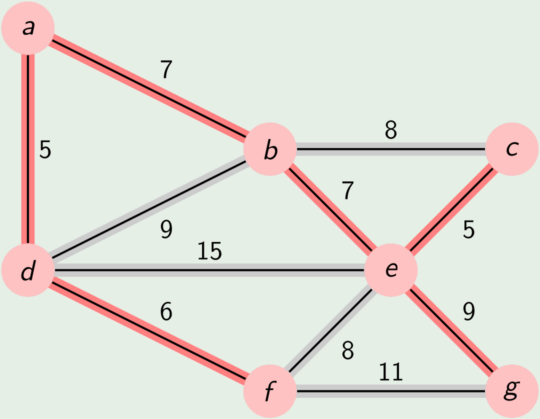
Escolhemos o vértice $d$.



# O algoritmo de Prim

## Exemplo

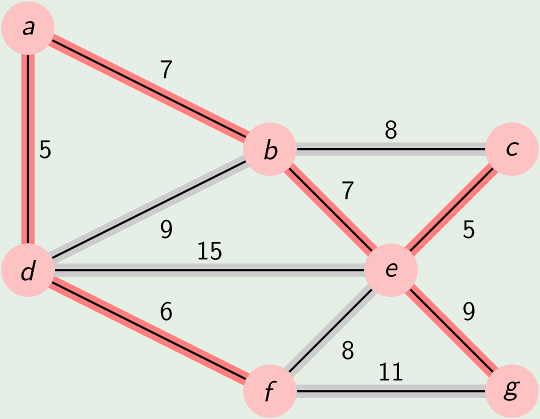
Escolhemos o vértice *d*.



# O algoritmo de Prim

## Exemplo

Escolhemos o vértice $d$.




Grafos em LaTeX e `tikz`:
`http://www.texample.net/tikz/examples/prims-algorithm/`