Matemática Discreta

Ano Letivo 2024/2025 (Versão: 27 de Maio de 2025)

Departamento de Matemática, Universidade de Aveiro
https://elearning.ua.pt/

Capítulo V
Elementos de Teoria dos Grafos

Parte III
Árvores e florestas

1. Árvores e florestas

Definição

Um grafo simples G diz-se uma **floresta** se G não contém ciclos[a]. Uma floresta conexa designa-se por **árvore**.

[a] Equivalentemente: não contém circuitos.

Definição

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

Nota

Uma floresta é um grafo simples cujas componentes conexas são árvores.

Mais intuitiva: Uma floresta é uma coleção de árvores.

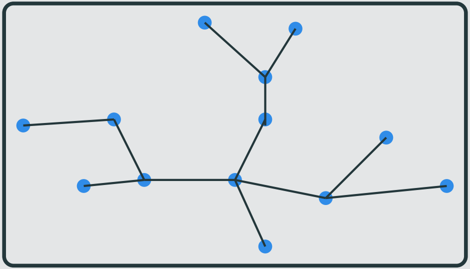
Definição

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

Nota

Uma floresta é um grafo simples cujas componentes conexas são árvores.

Exemplo (Árvore)



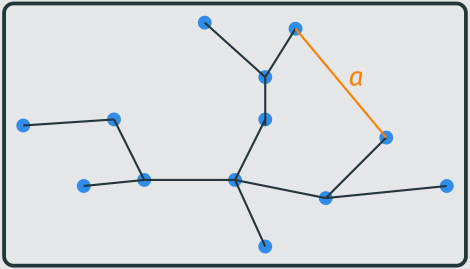
Definição

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

Nota

Uma floresta é um grafo simples cujas componentes conexas são árvores.

Exemplo (Árvore)



Acrescentando a aresta a, o grafo já não é uma árvore.

Teorema

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

(i) *G é uma árvore.*

Teorema

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

(i) *G é uma árvore.*

(ii) *G não tem lacetes e entre cada par de vértices em G existe um único caminho.*

Nota*: Em particular, G é simples.*

Teorema

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

(i) *G é uma árvore.*

(ii) *G não tem lacetes e entre cada par de vértices em G existe um único caminho.*

(iii) *G é «minimamente conexo», ou seja, G é conexo e cada aresta é uma ponte.*

Teorema

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

(i) *G é uma árvore.*

(ii) *G não tem lacetes e entre cada par de vértices em G 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.*

Teorema

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

(i) *G é uma árvore.*

(ii) *G é «minimamente conexo», ou seja, G é conexo e cada aresta é uma ponte.*

(iii) *G é «maximamente acíclico», ou seja, G não contém ciclos, mas acrescentando uma aresta obtém-se um ciclo.*

Teorema

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

(i) *G é uma árvore.*

(ii) *G é «minimamente conexo», ou seja, G é conexo e cada aresta é uma ponte.*

(iii) *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.

Teorema

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

(i) *G é uma árvore.*

(ii) *G é «minimamente conexo», ou seja, G é conexo e cada aresta é uma ponte.*

(iii) *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. (Por exemplo, podemos escolher um subgrafo «maximamente acíclico».)

Propriedades de árvores (6)

Lema

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

Lema

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

Demonstração.

(Ver o exercício 26 da folha 5.)

Considere, por exemplo, os vértices extremos do caminho mais comprido do grafo. □

Lema

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

Lema

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

Lema

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

Lema

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

Demonstração.

Indução sobre o número n de vértices da árvore T.

$\square$

Lema

Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 *(designados por* ***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!!

$\square$

Lema

Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (designados por ***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.

□

Lema

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (designados por **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;

$\square$

Lema

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (designados por **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.

$\square$

Lema

Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (designados por ***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.

$\square$

Lema

Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 *(designados por* ***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.*

Lema

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (designados por **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

$\square$

Lema

Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (designados por ***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.

$\square$

Lema

Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 *(designados por* ***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, $\square$

Lema

Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 *(designados por **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. $\square$

Lema

Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 *(designados por* ***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.

Lema

*Cada árvore finita com pelo menos dois vértices tem pelo menos dois vértices de grau 1 (designados por **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 ... mas ver seguinte teorema). $\square$

Teorema

Um grafo finito G é uma floresta se e só se

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

Teorema

Um grafo finito G é uma floresta se e só se

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

Nota

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

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

Portanto, num grafo conexo temos

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

Teorema

Um grafo finito G é uma floresta se e só se

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

Demonstração.

$\square$

Teorema

Um grafo finito G é uma floresta se e só se

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

Demonstração.

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

$\square$

Teorema

Um grafo finito G é uma floresta se e só se

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

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

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

□

Teorema

Um grafo finito G é uma floresta se e só se

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

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

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

Para cada $i = 1, 2, \ldots, k$, $\varepsilon(G_i) = \nu(G_i) - 1$ (o lema anterior para árvores),

$\square$

Teorema

Um grafo finito G é uma floresta se e só se

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

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

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

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

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

$\square$

Teorema

Um grafo finito G é uma floresta se e só se

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

Demonstração.

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

$\square$

Teorema

Um grafo finito G é uma floresta se e só se

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

Demonstração.

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

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

□

Teorema

Um grafo finito G é uma floresta se e só se

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

Demonstração.

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

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

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

$\square$

Teorema

Um grafo finito G é uma floresta se e só se

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

Demonstração.

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

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

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

$\square$

Definição

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

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.

Definição

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

Alguns casos particulares

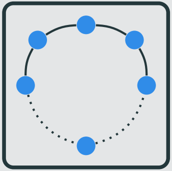
- $\tau(G) = 0 \iff G$ é desconexo.
- $\tau(G) = 1 \impliedby G$ é uma árvore.

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 $\tau(G) = k$



As árvores abrangentes de G são da forma $G - a$.

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 paralelas), então $\tau(G) = k$.

  As árvores abrangentes de G são precisamente as arestas de G.

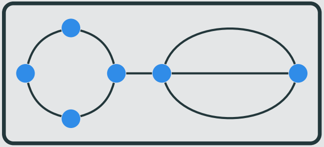
O NÚMERO DE ÁRVORES ABRANGENTES (8)

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 paralelas), então $\tau(G) = k$.
- Se G é constituído por dois subgrafos G_1 e G_2 unidos por uma ponte ou por um único vértice em comum, então $\tau(G) = \tau(G_1)\,\tau(G_2)$.



O NÚMERO DE ÁRVORES ABRANGENTES (8)

Definição

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

Alguns casos particulares

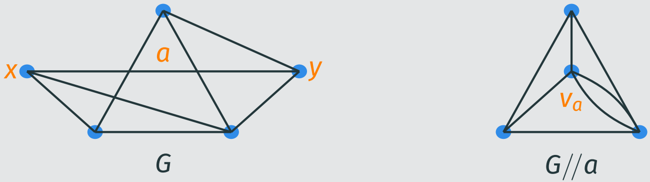
- $\tau(G) = 0 \iff G$ é desconexo.
- $\tau(G) = 1 \impliedby G$ é uma árvore.
- Se G é um ciclo com k arestas, então $\tau(G) = k$
- Se $G =$  (k arestas paralelas), então $\tau(G) = k$.
- Se G é constituído por dois subgrafos G_1 e G_2 unidos por uma ponte ou por um único vértice em comum, então $\tau(G)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

  De facto, 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.

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



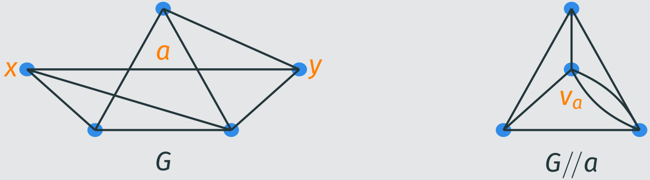
Fusão de extremos de uma aresta

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 dado por $\psi(e)$ com v_a no lugar de x e y (que se fundem no vértice v_a).

Exemplo



Nota

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

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

Nota

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

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

Teorema

Seja G um grafo finito e sejam a, b arestas distintas 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.

Teorema

Seja G um grafo finito e conexo, e a uma aresta de G que não é um lacete. Então,

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

Teorema

Seja G um grafo finito e conexo, e a uma aresta de G que não é um lacete. Então,

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

Demonstração.

Temos

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

$\square$

Teorema

Seja G um grafo finito e conexo, e a uma aresta de G que não é um lacete. Então,

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

Demonstração.

Temos

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

□

Teorema

Seja G um grafo finito e conexo, e a uma aresta de G que não é um lacete. Então,

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

Demonstração.

Temos

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

□

Teorema

Seja G um grafo finito e conexo, e a uma aresta de G que não é um lacete. Então,

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

Demonstração.

Temos

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

□

Nota

- Se a é um lacete em G, então $\tau(G) = \tau(G - a)$.

Uma Fórmula recursiva para o cálculo de $\tau(G)$

Teorema

Seja G um grafo finito e conexo, e a uma aresta de G que não é um lacete. Então,

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

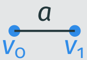
Demonstração.

Temos

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

□

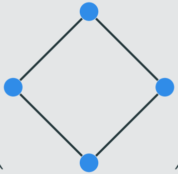
Nota

- Se a é um lacete em G, então $\tau(G) = \tau(G - a)$.
- Para 

em G com $d(v_1) = 1$: $\tau(G) = \tau(G - v_1)$.

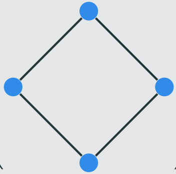
Exemplos

Não escrevemos aqui os «nomes» dos vértices.

$\tau \Bigg($  $\Bigg) =$

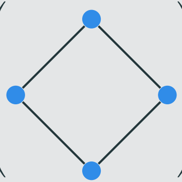
Exemplos

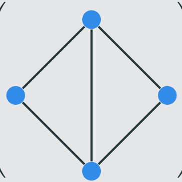
Não escrevemos aqui os «nomes» dos vértices.

τ () $= 4.$

Exemplos

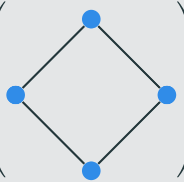
Não escrevemos aqui os «nomes» dos vértices.

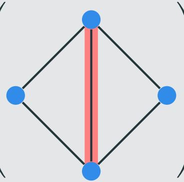
$\tau\Big($  $\Big) = 4.$

$\tau\Big($  $\Big) =$

Exemplos

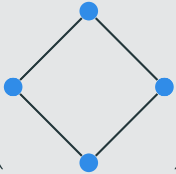
Não escrevemos aqui os «nomes» dos vértices.

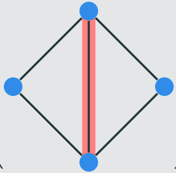
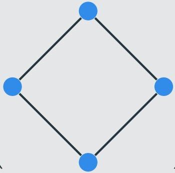
τ () $= 4.$

τ () $=$

Exemplos

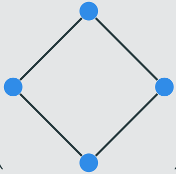
Não escrevemos aqui os «nomes» dos vértices.

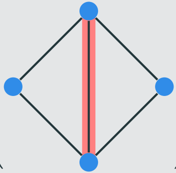
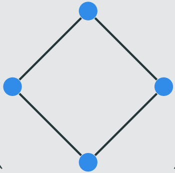
τ () $= 4.$

τ () $= \tau$ () $+$

Exemplos

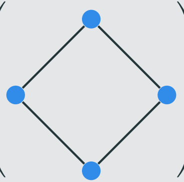
Não escrevemos aqui os «nomes» dos vértices.

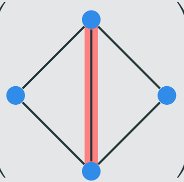
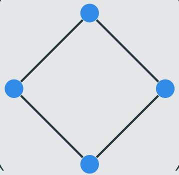
$\tau\Bigg($  $\Bigg) = 4.$

$\tau\Bigg($  $\Bigg) = \tau\Bigg($  $\Bigg) + \tau\big($  $\big)$

Exemplos

Não escrevemos aqui os «nomes» dos vértices.

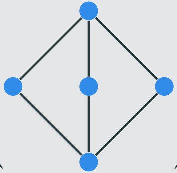
τ () $= 4.$

τ () $= \tau$ () $+\ \tau$ ()

$$= 4 + 2 \cdot 2 = 8.$$

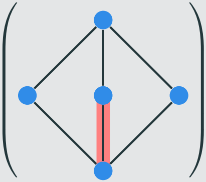
Exemplos

Não escrevemos aqui os «nomes» dos vértices.

$\tau\Bigg($  $\Bigg) =$

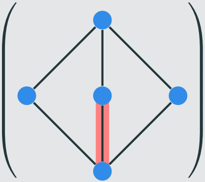
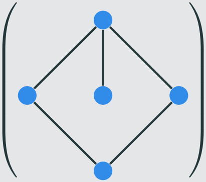
Exemplos

Não escrevemos aqui os «nomes» dos vértices.

τ  $=$

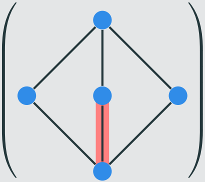
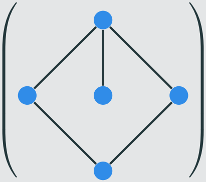
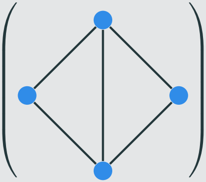
Exemplos

Não escrevemos aqui os «nomes» dos vértices.

τ  $= \tau$  $+$

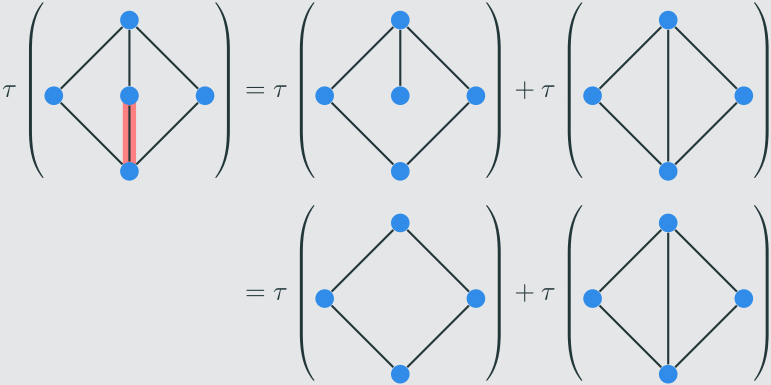
Exemplos

Não escrevemos aqui os «nomes» dos vértices.

τ  $= \tau$  $+ \tau$ 

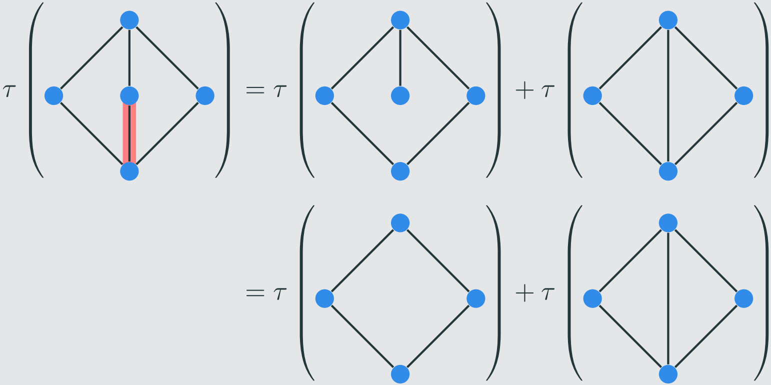
Exemplos

Não escrevemos aqui os «nomes» dos vértices.



Exemplos

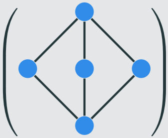
Não escrevemos aqui os «nomes» dos vértices.



$$= 4 + 8 = 12.$$

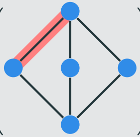
Exemplos

Não escrevemos aqui os «nomes» dos vértices.

τ  $=$

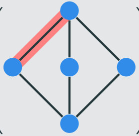
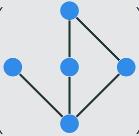
Exemplos

Não escrevemos aqui os «nomes» dos vértices.

$\tau\Bigg($  $\Bigg) =$

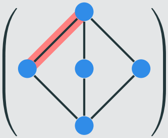
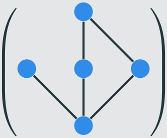
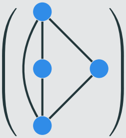
Exemplos

Não escrevemos aqui os «nomes» dos vértices.

τ () $= \tau$ () $+$

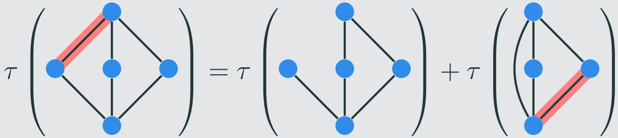
Exemplos

Não escrevemos aqui os «nomes» dos vértices.

τ  $= \tau$  $+ \tau$ 

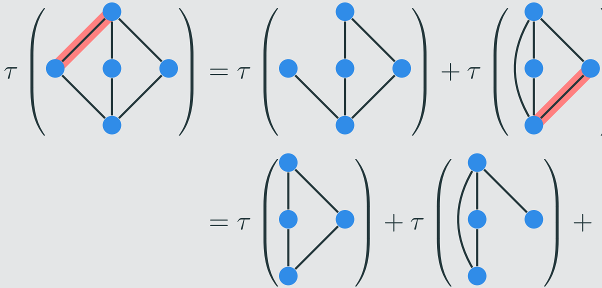
Exemplos

Não escrevemos aqui os «nomes» dos vértices.



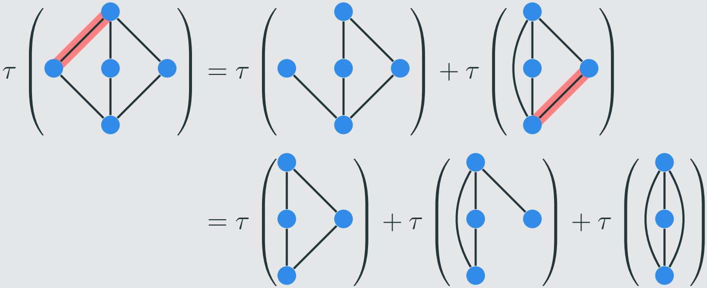
Exemplos

Não escrevemos aqui os «nomes» dos vértices.



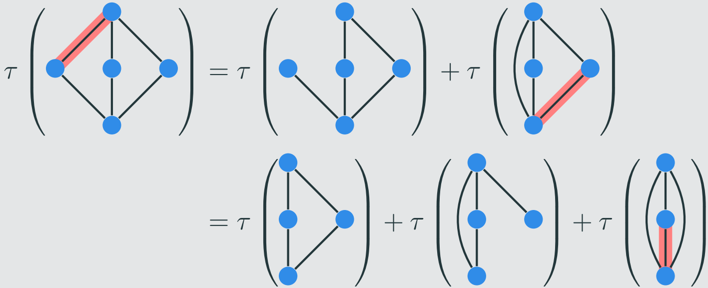
Exemplos

Não escrevemos aqui os «nomes» dos vértices.



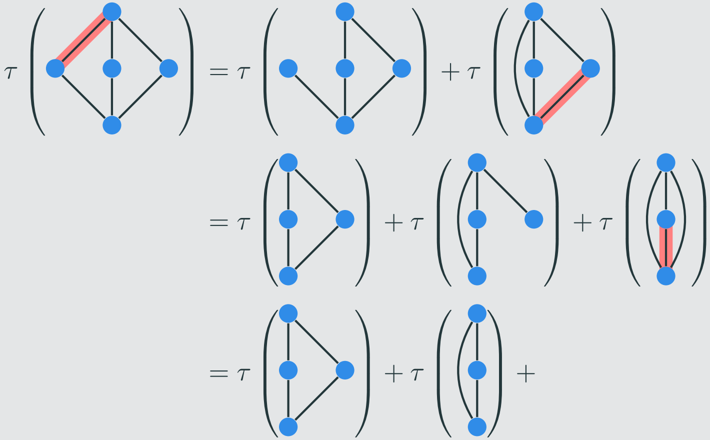
Exemplos

Não escrevemos aqui os «nomes» dos vértices.



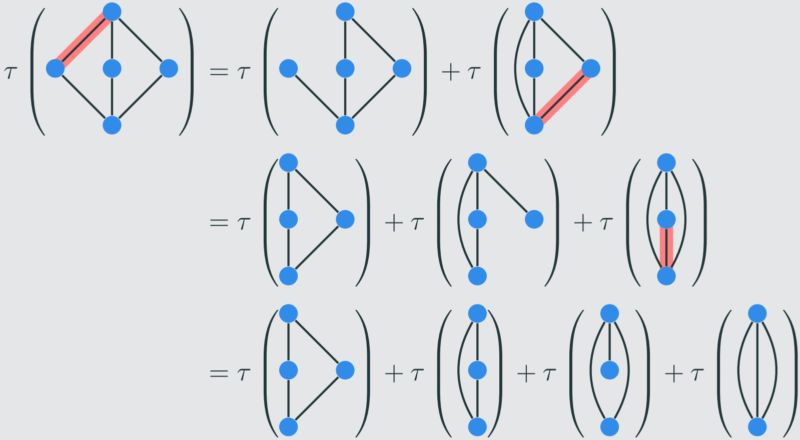
Exemplos

Não escrevemos aqui os «nomes» dos vértices.



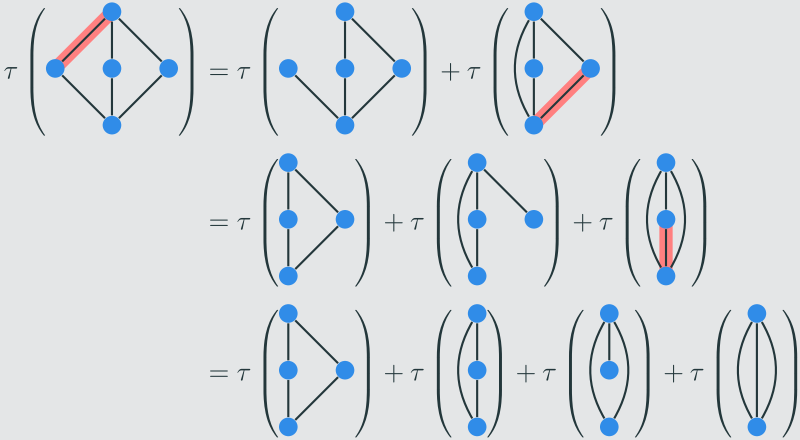
Exemplos

Não escrevemos aqui os «nomes» dos vértices.



Exemplos

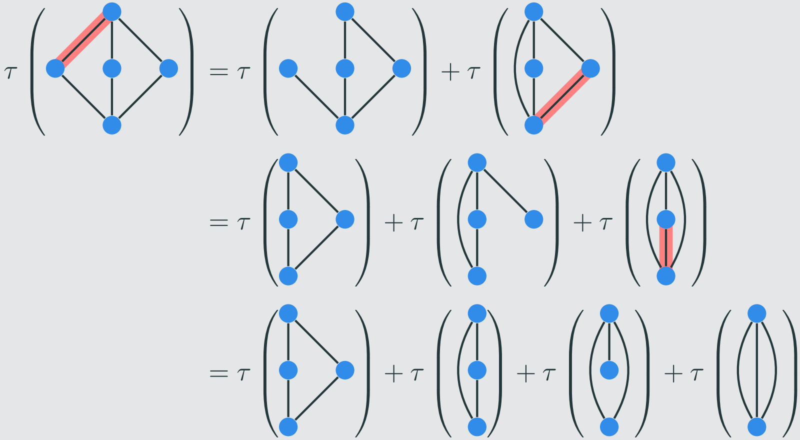
Não escrevemos aqui os «nomes» dos vértices.



$$= 4 +$$

Exemplos

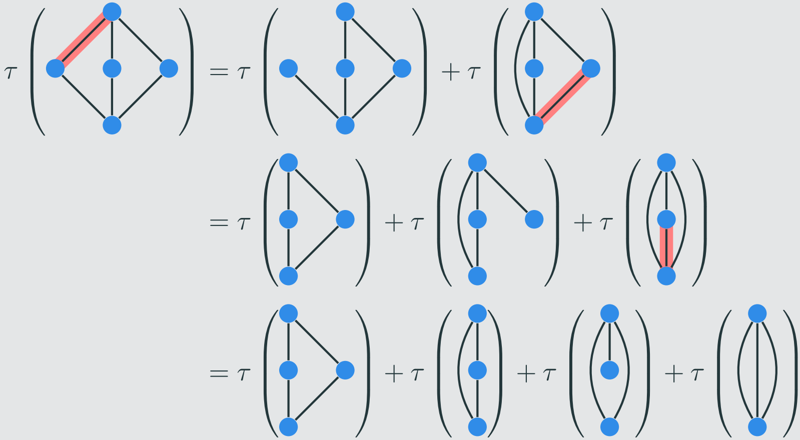
Não escrevemos aqui os «nomes» dos vértices.



$$= 4 + 3 +$$

Exemplos

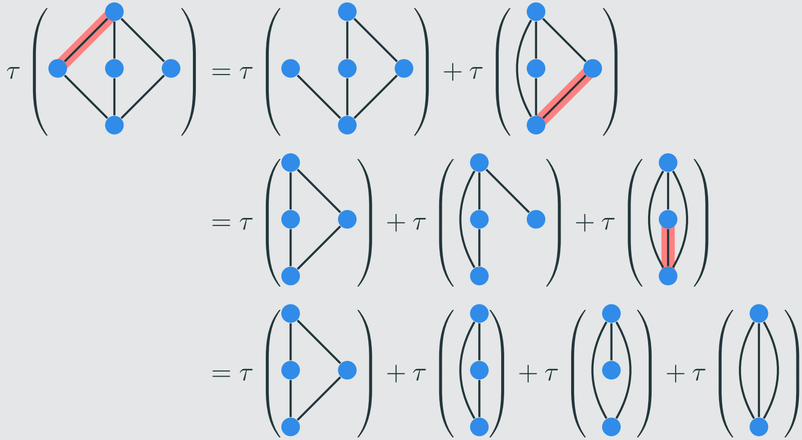
Não escrevemos aqui os «nomes» dos vértices.



$$= 4 + 3 + 2 +$$

Exemplos

Não escrevemos aqui os «nomes» dos vértices.

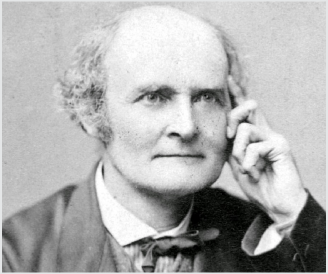


$$= 4 + 3 + 2 + 3 = 12.$$

Teorema (Fórmula de Cayley)

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

Referência

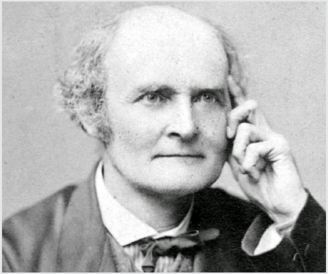


Arthur Cayley (1821 – 1895), matemático britânico.

Teorema (Fórmula de Cayley)

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

Referência



Arthur Cayley (1821 – 1895), matemático britânico.

Corolário

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

A ideia

Sejam $n \geq 2$ e V um conjunto de n elementos (tipicamente $V = \{1, 2, \ldots, n\}$). Tendo em conta o Teorema de Cayley definimos 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 $a_i \in V$.

Pru18.

A ideia

Sejam $n \geq 2$ e V um conjunto de n elementos (tipicamente $V = \{1, 2, \dots, n\}$). Tendo em conta o Teorema de Cayley definimos 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.

Pru18.

A ideia

Sejam $n \geq 2$ e V um conjunto de n elementos (tipicamente $V = \{1, 2, \dots, n\}$). Tendo em conta o Teorema de Cayley definimos 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}.

saltar

Pru18.

O procedimento

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

$$\texttt{pruefer}\colon \{\text{árvores em } V\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

de seguinte maneira.

O procedimento

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

$$\texttt{pruefer}\colon \{\text{árvores em } V\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

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

O procedimento

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

$$\texttt{pruefer}\colon \{\text{árvores em } V\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

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

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

O procedimento

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

$$\texttt{pruefer}\colon \{\text{árvores em } V\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

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**.

O procedimento

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

$$\texttt{pruefer}\colon \{\text{árvores em } V\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

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 (a menor folha).

O procedimento

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

$$\texttt{pruefer}\colon \{\text{árvores em } V\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

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 (a menor folha).
4. $a_i =$ o único vizinho de v.

O procedimento

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

$$\texttt{pruefer}\colon \{\text{árvores em } V\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

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 (a menor folha).
4. $a_i =$ o único vizinho de v.
5. $T = T - v$ (o que ainda é uma árvore!!) e $i = i + 1$.

O procedimento

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

$$\texttt{pruefer}\colon \{\text{árvores em } V\} \longrightarrow \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\}$$

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 (a menor 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.

O procedimento

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

$$\texttt{pruefer}\colon \{\text{árvores em } V\} \longrightarrow \{(a_1, \dots, a_{n-2}) \mid a_i \in V\}$$

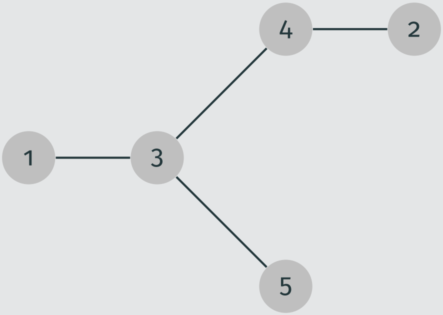
de seguinte maneira. Em primeiro lugar, escolhemos uma ordem total em V, e depois aplicamos o seguinte algoritmo:

Ou de forma recursiva:

$$\begin{aligned}
\texttt{pruefer}(\text{árvore de dois vértices}) &= \text{a lista vazia}\\
\texttt{pruefer}(T) &= (u, \texttt{pruefer}(T - v))\\
&\quad \text{onde}\\
&\quad v = \text{a menor folha de } T\\
&\quad u = \text{o único vizinho de v em } T
\end{aligned}$$

Exemplo

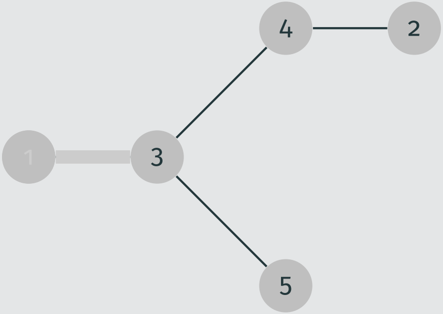
A árvore T:



O código de Prüfer de T: $\texttt{pruefer}(T) = (\qquad)$.

Exemplo

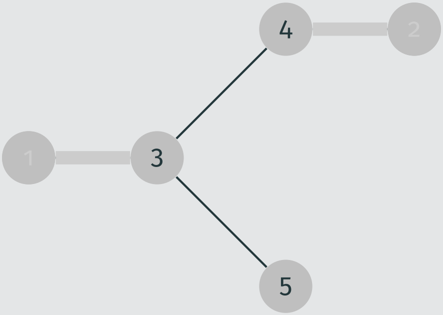
A árvore T:



O código de Prüfer de T: $\texttt{pruefer}(T) = (3, \quad)$.

Exemplo

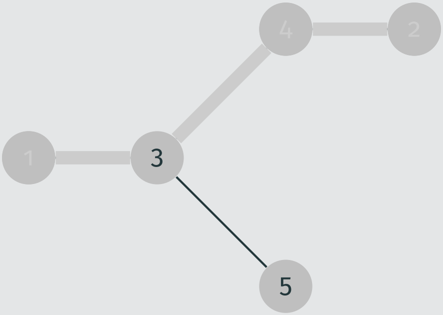
A árvore T:



O código de Prüfer de T: $\texttt{pruefer}(T) = (3, 4, \;)$.

Exemplo

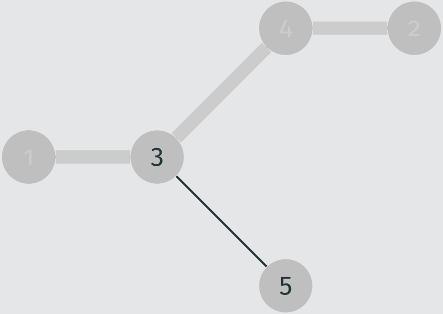
A árvore T:



O código de Prüfer de T: $\texttt{pruefer}(T) = (3, 4, 3)$.

Exemplo

A árvore T:



O código de Prüfer de T: $\texttt{pruefer}(T) = (3, 4, 3)$.

Nota

Cada vértice v aparece $d(v) - 1$ vezes em $(a_1, \dots, a_{n-2})$. Em particular, um vértice v é uma folha se e somente se v não ocorre em $(a_1, \dots, a_{n-2})$.

O procedimento

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

$$\mathtt{unpruefer}\colon \{(a_1, \dots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores em } V\}$$

de seguinte maneira:

O procedimento

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

$$\texttt{unpruefer}\colon \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores em } V\}$$

de seguinte maneira:

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

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

O procedimento

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

$$\texttt{unpruefer}\colon \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores em } V\}$$

de seguinte maneira:

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

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

(2) Se L tem comprimento dois (e portanto P tem comprimento zero), então ligar os dois vértices correspondentes e **Parar**.

O procedimento

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

$$\texttt{unpruefer}\colon \{(a_1, \dots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores em } V\}$$

de seguinte maneira:

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

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

(2) Se L tem comprimento dois (e portanto P tem comprimento zero), então ligar os dois vértices correspondentes e **Parar**.

(3) 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.

O procedimento

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

$$\texttt{unpruefer}\colon \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores em } V\}$$

de seguinte maneira:

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

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

(2) Se L tem comprimento dois (e portanto P tem comprimento zero), então ligar os dois vértices correspondentes e **Parar**.

(3) 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.

(4) **Voltar para** 2.

O procedimento

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

$$\texttt{unpruefer}\colon \{(a_1, \ldots, a_{n-2}) \mid a_i \in V\} \longrightarrow \{\text{árvores em } V\}$$

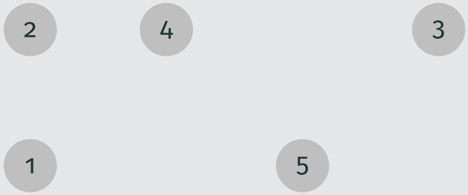
de seguinte maneira:

Ou de forma recursiva:

$$\begin{aligned} \texttt{unpruefer}(\text{a lista vazia}) &= \text{a árvore com dois vértices} \\ \texttt{unpruefer}((a, \text{resto})) &= \texttt{unpruefer}(\text{resto}) + \text{ligar } v \text{ e } a \\ &\qquad \text{onde} \\ &\qquad\qquad v = \text{o menor elemento de } V \text{ que} \\ &\qquad\qquad\qquad \text{não ocorre em } (a, \text{resto}) \end{aligned}$$

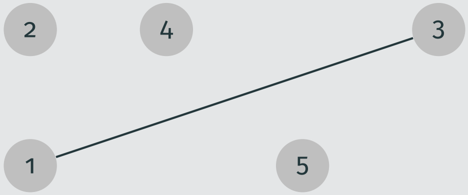
Exemplo

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



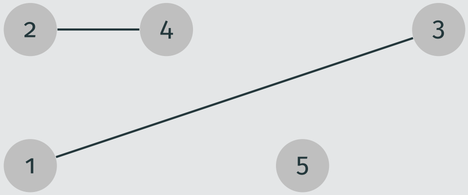
Exemplo

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



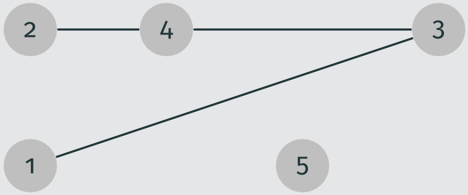
Exemplo

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



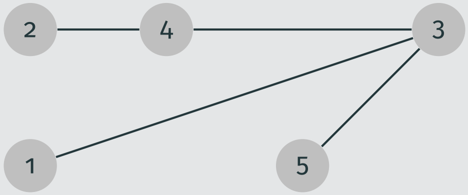
Exemplo

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



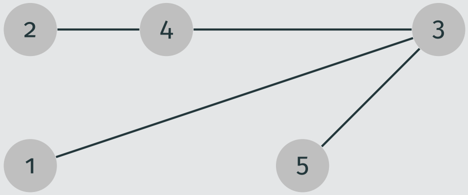
Exemplo

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

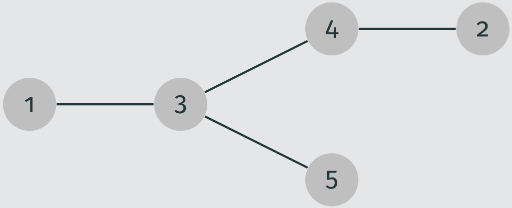


Exemplo

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

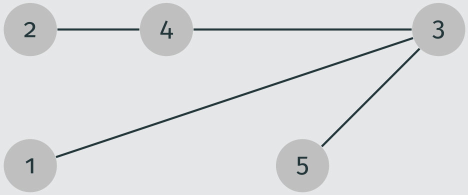


Para comparar



Exemplo

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



Teorema

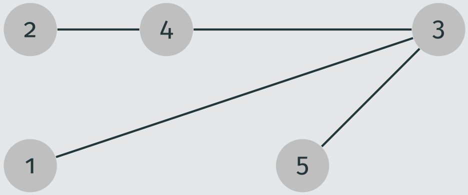
Verificam-se as igualdades

$$\texttt{pruefer} \circ \texttt{unpruefer} = \mathrm{id} \quad e \quad \texttt{unpruefer} \circ \texttt{pruefer} = \mathrm{id},$$

logo $\texttt{unpruefer} = \texttt{pruefer}^{-1}$

Exemplo

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



Teorema

Verificam-se as igualdades

$$\texttt{pruefer} \circ \texttt{unpruefer} = \mathrm{id} \quad e \quad \texttt{unpruefer} \circ \texttt{pruefer} = \mathrm{id},$$

logo $\texttt{unpruefer} = \texttt{pruefer}^{-1}$ *e por isso* `pruefer` *e* `unpruefer` *são funções bijetivas.*

2. Árvores abrangentes de custo mínimo

O contexto

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

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

de custos não negativos nas arestas.

O contexto

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

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

de custos não negativos 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 contexto

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

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

de custos não negativos 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 grafo conexo finito $G = (V, E, \psi)$ com $W \colon E \longrightarrow [0, \infty]$, encontrar uma árvore abrangente de custo mínimo.

Dois algoritmos

- O algoritmo de Kruskal.
- O algoritmo de Prim.

Joseph Bernard Kruskal (1928 – 2010) matemático, estatístico, informático e psicometrista estadunidense, e Robert Clay Prim (1921) matemático e informático estadunidense.

Descrição do algoritmo

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

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

Descrição do algoritmo

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

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.

Descrição do algoritmo

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

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. $E' = \varnothing$, $i = 1$.

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

Descrição do algoritmo

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

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.

Descrição do algoritmo

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

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.

Descrição do algoritmo

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

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

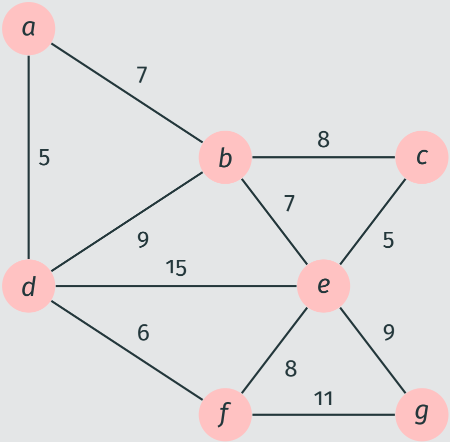
$$W(a_1) \leq W(a_2) \leq \cdots \leq W(a_m).$$

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.

Exemplo

Ordenar as arestas: ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

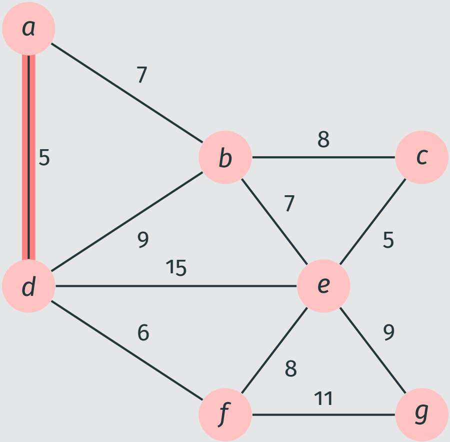
1. $E' = \varnothing$



Exemplo

Ordenar as arestas: ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

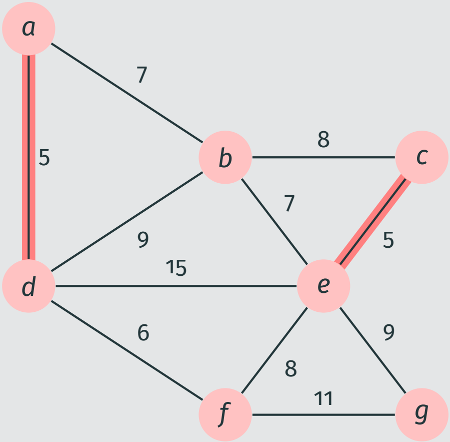
1. $E' = \varnothing$
2. $E' = \{\text{ad}\}$



Exemplo

Ordenar as arestas: ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

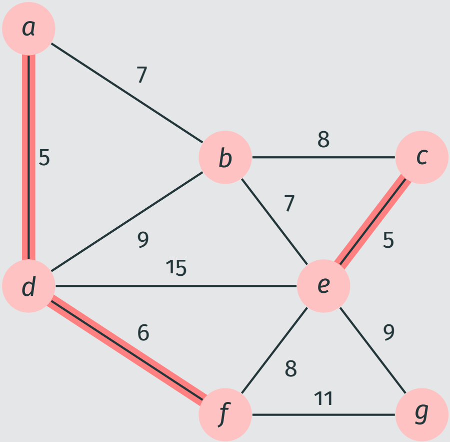
1. $E' = \varnothing$
2. $E' = \{\text{ad}\}$
3. $E' = \{\text{ad,ce}\}$



Exemplo

Ordenar as arestas: ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

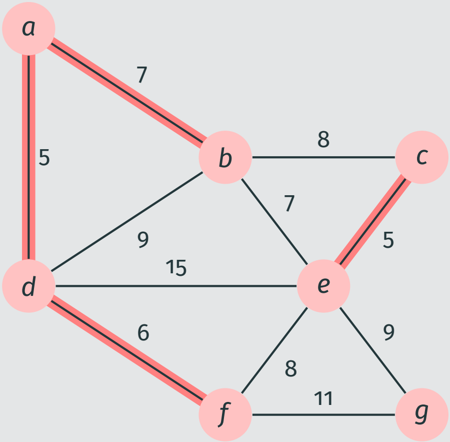
1. $E' = \varnothing$
2. $E' = \{\text{ad}\}$
3. $E' = \{\text{ad,ce}\}$
4. $E' = \{\text{ad,ce,df}\}$



Exemplo

Ordenar as arestas: ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

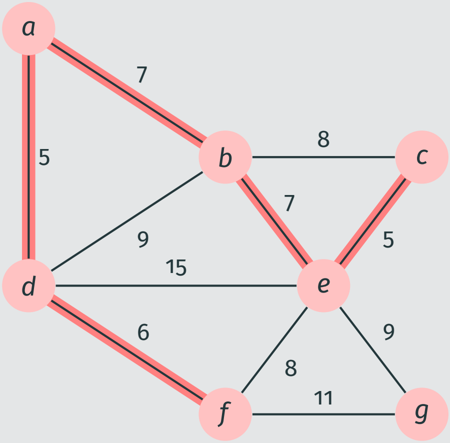
1. $E' = \varnothing$
2. $E' = \{\text{ad}\}$
3. $E' = \{\text{ad,ce}\}$
4. $E' = \{\text{ad,ce,df}\}$
5. $E' = \{\text{ad,ce,df,ab}\}$



Exemplo

Ordenar as arestas: ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

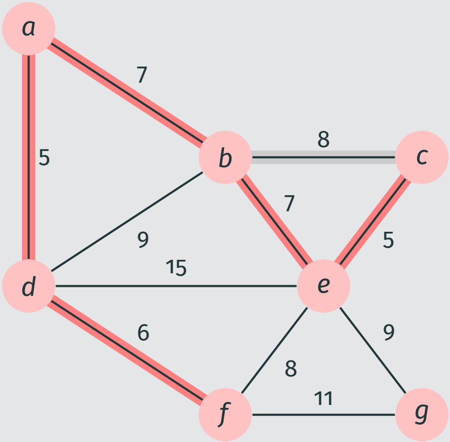
1. $E' = \varnothing$
2. $E' = \{\text{ad}\}$
3. $E' = \{\text{ad,ce}\}$
4. $E' = \{\text{ad,ce,df}\}$
5. $E' = \{\text{ad,ce,df,ab}\}$
6. $E' = \{\text{ad,ce,df,ab,be}\}$



Exemplo

Ordenar as arestas: ad, ce, df, ab, be, bc, ef, bd, eg, fg, de.

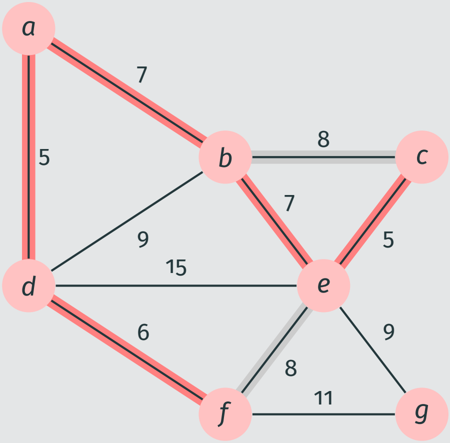
1. $E' = \varnothing$
2. $E' = \{\text{ad}\}$
3. $E' = \{\text{ad,ce}\}$
4. $E' = \{\text{ad,ce,df}\}$
5. $E' = \{\text{ad,ce,df,ab}\}$
6. $E' = \{\text{ad,ce,df,ab,be}\}$
7. $E' = \{\text{ad,ce,df,ab,be}\}$, bc $\notin E'$



Exemplo

Ordenar as arestas: ad, ce, df, ab, be, ~~bc~~, ef, bd, eg, fg, de.

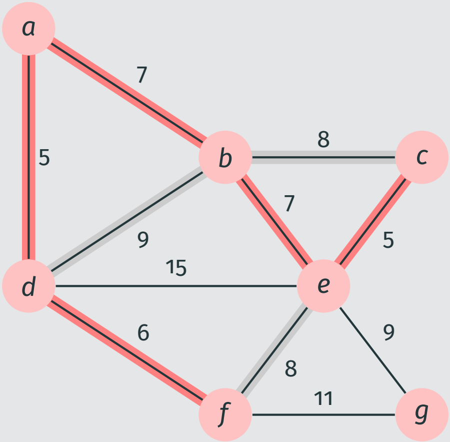
1. $E' = \varnothing$
2. $E' = \{\text{ad}\}$
3. $E' = \{\text{ad,ce}\}$
4. $E' = \{\text{ad,ce,df}\}$
5. $E' = \{\text{ad,ce,df,ab}\}$
6. $E' = \{\text{ad,ce,df,ab,be}\}$
7. $E' = \{\text{ad,ce,df,ab,be}\}$, bc $\notin E'$
8. $E' = \{\text{ad,ce,df,ab,be}\}$, ef $\notin E'$



Exemplo

Ordenar as arestas: ad, ce, df, ab, be, ~~bc~~, ~~ef~~, bd, eg, fg, de.

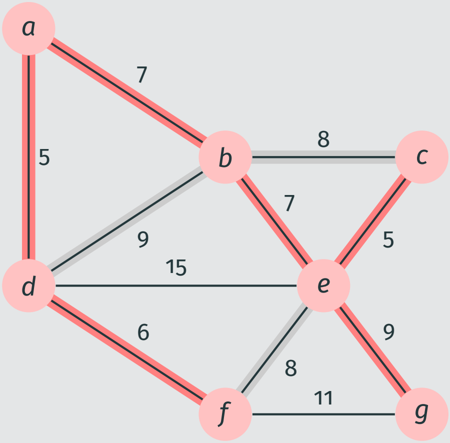
1. $E' = \varnothing$
2. $E' = \{\text{ad}\}$
3. $E' = \{\text{ad,ce}\}$
4. $E' = \{\text{ad,ce,df}\}$
5. $E' = \{\text{ad,ce,df,ab}\}$
6. $E' = \{\text{ad,ce,df,ab,be}\}$
7. $E' = \{\text{ad,ce,df,ab,be}\}$, bc $\notin E'$
8. $E' = \{\text{ad,ce,df,ab,be}\}$, ef $\notin E'$
9. $E' = \{\text{ad,ce,df,ab,be}\}$, bd $\notin E'$



Exemplo

Ordenar as arestas: ad, ce, df, ab, be, ~~bc~~, ~~ef~~, ~~bd~~, eg, fg, de.

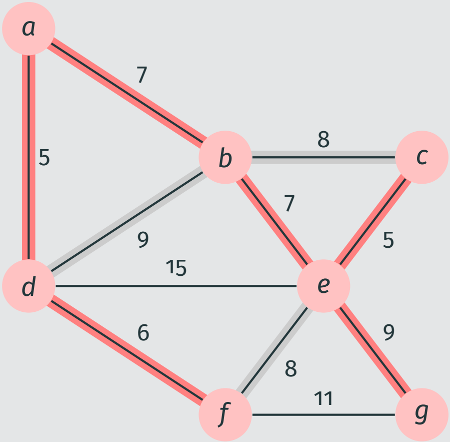
1. $E' = \varnothing$
2. $E' = \{\text{ad}\}$
3. $E' = \{\text{ad,ce}\}$
4. $E' = \{\text{ad,ce,df}\}$
5. $E' = \{\text{ad,ce,df,ab}\}$
6. $E' = \{\text{ad,ce,df,ab,be}\}$
7. $E' = \{\text{ad,ce,df,ab,be}\}$, bc $\notin E'$
8. $E' = \{\text{ad,ce,df,ab,be}\}$, ef $\notin E'$
9. $E' = \{\text{ad,ce,df,ab,be}\}$, bd $\notin E'$
10. $E' = \{\text{ad,ce,df,ab,be,eg}\}$



Exemplo

Ordenar as arestas: ad, ce, df, ab, be, ~~bc~~, ~~ef~~, ~~bd~~, eg, fg, de.

1. $E' = \varnothing$
2. $E' = \{\text{ad}\}$
3. $E' = \{\text{ad,ce}\}$
4. $E' = \{\text{ad,ce,df}\}$
5. $E' = \{\text{ad,ce,df,ab}\}$
6. $E' = \{\text{ad,ce,df,ab,be}\}$
7. $E' = \{\text{ad,ce,df,ab,be}\}$, bc $\notin E'$
8. $E' = \{\text{ad,ce,df,ab,be}\}$, ef $\notin E'$
9. $E' = \{\text{ad,ce,df,ab,be}\}$, bd $\notin E'$
10. $E' = \{\text{ad,ce,df,ab,be,eg}\}$



Terminar: O grafo $T = (V, E')$ é conexo. $W(T) = 39$.

Descrição do algoritmo

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

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

Descrição do algoritmo

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

1. Escolher um vértice $u \in V$.

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

Descrição do algoritmo

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

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.

Descrição do algoritmo

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

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.

Descrição do algoritmo

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

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.

Descrição do algoritmo

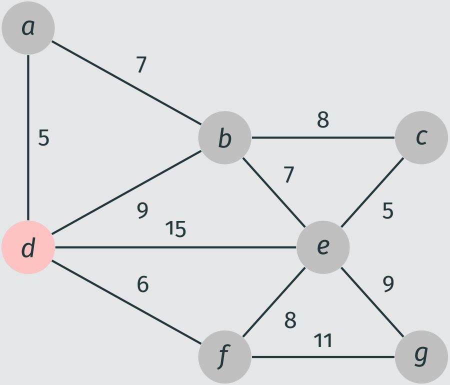
Consideramos um grafo conexo $G = (V, E, \psi)$ e $W\colon E \longrightarrow [0, \infty]$.

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 $w^* \notin V'$.
   - $V' = V' \cup \{w^*\}$, $E' = E' \cup \{e^*\}$.
   - **Saltar para** o início de 3.
4. Devolver a árvore abrangente (V, E') de G de custo mínimo.

Exemplo

Escolhemos o vértice d.

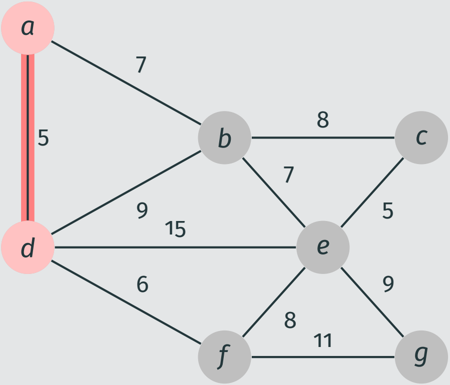
1. $V' = \{d\}, E' = \varnothing$



Exemplo

Escolhemos o vértice d.

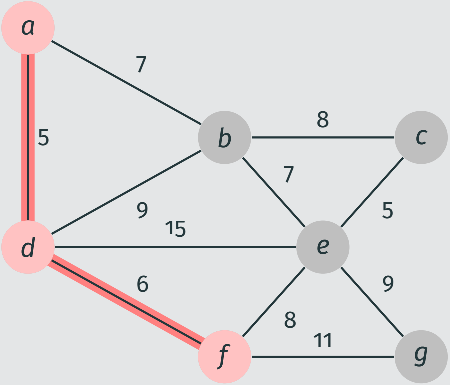
1. $V' = \{d\}, E' = \varnothing$
2. $V' = \{d, a\}, E' = \{ad\}$



Exemplo

Escolhemos o vértice d.

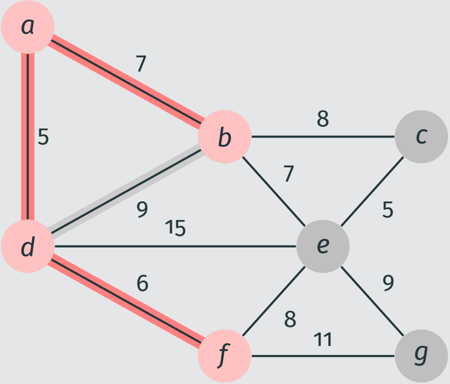
1. $V' = \{d\}, E' = \varnothing$
2. $V' = \{d, a\}, E' = \{ad\}$
3. $V' = \{d, a, f\}, E' = \{ad, df\}$



Exemplo

Escolhemos o vértice d.

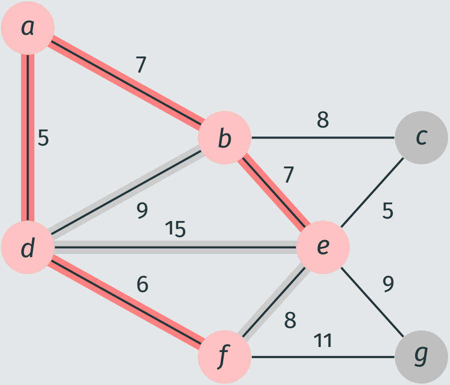
1. $V' = \{d\}, E' = \varnothing$
2. $V' = \{d, a\}, E' = \{ad\}$
3. $V' = \{d, a, f\}, E' = \{ad, df\}$
4. $V' = \{d, a, f, b\}$,
   $E' = \{ad, df, ab\}$



Exemplo

Escolhemos o vértice d.

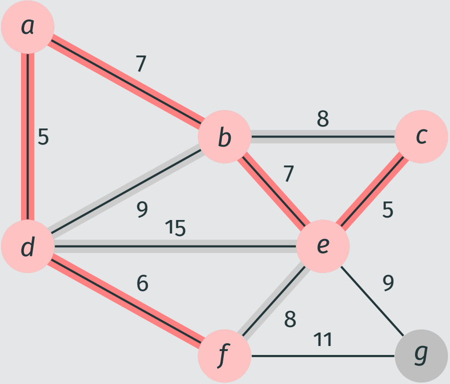
1. $V' = \{d\}, E' = \varnothing$
2. $V' = \{d, a\}, E' = \{ad\}$
3. $V' = \{d, a, f\}, E' = \{ad, df\}$
4. $V' = \{d, a, f, b\}$,
   $E' = \{ad, df, ab\}$
5. $V' = \{d, a, f, b, e\}$,
   $E' = \{ad, df, ab, be\}$



Exemplo

Escolhemos o vértice d.

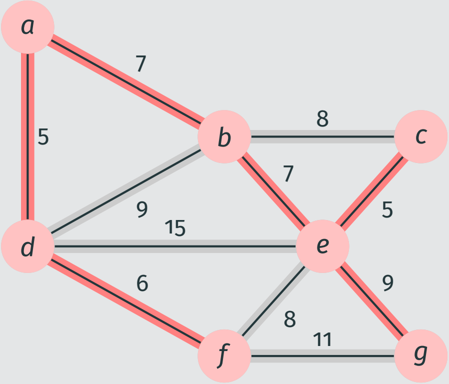
1. $V' = \{d\}, E' = \varnothing$
2. $V' = \{d, a\}, E' = \{ad\}$
3. $V' = \{d, a, f\}, E' = \{ad, df\}$
4. $V' = \{d, a, f, b\}$,
   $E' = \{ad, df, ab\}$
5. $V' = \{d, a, f, b, e\}$,
   $E' = \{ad, df, ab, be\}$
6. $V' = \{d, a, f, b, e, c\}$,
   $E' = \{ad, df, ab, be, ec\}$



Exemplo

Escolhemos o vértice d.

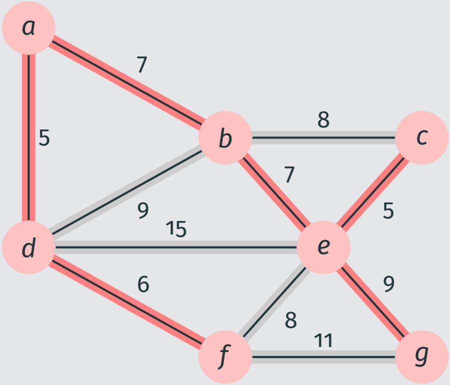
1. $V' = \{d\}, E' = \varnothing$
2. $V' = \{d, a\}, E' = \{ad\}$
3. $V' = \{d, a, f\}, E' = \{ad, df\}$
4. $V' = \{d, a, f, b\}$,
   $E' = \{ad, df, ab\}$
5. $V' = \{d, a, f, b, e\}$,
   $E' = \{ad, df, ab, be\}$
6. $V' = \{d, a, f, b, e, c\}$,
   $E' = \{ad, df, ab, be, ec\}$
7. $V' = \{d, a, f, b, e, c, g\}$,
   $E' = \{ad, df, ab, be, ec, eg\}$



Exemplo

Escolhemos o vértice d.

1. $V' = \{d\}, E' = \varnothing$
2. $V' = \{d, a\}, E' = \{ad\}$
3. $V' = \{d, a, f\}, E' = \{ad, df\}$
4. $V' = \{d, a, f, b\}$,
   $E' = \{ad, df, ab\}$
5. $V' = \{d, a, f, b, e\}$,
   $E' = \{ad, df, ab, be\}$
6. $V' = \{d, a, f, b, e, c\}$,
   $E' = \{ad, df, ab, be, ec\}$
7. $V' = \{d, a, f, b, e, c, g\}$,
   $E' = \{ad, df, ab, be, ec, eg\}$

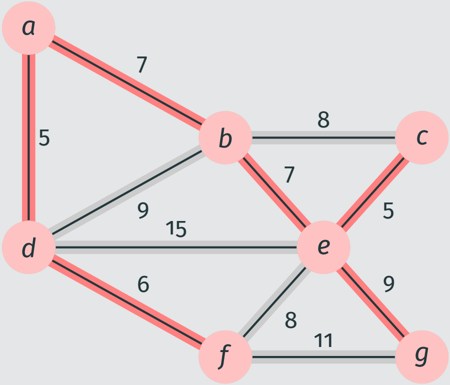


Terminar: $V' = V$. $W(V, E') = 39$.

Exemplo

Escolhemos o vértice d.

1. $V' = \{d\}, E' = \varnothing$
2. $V' = \{d, a\}, E' = \{ad\}$
3. $V' = \{d, a, f\}, E' = \{ad, df\}$
4. $V' = \{d, a, f, b\}$,
   $E' = \{ad, df, ab\}$
5. $V' = \{d, a, f, b, e\}$,
   $E' = \{ad, df, ab, be\}$
6. $V' = \{d, a, f, b, e, c\}$,
   $E' = \{ad, df, ab, be, ec\}$
7. $V' = \{d, a, f, b, e, c, g\}$,
   $E' = \{ad, df, ab, be, ec, eg\}$



Terminar: $V' = V$. $W(V, E') = 39$.

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