Teoria dos Grafos

ÁRVORES, PONTES E ÁRVORES *SPANNING*

Prof. Tiago Eugenio de Melo
tmelo@uea.edu.br

www.tiagodemelo.info

Observações

Observações

- As definições e teorias apresentadas aqui foram baseadas no livro *Fundamentos da Teoria dos Grafos para Computação* (LTC, 2018).

Observações

- As definições e teorias apresentadas aqui foram baseadas no livro *Fundamentos da Teoria dos Grafos para Computação* (LTC, 2018).
- A numeração das definições e teoremas seguem as mesmas referências adotadas no livro para facilitar a localização.

ÁRVORES

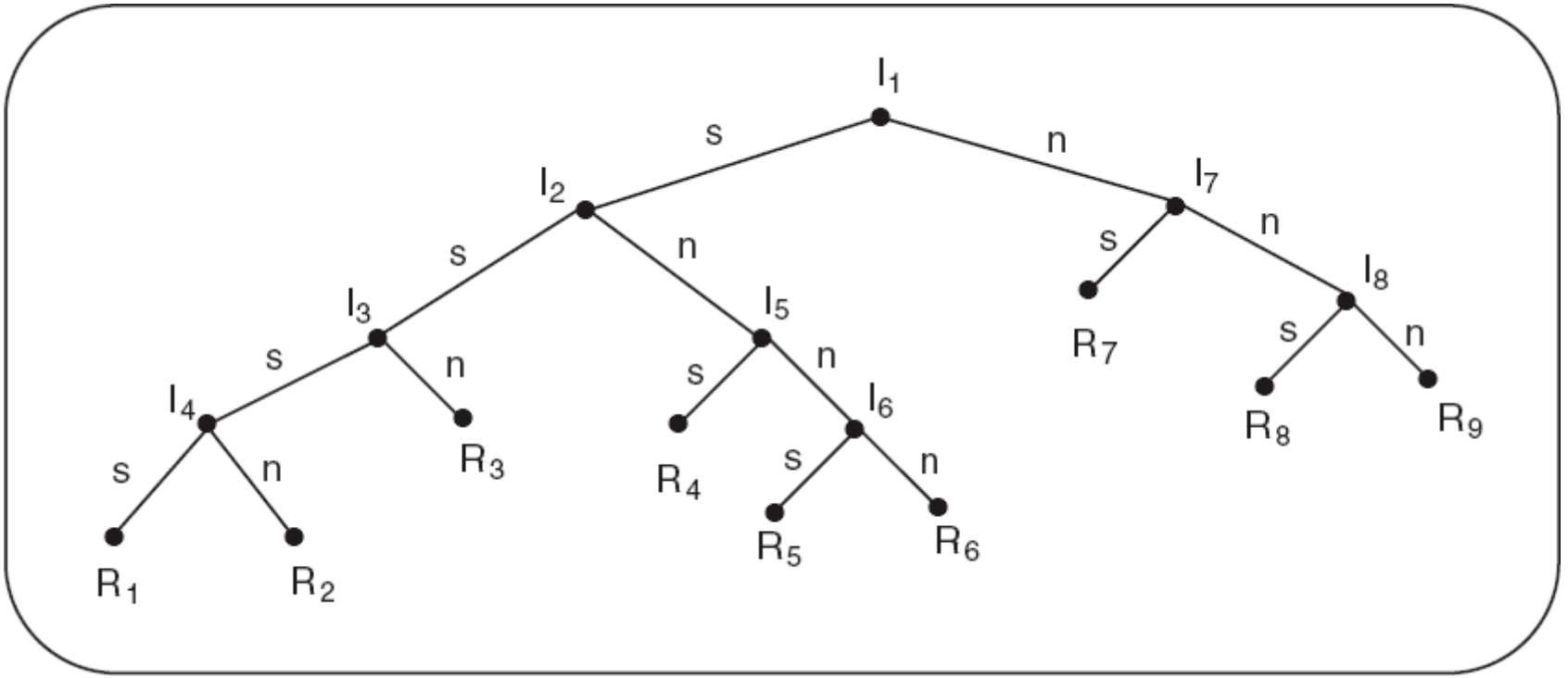
Introdução

Introdução

- O conceito de árvore é provavelmente o mais importante em Teoria dos Grafos.
- Principalmente pelo seu emprego em muitas aplicações computacionais.

Introdução

- O conceito de árvore é provavelmente o mais importante em Teoria dos Grafos.
- Principalmente pelo seu emprego em muitas aplicações computacionais.



Árvores

Árvores

- **Definição 6.1**

Árvores

- **Definição 6.1**
  - Seja G = (V,E) um grafo.

Árvores

- **Definição 6.1**
  - Seja G = (V,E) um grafo.
  - G é acíclico se G não contém ciclos.

Árvores

- **Definição 6.1**
  - Seja G = (V,E) um grafo.
  - G é acíclico se G não contém ciclos.
  - G é uma árvore se G for um grafo acíclico conexo.

Árvores

- **Definição 6.1**
  - Seja G = (V,E) um grafo.
  - G é acíclico se G não contém ciclos.
  - G é uma árvore se G for um grafo acíclico conexo.
  - G é uma floresta se G for acíclico, independentemente de ser conexo ou não.

Árvores

- **Definição 6.1**
  - Seja G = (V,E) um grafo.
  - G é acíclico se G não contém ciclos.
  - G é uma árvore se G for um grafo acíclico conexo.
  - G é uma floresta se G for acíclico, independentemente de ser conexo ou não.
- Como loops são ciclos de comprimento um e um par de arestas paralelas é um ciclo de comprimento dois, qualquer grafo acíclico deve ser simples.

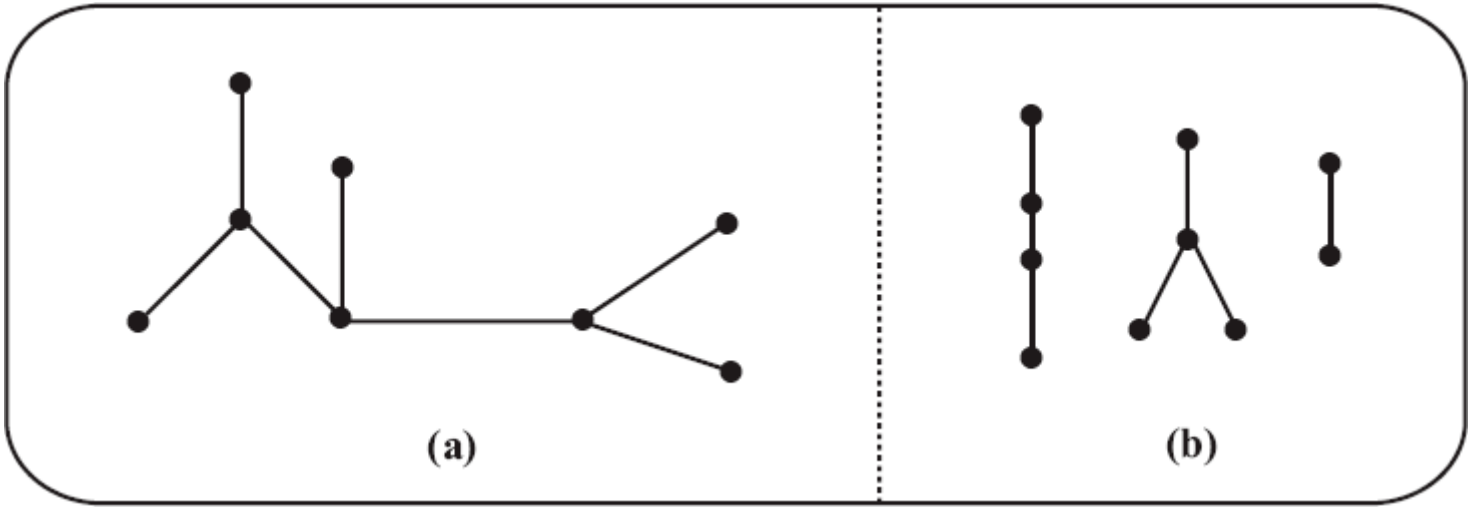
Exemplo

Exemplo

- O grafo da figura (a) é uma árvore.

Exemplo

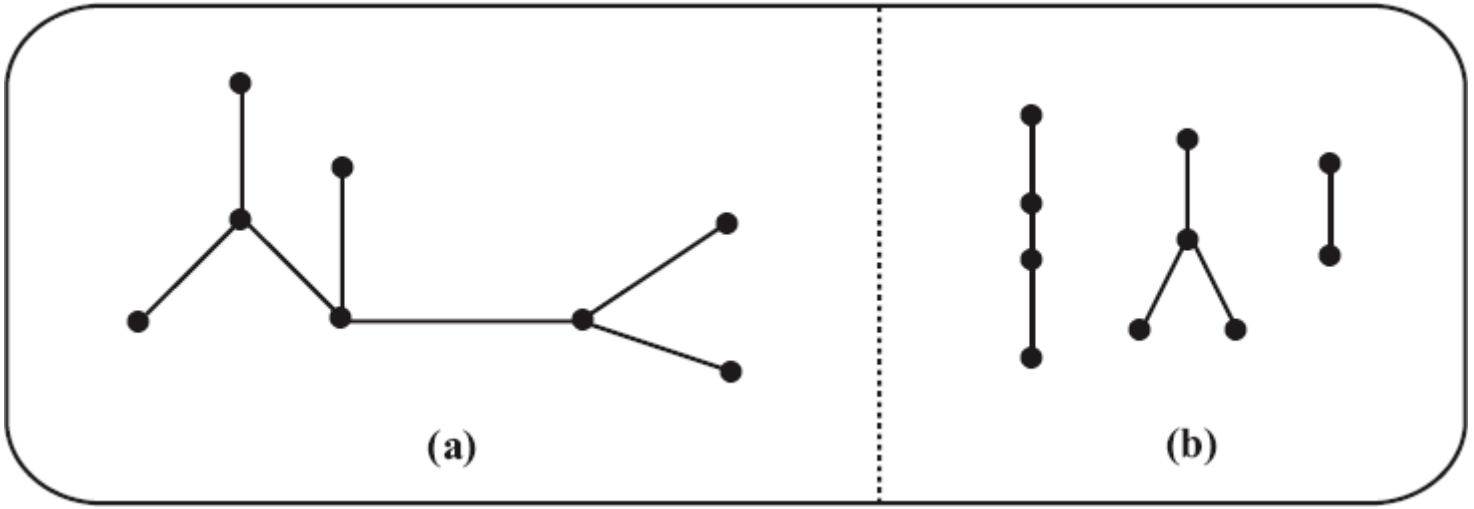
- O grafo da figura (a) é uma árvore.



Árvore Floresta

Exemplo

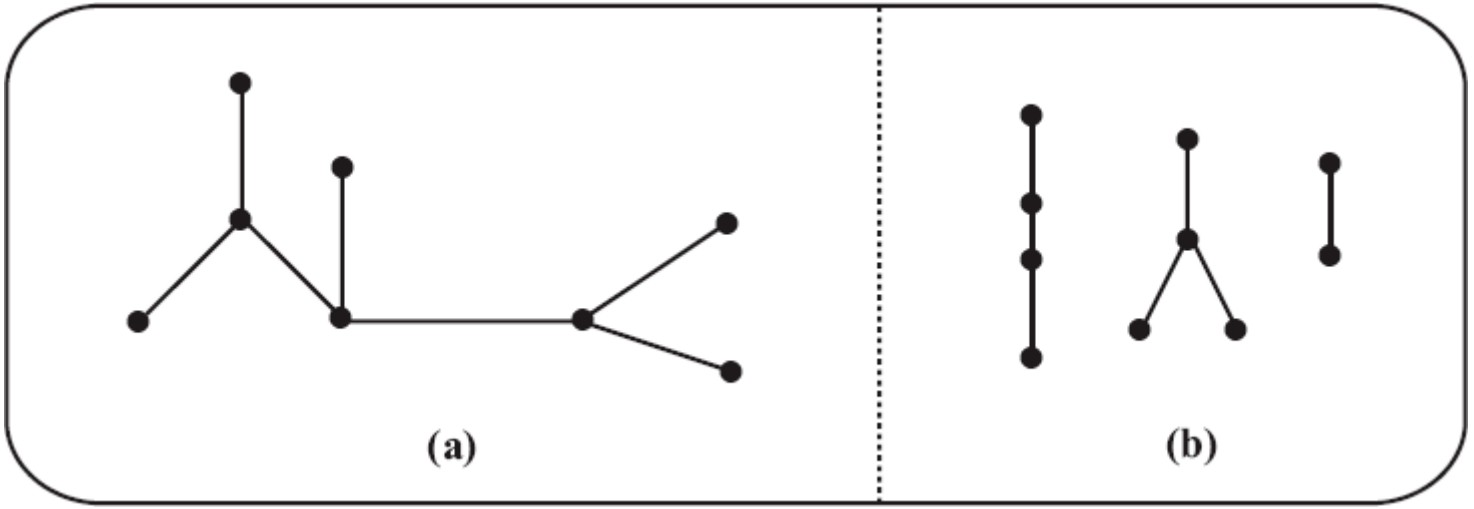
- O grafo da figura (a) é uma árvore.
- O grafo da figura (b) é uma floresta, mas não é uma árvore.



Árvore Floresta

Exemplo

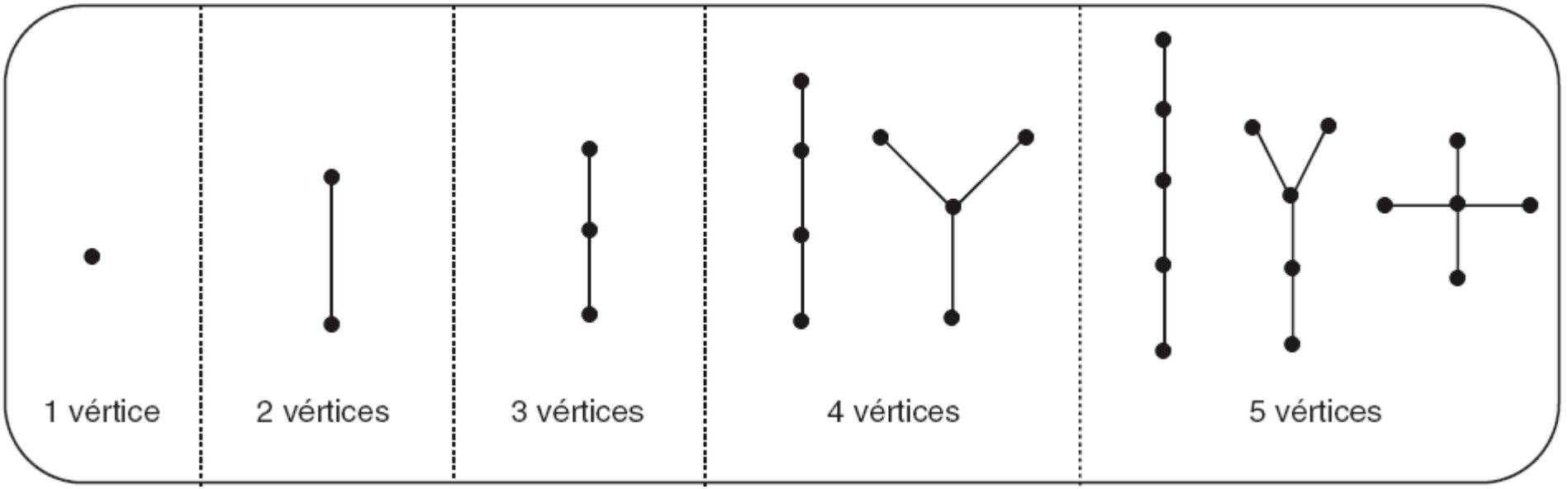
- O grafo da figura (a) é uma árvore.
- O grafo da figura (b) é uma floresta, mas não é uma árvore.
- Note que toda árvore é também uma floresta.



Árvore Floresta

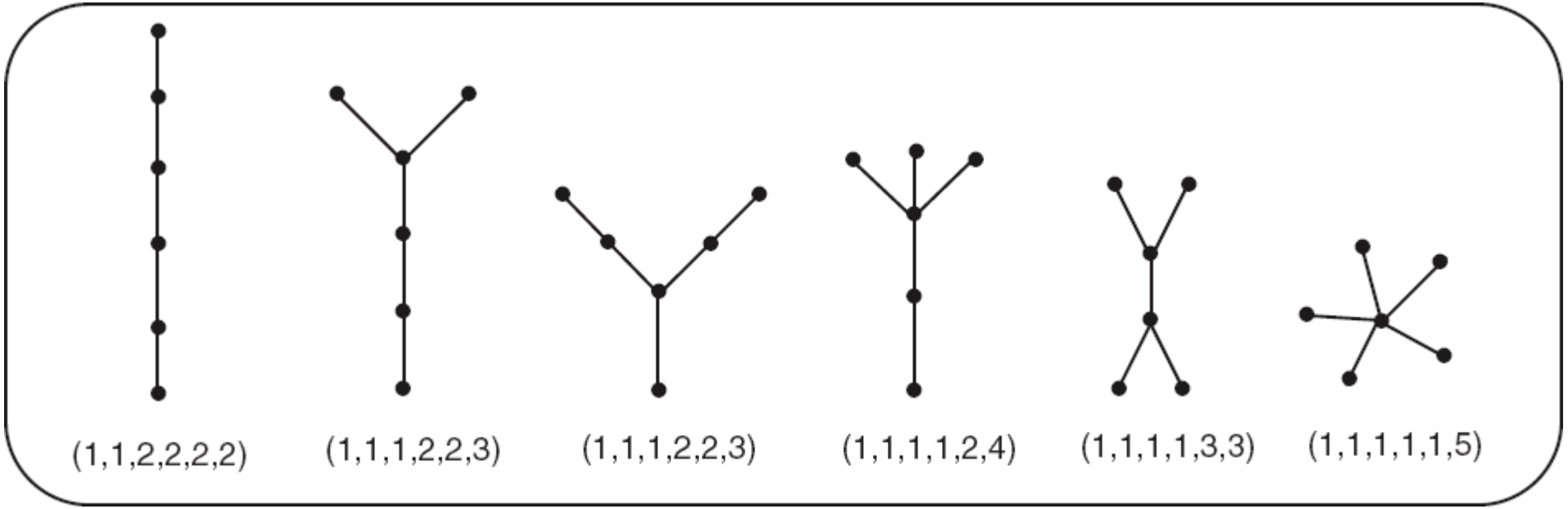
Exemplo

- A figura mostra todas as árvores com até cinco vértices.



Exemplo

- A figura mostra todas as árvores com seis vértices.



Árvore

Árvore

- **Teorema 6.1**

Árvore

- **Teorema 6.1**

  A) Sejam u e v vértices distintos de uma árvore T. Então existe precisamente um único caminho de u a v.

Árvore

- **Teorema 6.1**

  A) Sejam u e v vértices distintos de uma árvore T. Então existe precisamente um único caminho de u a v.

  B) Seja G um grafo sem nenhum loop. Se para todo par de vértices distintos u e v de G existe precisamente um caminho de u a v, então G é uma árvore.

Árvore

Árvore

- **Teorema 6.2**

Árvore

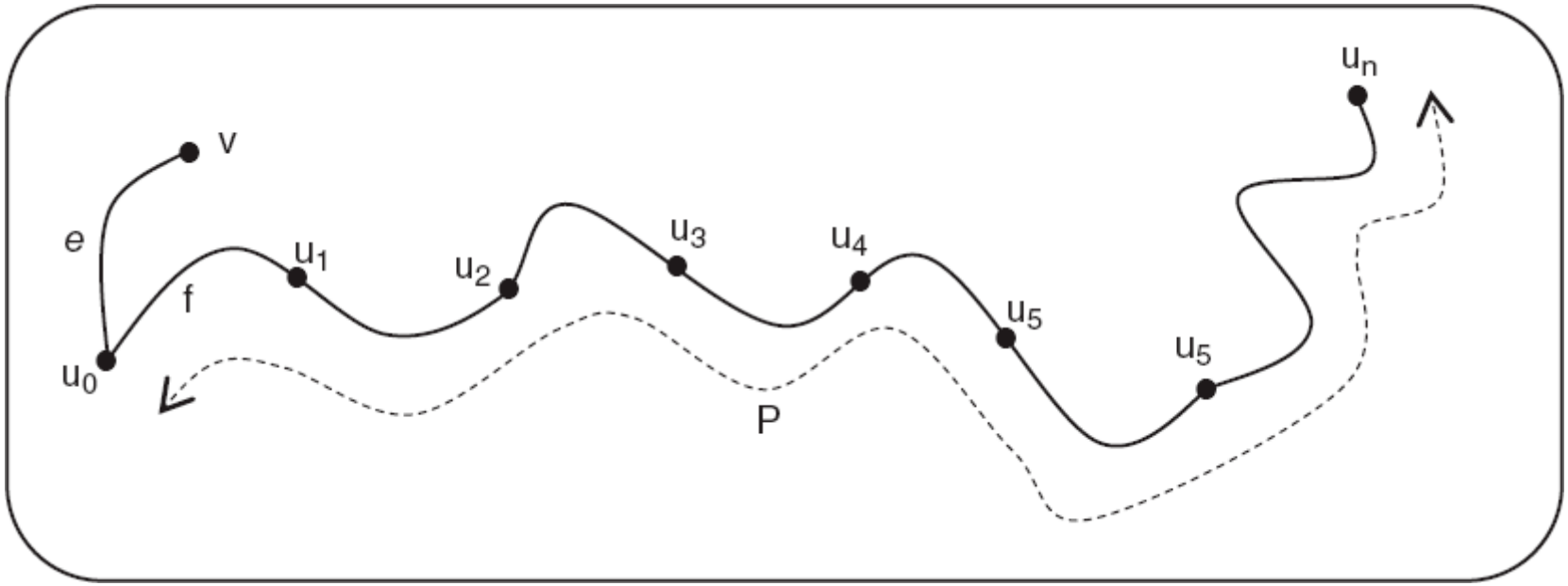
- **Teorema 6.2**
  - Seja T uma árvore com pelo menos dois vértices e seja P = u0u1...un o caminho mais longo em T (isto é, não existe em T um caminho com comprimento maior que n).

Árvore

- **Teorema 6.2**
  - Seja T uma árvore com pelo menos dois vértices e seja P = u0u1...un o caminho mais longo em T (isto é, não existe em T um caminho com comprimento maior que n).
  - Então u0 e un têm ambos grau 1, isto é, d(u0) = d(un) = 1.

Árvore

- **Teorema 6.2**
  - Dado o grafo abaixo:



  - Se P é o caminho mais longo na árvore T, o vértice v (e a aresta e) não podem existir.

Árvore

Árvore

- **Corolário do Teorema 6.2**

Árvore

- **Corolário do Teorema 6.2**
  - Uma árvore T com pelo menos dois vértices tem mais do que um vértice de grau 1.

Árvore

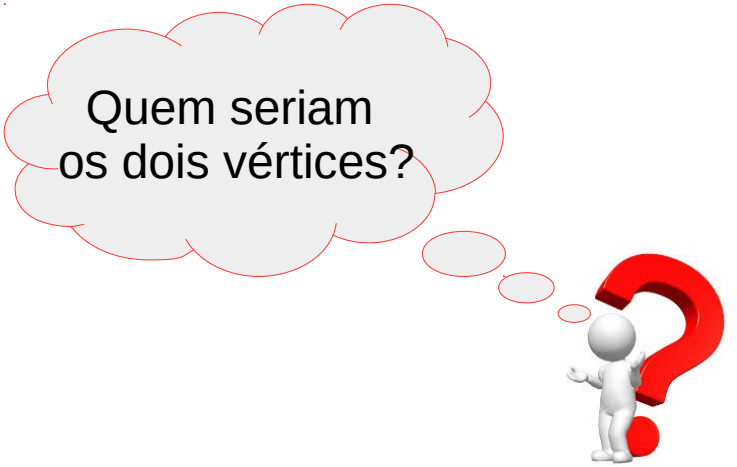
- **Corolário do Teorema 6.2**
  - Uma árvore T com pelo menos dois vértices tem mais do que um vértice de grau 1.
- **Prova:**

Árvore

- **Corolário do Teorema 6.2**
  - Uma árvore T com pelo menos dois vértices tem mais do que um vértice de grau 1.
- **Prova:**
  - Nessa tal árvore T existe um caminho mais longo P (de comprimento maior que 0); assim, o Teorema 6.2 garante a existência de, pelo menos, dois vértices com grau 1.

Árvore

- **Corolário do Teorema 6.2**
  - Uma árvore T com pelo menos dois vértices tem mais do que um vértice de grau 1.
- **Prova:**
  - Nessa tal árvore T existe um caminho mais longo P (de comprimento maior que 0); assim, o Teorema 6.2 garante a existência de, pelo menos, dois vértices com grau 1.



Árvore

- **Teorema 6.3**
  - Se T é uma árvore com n vértices, então T tem precisamente n−1 arestas.

PONTES

Pontes

Pontes

- **Definição 6.2**

Pontes

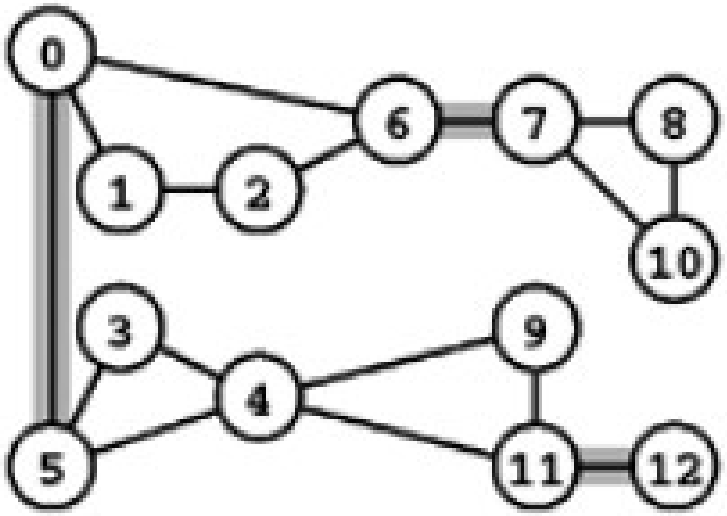
- **Definição 6.2**
  - Seja G = (V,E) um grafo.

Pontes

- **Definição 6.2**
  - Seja G = (V,E) um grafo.
  - Um vértice v de G é chamado de **vértice de corte** (**articulação**) se o subgrafo G−v tiver mais componentes conexos que G.

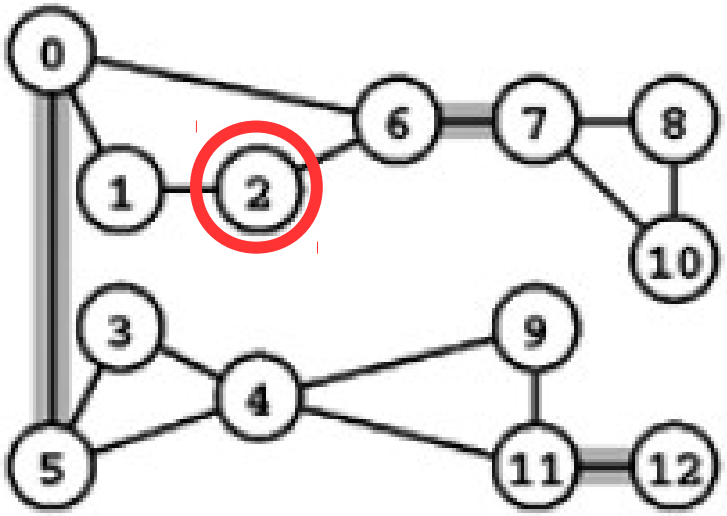
Pontes

- Exemplo



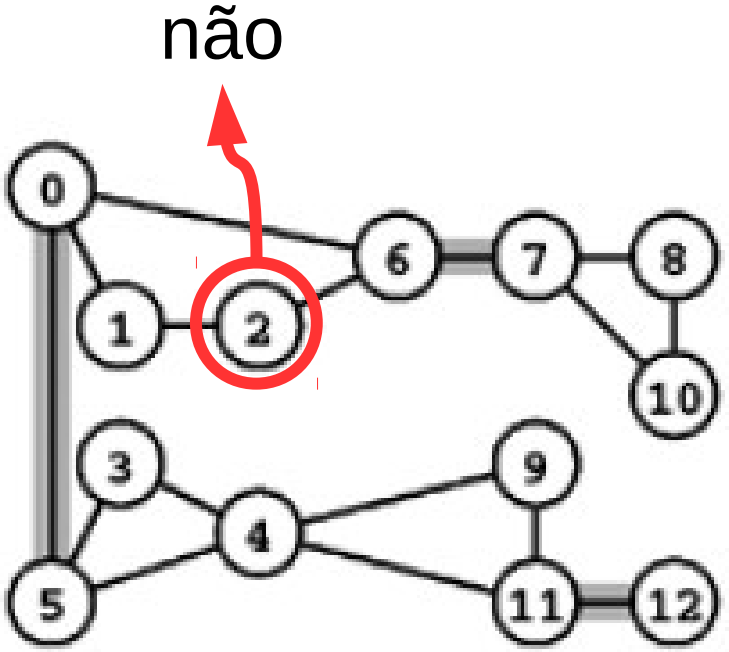
Pontes

- Exemplo



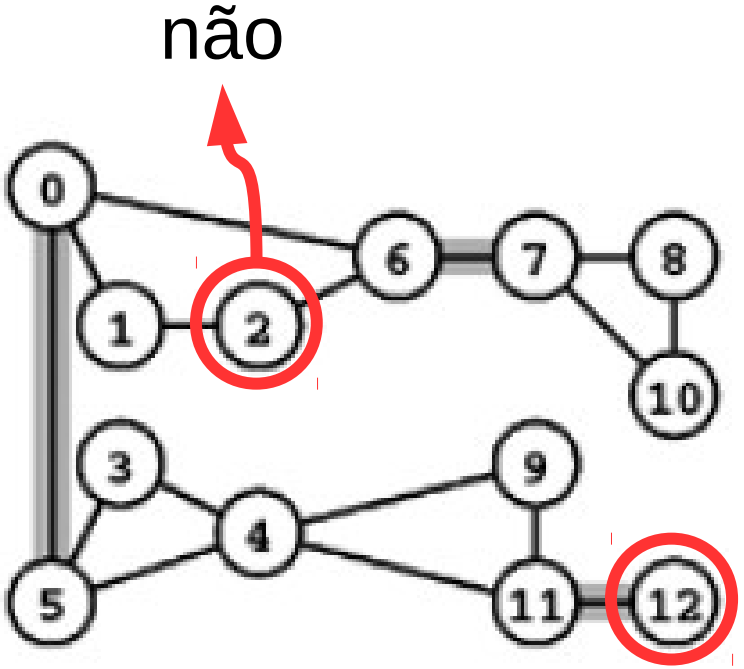
Pontes

- Exemplo



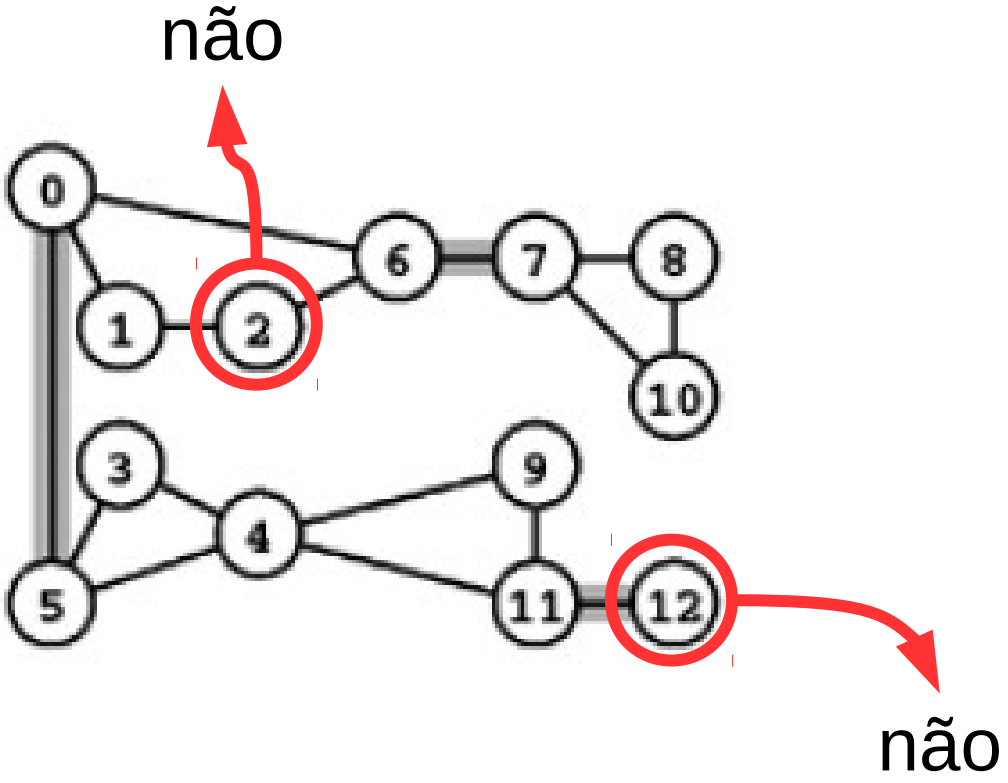
Pontes

- Exemplo



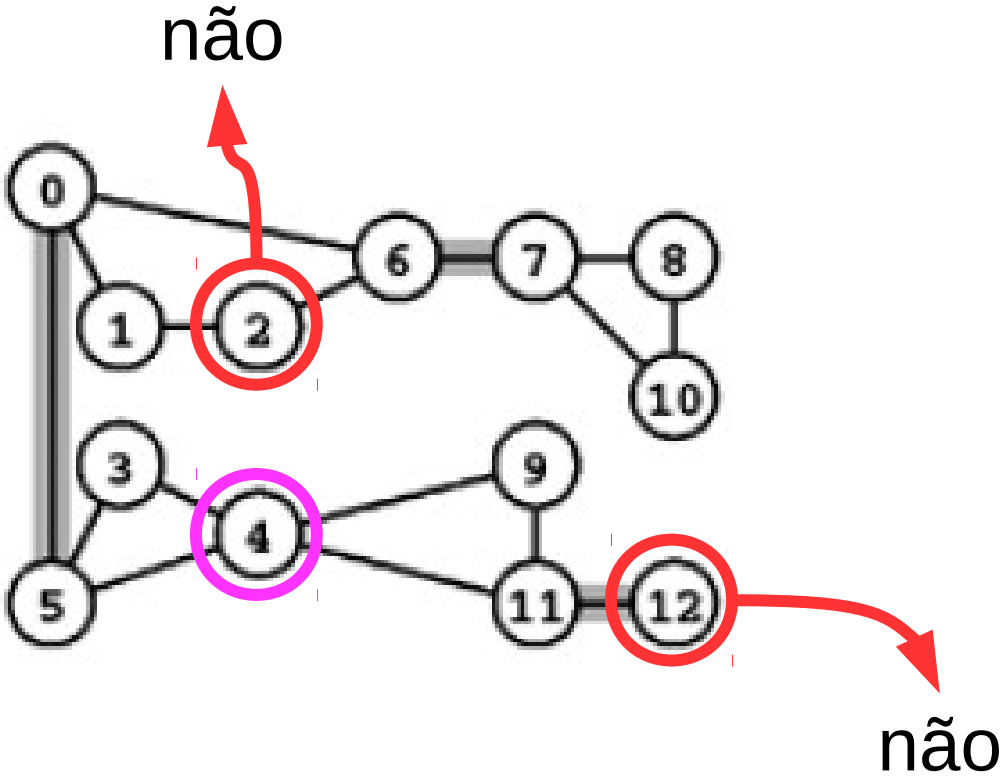
Pontes

- Exemplo



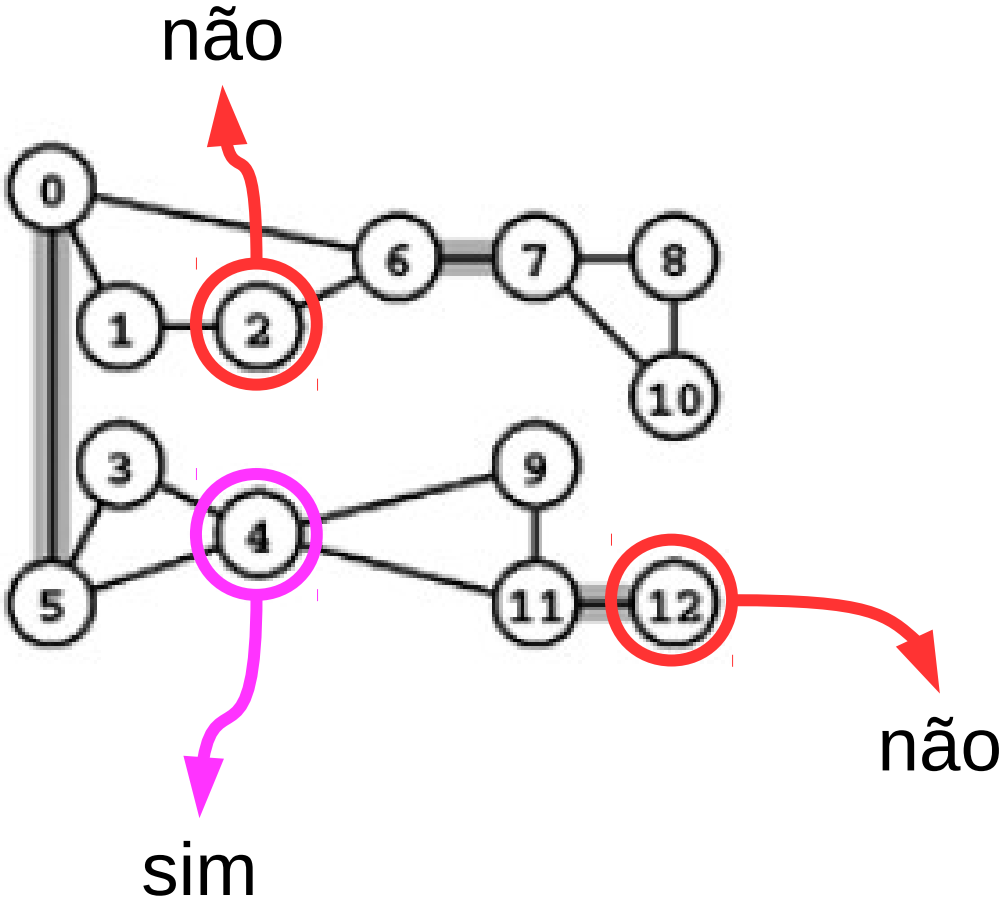
Pontes

- Exemplo



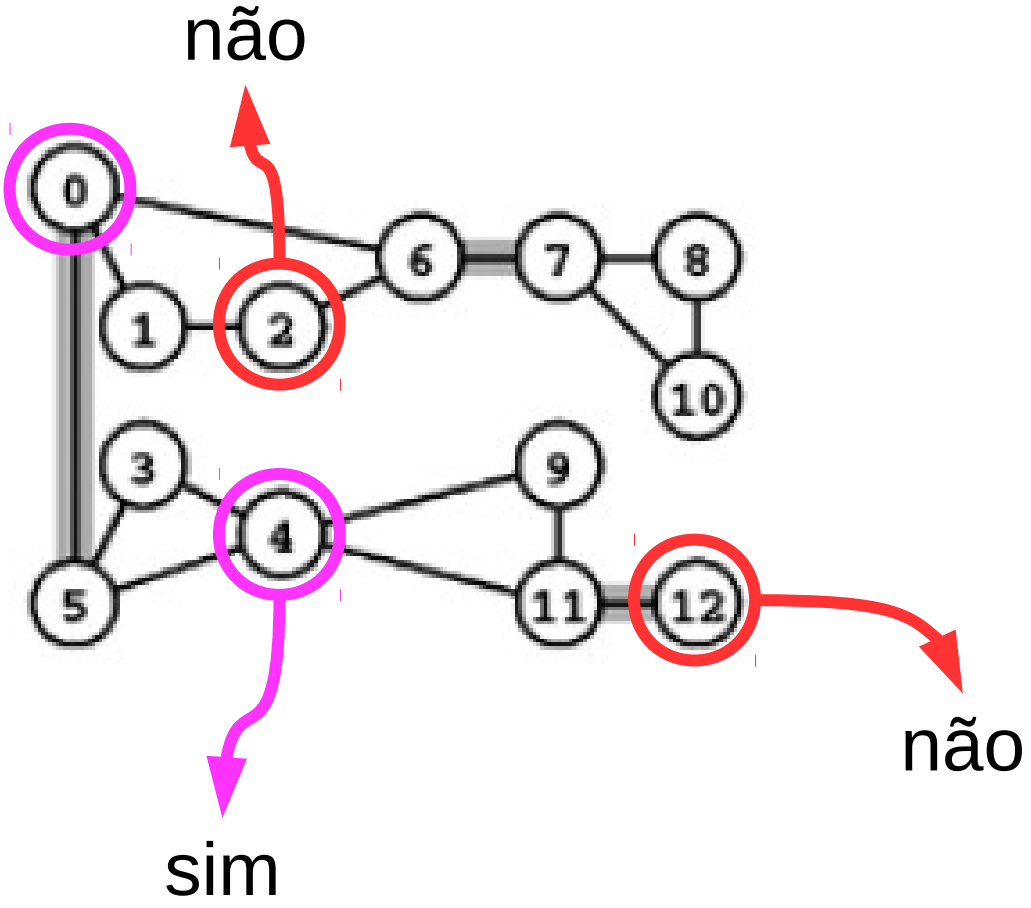
Pontes

- Exemplo



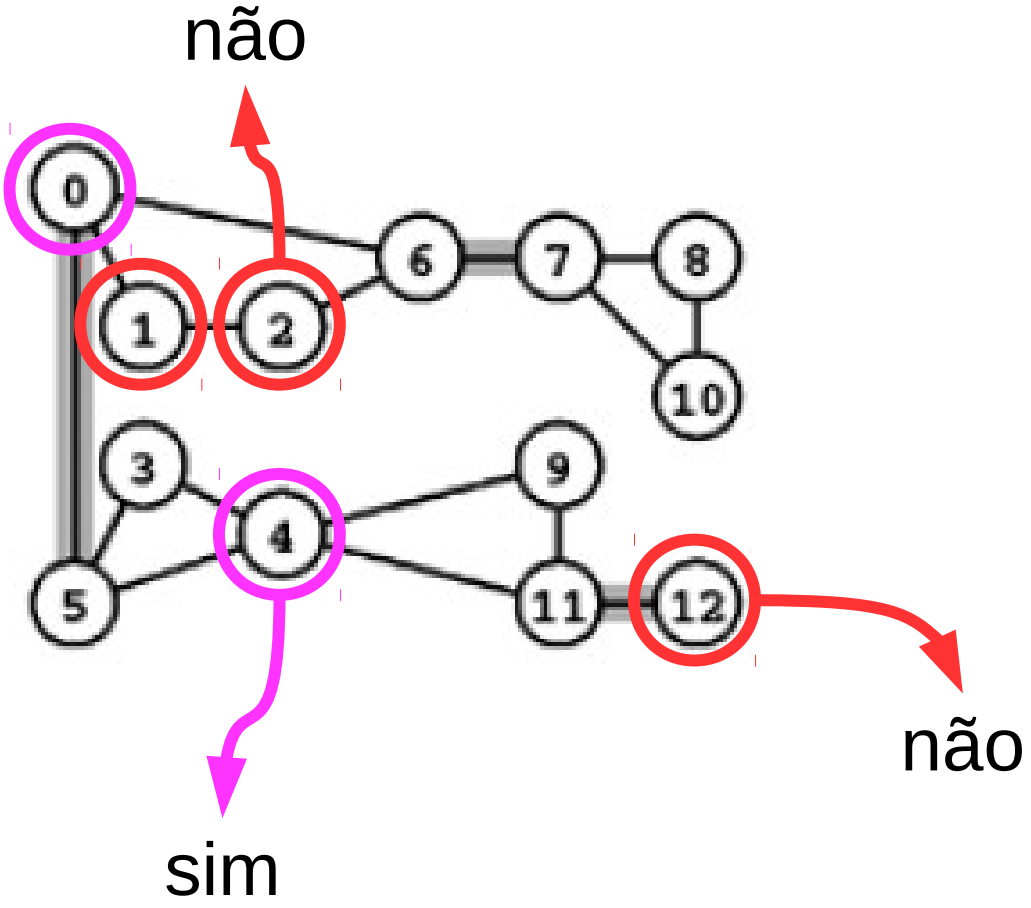
Pontes

- Exemplo



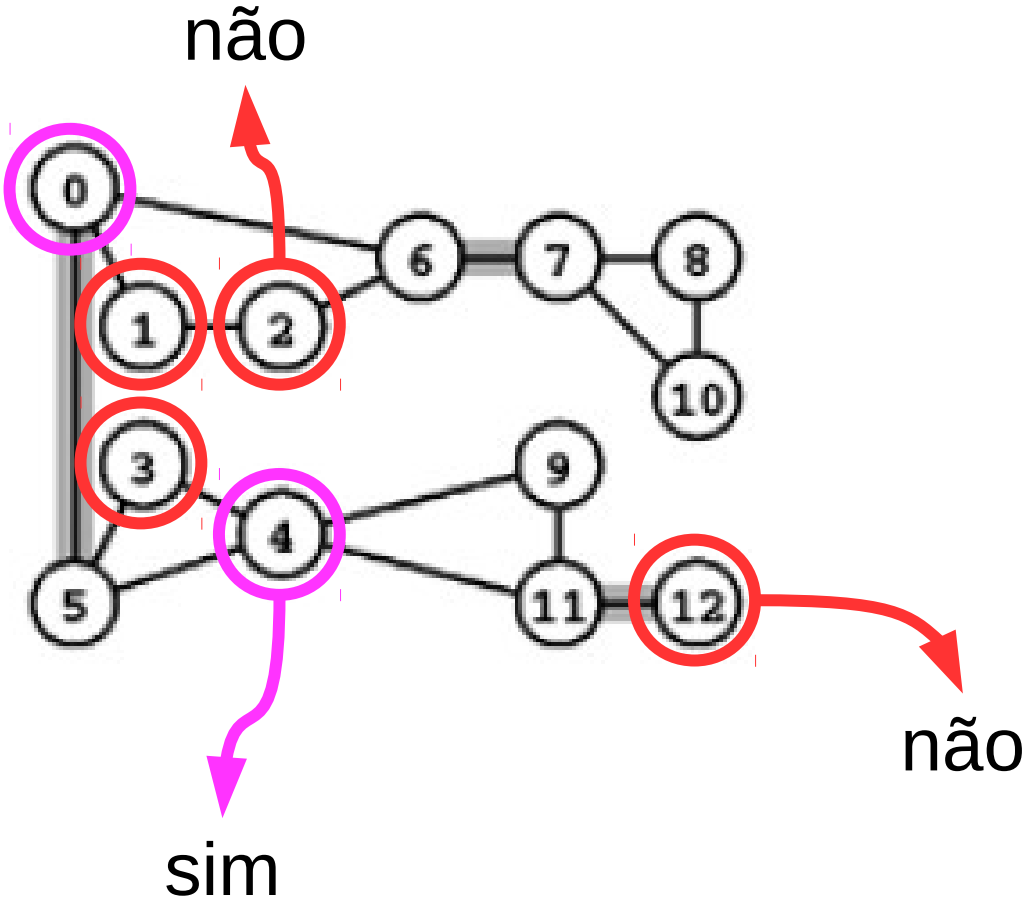
Pontes

- Exemplo



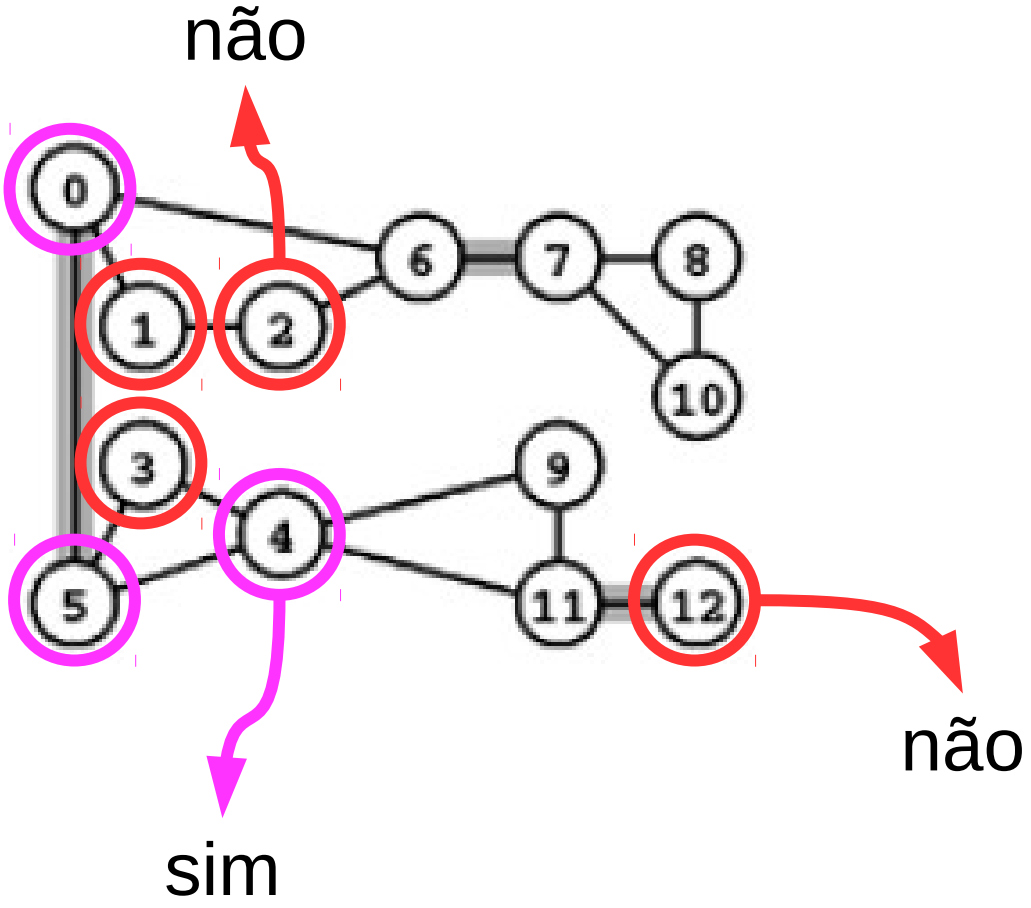
Pontes

- Exemplo



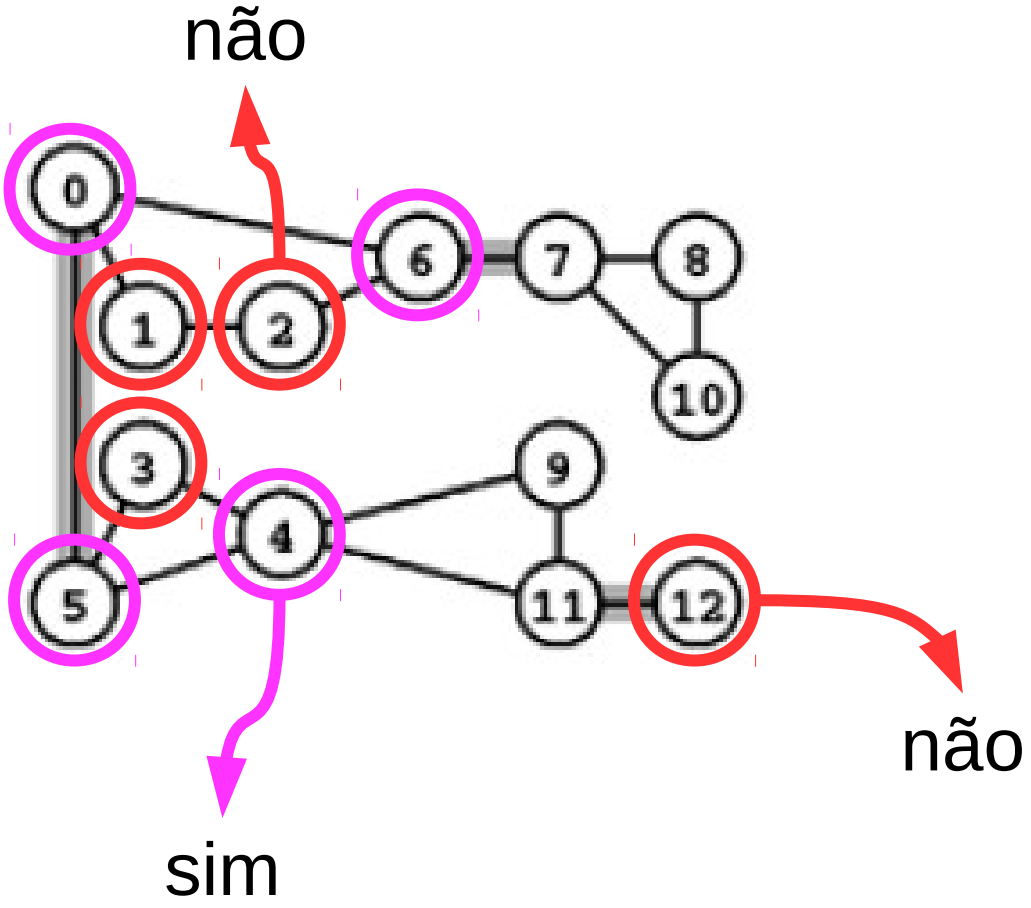
Pontes

- Exemplo



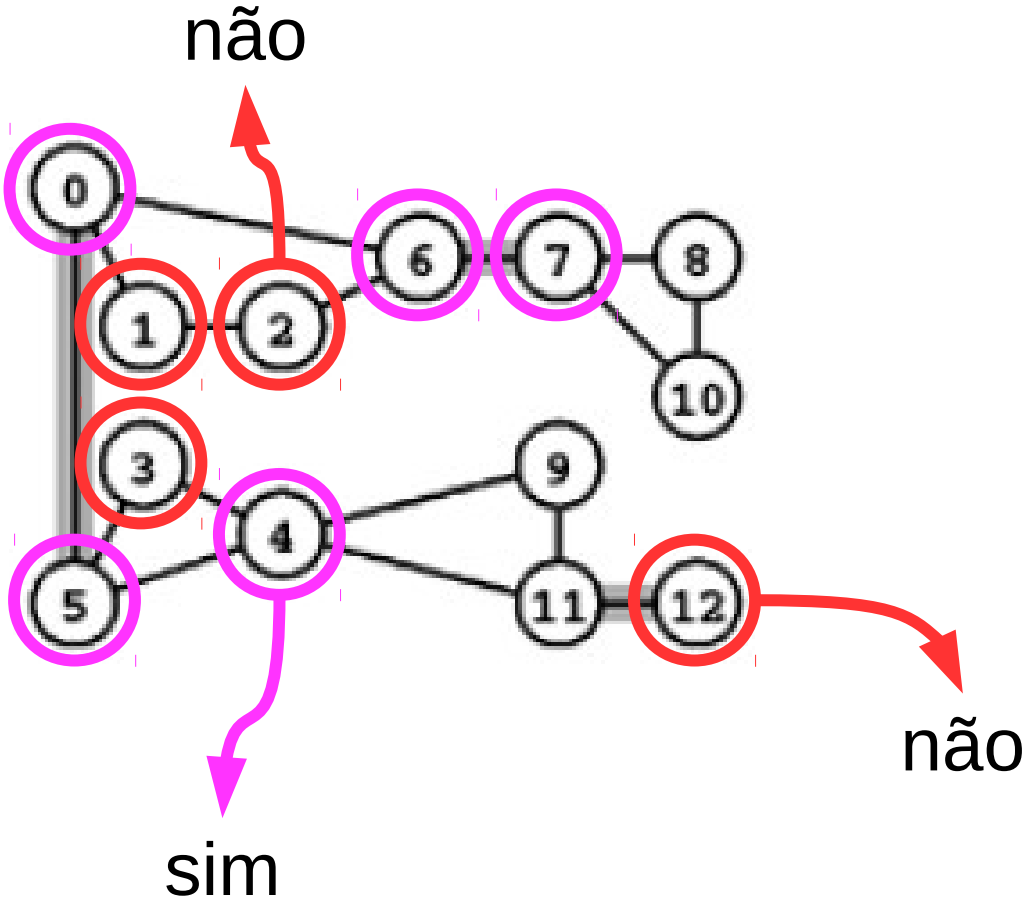
Pontes

- Exemplo



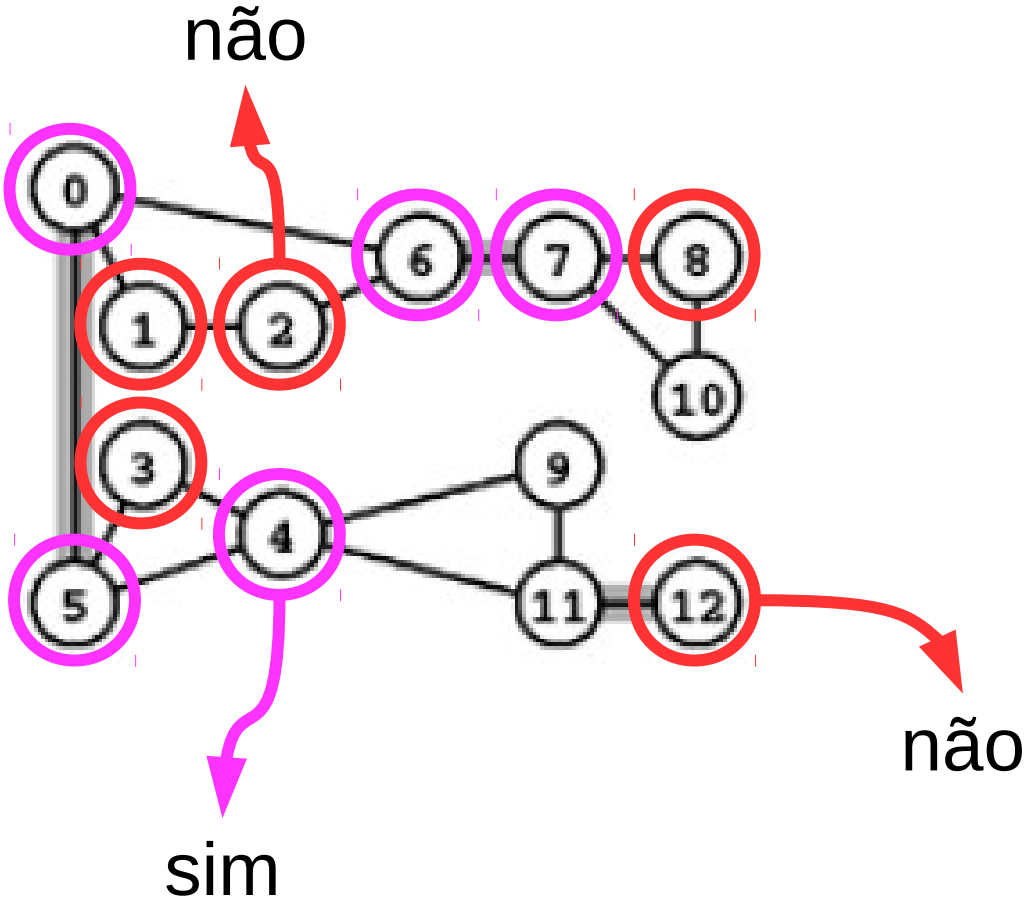
Pontes

- Exemplo



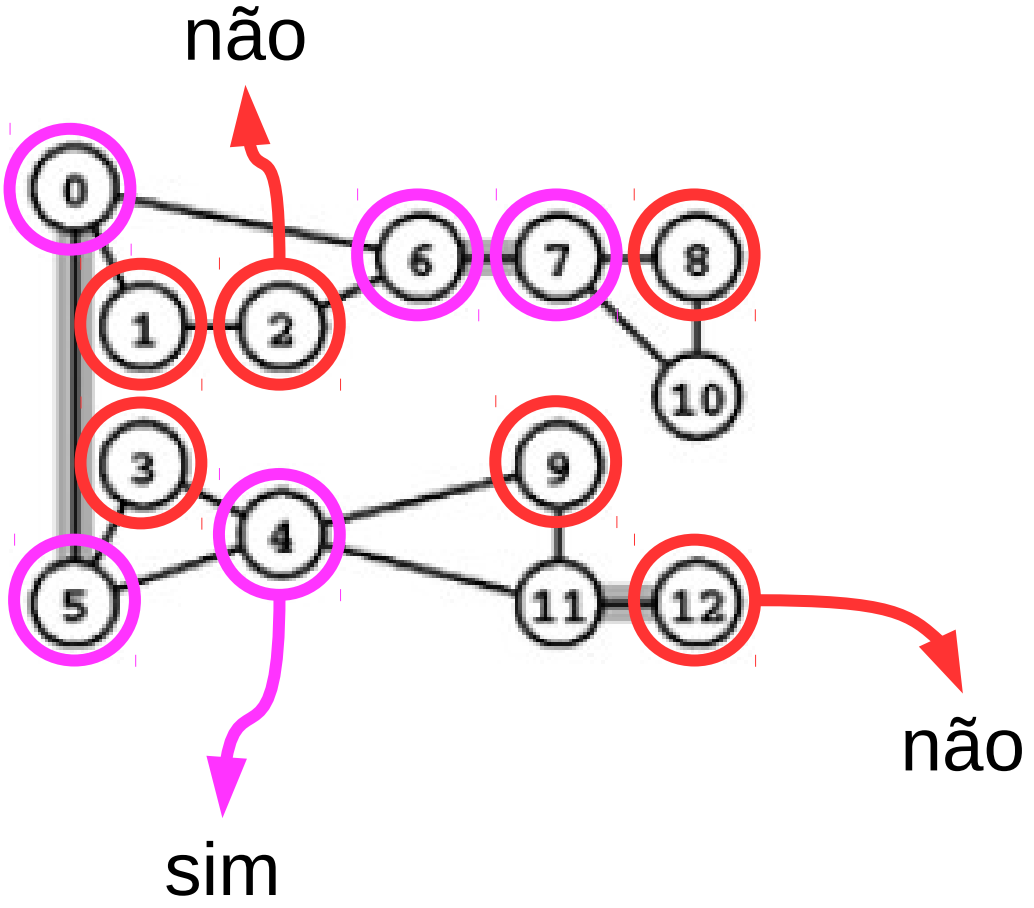
Pontes

- Exemplo



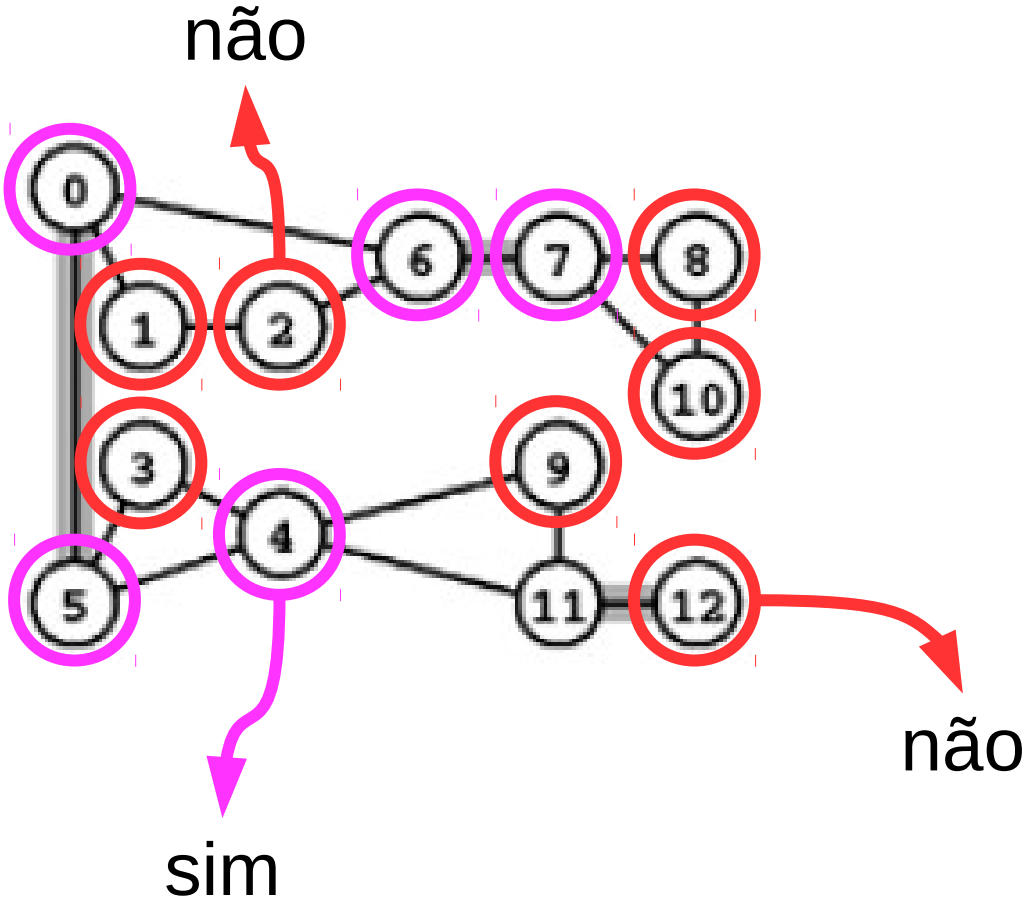
Pontes

- Exemplo



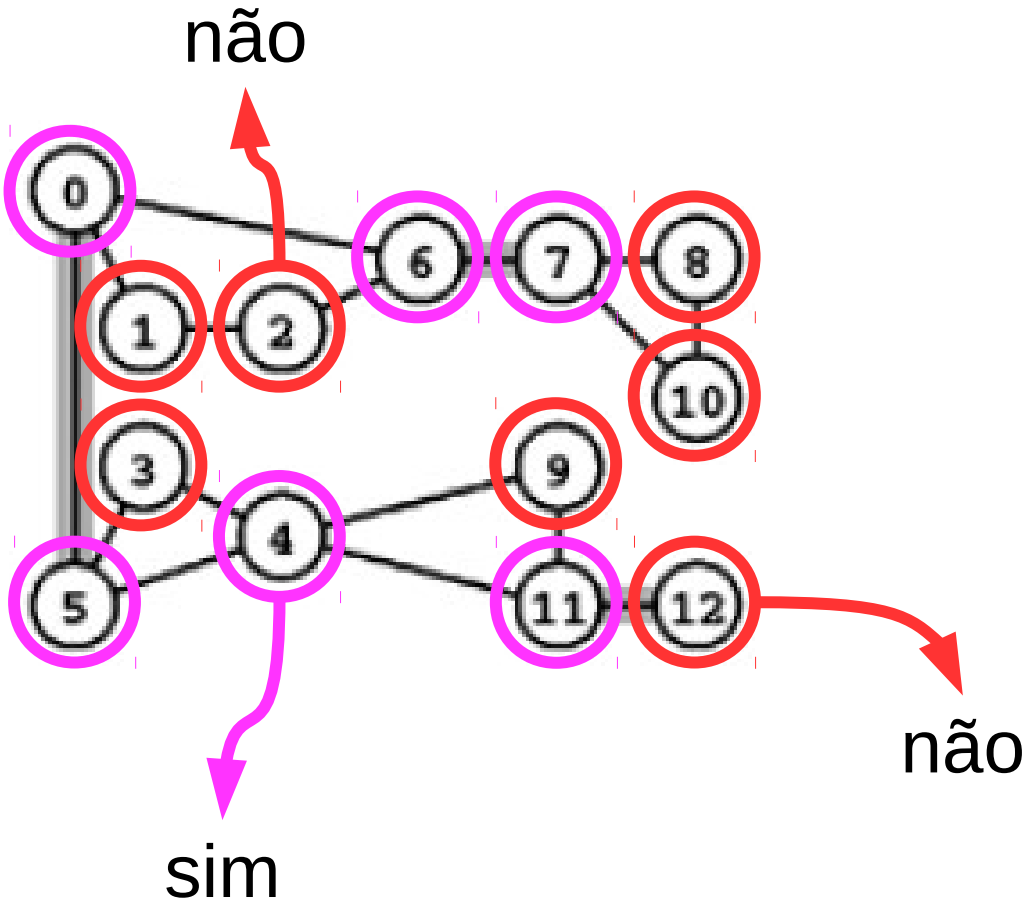
Pontes

- Exemplo



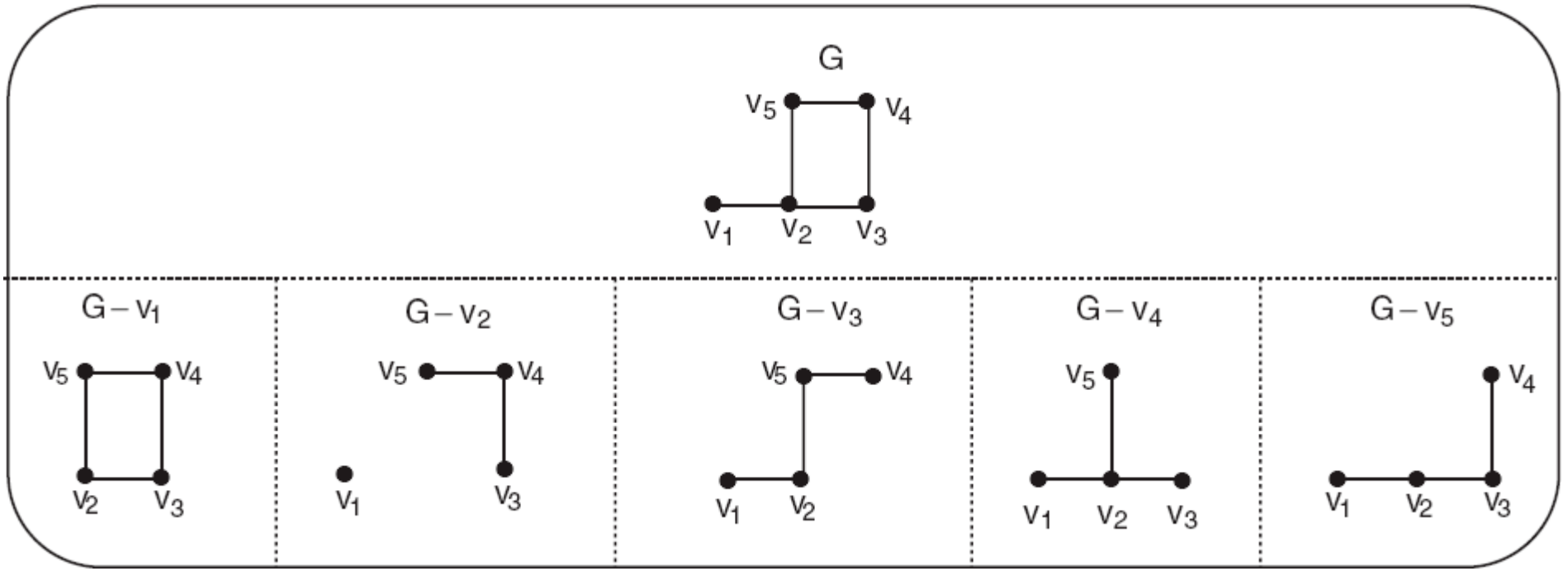
Pontes

- Exemplo



Pontes

- Dado o grafo G:



O único vértice de corte de G é v2.

Pontes

Pontes

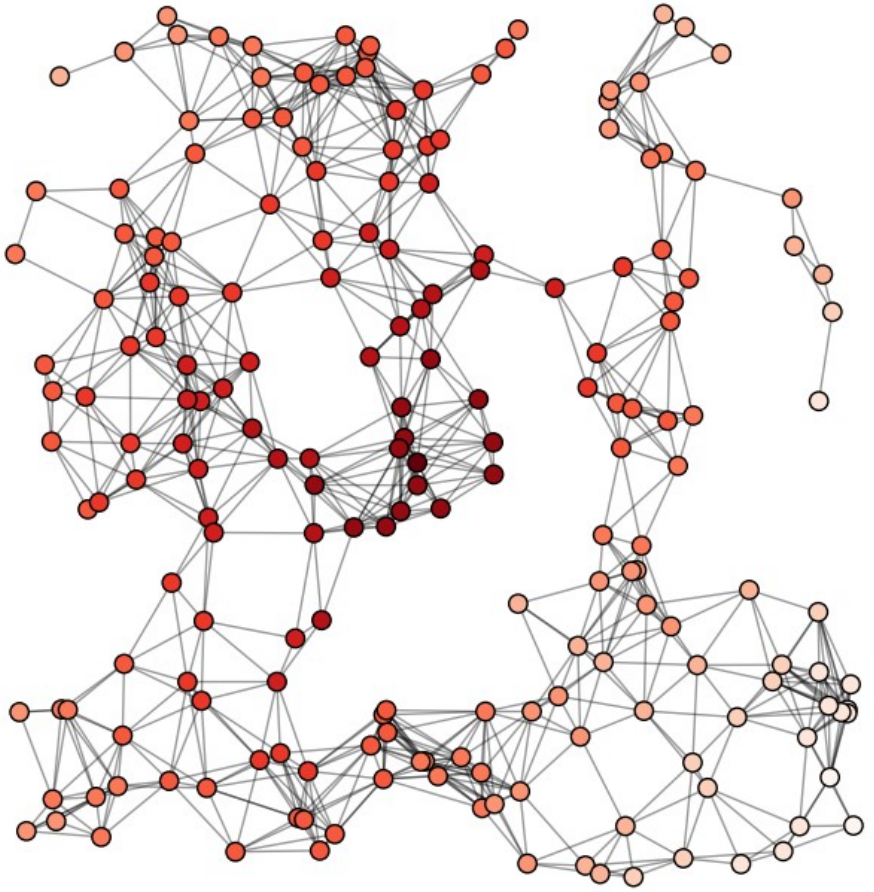
- O problema das articulações.

Pontes

- O problema das articulações.
- Embora possamos decidir rapidamente se um dado vértice de um grafo não-dirigido é uma articulação, o seguinte problema é mais difícil:

Pontes

- Problema: encontrar **todas** as articulações de um grafo não-dirigido.



Pontes

Pontes

- **Definição 6.3**

Pontes

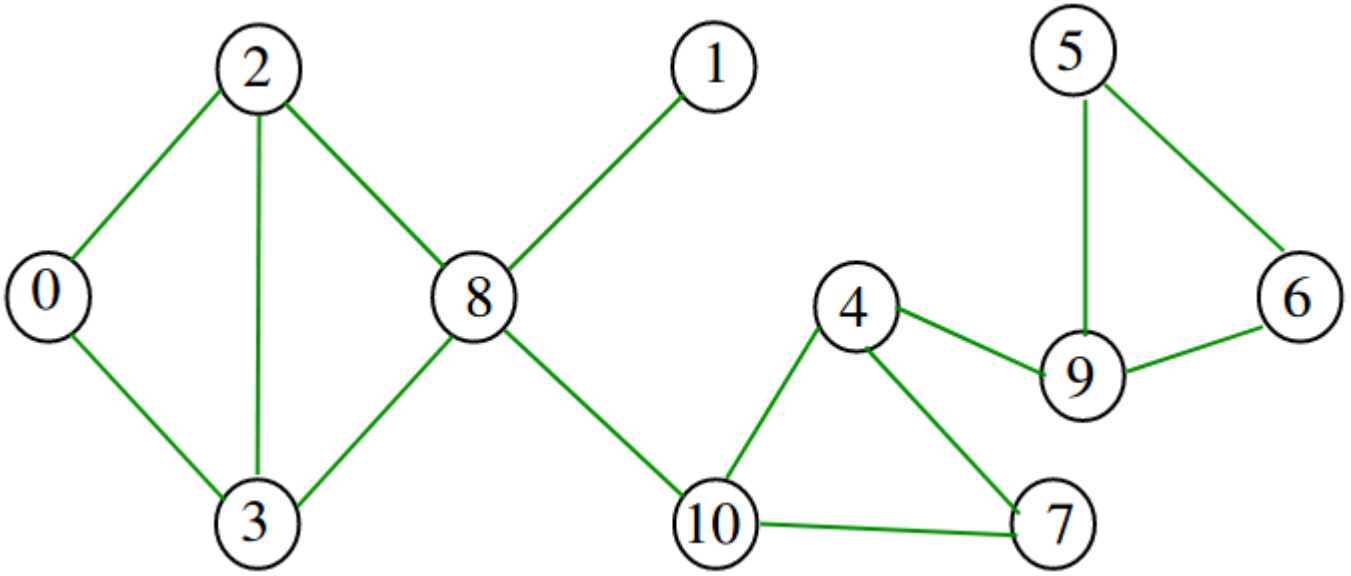
- **Definição 6.3**
  - Seja G = (V,E) um grafo.

Pontes

- **Definição 6.3**
  - Seja G = (V,E) um grafo.
  - Uma aresta e de G é chamada de **ponte** (ou **aresta de corte**) se o subgrafo G−e tem mais componentes conexos que G.

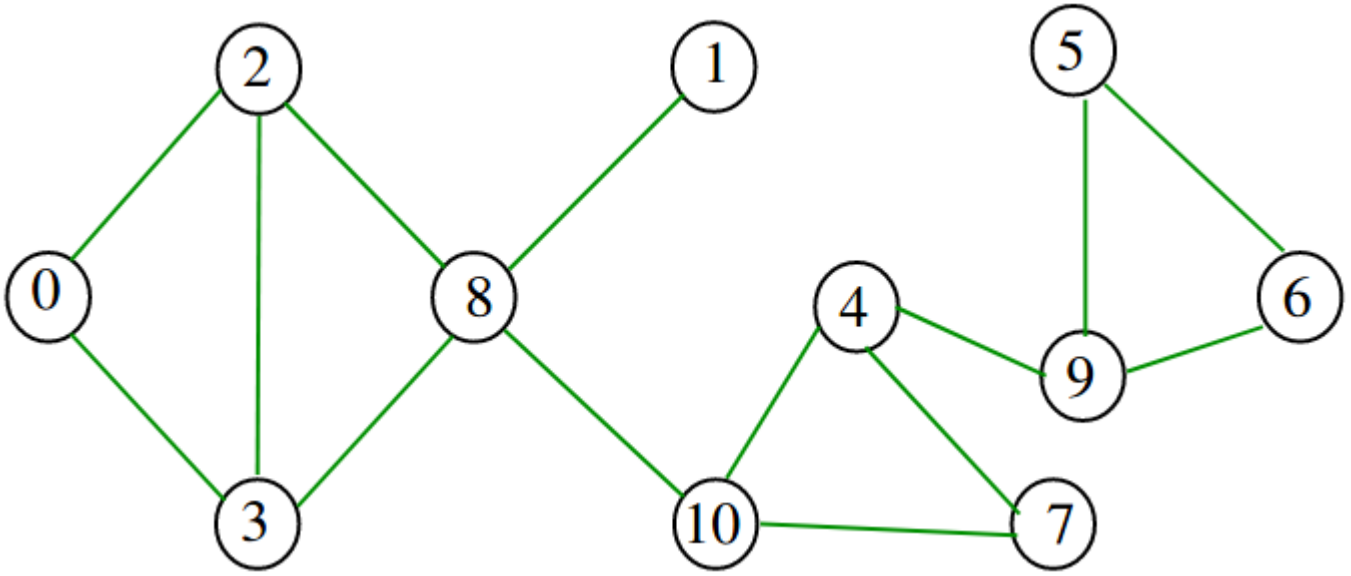
Pontes

- Exemplo



Pontes

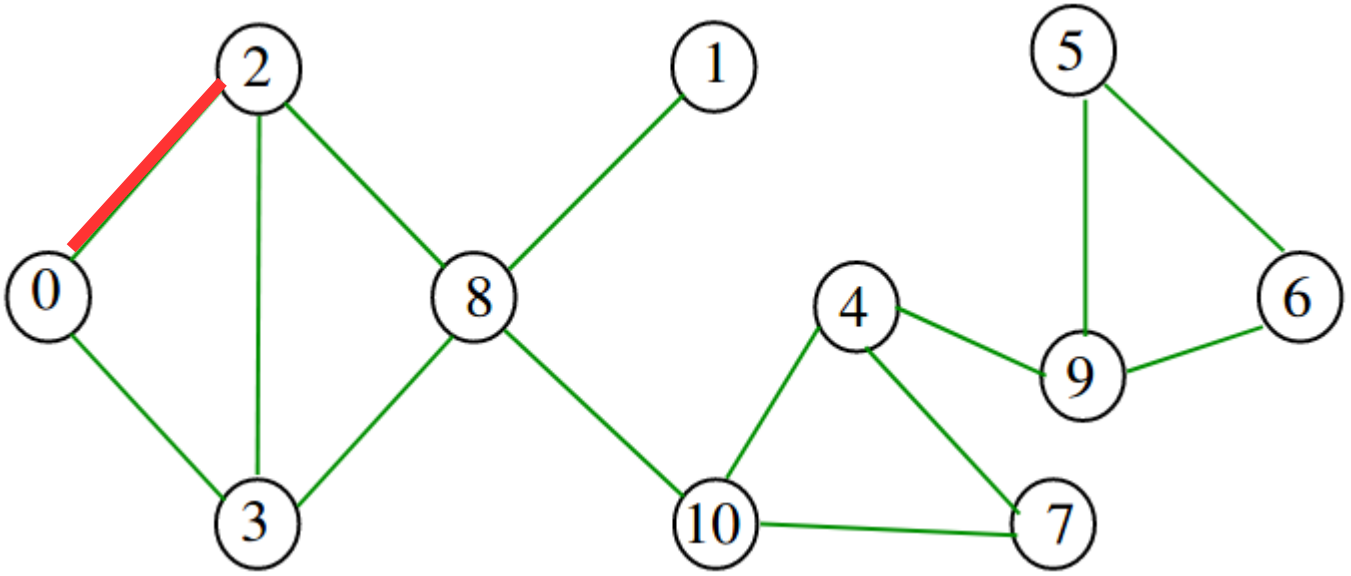
- Exemplo



Quais arestas são pontes?

Pontes

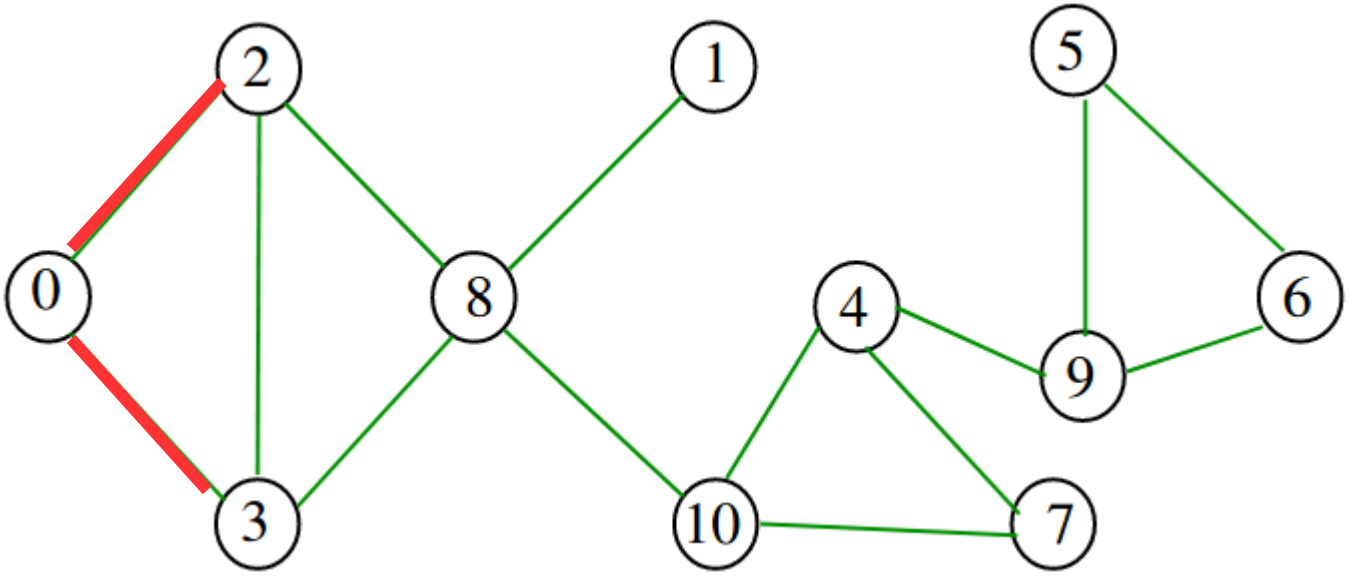
- Exemplo



Quais arestas são pontes?

Pontes

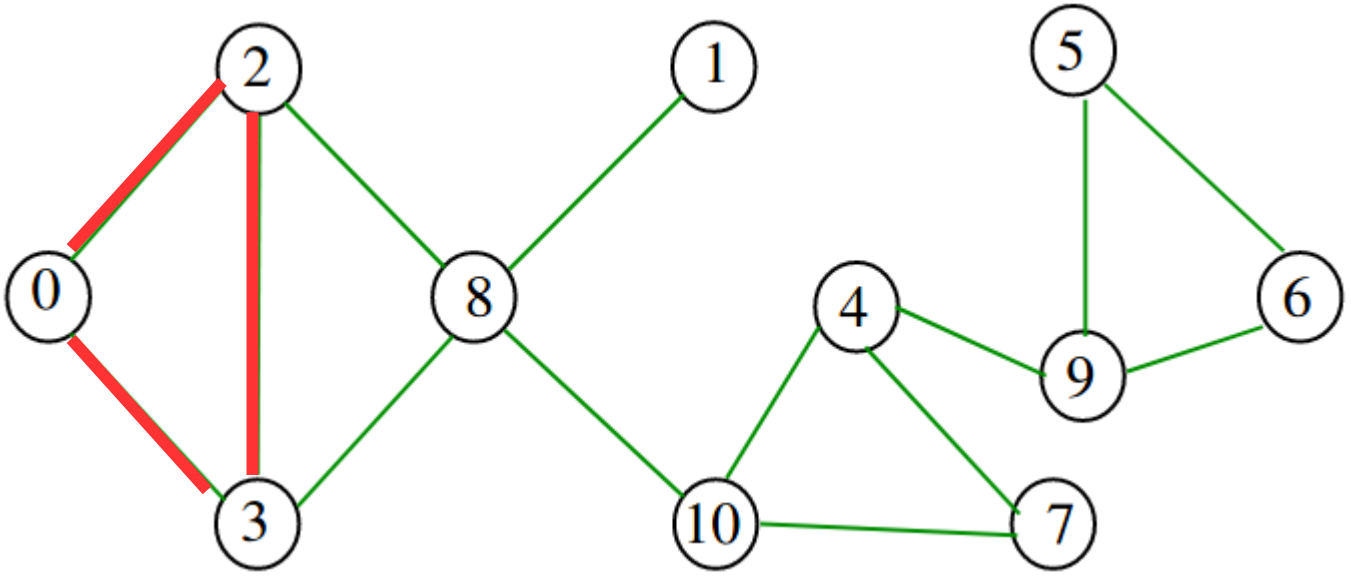
- Exemplo



Quais arestas são pontes?

Pontes

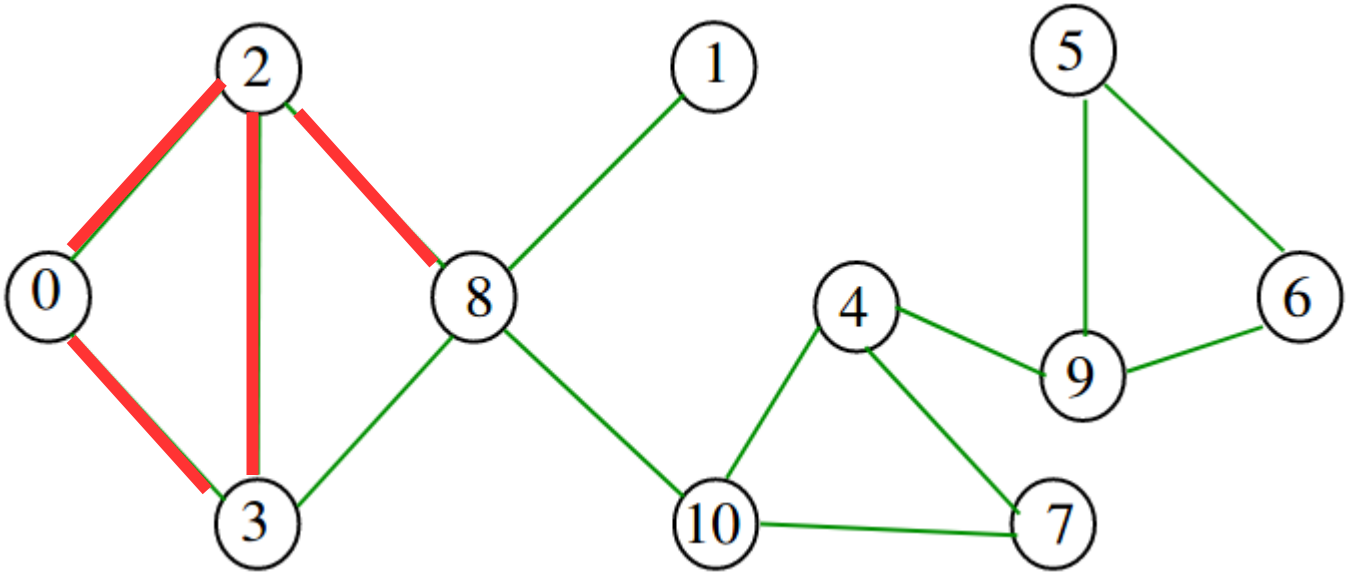
- Exemplo



Quais arestas são pontes?

Pontes

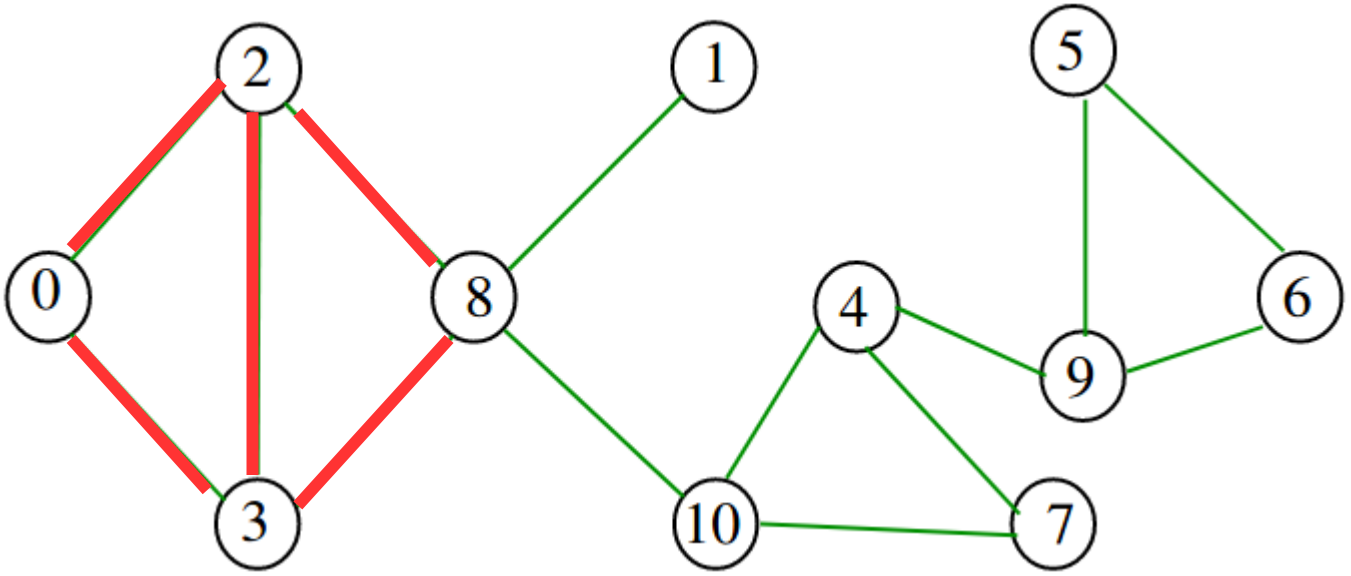
- Exemplo



Quais arestas são pontes?

Pontes

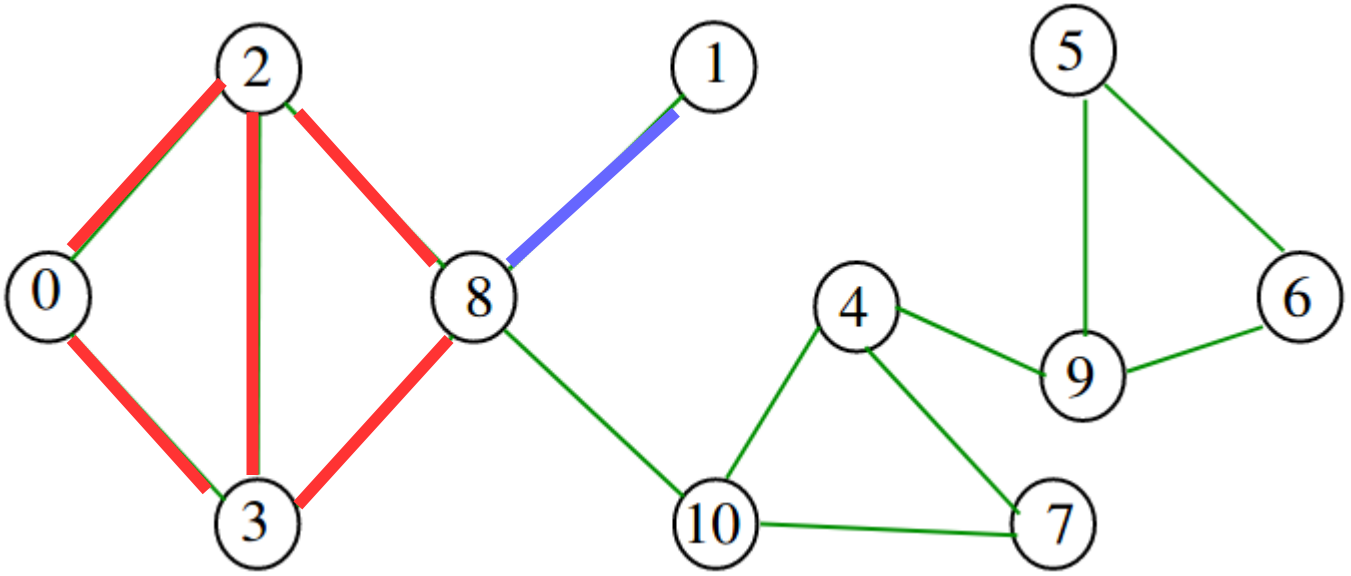
- Exemplo



Quais arestas são pontes?

Pontes

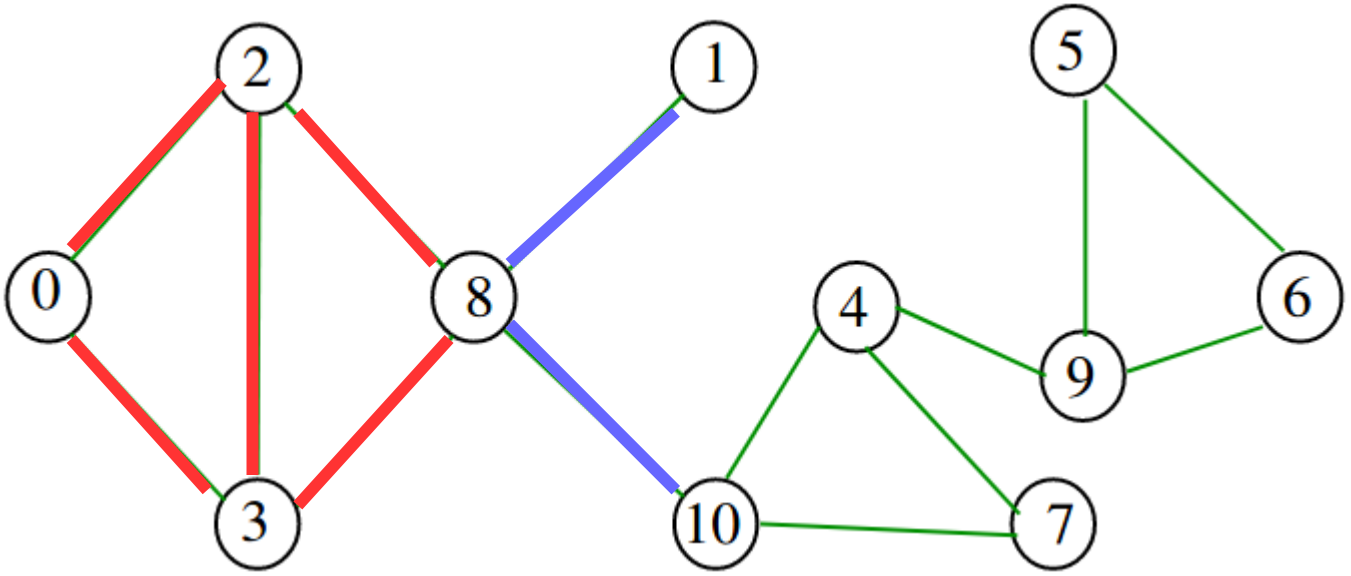
- Exemplo



Quais arestas são pontes?

Pontes

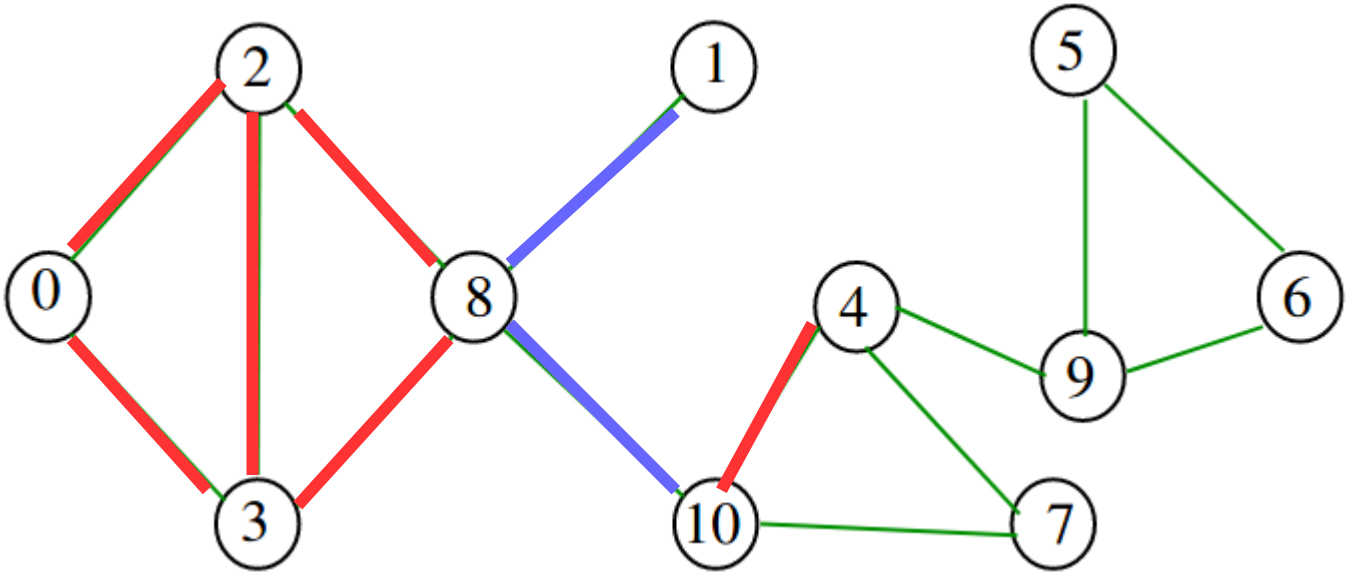
- Exemplo



Quais arestas são pontes?

Pontes

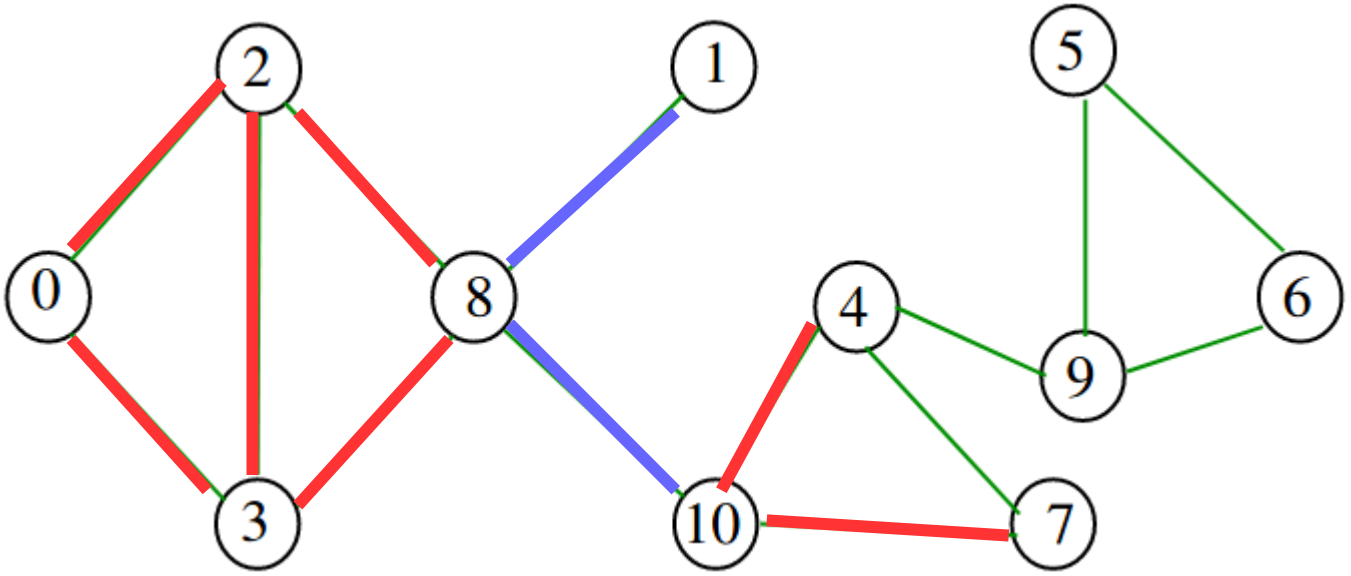
- Exemplo



Quais arestas são pontes?

Pontes

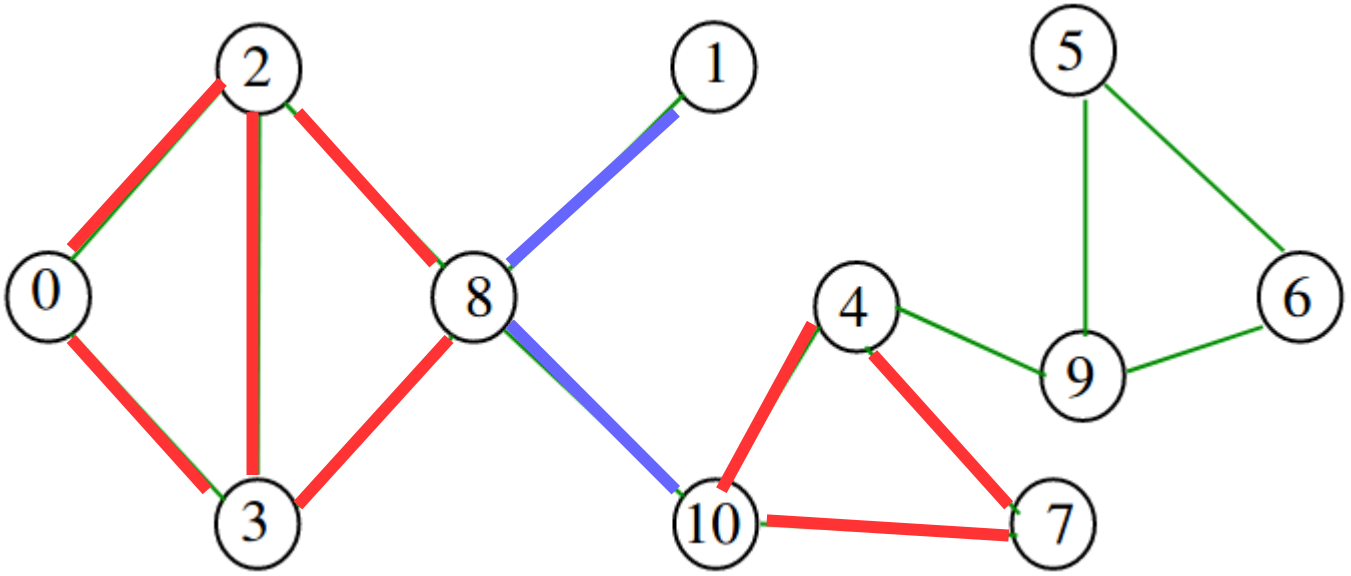
- Exemplo



Quais arestas são pontes?

Pontes

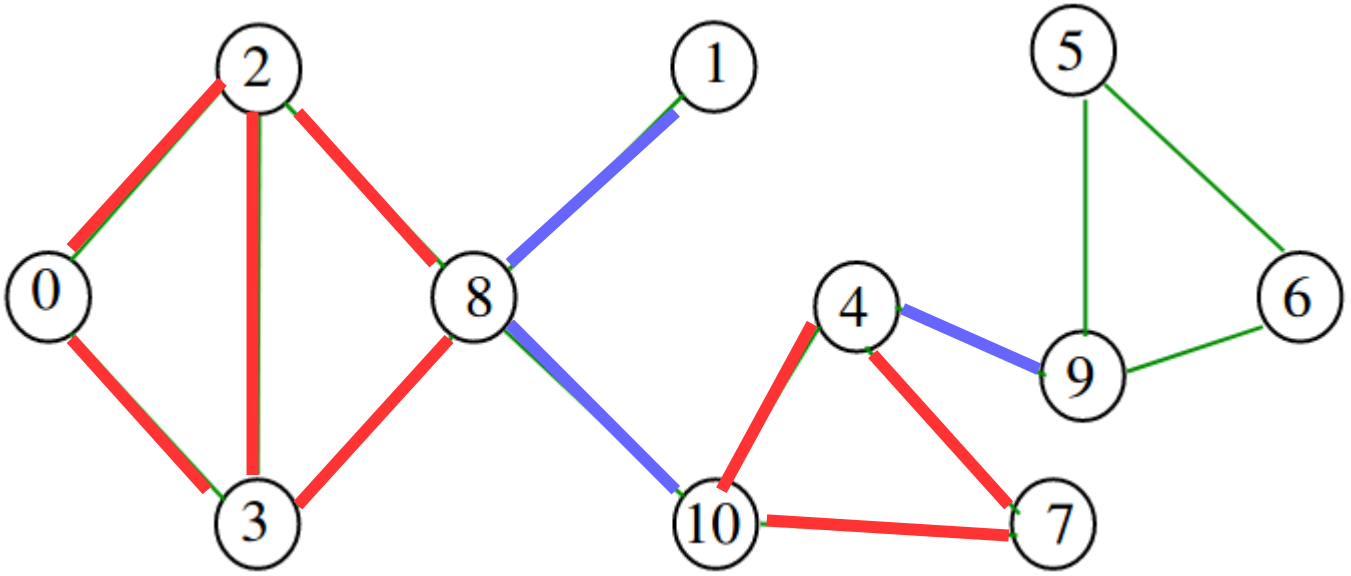
- Exemplo



Quais arestas são pontes?

Pontes

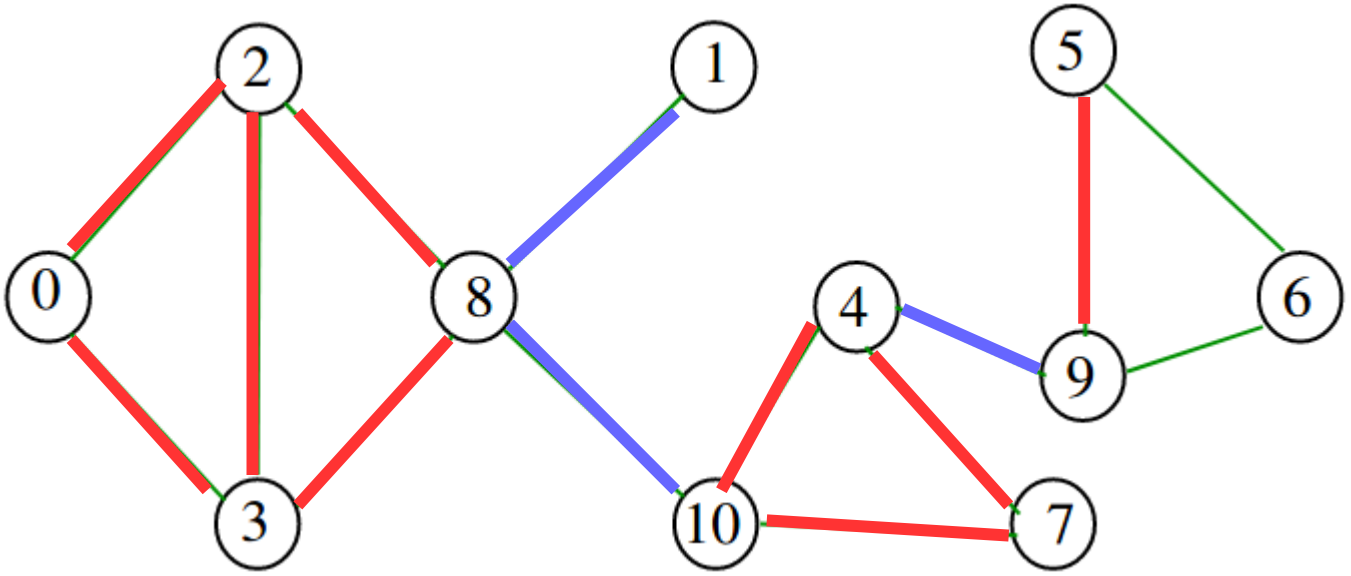
- Exemplo



Quais arestas são pontes?

Pontes

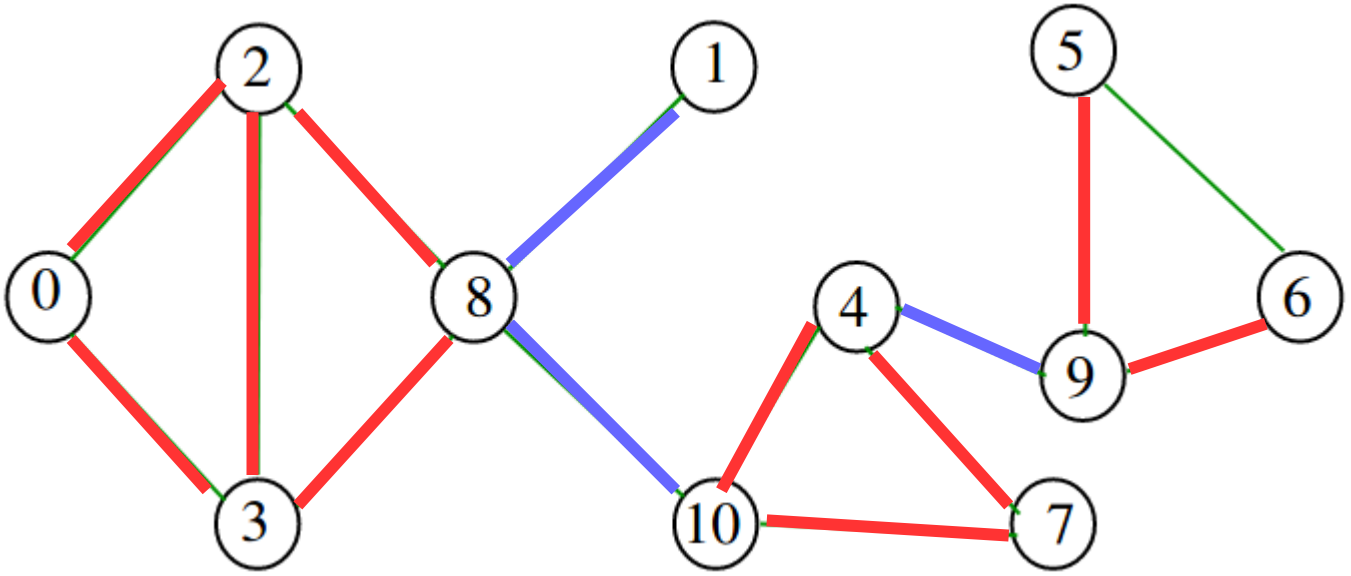
- Exemplo



Quais arestas são pontes?

Pontes

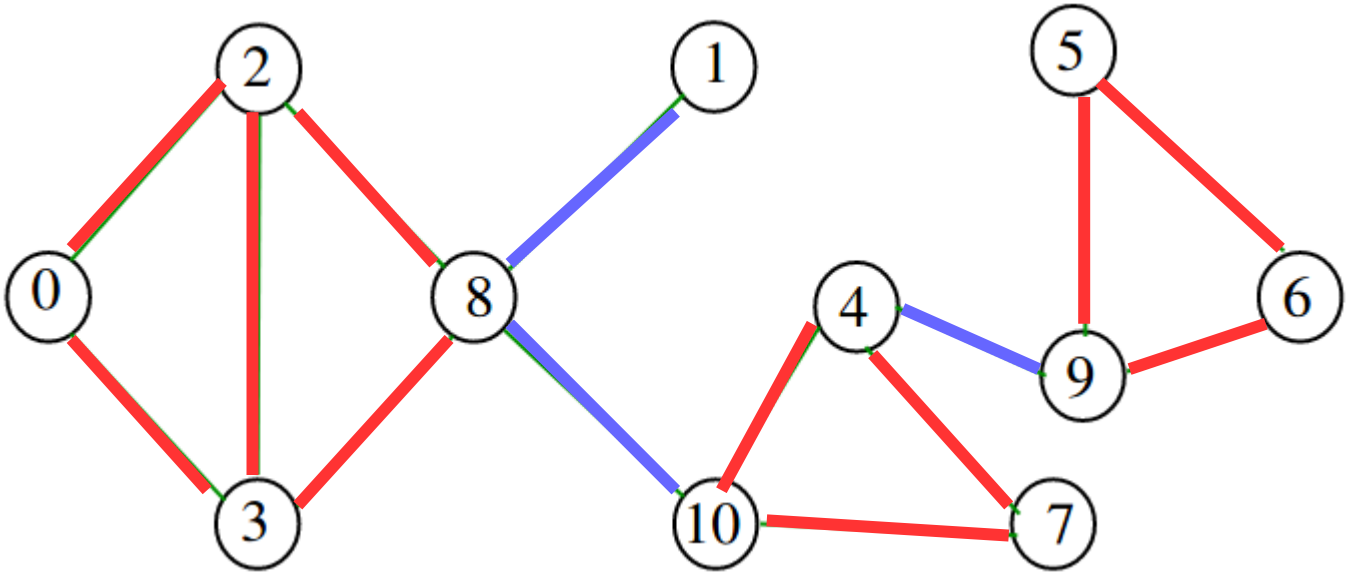
- Exemplo



Quais arestas são pontes?

Pontes

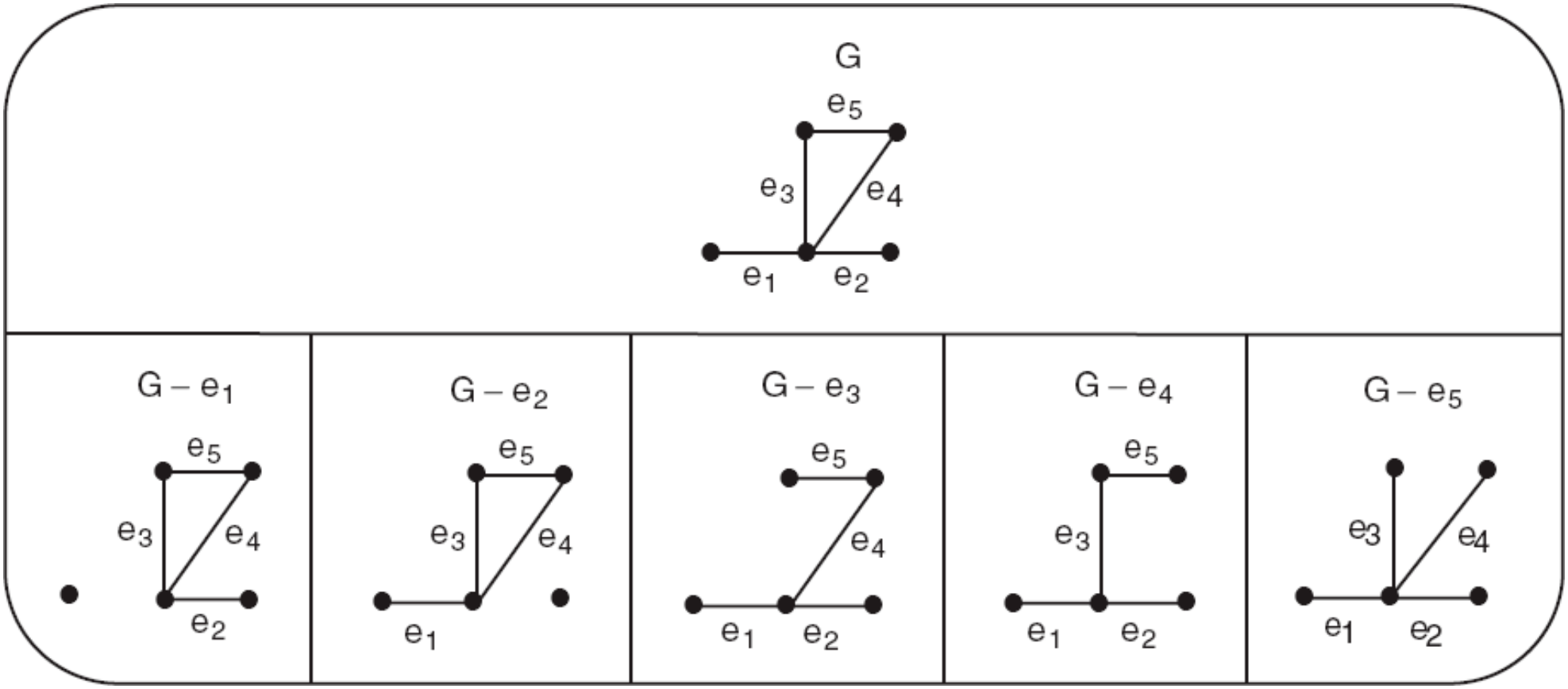
- Exemplo



Quais arestas são pontes?

Pontes

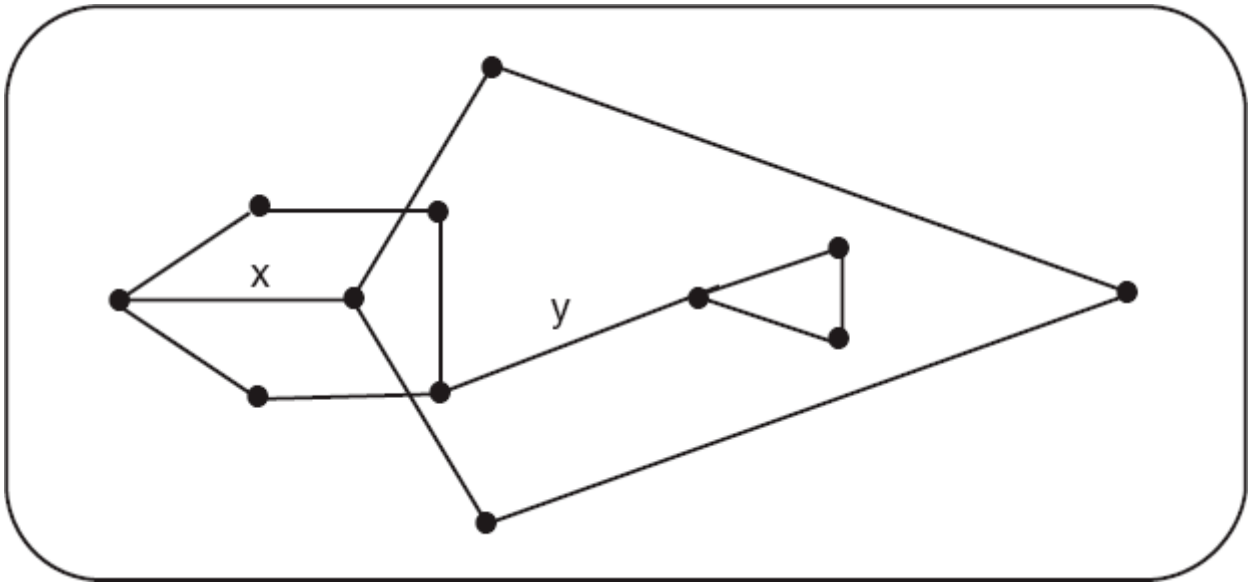
- Dado o grafo G:



As arestas **`e1`** e **`e2`** são arestas de corte (pontes) em G.

Pontes

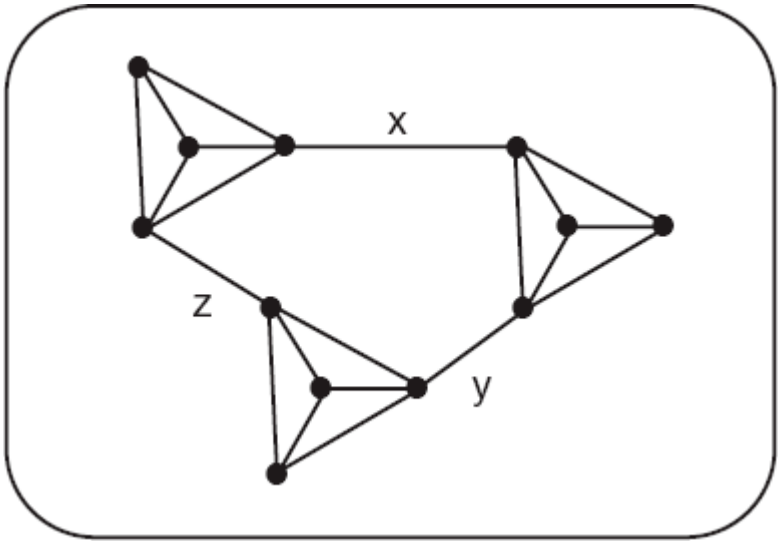
- Dado o grafo G:



As arestas **x** e **y** são arestas de corte (pontes) em G.

Pontes

- Dado o grafo G:



As arestas **`x, y`** e **`z`** são arestas de corte (pontes) em G.

Pontes

Pontes

- **Teorema 6.6**

Pontes

- **Teorema 6.6**
  - Seja G = (V,E) um grafo.

Pontes

- **Teorema 6.6**
  - Seja G = (V,E) um grafo.
  - Uma aresta e de G é uma ponte se e somente se e não fizer parte de nenhum ciclo em G.

Pontes

Pontes

- **Teorema 6.7**

Pontes

- **Teorema 6.7**
  - Seja G um grafo conexo.

Pontes

- **Teorema 6.7**
  - Seja G um grafo conexo.
  - G é uma árvore se e somente se toda aresta de G é uma ponte, ou seja, se e somente se para toda aresta e de G o subgrafo G−e tem dois componentes.

Pontes

Pontes

- **Teorema 6.8**

Pontes

- **Teorema 6.8**
  - Seja G um grafo com n vértices. As seguintes assertivas são equivalentes:

Pontes

- **Teorema 6.8**
  - Seja G um grafo com n vértices. As seguintes assertivas são equivalentes:
    - G é uma árvore.

Pontes

- **Teorema 6.8**
  - Seja G um grafo com n vértices. As seguintes assertivas são equivalentes:
    - G é uma árvore.
    - G é um grafo acíclico com n−1 arestas.

Pontes

- **Teorema 6.8**
  - Seja G um grafo com n vértices. As seguintes assertivas são equivalentes:
    - G é uma árvore.
    - G é um grafo acíclico com n−1 arestas.
    - G é um grafo conexo com n−1 arestas.

Pontes

- **Teorema 6.8**
  - Seja G um grafo com n vértices. As seguintes assertivas são equivalentes:
    - G é uma árvore.
    - G é um grafo acíclico com n−1 arestas.
    - G é um grafo conexo com n−1 arestas.
    - T é conexo e cada aresta é uma ponte.

Pontes

- **Teorema 6.8**
  - Seja G um grafo com n vértices. As seguintes assertivas são equivalentes:
    - G é uma árvore.
    - G é um grafo acíclico com n−1 arestas.
    - G é um grafo conexo com n−1 arestas.
    - T é conexo e cada aresta é uma ponte.
    - quaisquer dois vértices de T são conectados por exatamente um caminho.

Pontes

- **Teorema 6.8**
  - Seja G um grafo com n vértices. As seguintes assertivas são equivalentes:
    - G é uma árvore.
    - G é um grafo acíclico com n−1 arestas.
    - G é um grafo conexo com n−1 arestas.
    - T é conexo e cada aresta é uma ponte.
    - quaisquer dois vértices de T são conectados por exatamente um caminho.
    - T não contém ciclos, mas a adição de qualquer nova aresta cria exatamente um ciclo.

ÁRVORES SPANNING E O PROBLEMA DA ÁRVORE SPANNING MINIMAL

Spanning Tree

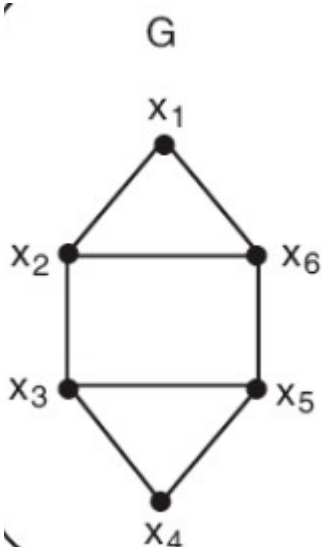
- Definição 3.12
  - G1 = (V1,E1) é um subgrafo spanning de G = (V,E) se G1 for um subgrafo de G, tal que V1 = V, ou seja, G1 e G têm exatamente o mesmo conjunto de vértices.

Spanning Tree

- Definição 3.12
  - G1 = (V1,E1) é um subgrafo spanning de G = (V,E) se G1 for um subgrafo de G, tal que V1 = V, ou seja, G1 e G têm exatamente o mesmo conjunto de vértices.
- Exemplos:

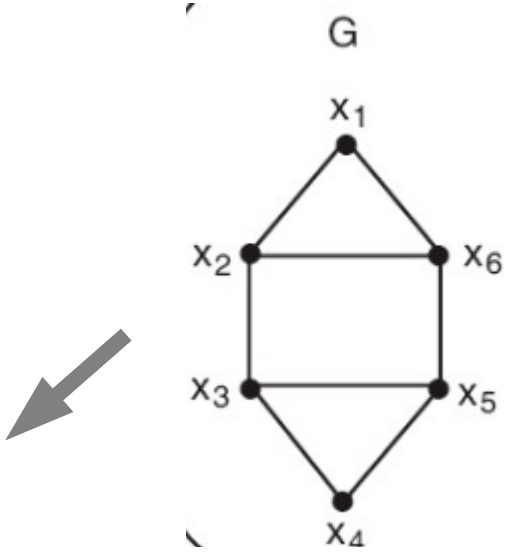
Spanning Tree

- Definição 3.12
  - G1 = (V1,E1) é um subgrafo spanning de G = (V,E) se G1 for um subgrafo de G, tal que V1 = V, ou seja, G1 e G têm exatamente o mesmo conjunto de vértices.
- Exemplos:



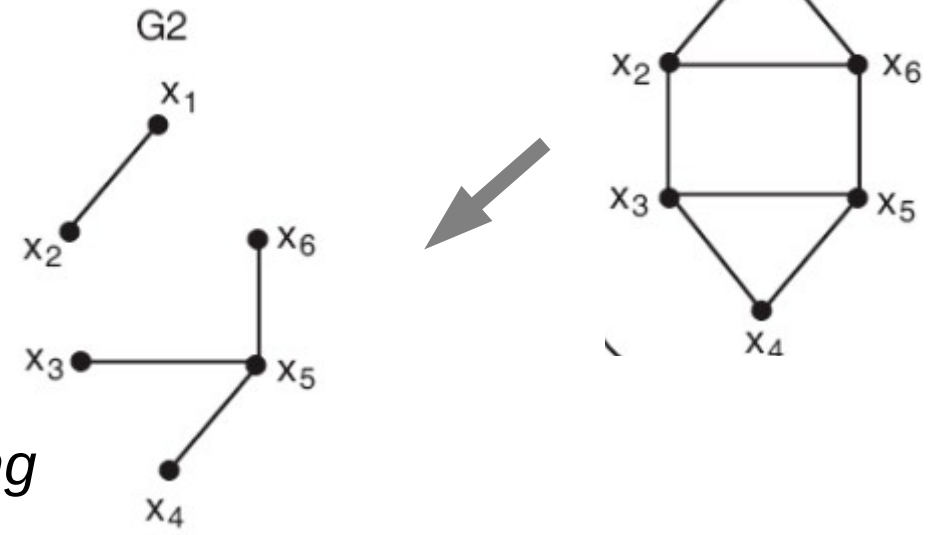
Spanning Tree

- Definição 3.12
  - G1 = (V1,E1) é um subgrafo spanning de G = (V,E) se G1 for um subgrafo de G, tal que V1 = V, ou seja, G1 e G têm exatamente o mesmo conjunto de vértices.
- Exemplos:



Spanning Tree

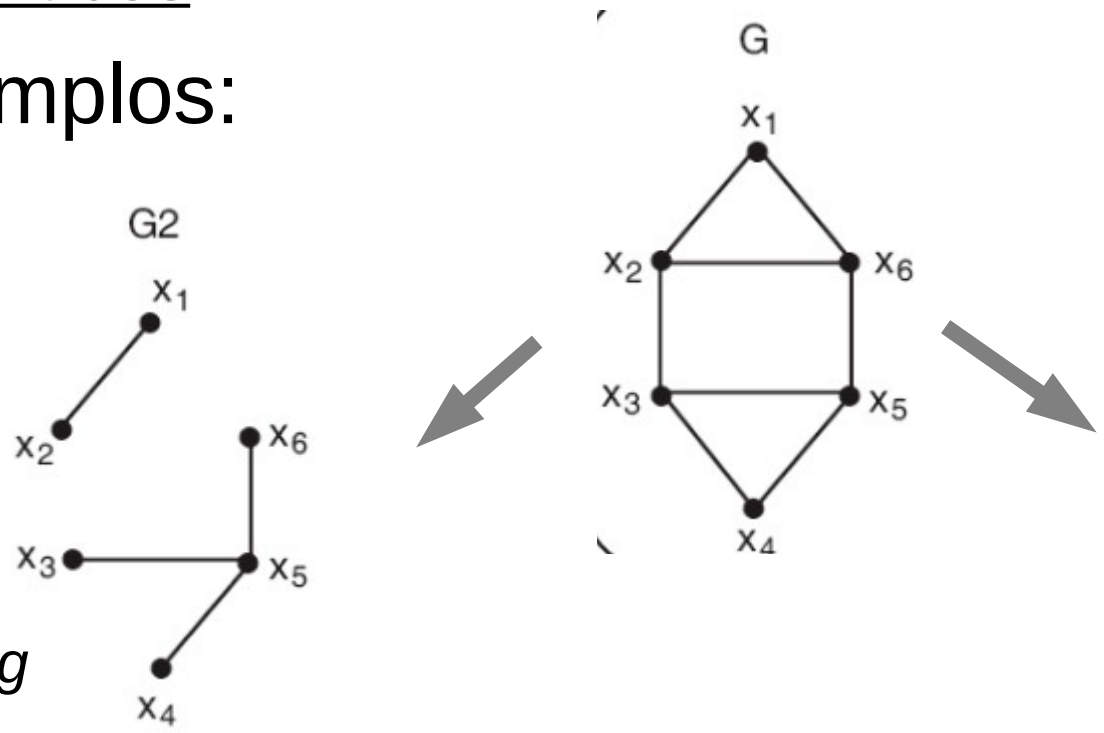
- Definição 3.12
  - G1 = (V1,E1) é um subgrafo spanning de G = (V,E) se G1 for um subgrafo de G, tal que V1 = V, ou seja, G1 e G têm exatamente o mesmo conjunto de vértices.
- Exemplos:



spanning

Spanning Tree

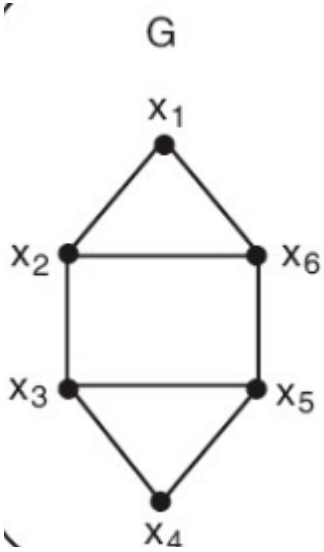
- Definição 3.12
  - G1 = (V1,E1) é um subgrafo spanning de G = (V,E) se G1 for um subgrafo de G, tal que V1 = V, ou seja, G1 e G têm exatamente o mesmo conjunto de vértices.
- Exemplos:

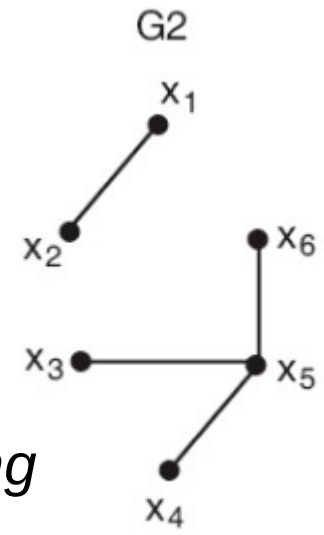


spanning

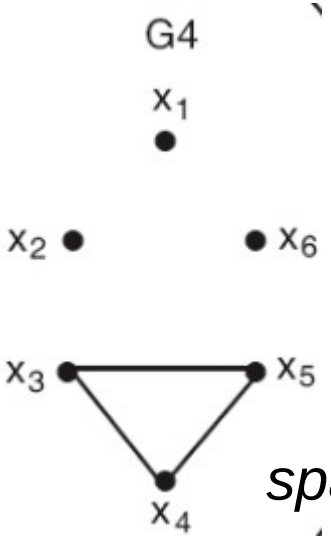
Spanning Tree

- Definição 3.12
  - G1 = (V1,E1) é um subgrafo spanning de G = (V,E) se G1 for um subgrafo de G, tal que V1 = V, ou seja, G1 e G têm exatamente o mesmo conjunto de vértices.
- Exemplos:





spanning



spanning

Spanning Tree

Spanning Tree

- **Definição 6.4**

Spanning Tree

- **Definição 6.4**
  - Seja G = (V,E) um grafo. Uma árvore *spanning* do grafo G é um subgrafo spanning de G, que é uma árvore.

Spanning Tree

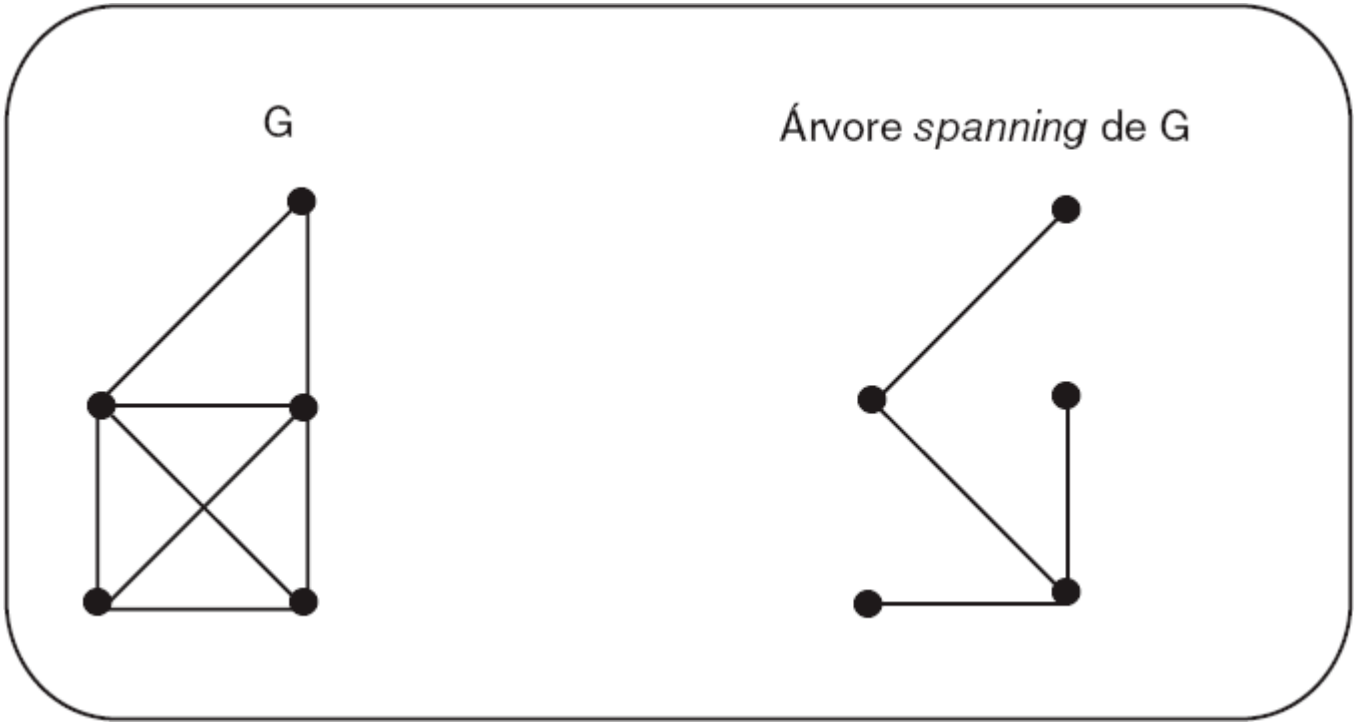
- **Definição 6.4**
  - Seja G = (V,E) um grafo. Uma árvore *spanning* do grafo G é um subgrafo spanning de G, que é uma árvore.
- **Teorema 6.9**

Spanning Tree

- **Definição 6.4**
  - Seja G = (V,E) um grafo. Uma árvore *spanning* do grafo G é um subgrafo spanning de G, que é uma árvore.
- **Teorema 6.9**
  - Um grafo G é conexo se e somente se tem uma árvore *spanning*.

Spanning Tree

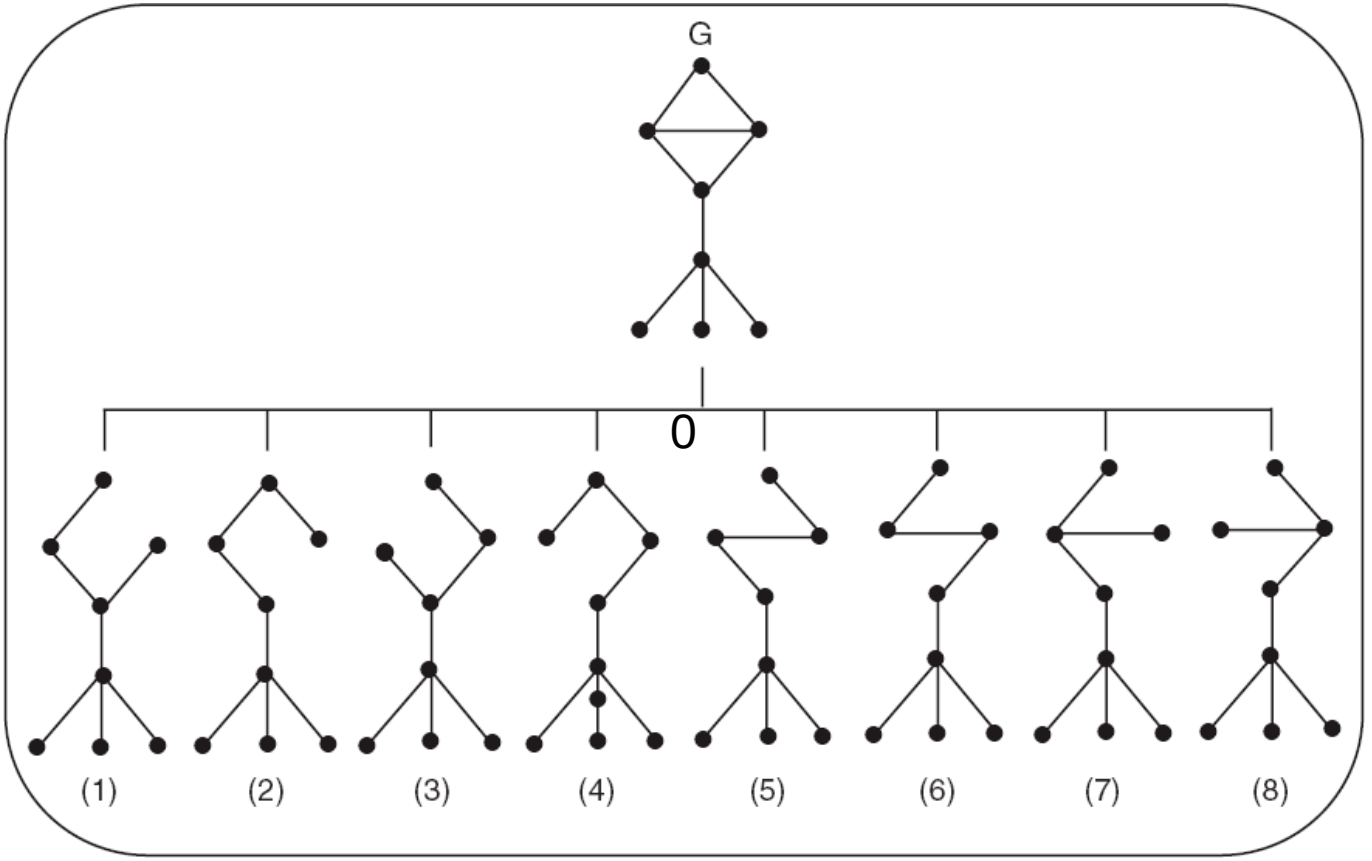
- Exemplo



Um grafo G e uma árvore spanning de G.

Spanning Tree

- Exemplo
  - Conjunto de todas as árvores *spanning*, inclusive as isomorfas, do grafo G.



Spanning Tree

Spanning Tree

- **Teorema 6.10 (Teorema de Cayley, 1889)**

Spanning Tree

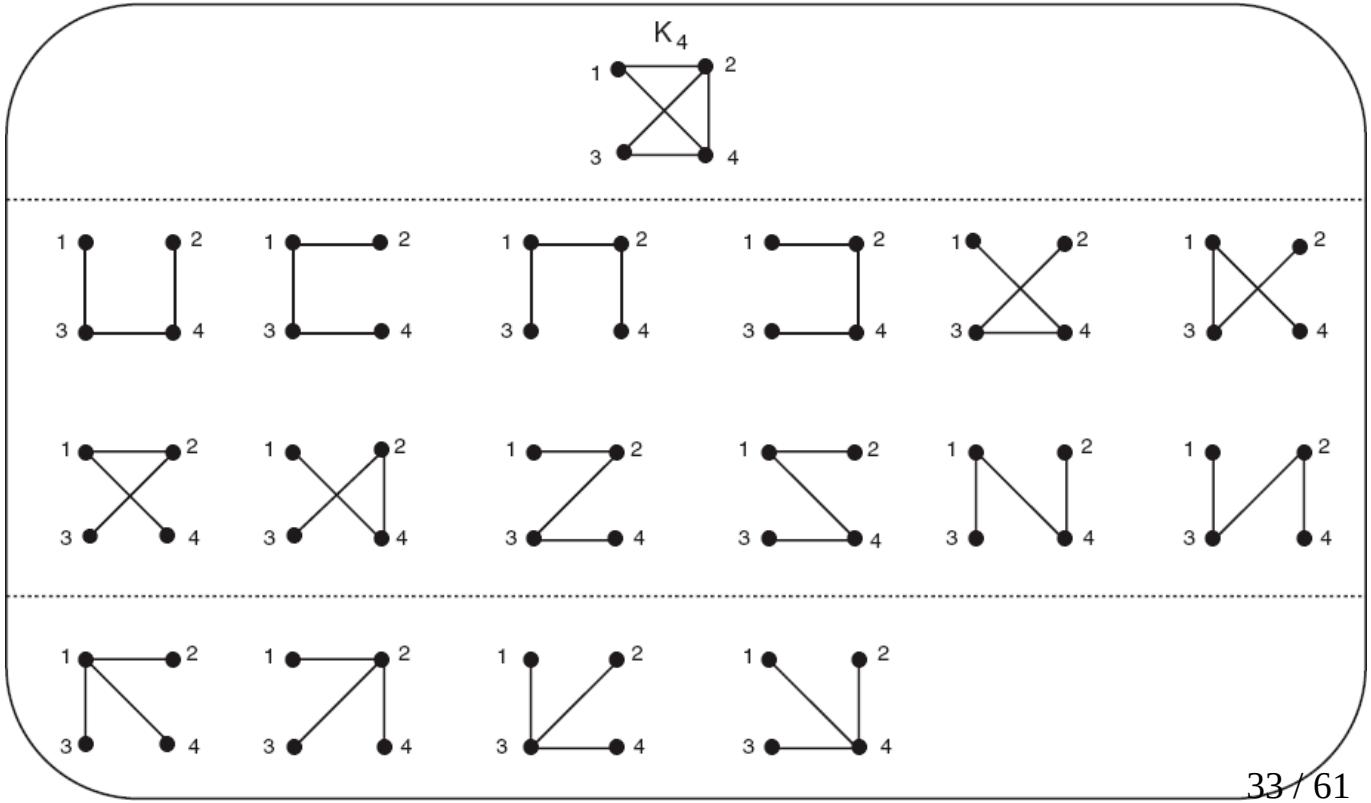
- **Teorema 6.10 (Teorema de Cayley, 1889)**
  - Um grafo completo K_n tem n^{n-2} árvores spanning diferentes.

Spanning Tree

- **Teorema 6.10 (Teorema de Cayley, 1889)**
  - Um grafo completo K_n tem n^{n-2} árvores spanning diferentes.
- Exemplo:

Spanning Tree

- **Teorema 6.10 (Teorema de Cayley, 1889)**
  - Um grafo completo K_n tem n^{n-2} árvores spanning diferentes.
- Exemplo:



Spanning Tree

Spanning Tree

- **Definição 6.5**

Spanning Tree

- **Definição 6.5**
  - Seja G = (V,E) um grafo, com |E| = m.G é chamado **grafo ponderado** se cada aresta e ∈ E tem um número real associado a ela, p(e), chamado peso de e.

Spanning Tree

- **Definição 6.5**
  - Seja G = (V,E) um grafo, com |E| = m.G é chamado **grafo ponderado** se cada aresta e ∈ E tem um número real associado a ela, p(e), chamado peso de e.
  - A soma dos pesos de todas as arestas do grafo, p(e1) + p(e2) + ... + p(em), é o peso de G, notado por p(G).

Spanning Tree

- **Definição 6.5**
  - Seja G = (V,E) um grafo, com |E| = m.G é chamado **grafo ponderado** se cada aresta e ∈ E tem um número real associado a ela, p(e), chamado peso de e.
  - A soma dos pesos de todas as arestas do grafo, p(e1) + p(e2) + ... + p(em), é o peso de G, notado por p(G).
  - Muitos problemas de otimização podem ser modelados no problema de encontrar, em um grafo ponderado, um certo tipo de subgrafo (por exemplo, um subgrafo que tenha todos os vértices do grafo original) com um peso mínimo (ou máximo).

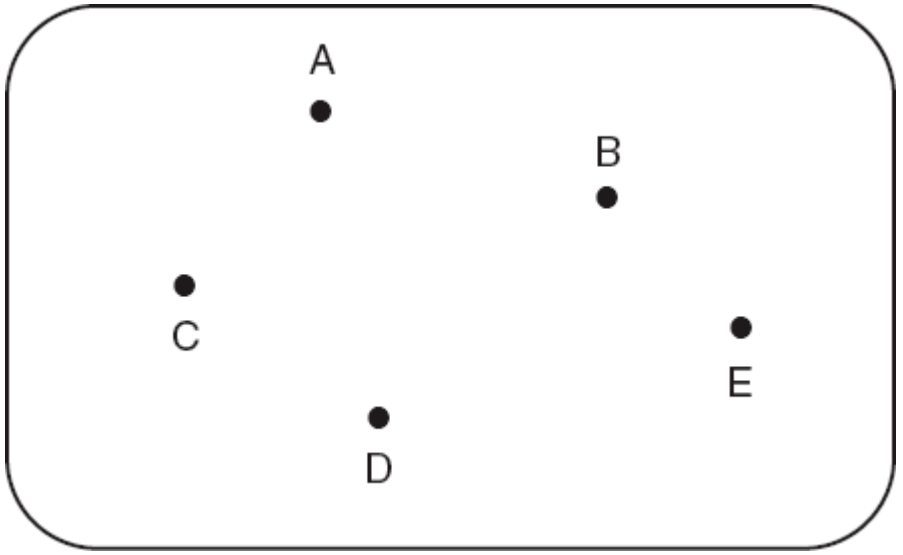
Spanning Tree

Spanning Tree

- Exemplo
  - Suponha que um certo Departamento de Transportes tenha decidido construir um sistema ferroviário entre cinco cidades que, geograficamente, estão localizadas como mostrado na figura abaixo e identificadas como A, B, C, D e E.

Spanning Tree

- Exemplo
  - Suponha que um certo Departamento de Transportes tenha decidido construir um sistema ferroviário entre cinco cidades que, geograficamente, estão localizadas como mostrado na figura abaixo e identificadas como A, B, C, D e E.



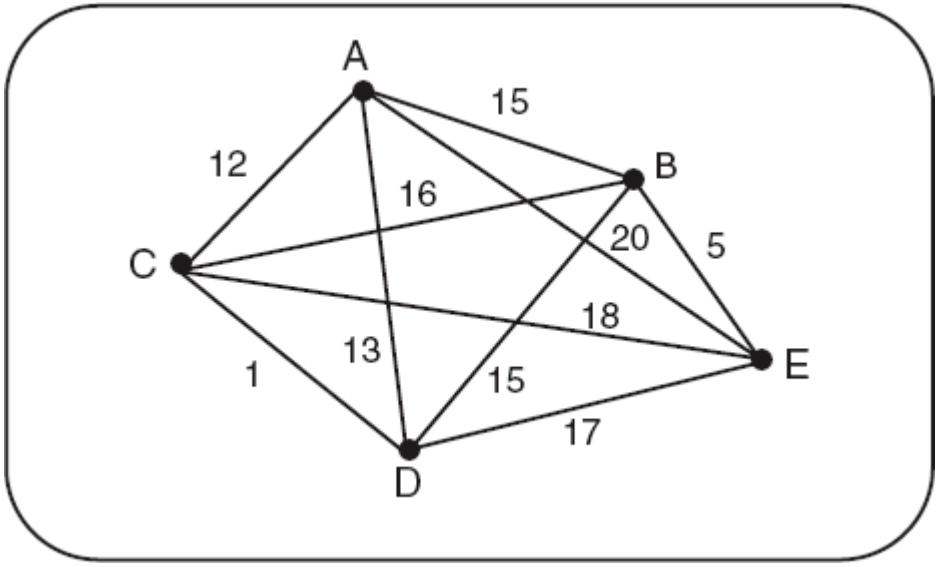
Spanning Tree

Spanning Tree

- Exemplo (cont.)
  - O primeiro passo do Departamento é determinar qual o custo de construir trilhos entre cada par das cidades consideradas.

Spanning Tree

- Exemplo (cont.)
  - O primeiro passo do Departamento é determinar qual o custo de construir trilhos entre cada par das cidades consideradas.



Spanning Tree

Spanning Tree

- Exemplo (cont.)

Spanning Tree

- Exemplo (cont.)
  - O problema de determinar quais conexões o Departamento deve construir, de maneira a manter o custo mínimo, pode ser caracterizado como o problema de encontrar um subgrafo com menor peso associado, que seja conectado e que tenha o mesmo conjunto de vértices que o grafo original G.

Spanning Tree

- Exemplo (cont.)
  - O problema de determinar quais conexões o Departamento deve construir, de maneira a manter o custo mínimo, pode ser caracterizado como o problema de encontrar um subgrafo com menor peso associado, que seja conectado e que tenha o mesmo conjunto de vértices que o grafo original G.
  - Não é difícil ver que tal subgrafo deve ser uma árvore *spanning* do grafo.

Spanning Tree

- Exemplo (cont.)
  - Note que, se tal subgrafo tivesse um **ciclo**, qualquer uma das arestas desse ciclo poderia ser removida e o subgrafo resultante teria um peso menor e ainda manteria acesso a todos os vértices.

Spanning Tree

Spanning Tree

- Exemplo (cont.)

Spanning Tree

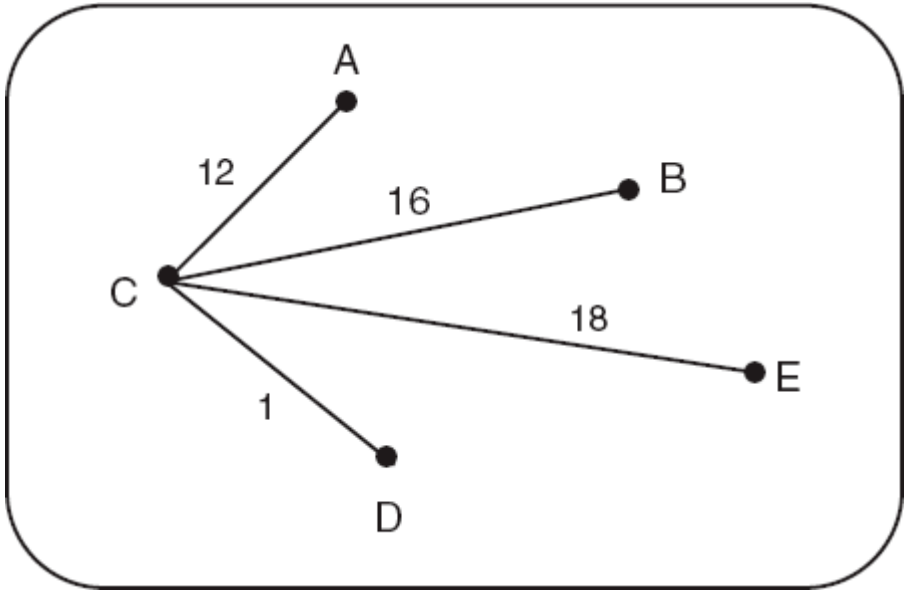
- Exemplo (cont.)
  - Entretanto, que nem toda árvore *spanning* do grafo solucionará o problema da árvore *spanning* mínima.

Spanning Tree

- Exemplo (cont.)
  - Entretanto, que nem toda árvore *spanning* do grafo solucionará o problema da árvore *spanning* mínima.
  - Algumas arestas presentes em algumas das árvores *spanning* terão peso maior do que outras

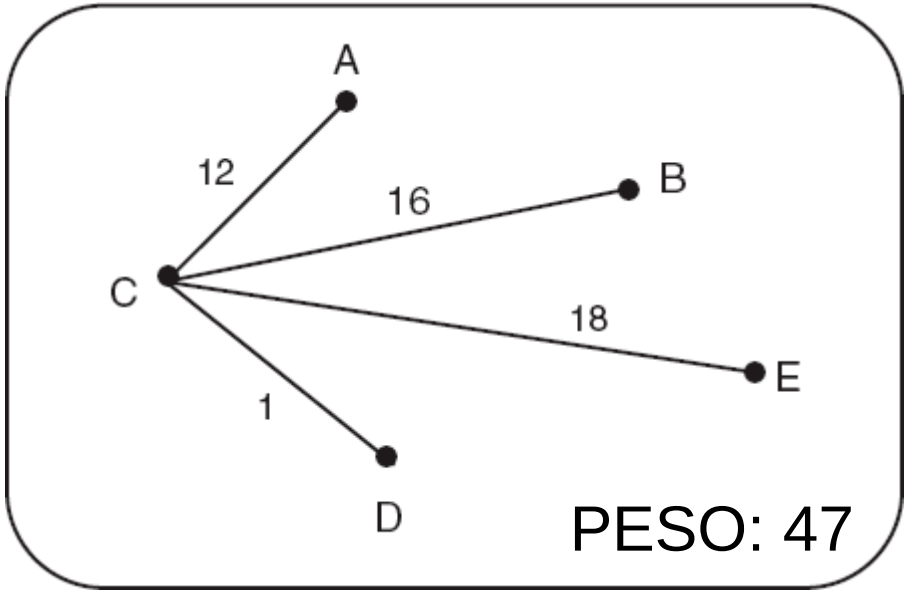
Spanning Tree

- Exemplo (cont.)
  - Entretanto, que nem toda árvore *spanning* do grafo solucionará o problema da árvore *spanning* mínima.
  - Algumas arestas presentes em algumas das árvores *spanning* terão peso maior do que outras



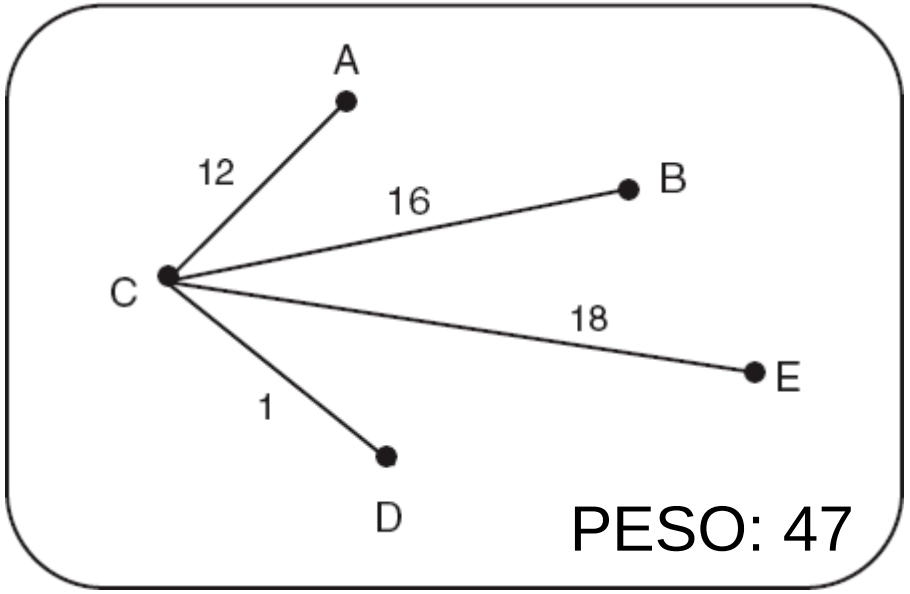
Spanning Tree

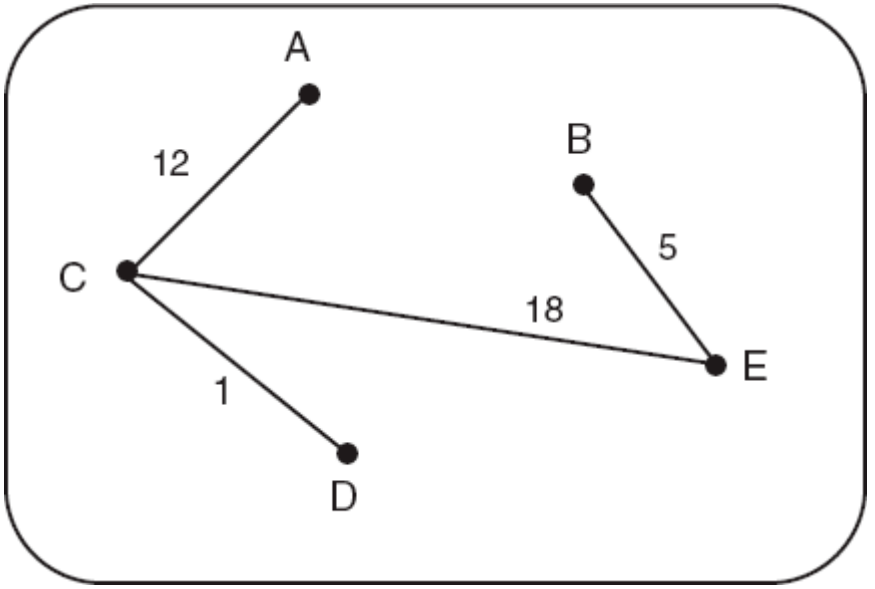
- Exemplo (cont.)
  - Entretanto, que nem toda árvore *spanning* do grafo solucionará o problema da árvore *spanning* mínima.
  - Algumas arestas presentes em algumas das árvores *spanning* terão peso maior do que outras



Spanning Tree

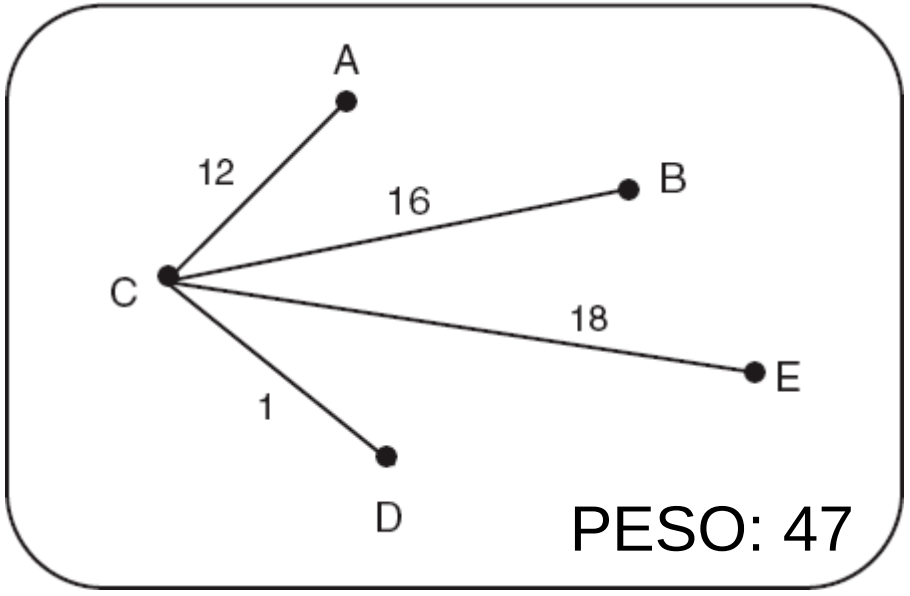
- Exemplo (cont.)
  - Entretanto, que nem toda árvore *spanning* do grafo solucionará o problema da árvore *spanning* mínima.
  - Algumas arestas presentes em algumas das árvores *spanning* terão peso maior do que outras

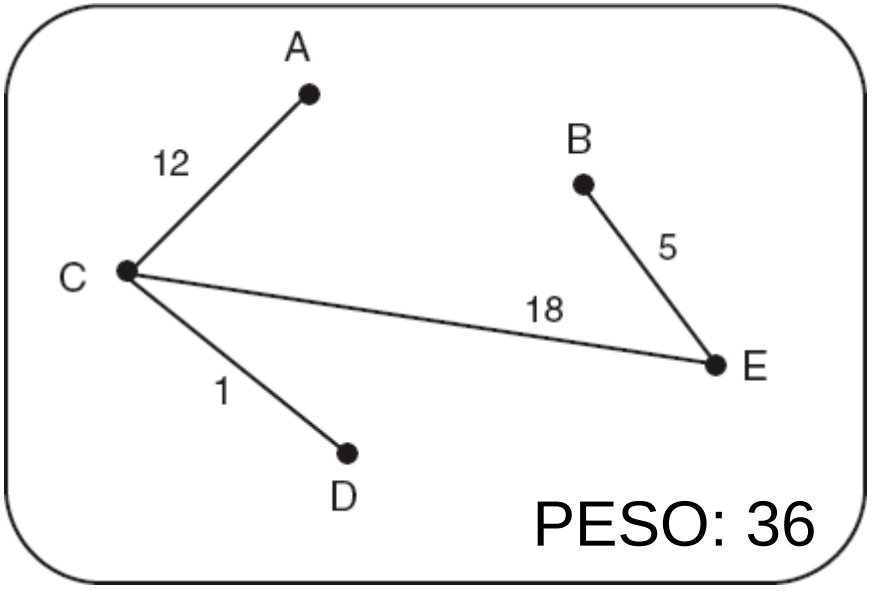




Spanning Tree

- Exemplo (cont.)
  - Entretanto, que nem toda árvore *spanning* do grafo solucionará o problema da árvore *spanning* mínima.
  - Algumas arestas presentes em algumas das árvores *spanning* terão peso maior do que outras





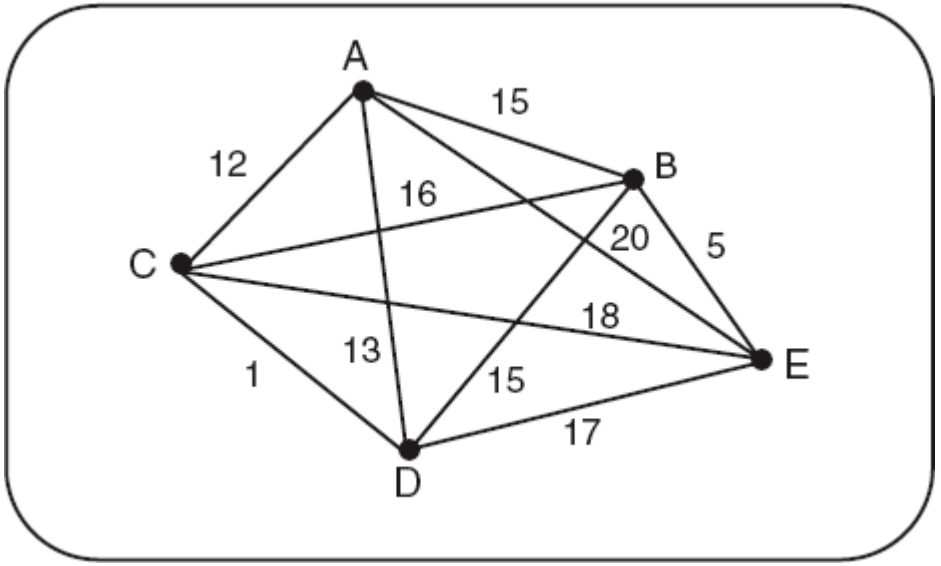
Spanning Tree

Spanning Tree

- Exemplo (cont.)
  - Árvore spanning mínima do grafo G.

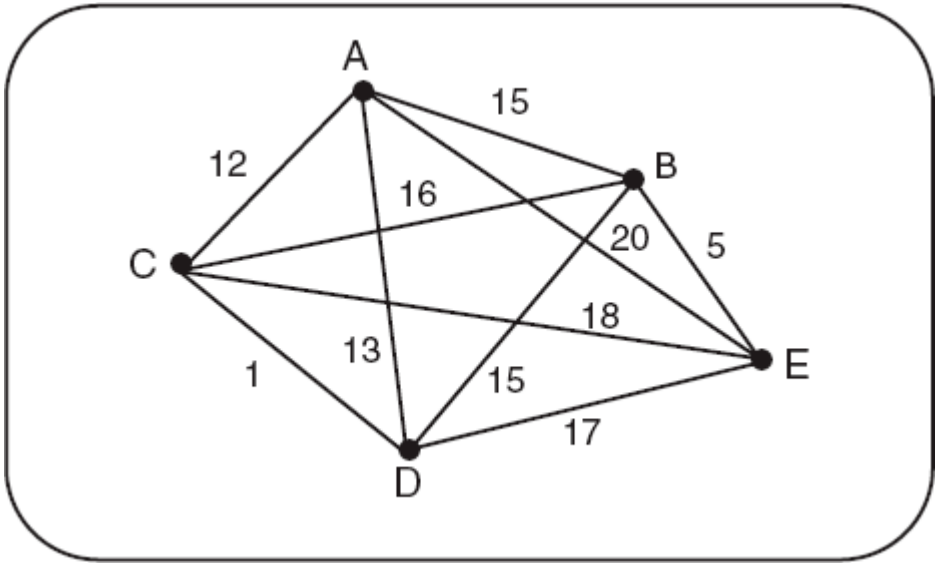
Spanning Tree

- Exemplo (cont.)
  - Árvore spanning mínima do grafo G.



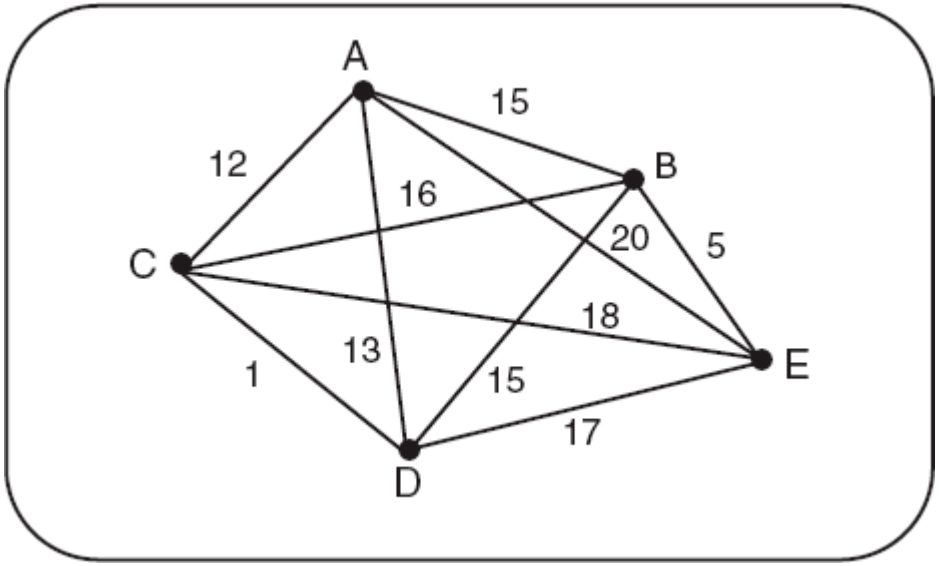
Spanning Tree

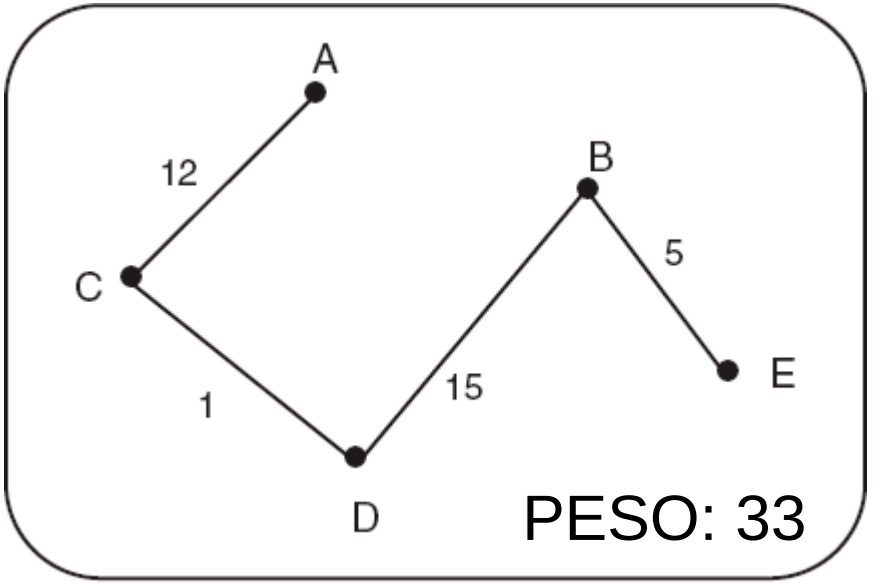
- Exemplo (cont.)
  - Árvore spanning mínima do grafo G.



Spanning Tree

- Exemplo (cont.)
  - Árvore spanning mínima do grafo G.





Spanning Tree

Spanning Tree

- Existem vários algoritmos para encontrar a árvore spanning mínima em um grafo.

Spanning Tree

- Existem vários algoritmos para encontrar a árvore spanning mínima em um grafo.
- Dois deles, o algoritmo de Kruskal (Kruskal, 1956) e o algoritmo de Prim (Prim, 1957).

Spanning Tree

- Existem vários algoritmos para encontrar a árvore spanning mínima em um grafo.
- Dois deles, o algoritmo de Kruskal (Kruskal, 1956) e o algoritmo de Prim (Prim, 1957).
- Esses dois algoritmos são para pesos não negativos associados às arestas.

Spanning Tree

- Existem vários algoritmos para encontrar a árvore spanning mínima em um grafo.
- Dois deles, o algoritmo de Kruskal (Kruskal, 1956) e o algoritmo de Prim (Prim, 1957).
- Esses dois algoritmos são para pesos não negativos associados às arestas.
- Dois teoremas, apresentados e provados em Clark & Holton (1998), garantem que os algoritmos de Kruskal e de Prim constroem árvores spanning que são minimais.

Algoritmo de Kruskal

Algoritmo de Kruskal

- A ideia geral é, iterativamente, escolher as arestas que tenham o menor peso.

Algoritmo de Kruskal

- A ideia geral é, iterativamente, escolher as arestas que tenham o menor peso.
- Algoritmo geral:

Algoritmo de Kruskal

- A ideia geral é, iterativamente, escolher as arestas que tenham o menor peso.
- Algoritmo geral:
  - É escolhida uma aresta de G que tenha o menor peso entre as arestas de G que não são loops.

Algoritmo de Kruskal

- A ideia geral é, iterativamente, escolher as arestas que tenham o menor peso.
- Algoritmo geral:
  - É escolhida uma aresta de G que tenha o menor peso entre as arestas de G que não são loops.
  - É escolhida dentre as arestas restantes aquela de menor peso, tomando o cuidado para que a aresta escolhida não forme um ciclo com a aresta que já compõe a árvore.

Algoritmo de Kruskal

- A ideia geral é, iterativamente, escolher as arestas que tenham o menor peso.
- Algoritmo geral:
  - É escolhida uma aresta de G que tenha o menor peso entre as arestas de G que não são loops.
  - É escolhida dentre as arestas restantes aquela de menor peso, tomando o cuidado para que a aresta escolhida não forme um ciclo com a aresta que já compõe a árvore.
  - Repete-se o processo.

Algoritmo de Kruskal

Algoritmo de Kruskal

- Se o grafo G tem n vértices, o processo termina após a escolha de n−1 arestas.

Algoritmo de Kruskal

- Se o grafo G tem n vértices, o processo termina após a escolha de n−1 arestas.
- As n−1 arestas formam um subgrafo acíclico T de G, que é a *árvore spanning minimal* de G.

Algoritmo de Kruskal

- Algoritmo:

Entrada: Grafo conectado $G = (\{v_1, v_2, ..., v_n\}, \{e_1, e_2, ..., e_m\})$ e $\{p(e_1), p(e_2), ..., p(e_n)\}$

Saída: Árvore *spanning* minimal dada por $\{v_1, v_2, ..., v_n\}$ e $n-1$ arestas extraídas de $\{e_1, e_2, ..., e_m\}$

Passo 1. Escolha e_1, uma aresta de G tal que $p(e_1)$ seja o menor possível e e_1 não seja um *loop*.

Passo 2. S as arestas $e_1, e_2, ..., e_i$ foram já escolhidas, então escolha uma aresta e_{i+1} que não tenha sido escolhida ainda, tal que

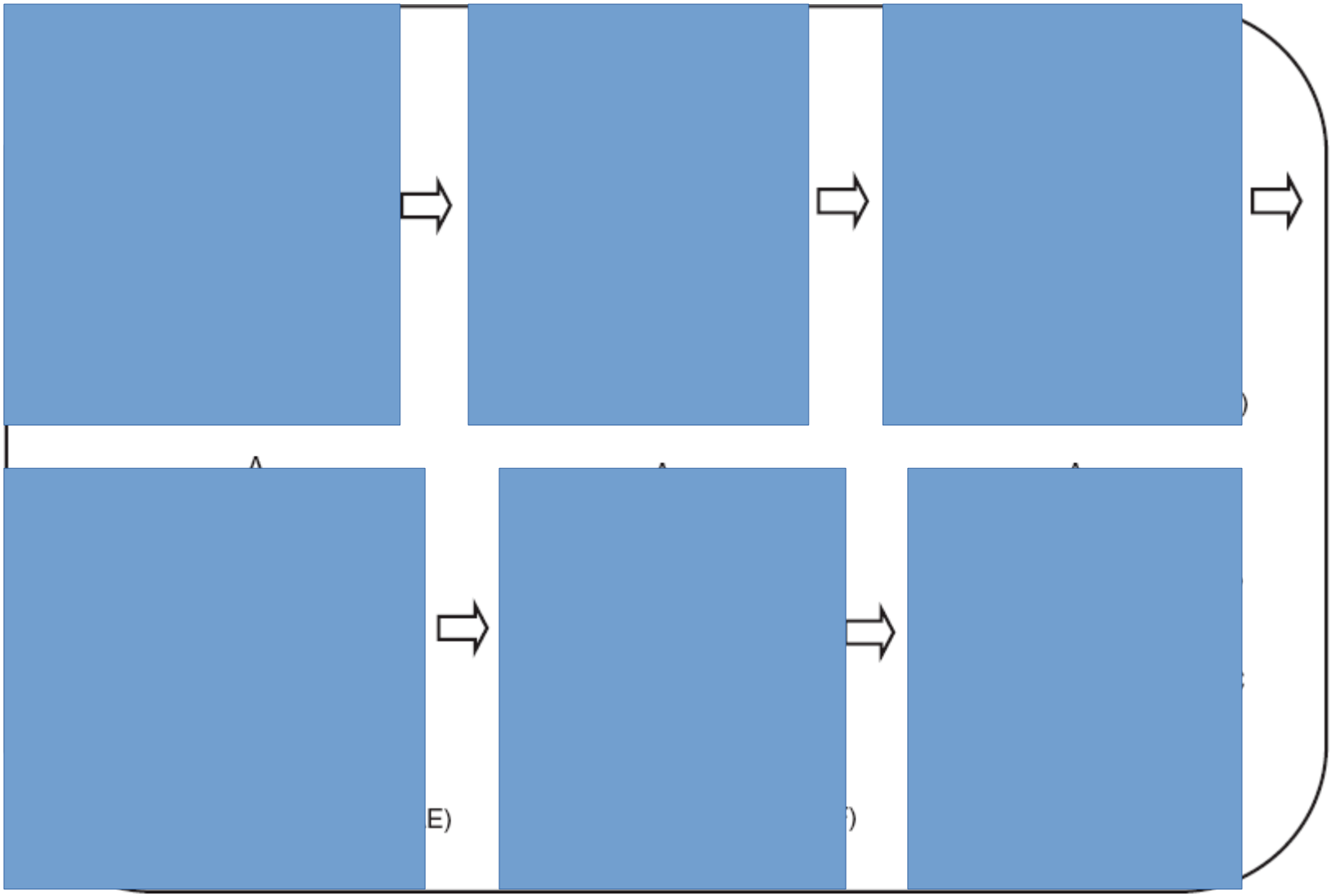
(i) o subgrafo induzido $G[\{e_1, e_2, ..., e_i, e_{i+1}\}]$ seja acíclico

(ii) $p(e_{i+1})$ seja o menor possível (sujeito à condição (i))

Passo 3. Se G tem n vértices, pare após a escolha de $n-1$ arestas; caso contrário, repita o Passo 2.

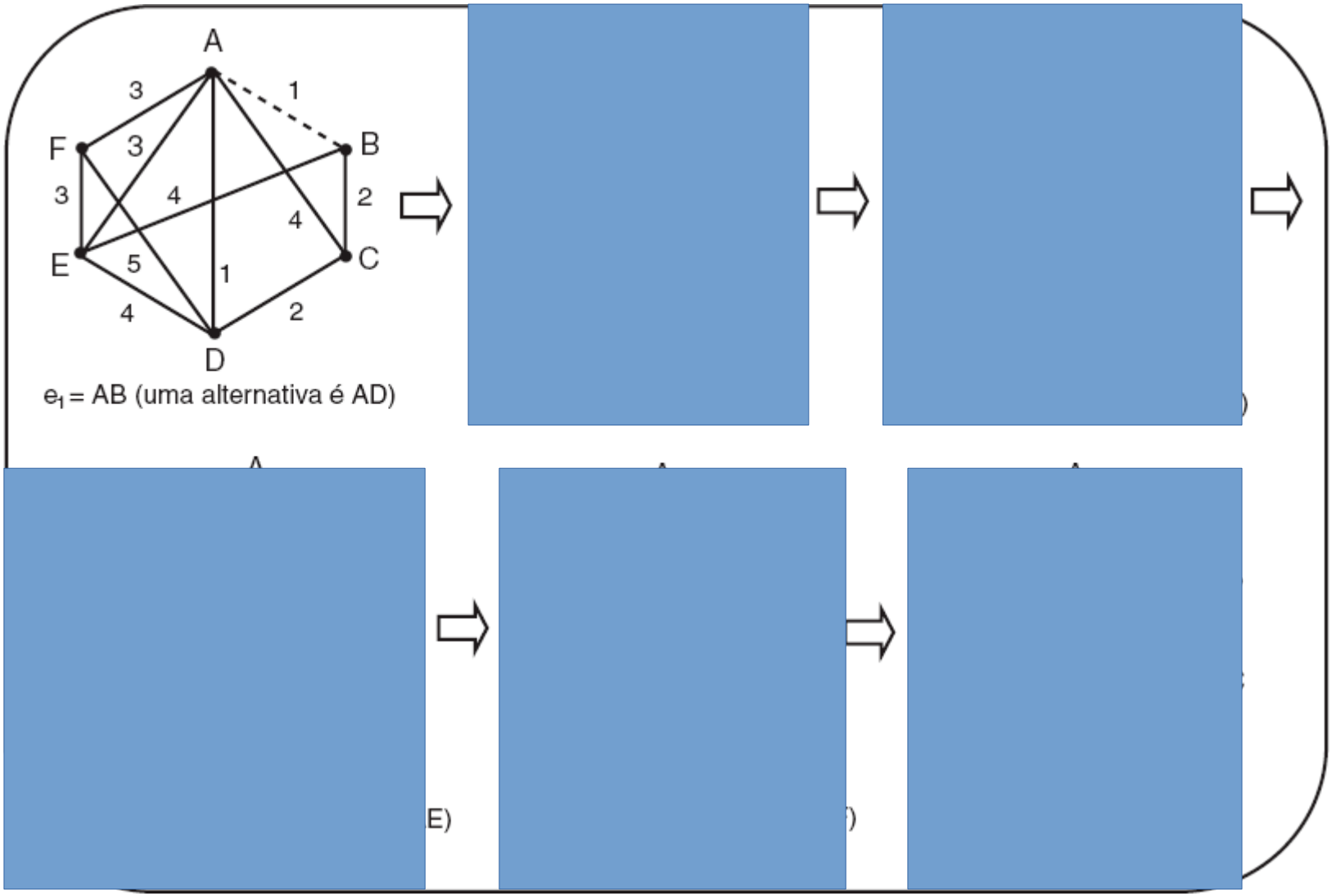
Algoritmo de Kruskal

- Construção da árvore



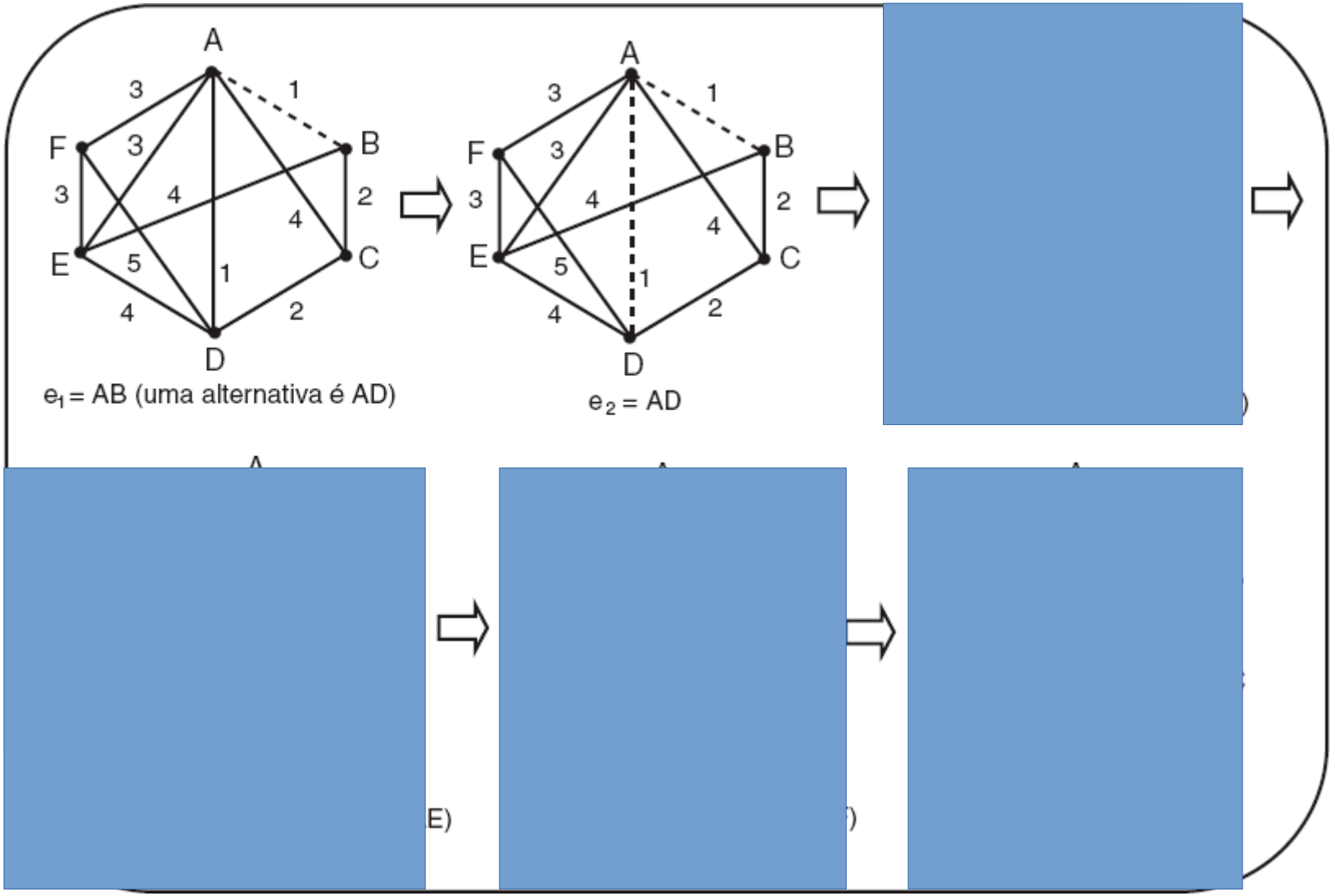
Algoritmo de Kruskal

- Construção da árvore



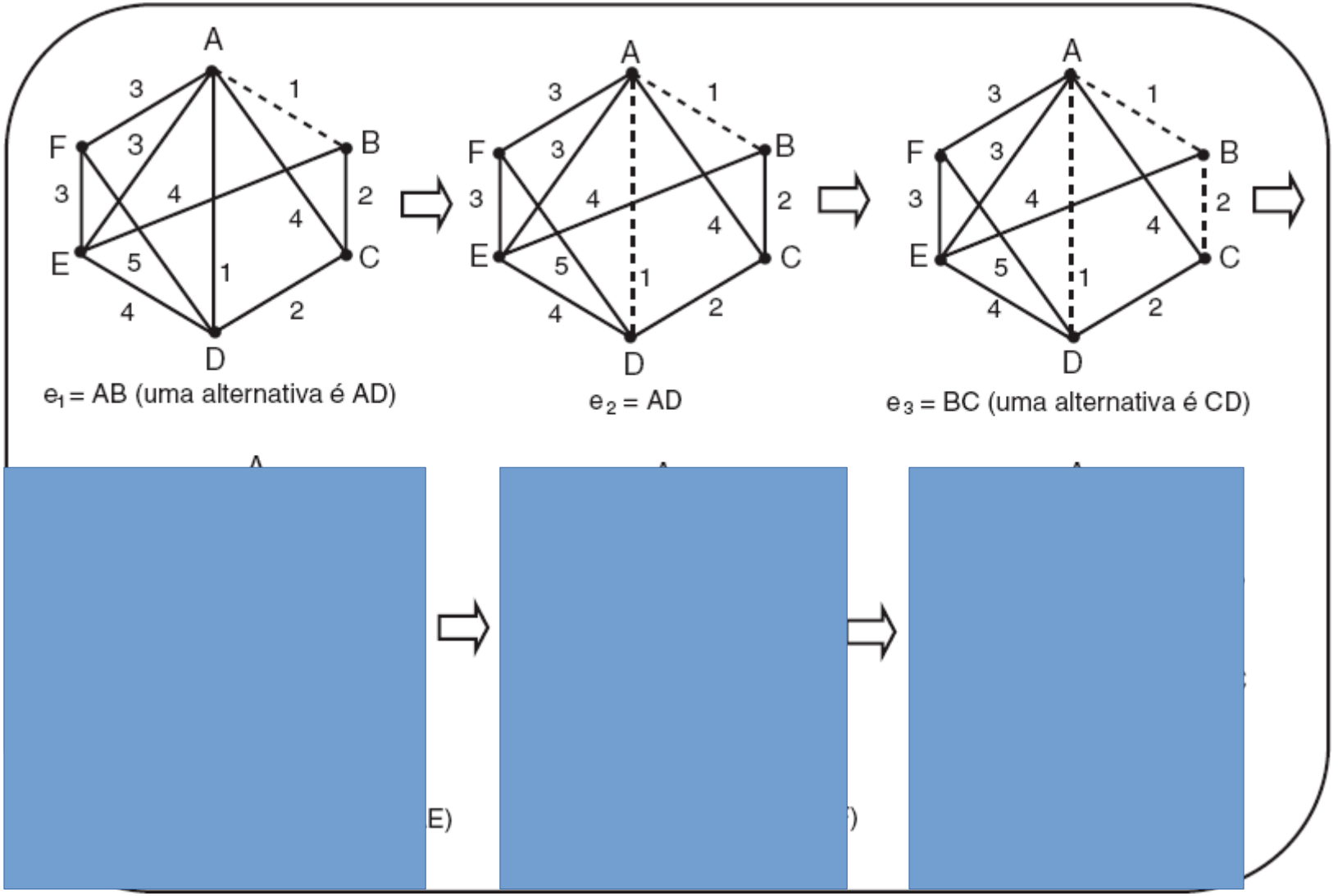
Algoritmo de Kruskal

- Construção da árvore



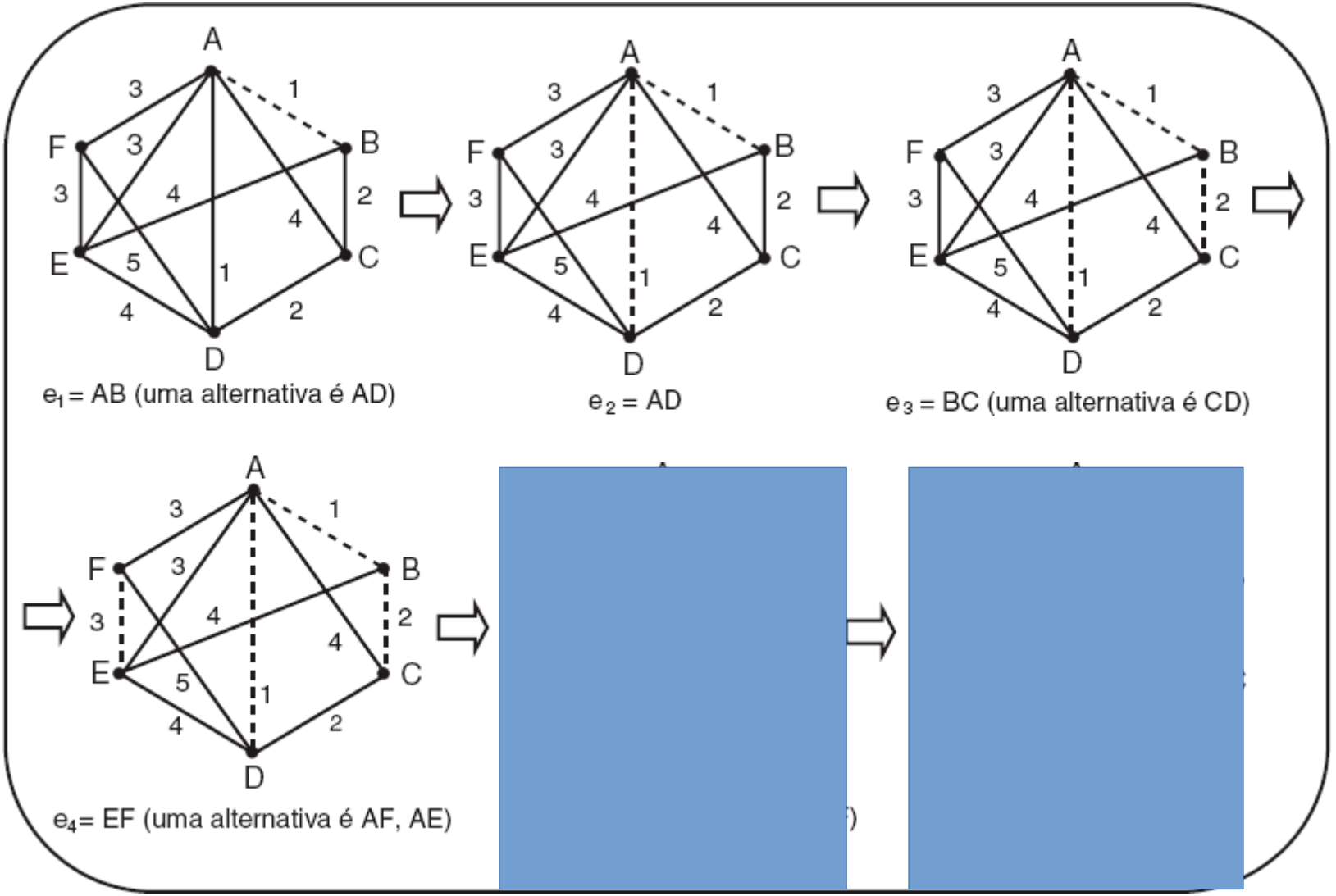
Algoritmo de Kruskal

- Construção da árvore



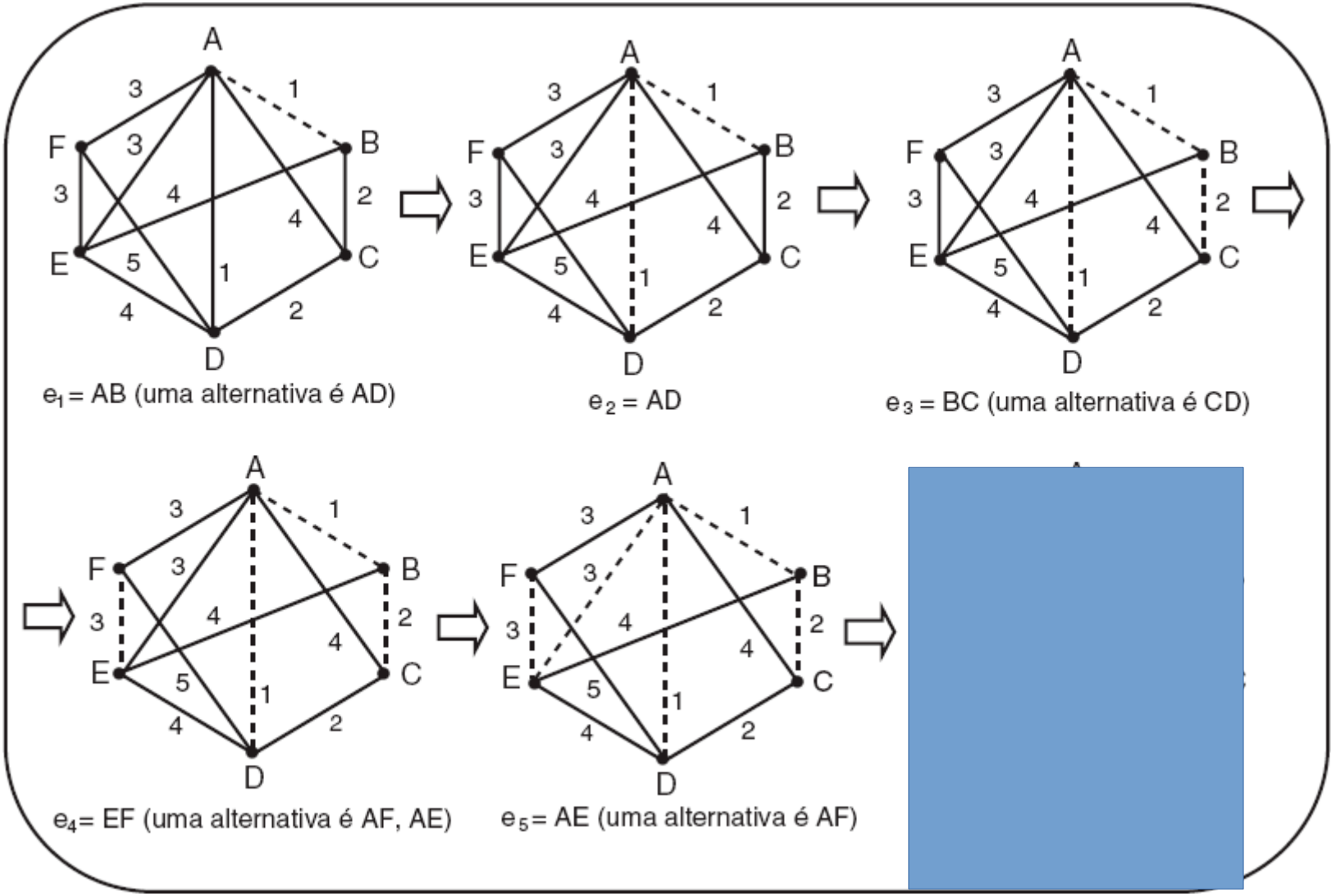
Algoritmo de Kruskal

- Construção da árvore



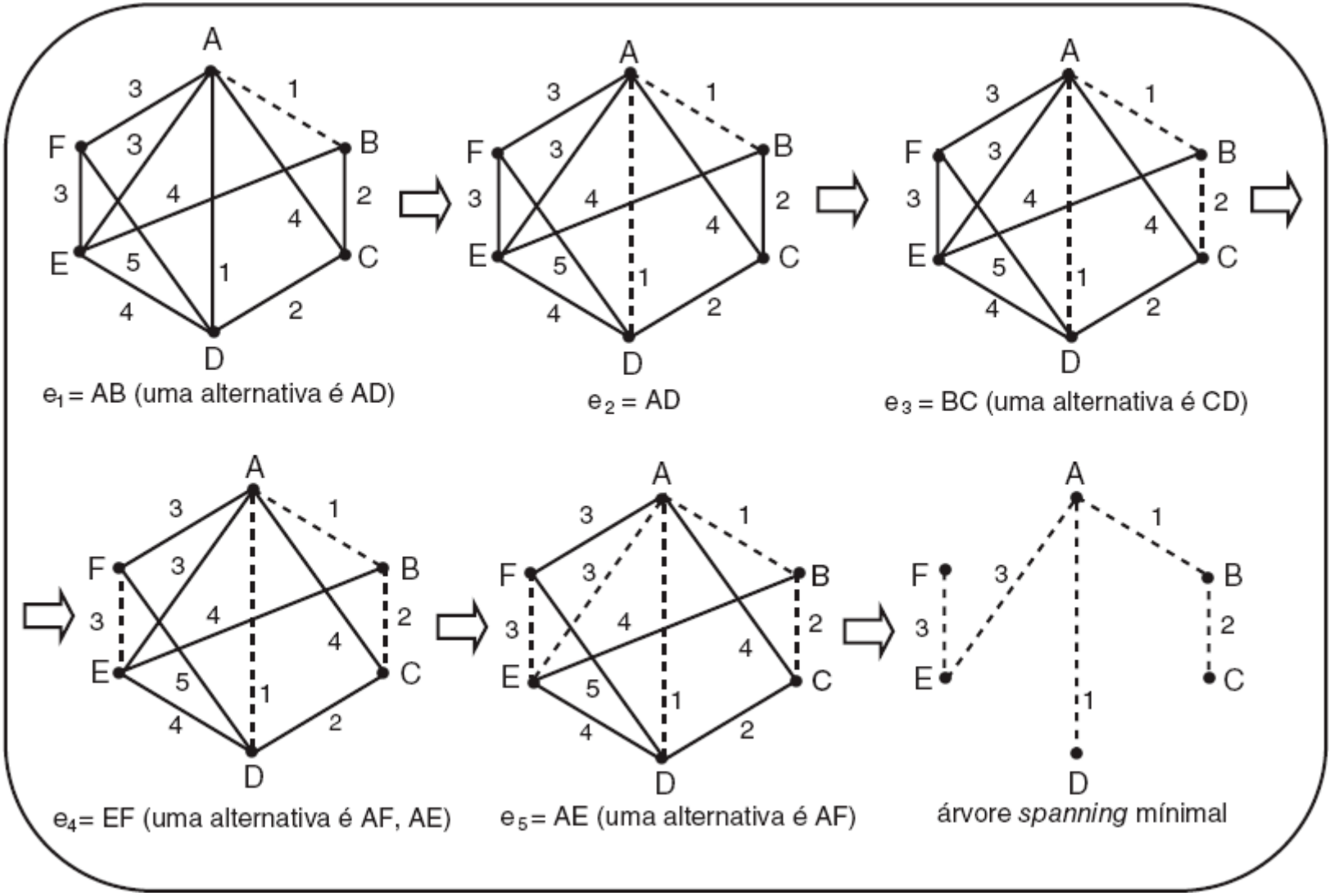
Algoritmo de Kruskal

- Construção da árvore



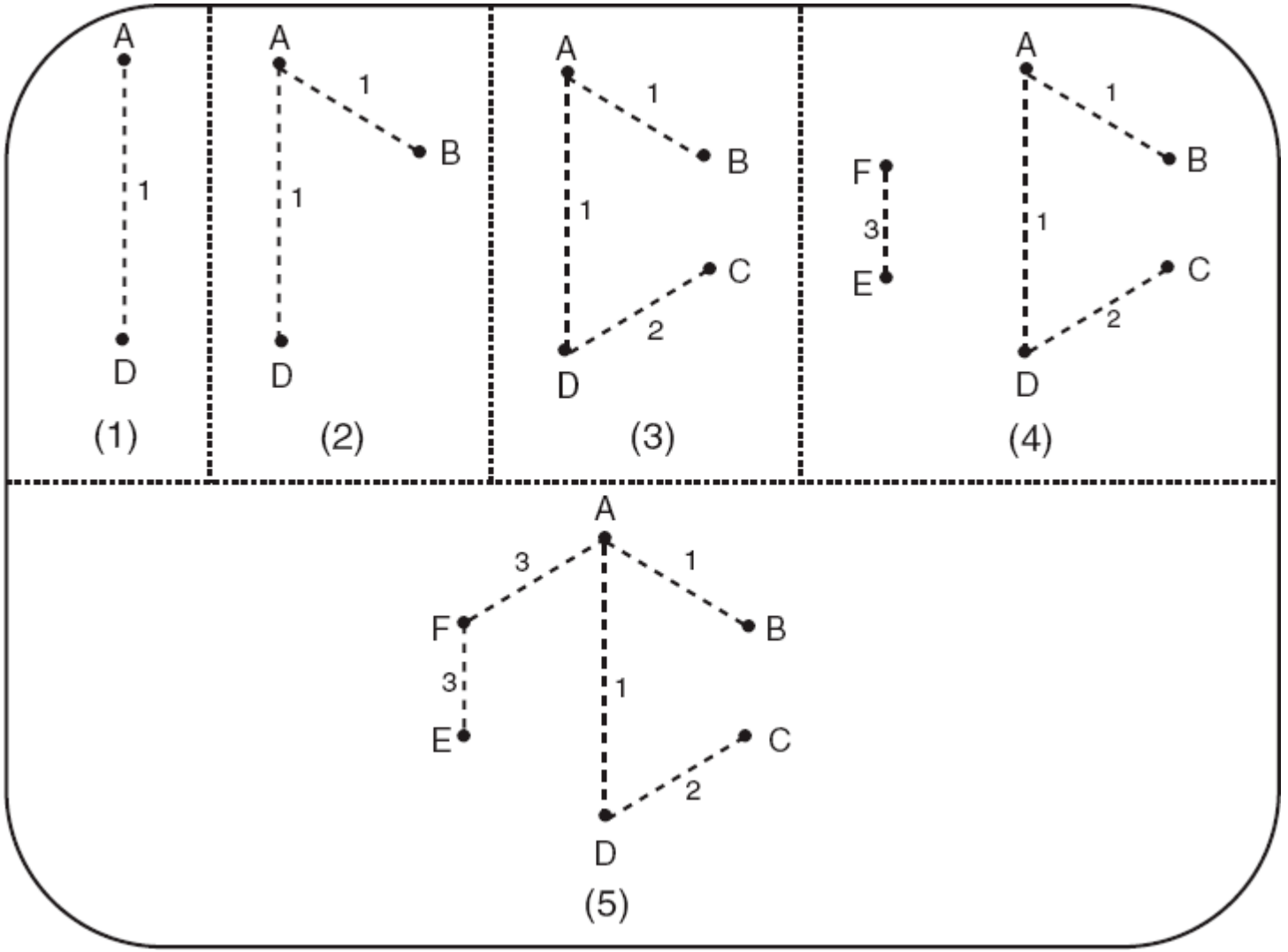
Algoritmo de Kruskal

- Construção da árvore



Algoritmo de Kruskal

- Contrução árvore com diferentes escolhas

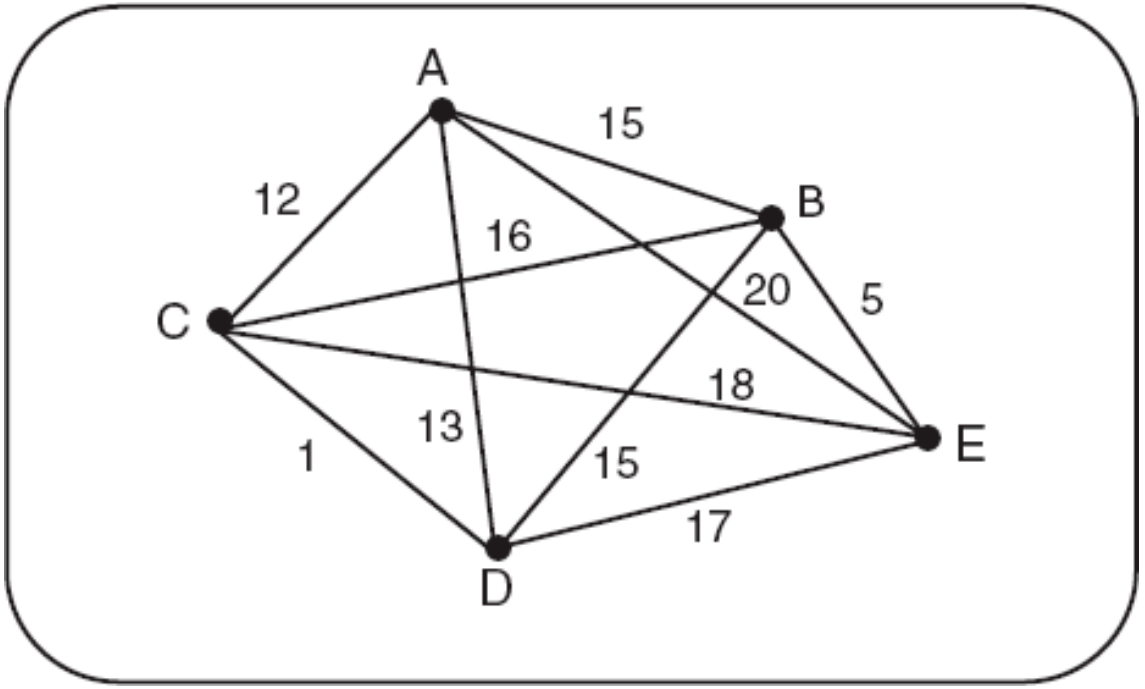


Algoritmo de Kruskal

- Exemplo

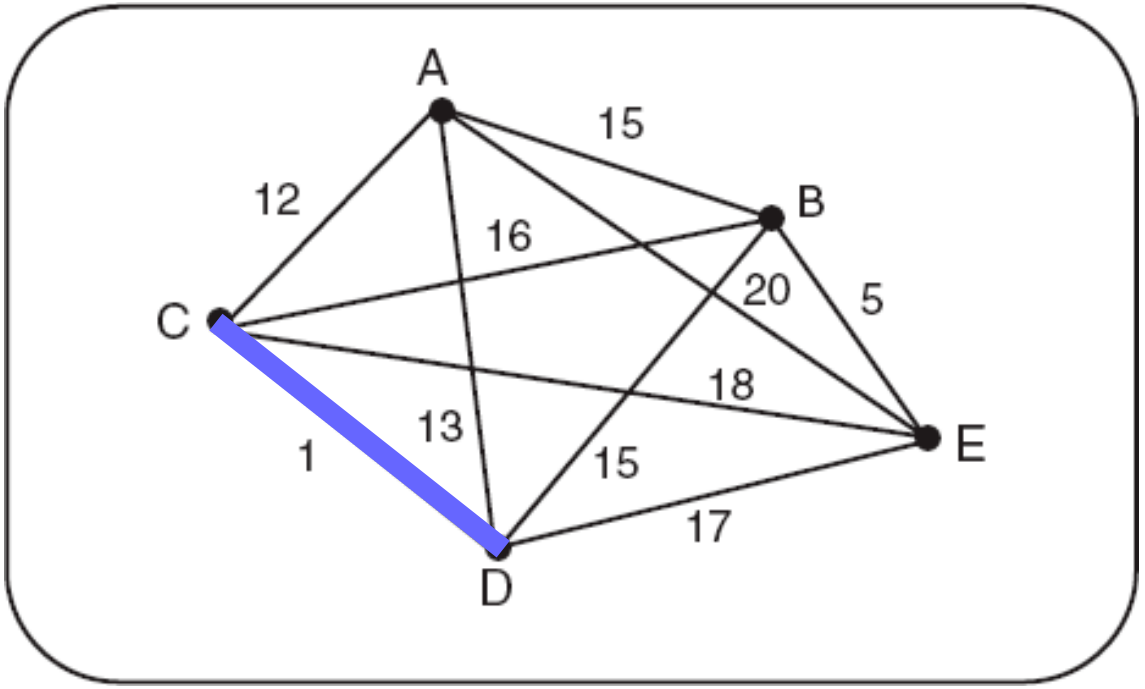
Algoritmo de Kruskal

- Exemplo



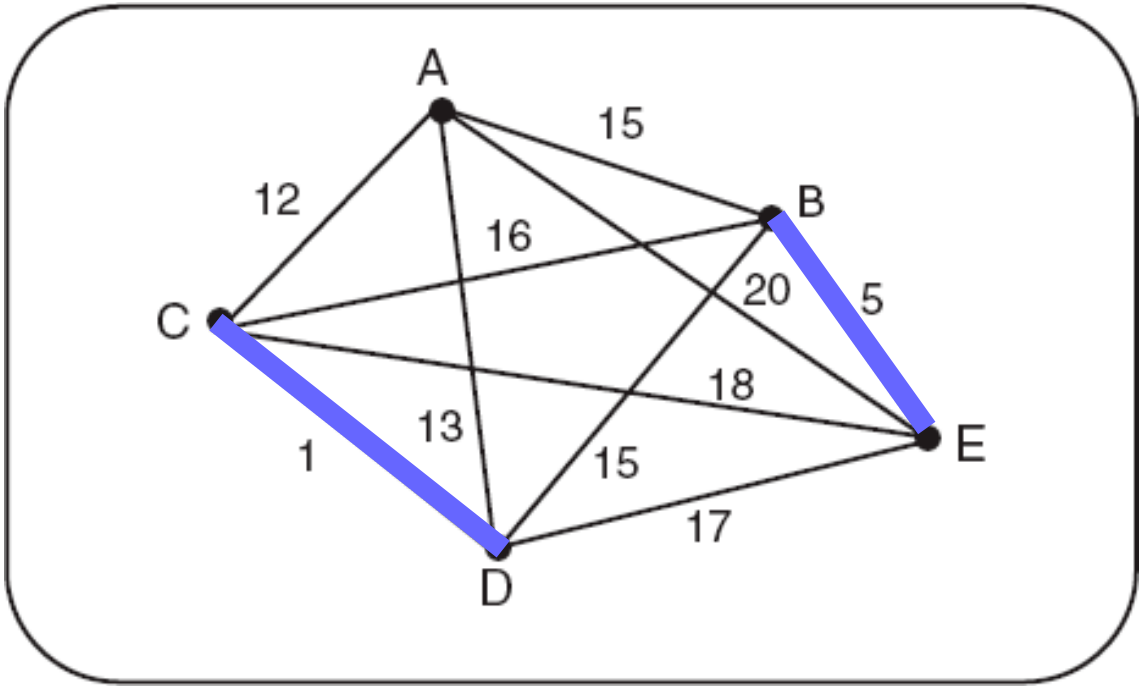
Algoritmo de Kruskal

- Exemplo



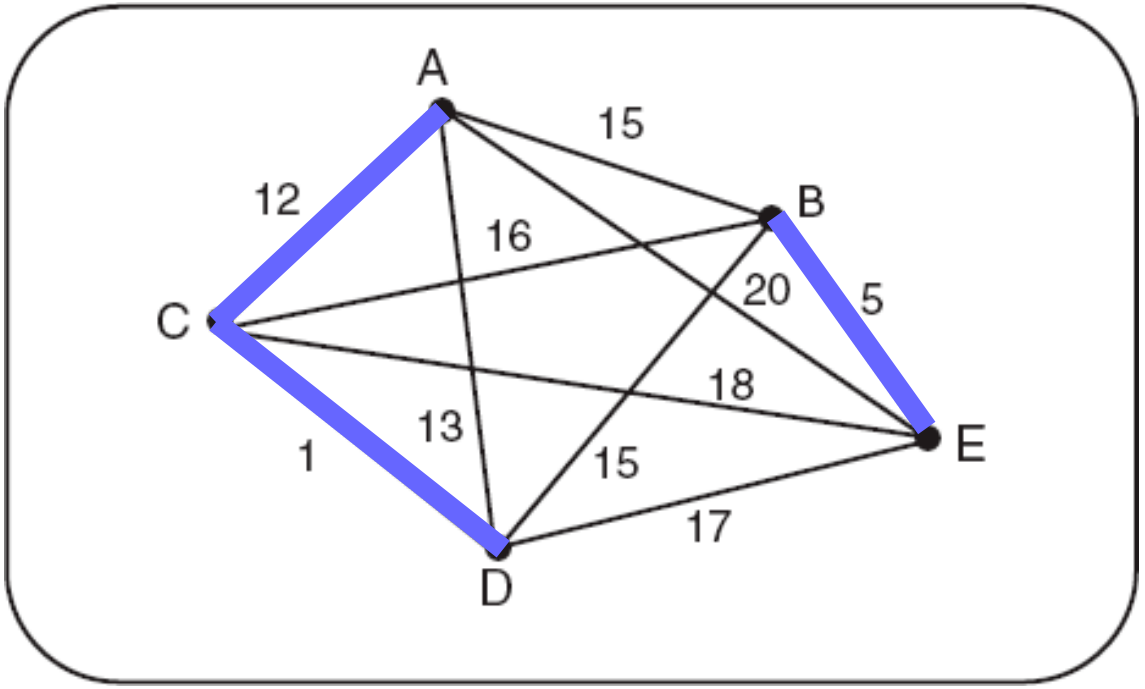
Algoritmo de Kruskal

- Exemplo



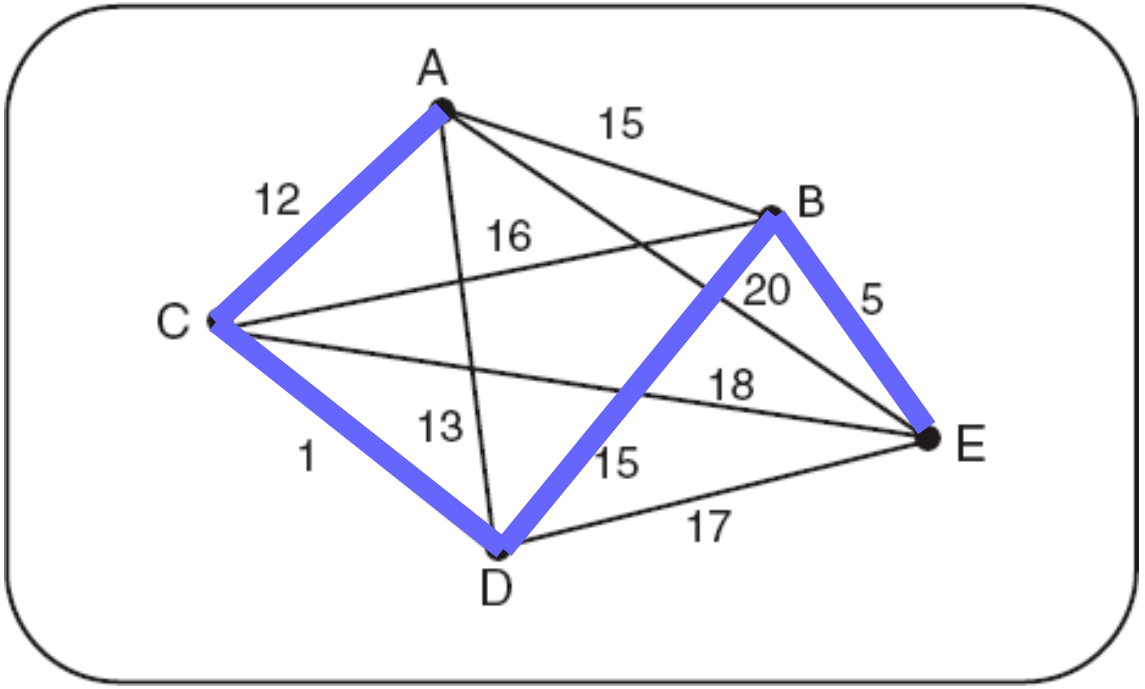
Algoritmo de Kruskal

- Exemplo



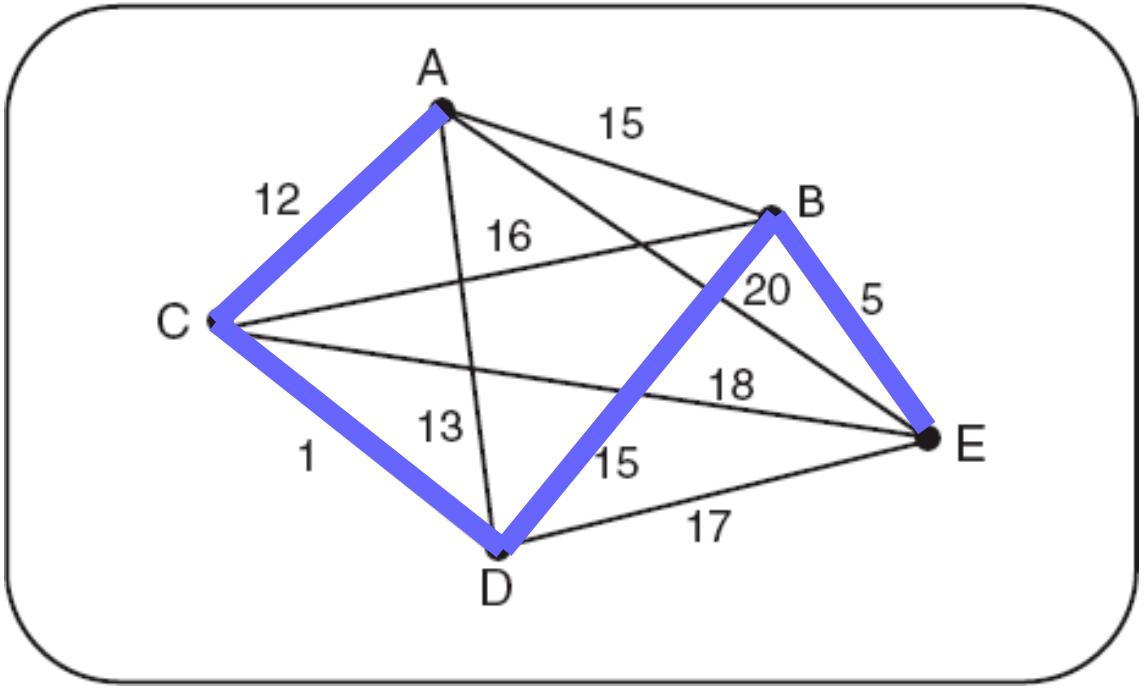
Algoritmo de Kruskal

- Exemplo



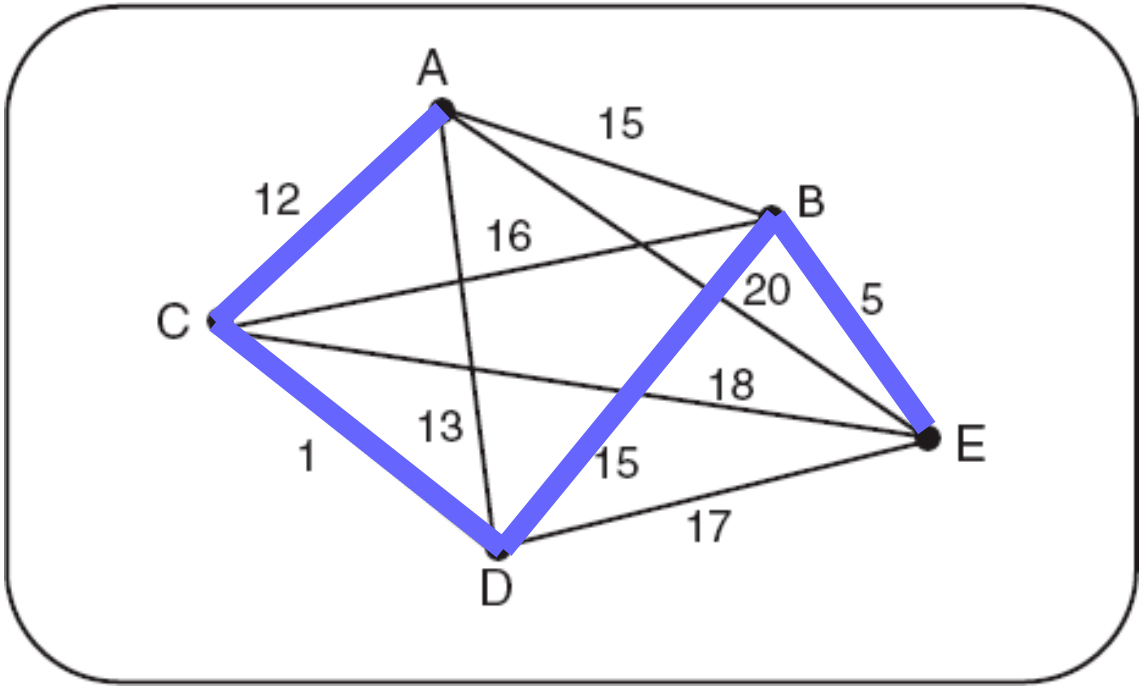
Algoritmo de Kruskal

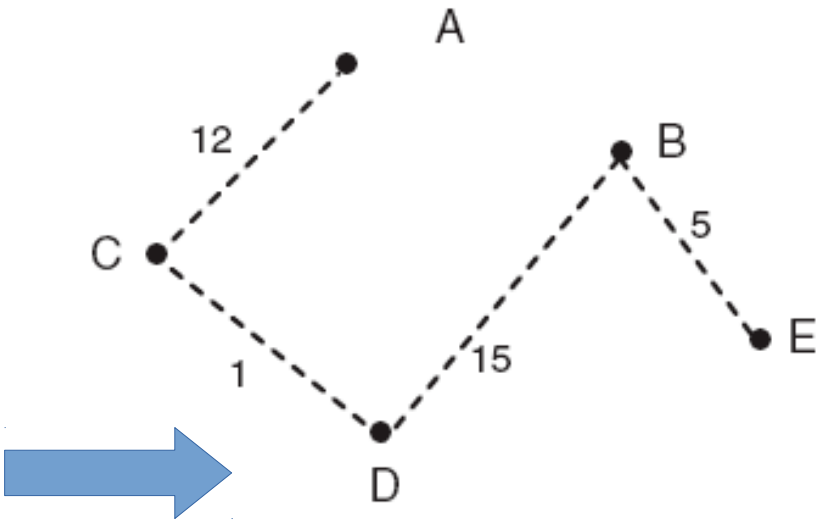
- Exemplo



Algoritmo de Kruskal

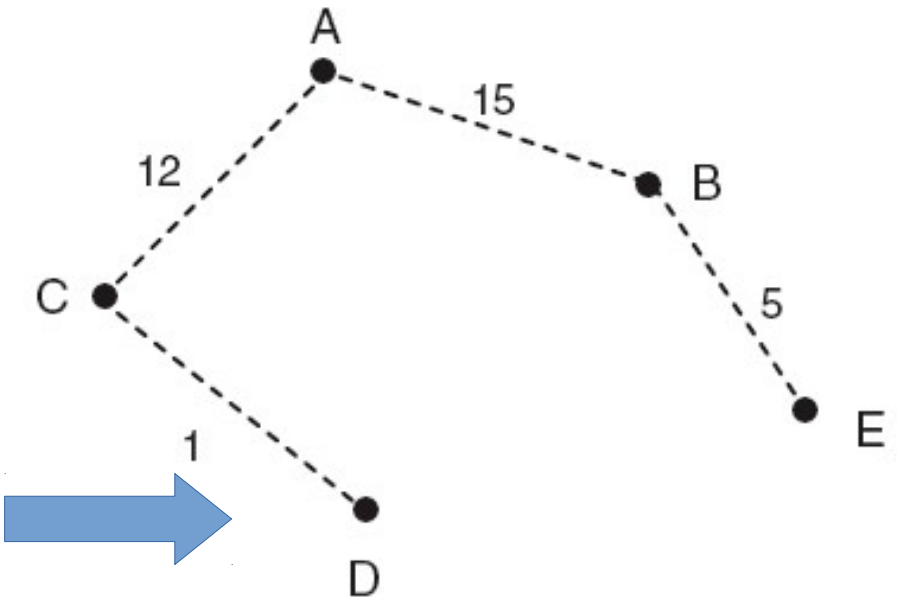
- Exemplo





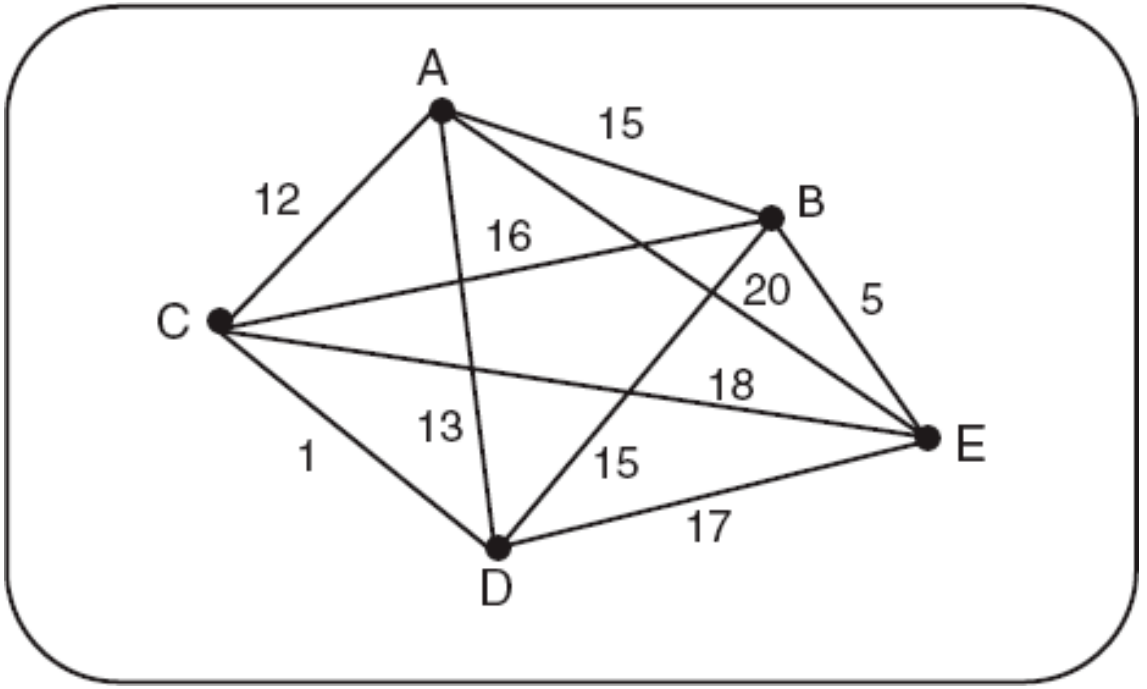
Algoritmo de Kruskal

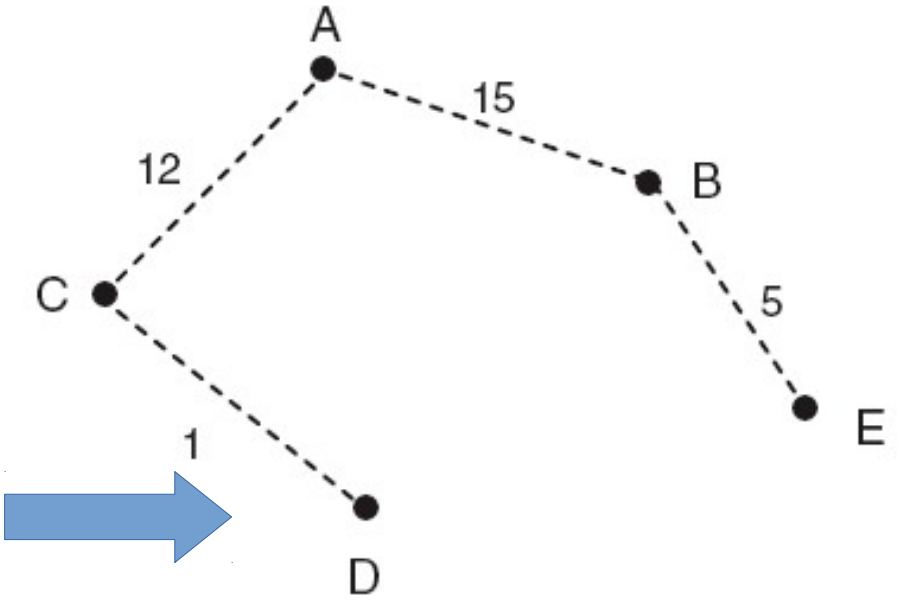
- Outra possível árvore spanning mínimal



Algoritmo de Kruskal

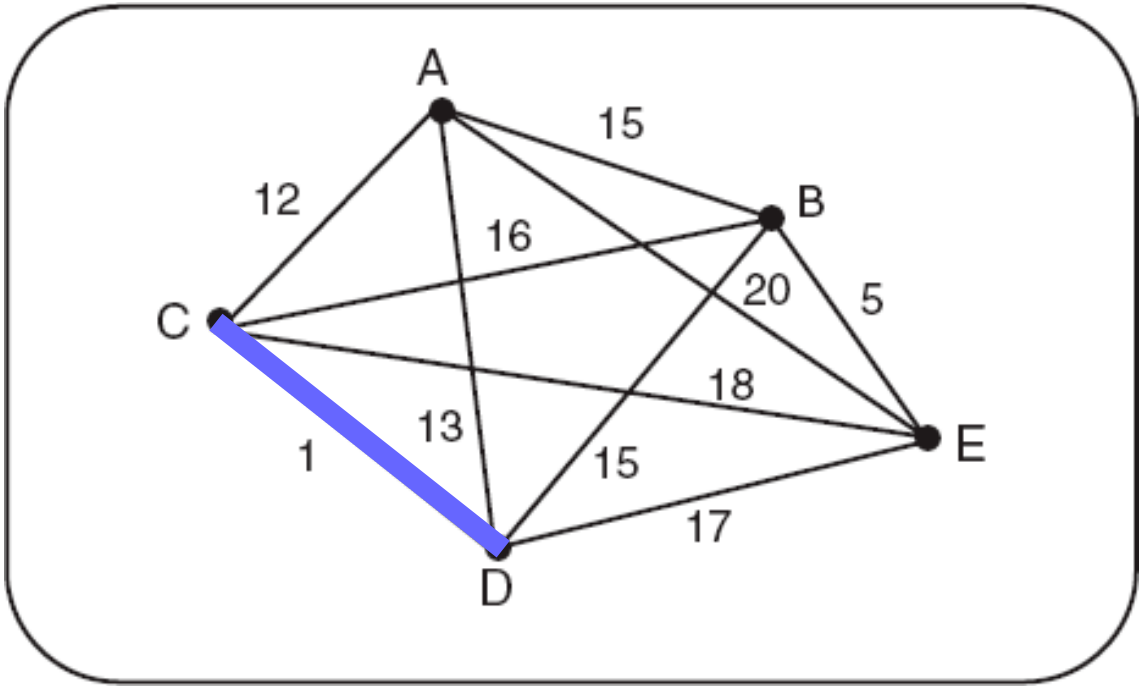
- Outra possível árvore spanning mínimal

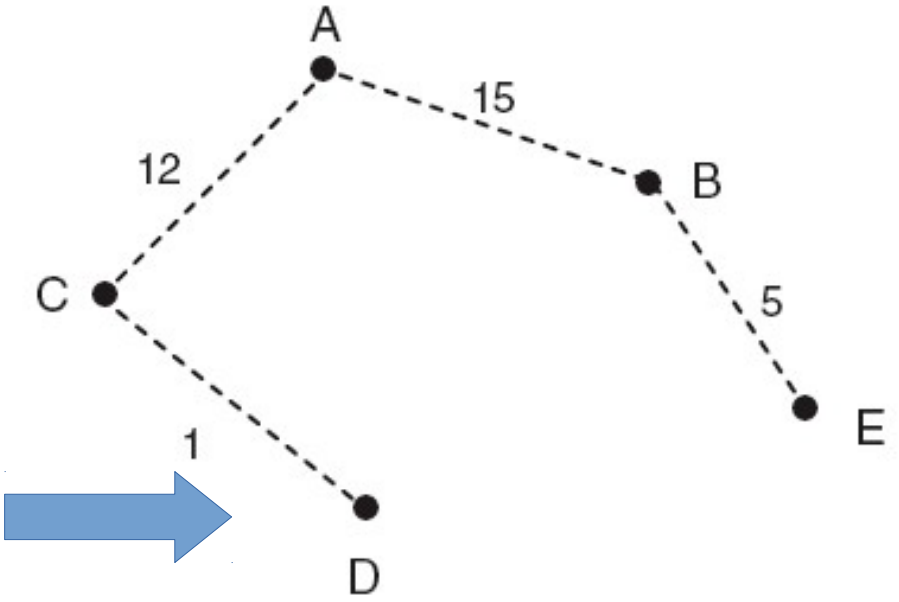




Algoritmo de Kruskal

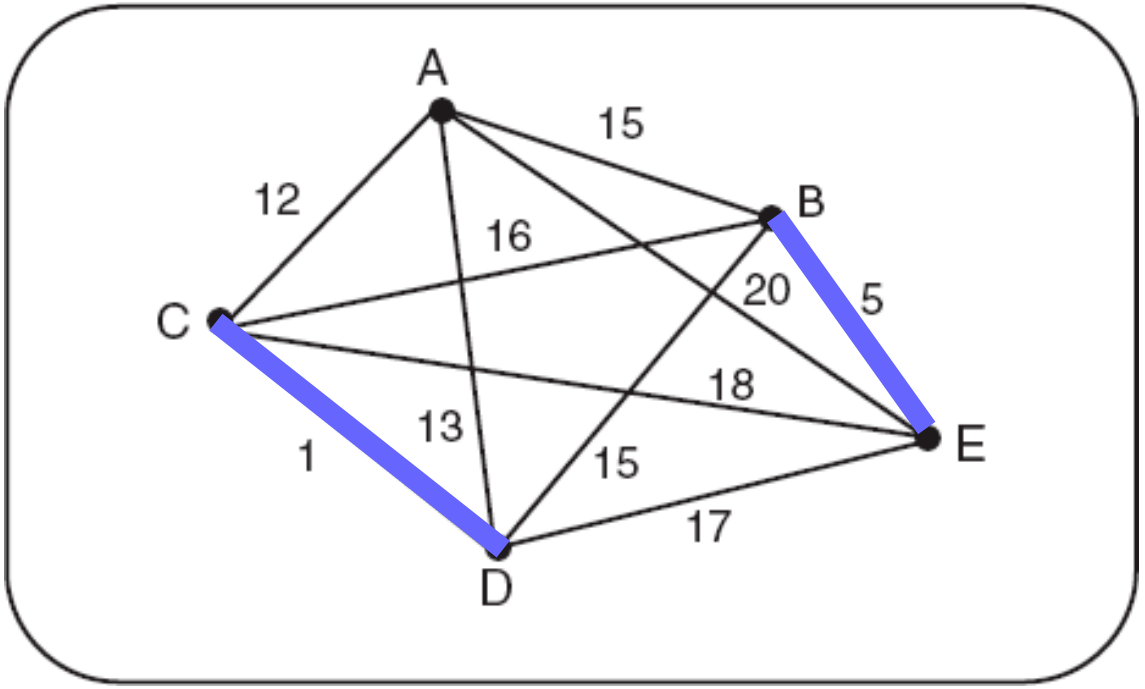
- Outra possível árvore spanning mínimal

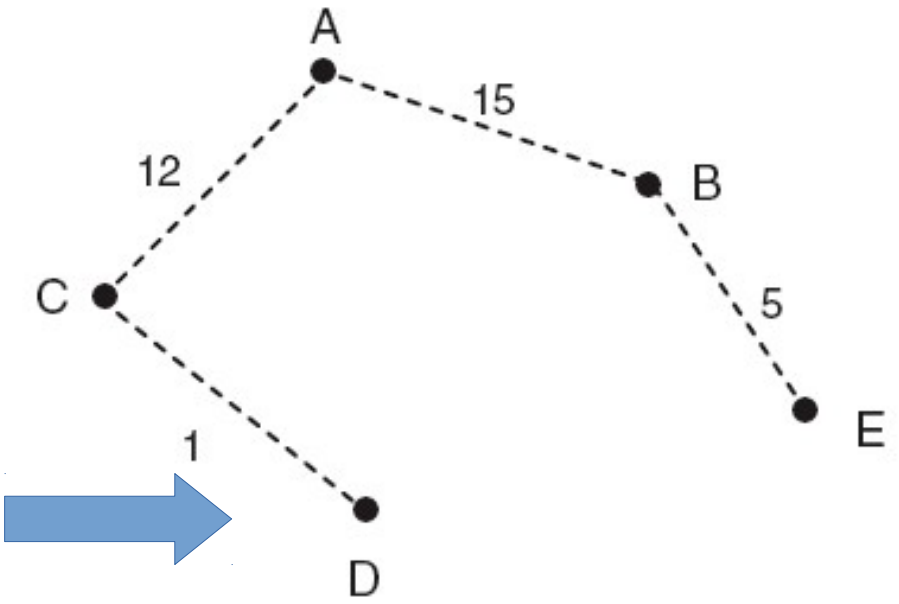




Algoritmo de Kruskal

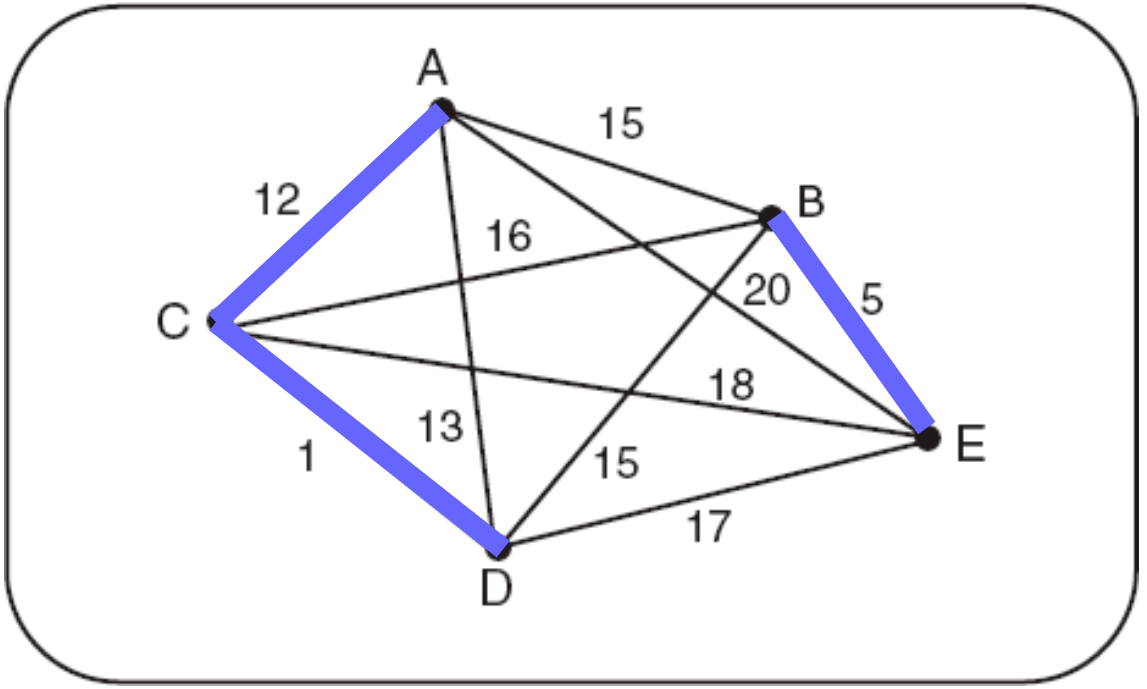
- Outra possível árvore spanning mínimal

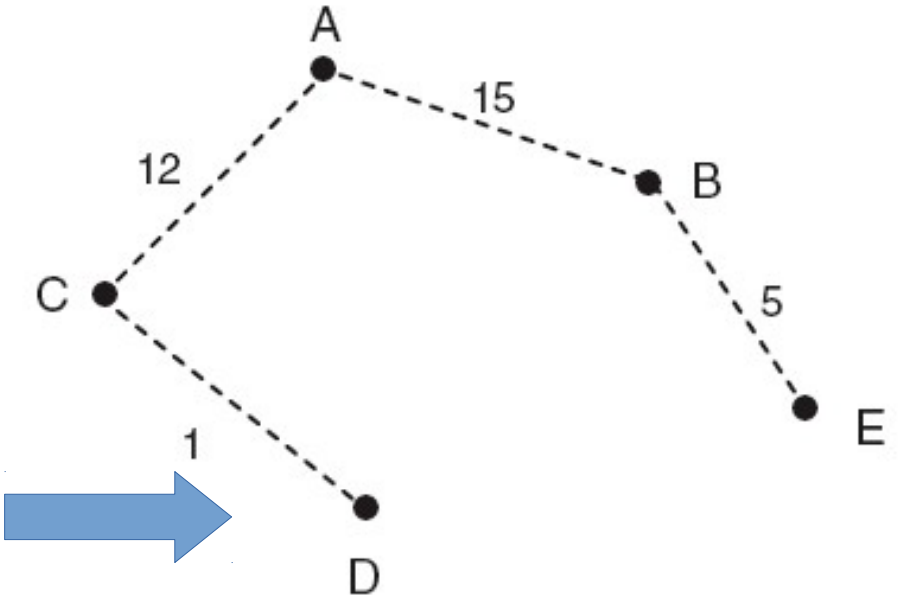




Algoritmo de Kruskal

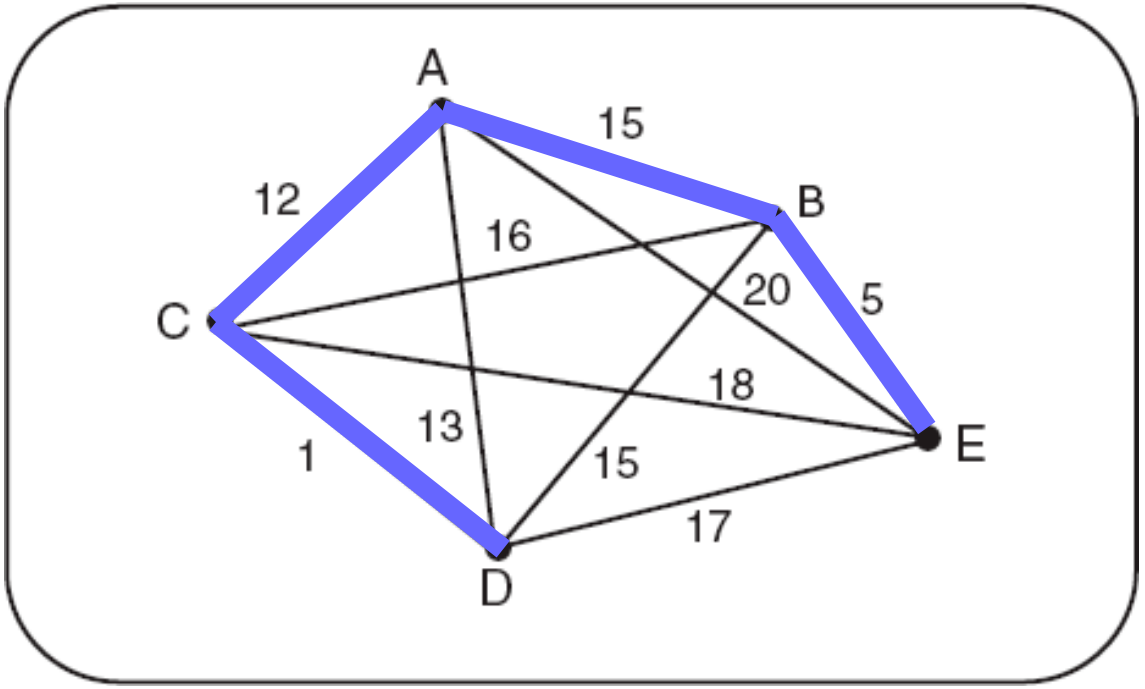
- Outra possível árvore spanning mínimal

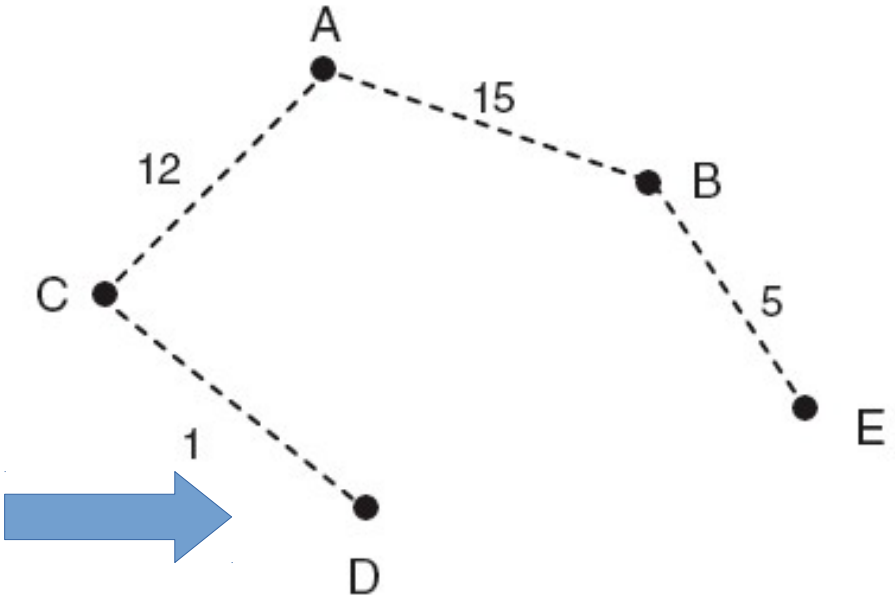




Algoritmo de Kruskal

- Outra possível árvore spanning mínimal





Algoritmo de Prim

Algoritmo de Prim

- O algoritmo escolhe um vértice qualquer vi de G.

Algoritmo de Prim

- O algoritmo escolhe um vértice qualquer vi de G.
- A seguir, é escolhida uma das arestas de G com o menor peso, que não seja um loop e que seja incidente a vi.

Algoritmo de Prim

- O algoritmo escolhe um vértice qualquer vi de G.
- A seguir, é escolhida uma das arestas de G com o menor peso, que não seja um loop e que seja incidente a vi.
- Seja essa aresta, por exemplo, ei = vivj.

Algoritmo de Prim

- O algoritmo escolhe um vértice qualquer vi de G.
- A seguir, é escolhida uma das arestas de G com o menor peso, que não seja um loop e que seja incidente a vi.
- Seja essa aresta, por exemplo, ei = vivj.
- É escolhida, a seguir, uma aresta de G com menor peso que seja incidente com vi ou com vj, mas cujo outro vértice-extremidade não seja nem vi e nem vj, ou seja, escolhe-se ek = vpvk, $p \in \{i,j\}$ e $vk \neq vi$, $vk \neq vj$.

Algoritmo de Prim

- O algoritmo escolhe um vértice qualquer vi de G.
- A seguir, é escolhida uma das arestas de G com o menor peso, que não seja um loop e que seja incidente a vi.
- Seja essa aresta, por exemplo, ei = vivj.
- É escolhida, a seguir, uma aresta de G com menor peso que seja incidente com vi ou com vj, mas cujo outro vértice-extremidade não seja nem vi e nem vj, ou seja, escolhe-se ek = vpvk, $p \in \{i,j\}$ e $vk \neq vi$, $vk \neq vj$.
- O processo de escolha da aresta com o menor peso, tal que um de seus vértices-extremidade é um dos vértices previamente escolhidos e o outro não, é repetido até que tenham sido escolhidas $n-1$ arestas.

Algoritmo de Prim

Algoritmo de Prim

- Entrada: Grafo conectado G = ({v1,v2, ...,vn}, {e1,e2, ...,em}) e {p(e1), p(e2), ..., p(en)}

Algoritmo de Prim

- Entrada: Grafo conectado G = ({v1,v2, ...,vn}, {e1,e2, ...,em}) e {p(e1), p(e2), ..., p(en)}
- Saída: Árvore spanning minimal dada por {v1,v2, ...,vn} e n−1 arestas extraídas de {e1,e2, ...,em}

Algoritmo de Prim

- Entrada: Grafo conectado G = ({v1,v2, ...,vn}, {e1,e2, ...,em}) e {p(e1), p(e2), ..., p(en)}
- Saída: Árvore spanning minimal dada por {v1,v2, ...,vn} e n−1 arestas extraídas de {e1,e2, ...,em}
- Passo 1. Escolha qualquer vértice v1 de G.

Algoritmo de Prim

- Entrada: Grafo conectado G = ({v1,v2, ...,vn}, {e1,e2, ...,em}) e {p(e1), p(e2), ..., p(en)}
- Saída: Árvore spanning minimal dada por {v1,v2, ...,vn} e n−1 arestas extraídas de {e1,e2, ...,em}
- Passo 1. Escolha qualquer vértice v1 de G.
- Passo 2. Escolha uma aresta e1 = v1v2 de G tal que v1 ≠ v2 e e1 tem o menor peso entre as arestas de G incidentes com v1.

Algoritmo de Prim

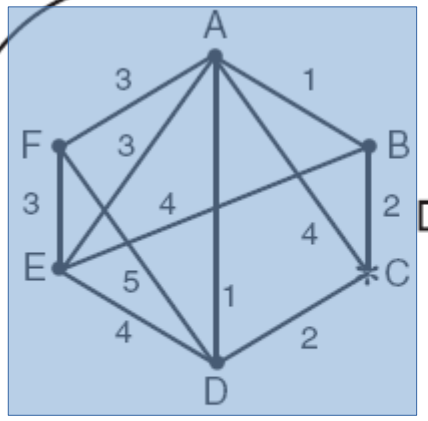
- Entrada: Grafo conectado G = ({v1,v2, ...,vn},{e1,e2, ...,em}) e {p(e1), p(e2), ..., p(en)}
- Saída: Árvore spanning minimal dada por {v1,v2, ...,vn} e n−1 arestas extraídas de {e1,e2, ...,em}
- Passo 1. Escolha qualquer vértice v1 de G.
- Passo 2. Escolha uma aresta e1 = v1v2 de G tal que v1 ≠ v2 e e1 tem o menor peso entre as arestas de G incidentes com v1.
- Passo 3. Se as arestas e1, e2, ..., ei já foram escolhidas envolvendo os vértices-extremidade v1, v2, .., vi+1, escolha uma aresta ei+1 = vjvk, com vj ∈ {v1, v2, ..., vi+1} e vk ∉ {v1, v2, ..., vi+1} tal que ei+1 tem o menor peso entre as arestas de G, com precisamente um vértice-extremidade em {v1, v2, ..., vi+1}.

Algoritmo de Prim

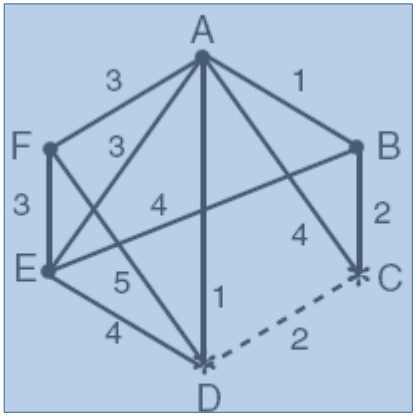
- Entrada: Grafo conectado G = ({v1,v2, ...,vn},{e1,e2, ...,em}) e {p(e1), p(e2), ..., p(en)}
- Saída: Árvore spanning minimal dada por {v1,v2, ...,vn} e n−1 arestas extraídas de {e1,e2, ...,em}
- Passo 1. Escolha qualquer vértice v1 de G.
- Passo 2. Escolha uma aresta e1 = v1v2 de G tal que v1 ≠ v2 e e1 tem o menor peso entre as arestas de G incidentes com v1.
- Passo 3. Se as arestas e1, e2, ..., ei já foram escolhidas envolvendo os vértices-extremidade v1, v2, .., vi+1, escolha uma aresta ei+1 = vjvk, com vj ∈ {v1, v2, ..., vi+1} e vk ∉ {v1, v2, ..., vi+1} tal que ei+1 tem o menor peso entre as arestas de G, com precisamente um vértice-extremidade em {v1, v2, ..., vi+1}.
- Passo 4. Pare após n−1 arestas terem sido escolhidas; caso contrário, repita o Passo 3.

Algoritmo de Prim

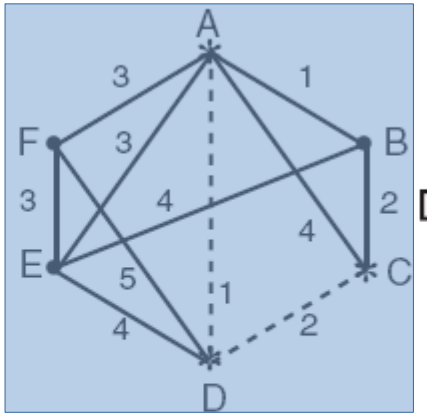
- Exemplo



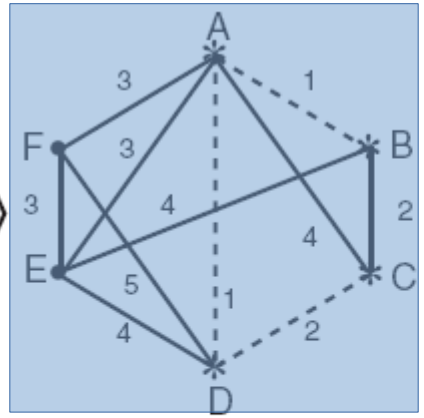
v_1 = C (poderia ser qualquer outro vértice)



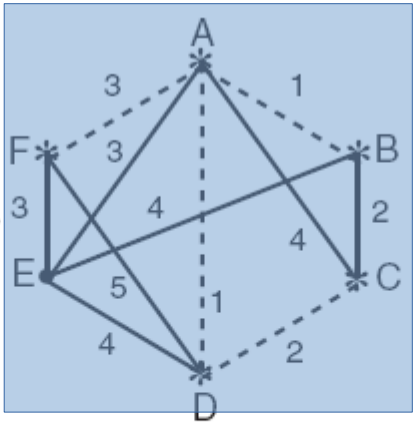
e_1 = CD tal que v_2 = D (uma alternativa é e_1 = CB)



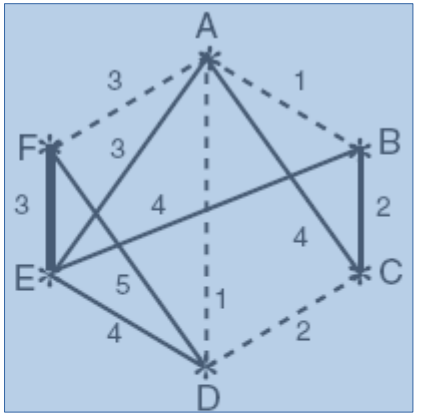
e_2 = DA tal que v_3 = A (nenhuma outra escolha nesse estágio)



e_3 = AB tal que v_4 = B (nenhuma outra alternativa nesse estágio)



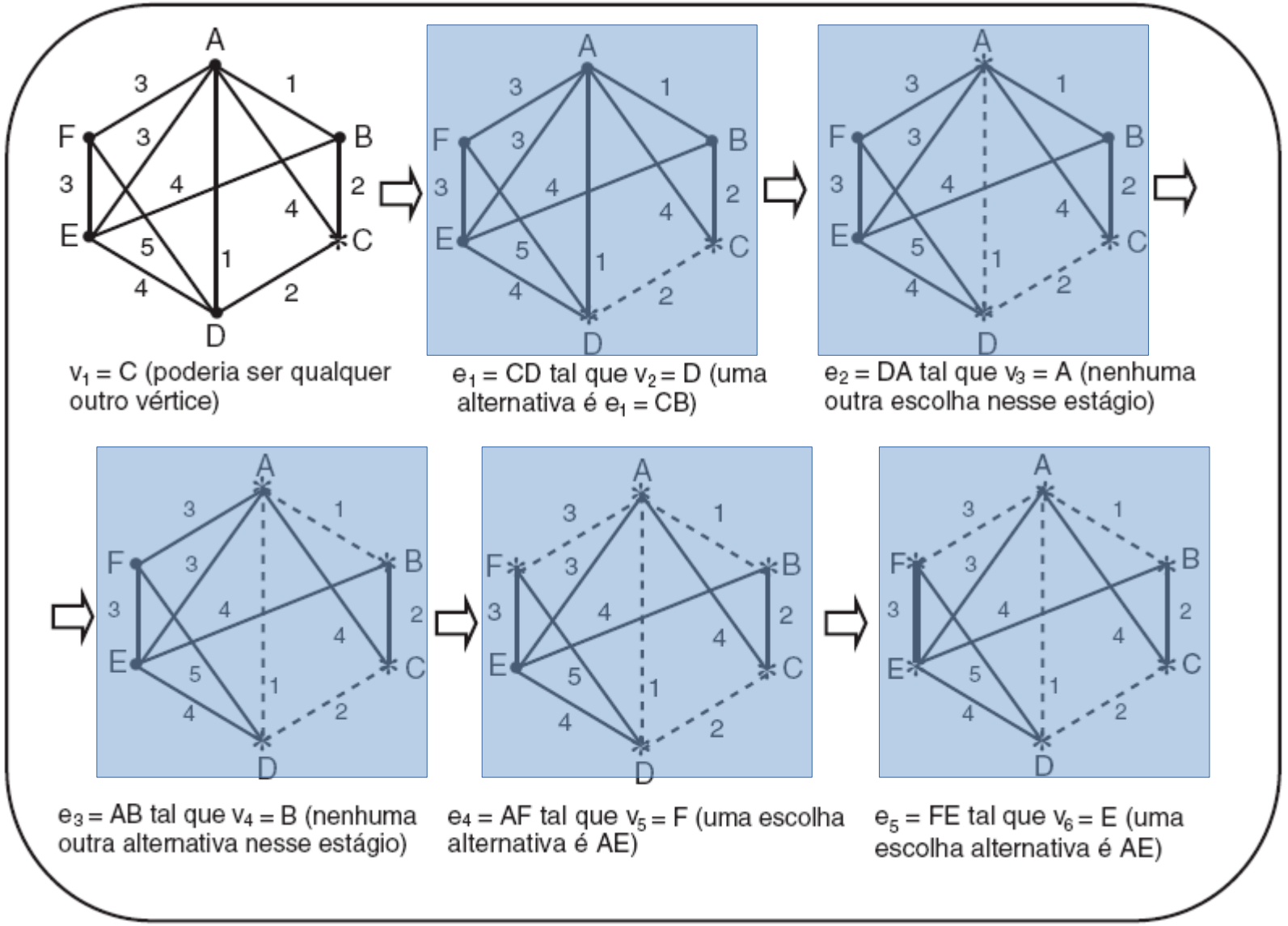
e_4 = AF tal que v_5 = F (uma escolha alternativa é AE)



e_5 = FE tal que v_6 = E (uma escolha alternativa é AE)

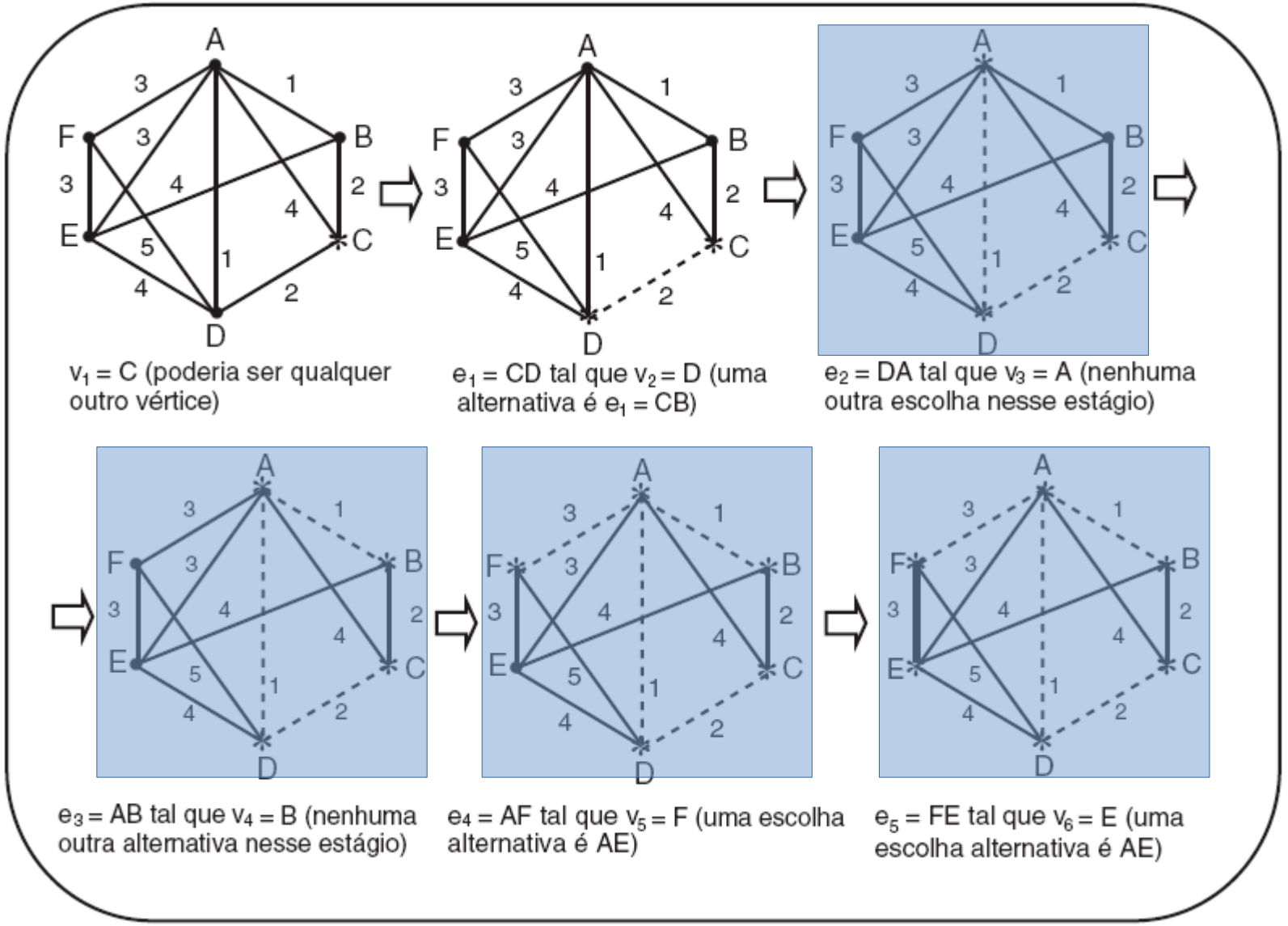
Algoritmo de Prim

- Exemplo



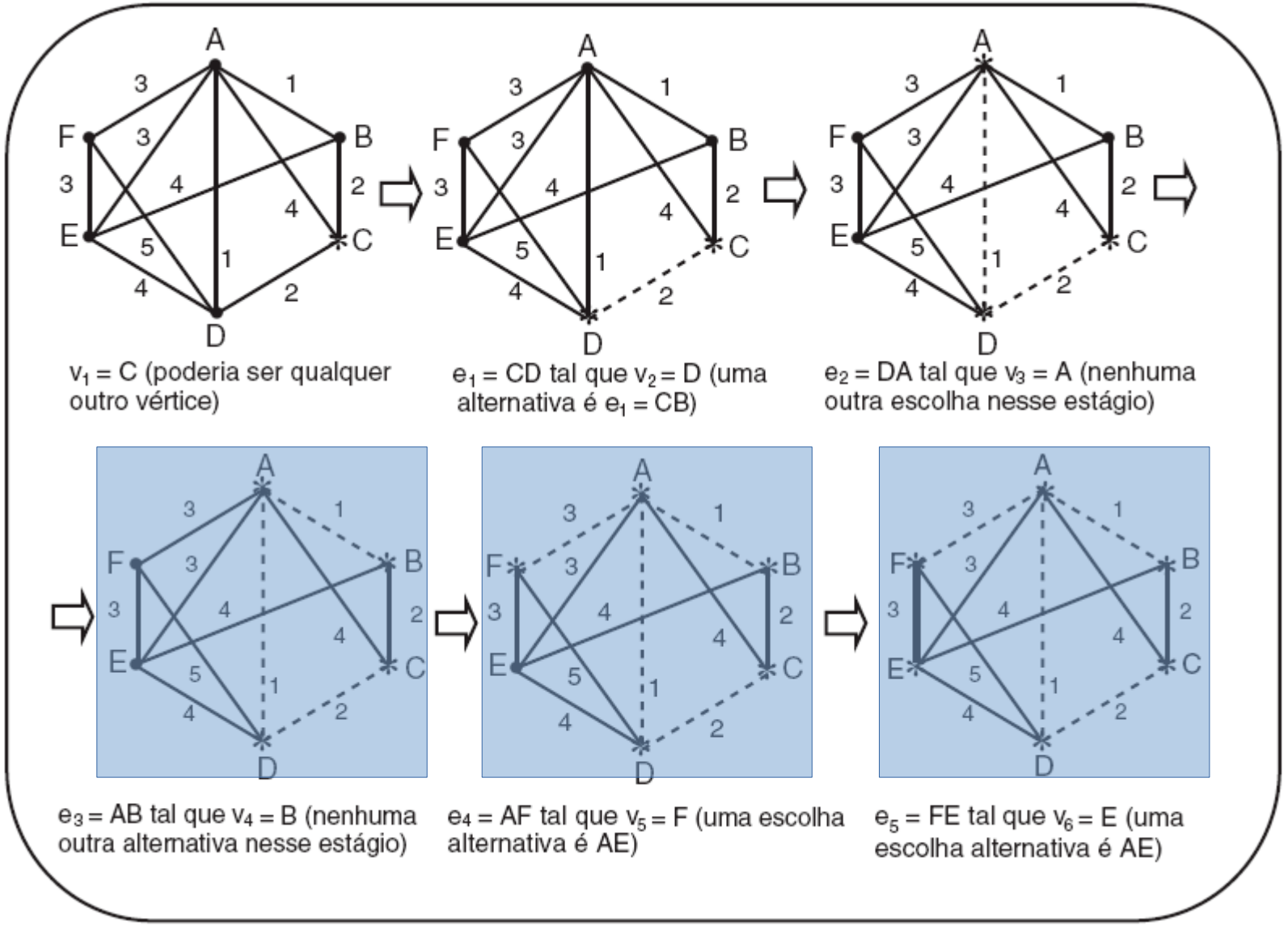
Algoritmo de Prim

- Exemplo



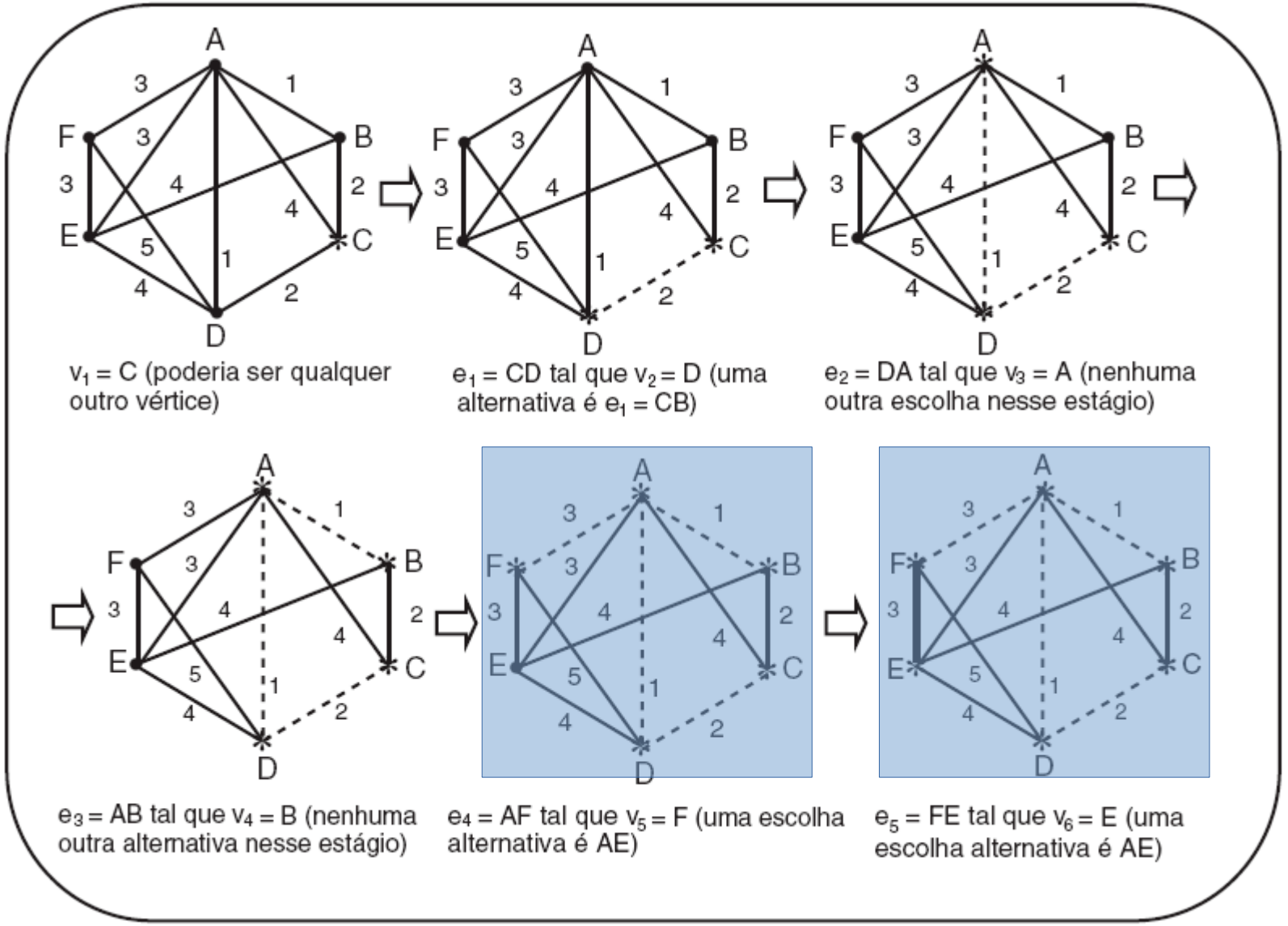
Algoritmo de Prim

- Exemplo



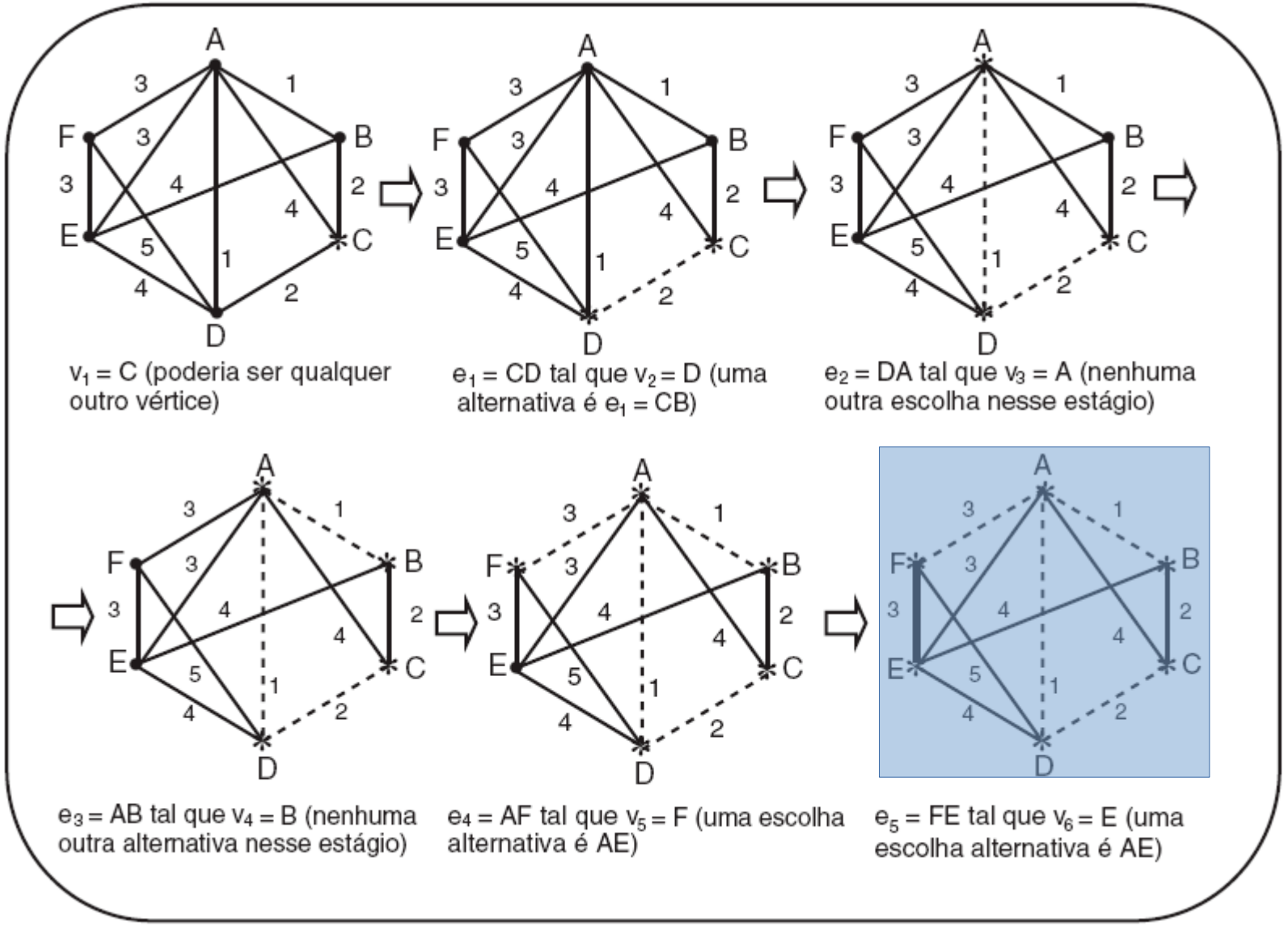
Algoritmo de Prim

- Exemplo



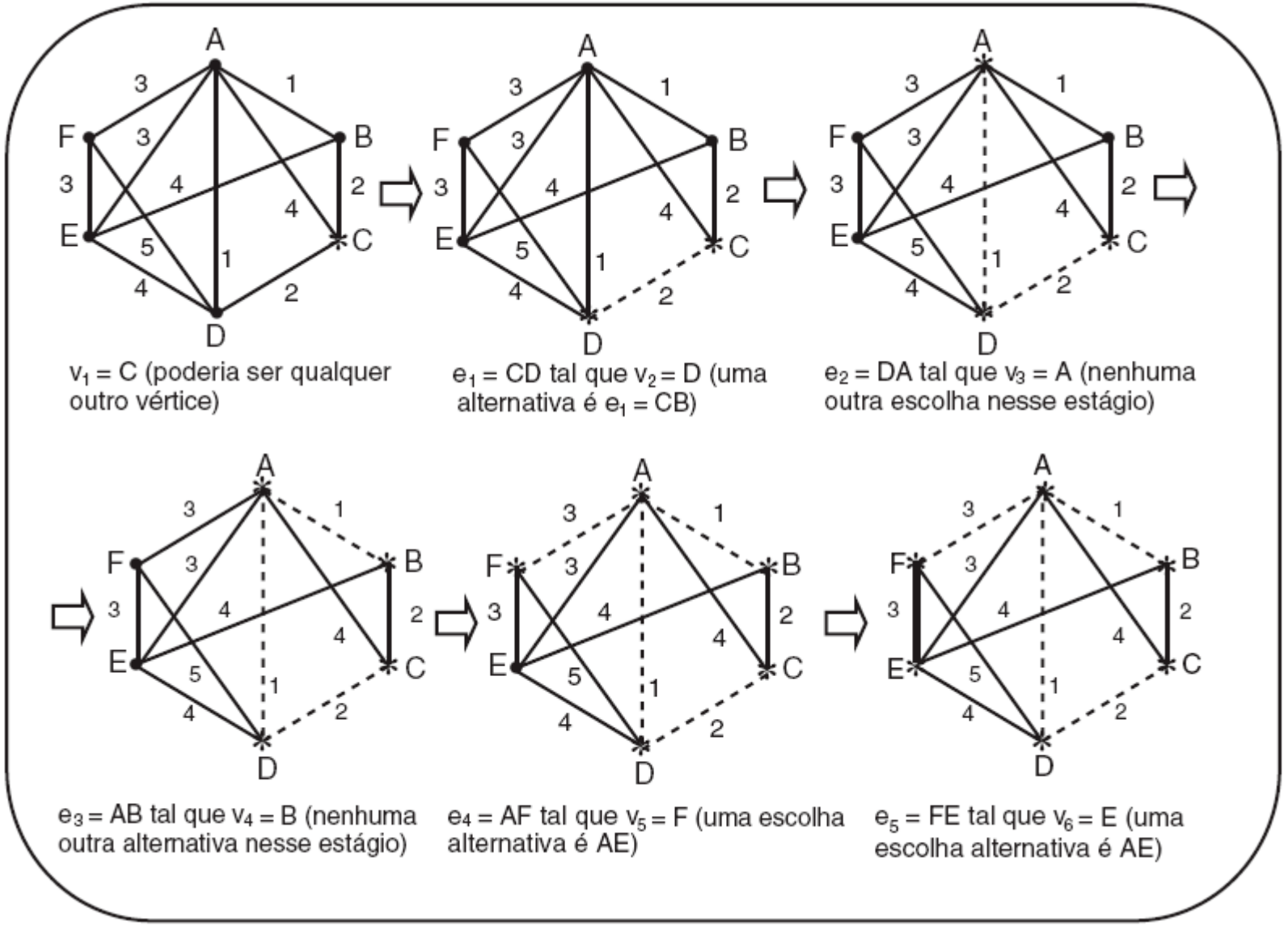
Algoritmo de Prim

- Exemplo



Algoritmo de Prim

- Exemplo



Algoritmo de Prim

Algoritmo de Prim

- As principais diferenças entre o algoritmo de Kruskal e de Prim são:

Algoritmo de Prim

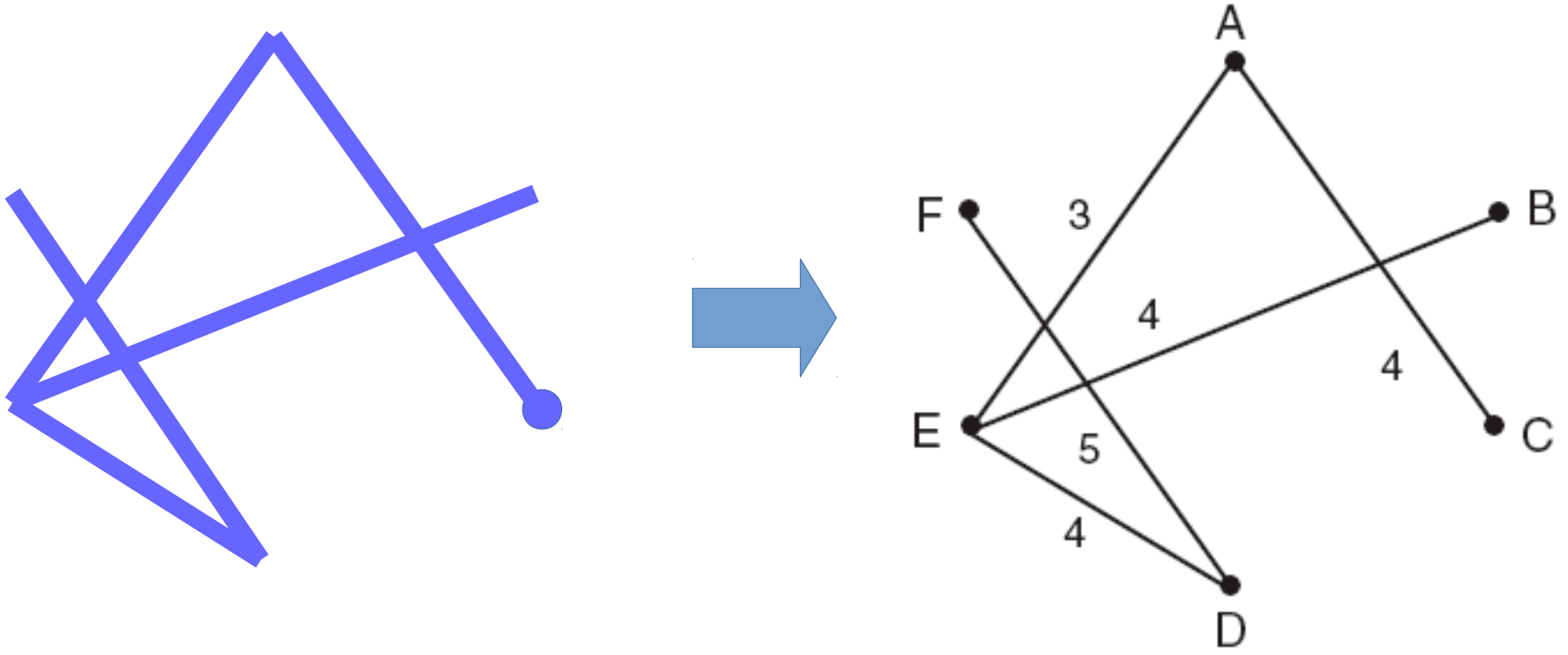
- As principais diferenças entre o algoritmo de Kruskal e de Prim são:
  - O algoritmo de Kruskal pode construir várias subárvores “simultaneamente” e, então, uni-las. O algoritmo de Prim “cresce” uma mesma subárvore a partir de um vértice inicial.

Algoritmo de Prim

- As principais diferenças entre o algoritmo de Kruskal e de Prim são:
  - O algoritmo de Kruskal pode construir várias subárvores “simultaneamente” e, então, uni-las. O algoritmo de Prim “cresce” uma mesma subárvore a partir de um vértice inicial.
  - O algoritmo de Kruskal depende da habilidade de detecção de ciclos, e o de Prim, de não escolher um vértice previamente escolhido. Essa característica torna uma implementação do algoritmo de Prim mais rápida do que uma do de Kruskal.

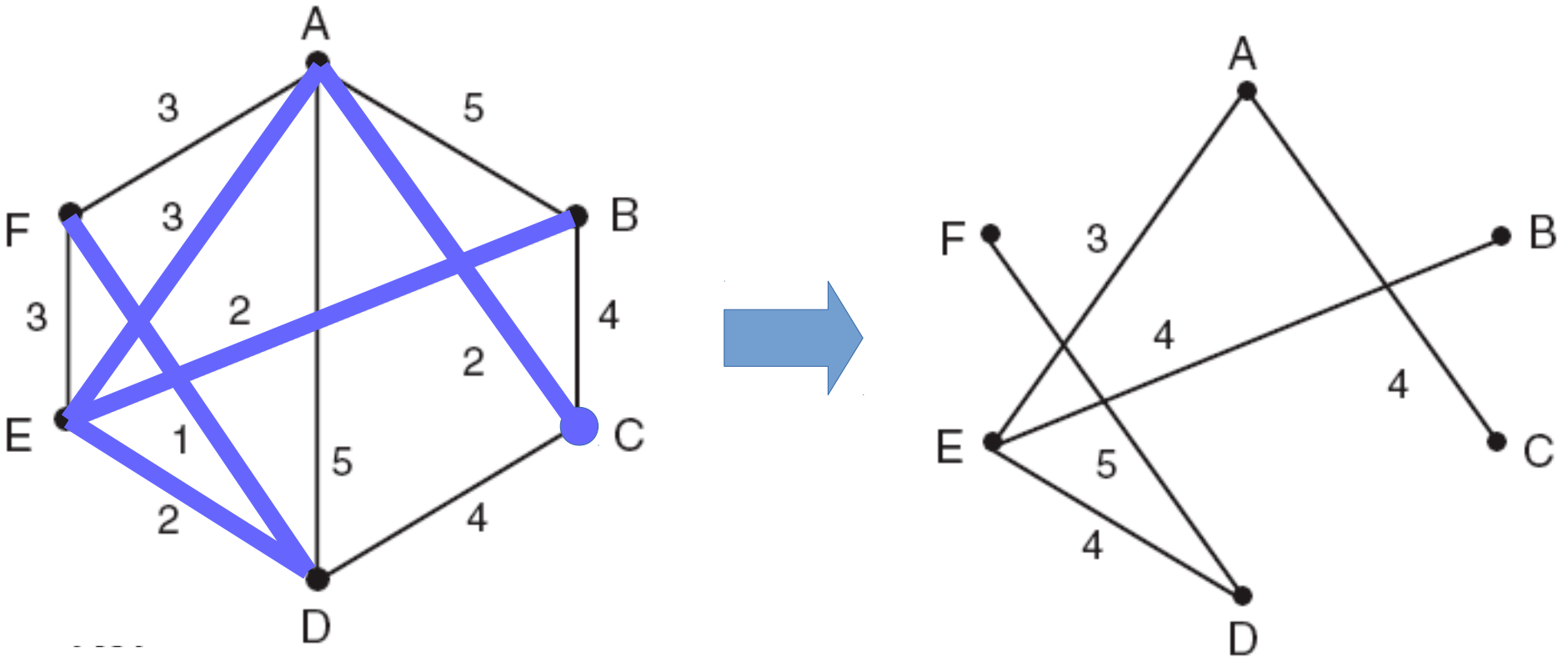
Algoritmo de Prim

- Construção da árvore



Algoritmo de Prim

- Construção da árvore



Spanning Tree

Spanning Tree

- Os dois algoritmos vistos anteriormente geram árvores spanning minimais.

Spanning Tree

- Os dois algoritmos vistos anteriormente geram árvores spanning minimais.
- Eles podem também ser usados para gerar árvores spanning maximais, ou seja, árvores que têm o maior peso possível.

Exercícios

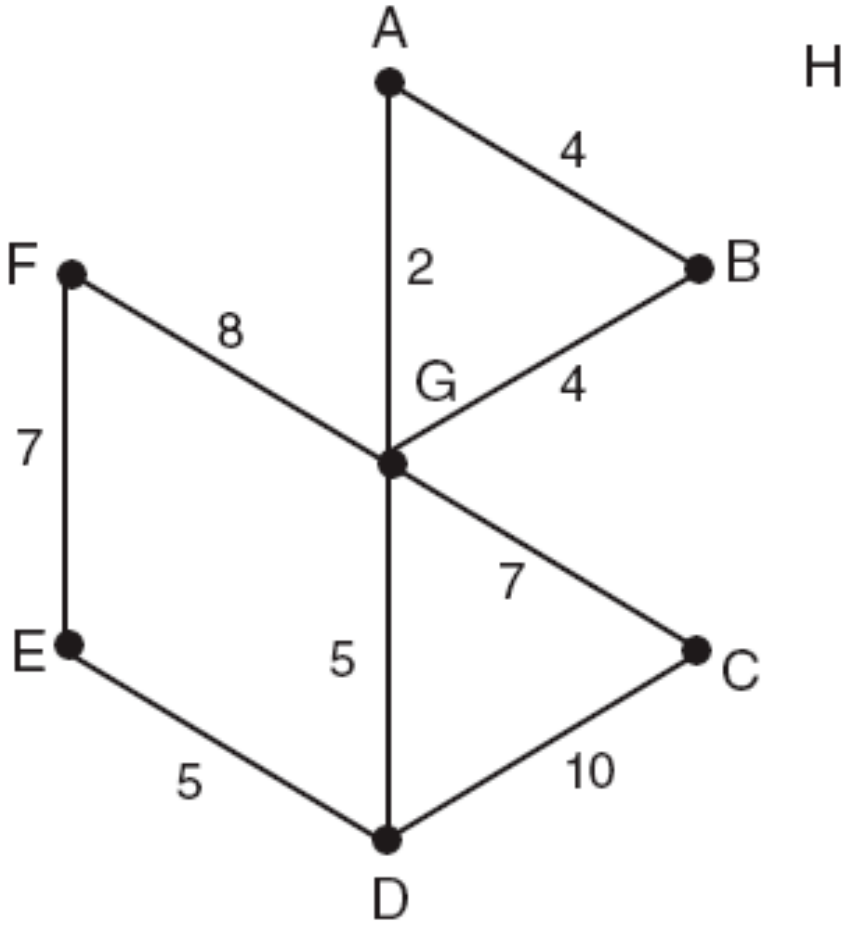
- Um vértice de corte em um grafo G pode ter grau igual a 1? Justifique a sua resposta e apresente um exemplo.
- Considere o grafo não-dirigido definido pelo conjunto de arestas 0-1 1-2 2-0 1-3 1-4 4-5 5-6 6-4. Faça uma lista de todas as articulações do grafo.
- Faça uma lista de todas as articulações do grafo não-dirigido definido pelo conjunto de arestas 8-9 3-7 1-4 7-8 0-5 5-2 3-8 2-9 0-6 4-9 2-6 6-4.

Exercícios

- Quantas articulações tem uma árvore?
- O que caracteriza uma *spanning tree*? Desenhe um exemplo.
- Considere uma árvore geradora com n vértices. Quantas arestas têm essa árvore? Justifique a sua resposta.
- É necessário verificar se o grafo é conexo antes de executar o algoritmo de Prim?
- Podemos afirmar que o caminho entre dois vértices de um grafo que faz parte da árvore geradora mínima é o caminho mais curto entre esses dois vértices?

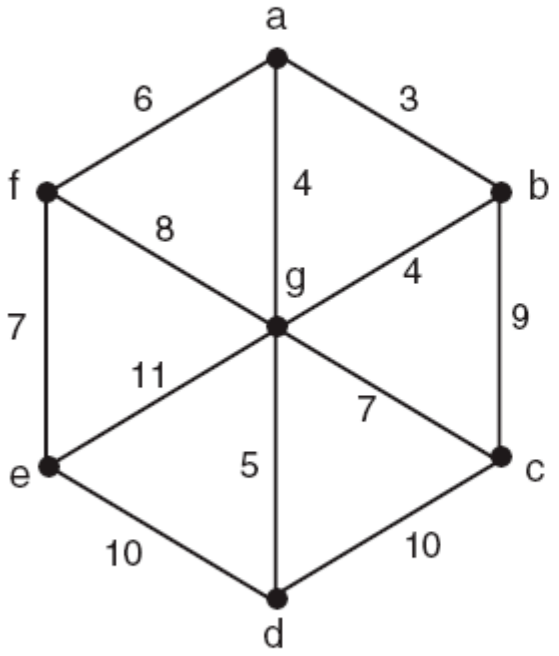
Exercícios

- Construa a árvore spanning do grafo H abaixo usando o algoritmo de Kruskal e de Prim.



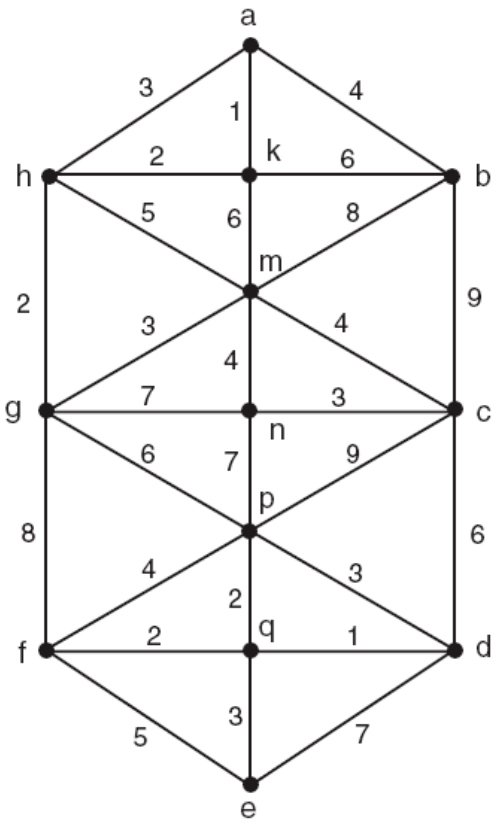
Exercícios

- Encontre a árvore spanning minimal e maximal para cada um dos grafos conectados ponderados usando os algoritmos de Kruskal e de Prim.



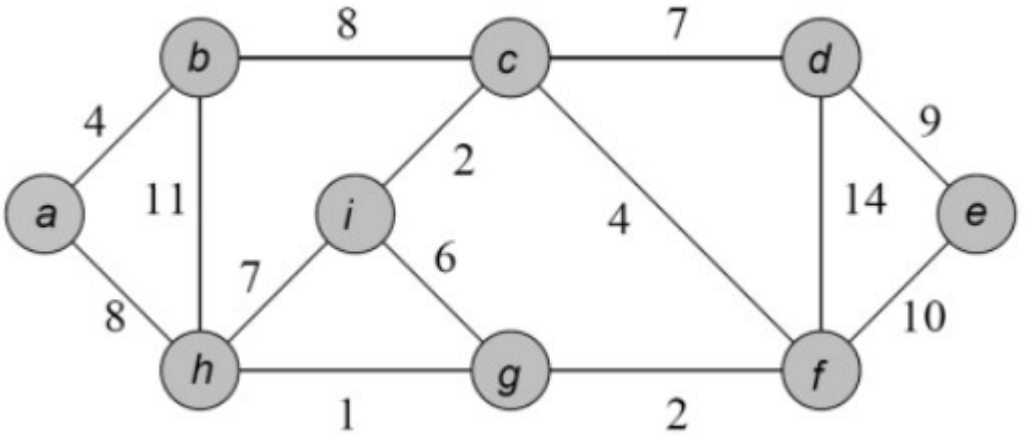
Exercícios

- Encontre a árvore spanning minimal e maximal para cada um dos grafos conectados ponderados usando os algoritmos de Kruskal e de Prim.



Exercícios

- Encontre árvore geradora mínima do seguinte grafo:
  - Usando o algoritmo de Kruskal.
  - Usando o algoritmo de Prim, começando a partir do vértice no extremo esquerdo.



Exercícios

- Ache a árvore geradora mínima do grafo abaixo utilizando o algoritmo de Prim começando pelo vértice A.

	A	B	C	D	E	F	G	H	I	J
A	0	15	10	19	0	0	0	0	0	0
B	15	0	0	7	17	0	0	0	0	0
C	10	0	0	16	0	14	0	0	0	0
D	19	7	16	0	12	6	3	0	0	0
E	0	17	0	12	0	0	20	13	0	0
F	0	0	14	6	0	0	9	0	5	0
G	0	0	0	3	20	9	0	4	1	11
H	0	0	0	0	13	0	4	0	0	2
I	0	0	0	0	0	5	1	0	0	18
J	0	0	0	0	0	0	11	2	18	0