

---

# Grafos: árvores geradoras mínimas

---

Graça Nunes

---

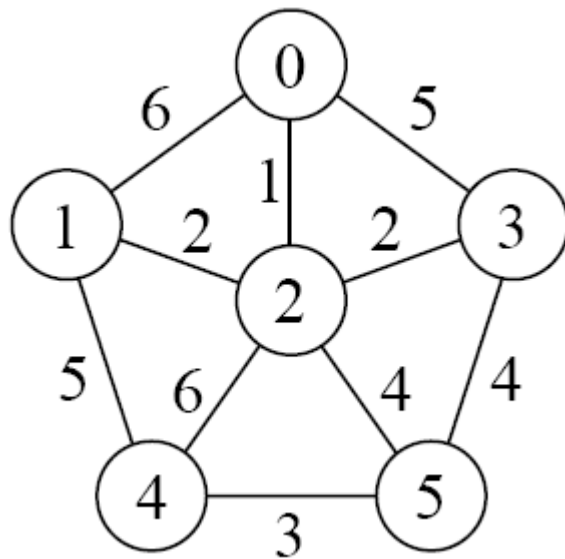
# Motivação

- Suponha que queremos construir **estradas** para interligar  **$n$**  cidades
  - Cada estrada direta entre as cidades  **$i$**  e  **$j$**  tem um custo associado
  - Nem todas as cidades precisam ser ligadas diretamente, desde que todas sejam acessíveis...
- Como determinar eficientemente **quais** estradas devem ser construídas de forma a **minimizar o custo total de interligação** das cidades?

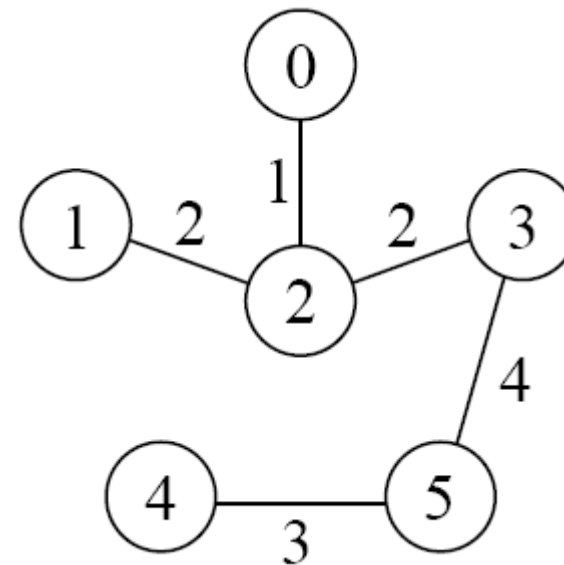
# Motivação

## ■ Exemplo

G

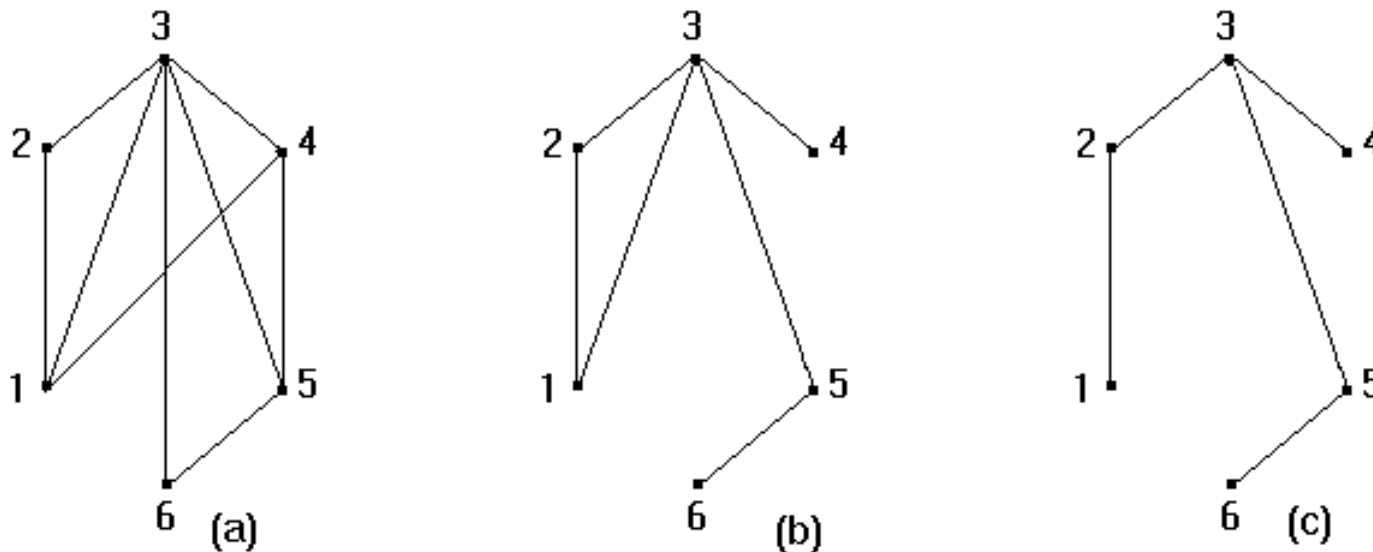


Árvore Geradora Mínima de G



# Subgrafo gerador

**Subgrafo Gerador** ou subgrafo de espalhamento de um grafo  $G_1(V_1, E_1)$  é um subgrafo  $G_2(V_2, E_2)$  de  $G_1$  tal que  $V_1 = V_2$  e  $E_2 \subseteq E_1$ . Quando o subgrafo gerador é uma **árvore**, ele recebe o nome de **árvore geradora** (ou de espalhamento).



**b** e **c** são subgrafos geradores de **a**  
**c** é árvore geradora de **a** e **b**

---

# Subgrafo gerador de custo mínimo

## ■ Formalmente

- Dado um grafo não orientado  $G(V,E)$ 
  - onde  $w: E \rightarrow \mathfrak{R}^+$  define os custos das arestas
  - queremos encontrar um subgrafo gerador conexo  $T$  de  $G$  tal que, para todo subgrafo gerador conexo  $T'$  de  $G$
  - $T$  é um subgrafo gerador de custo mínimo

$$\sum_{e \in T} w(e) \leq \sum_{e \in T'} w(e)$$

---

# Árvore geradora mínima

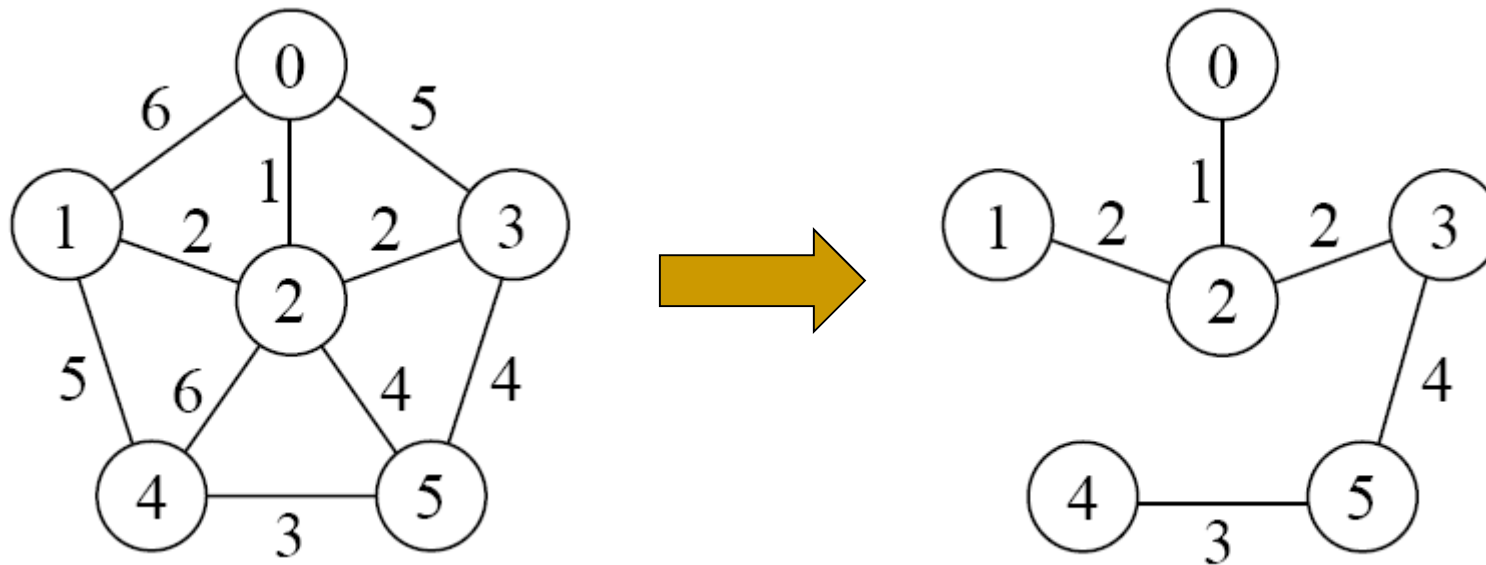
- Claramente, o problema só tem solução se  $G$  é conexo
  - A partir de agora, assumimos  $G$  conexo
- Também não é difícil ver que a solução para esse problema será sempre uma árvore
  - Basta notar que  $T$  não terá ciclos pois, caso contrário, poderíamos obter um outro sub-grafo  $T'$ , ainda conexo e com custo menor que o de  $T$ , removendo o ciclo!

# Árvore geradora mínima

- **Árvore Geradora** (*Spanning Tree*) de um grafo  $G$  é um subgrafo de  $G$  que contém **todos os seus vértices** (i.e. subgrafo gerador) e, ainda, é uma árvore
- **Árvore Geradora Mínima** (*Minimum Spanning Tree, MST*) é a árvore geradora de um grafo valorado cuja soma dos **pesos associados às arestas é mínimo**, i.e., é uma árvore geradora de custo mínimo

# Árvore geradora mínima

- Exemplo





---

# Árvore geradora mínima

- Como encontrar a árvore geradora mínima de um grafo  $G$  ?
  - Algoritmo genérico
  - Algoritmo de Prim
  - Algoritmo de Kruskal

# Árvore geradora mínima

## Algoritmo Genérico

```
procedimento genérico(G)
A = ∅
enquanto A não define uma árvore
    encontre uma aresta (u,v) segura para A
    A = A ∪ {(u,v)}
retorna A
```

$G$  conexo, não direcionado, ponderado

$A$  - conjunto de arestas

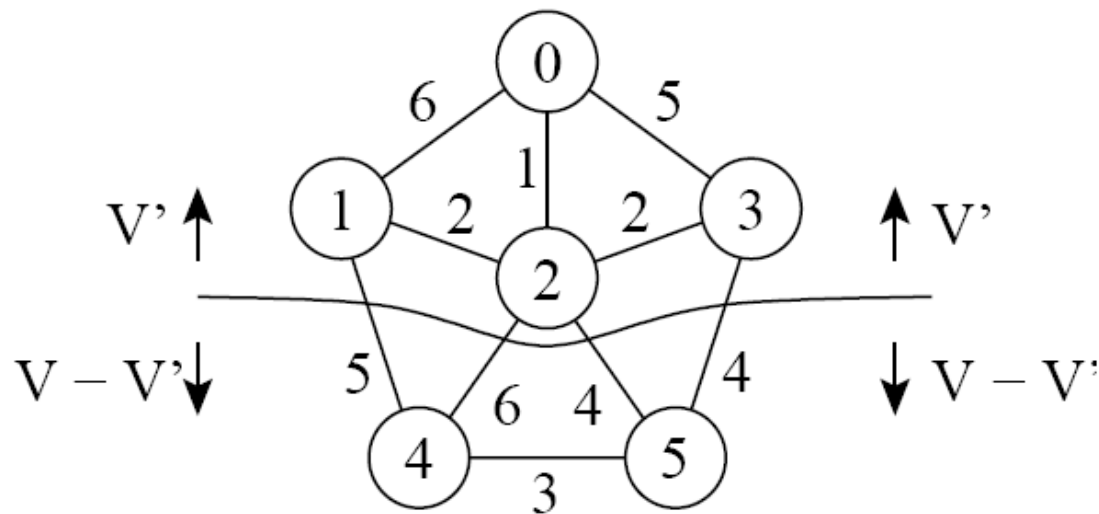
Abordagem 'gulosa' -> adiciona uma aresta segura a cada rodada

Aresta é 'segura' se mantém a condição de que, antes de cada iteração,  $A$  é uma árvore geradora mínima de um subconjunto de vértices

# Árvore geradora mínima

- Alguns conceitos

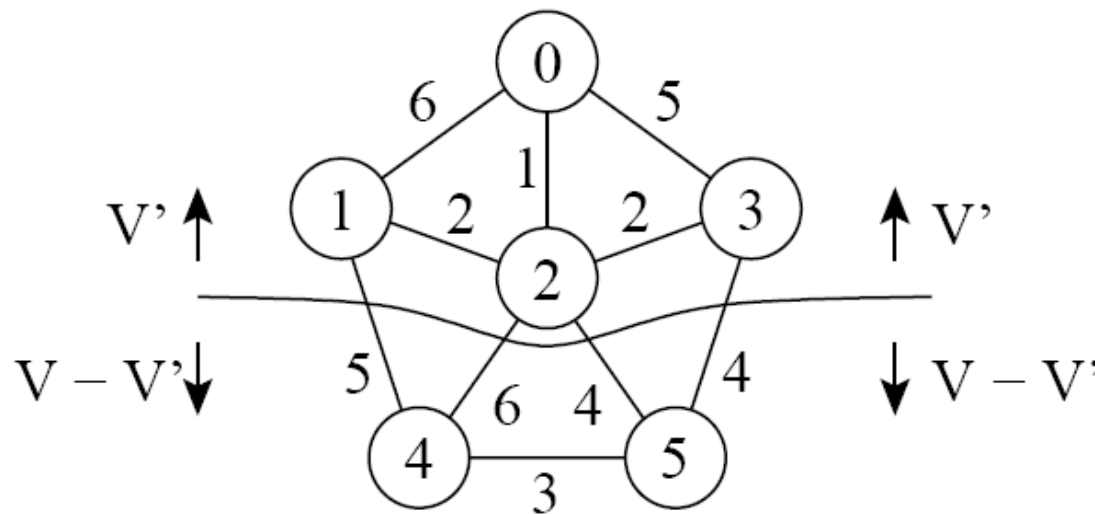
- Um **corte**  $(V'; V-V')$  de um grafo não direcionado  $G=(V;A)$  é uma partição de  $V$
- Uma aresta  $(u,v)$  **cruza o corte** se um vértice pertence a  $V'$  e o outro a  $V-V'$



# Árvore geradora mínima

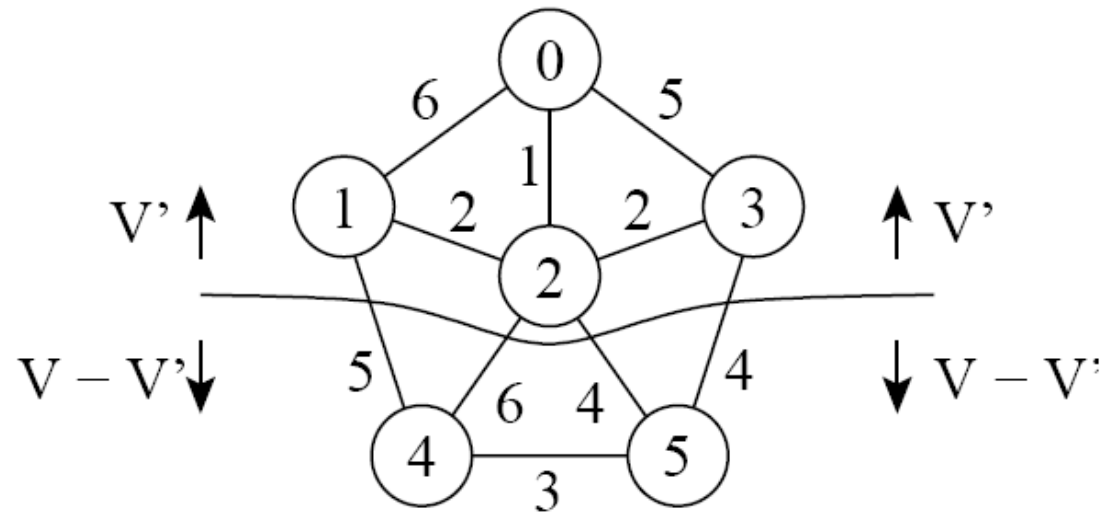
- Alguns conceitos

- Um corte *respeita* um conjunto  $S$  de arestas se não existirem arestas em  $S$  que o cruzem
- Uma aresta cruzando o corte que tenha custo mínimo em relação a todas as arestas cruzando o corte é uma *aresta leve*



# Árvore geradora mínima

- Exemplo



- Se  $S$  é uma árvore geradora mínima de um sub-grafo e há um corte  $(V'; V-V')$  que respeita  $S$ , a aresta leve  $(u,v)$  é uma aresta segura para  $S$

---

# Algoritmo de Prim

```
procedimento Prim(G)
```

```
  escolha um vértice  $s$  para iniciar a árvore
```

```
  enquanto há vértices que não estão na árvore
```

```
    selecione uma aresta segura
```

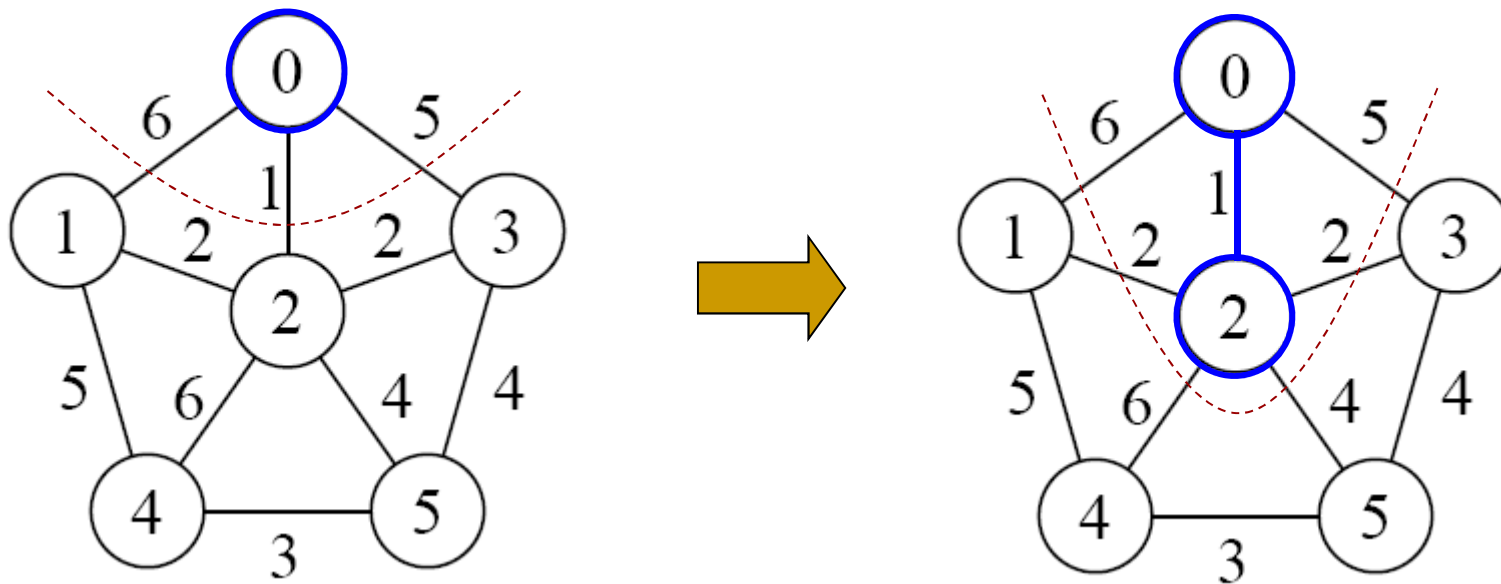
```
    insira a aresta e seu vértice na árvore
```

Ponto importante do algoritmo: [seleção de uma aresta segura](#)

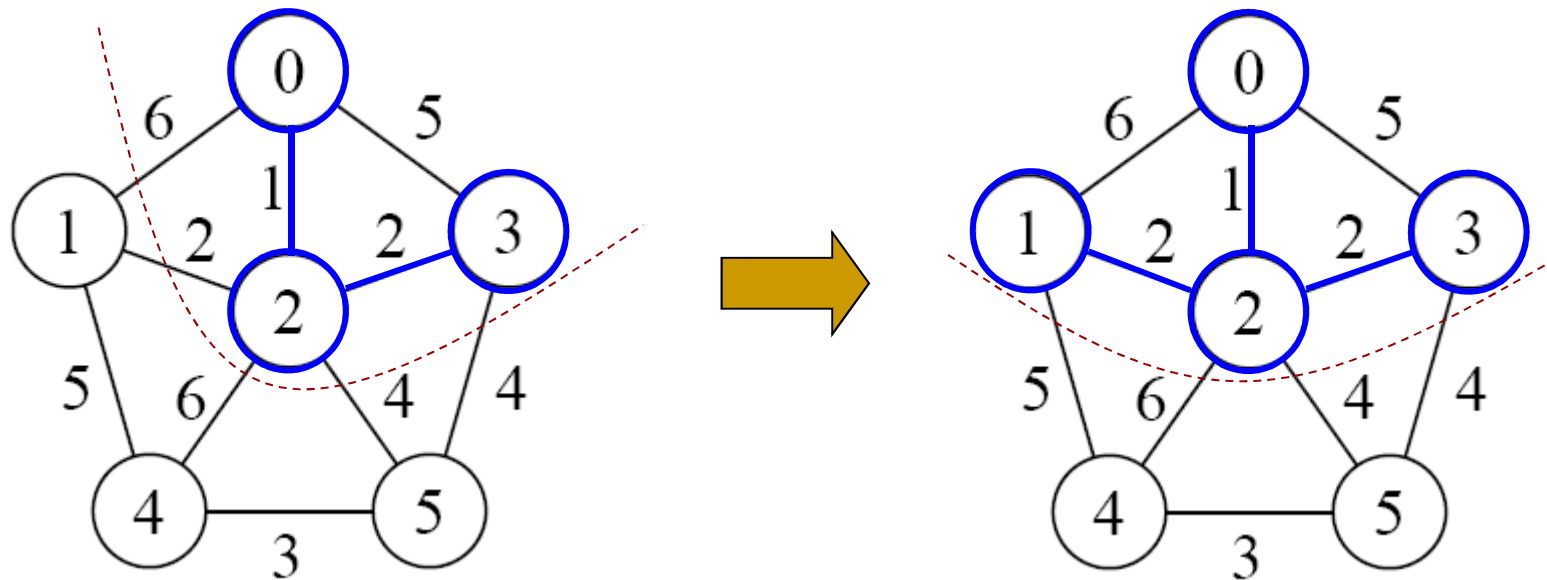
---

# Algoritmo de Prim

- Exemplo: iniciando o algoritmo pelo vértice 0



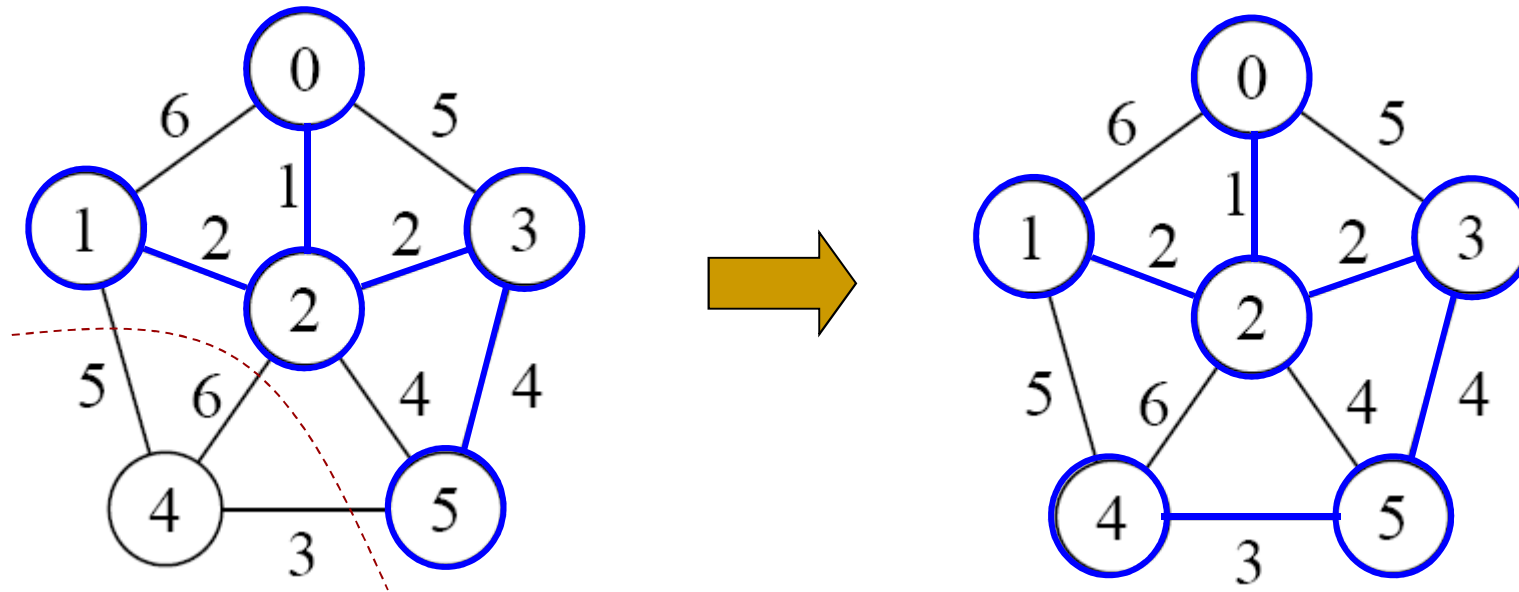
# Algoritmo de Prim





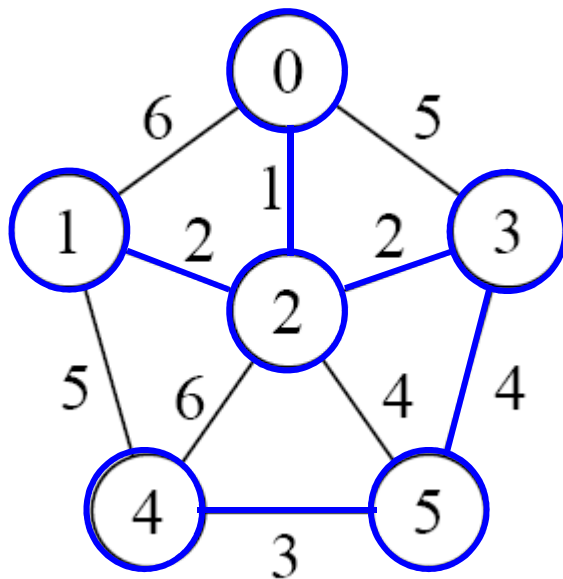
# Algoritmo de Prim

Árvore geradora mínima

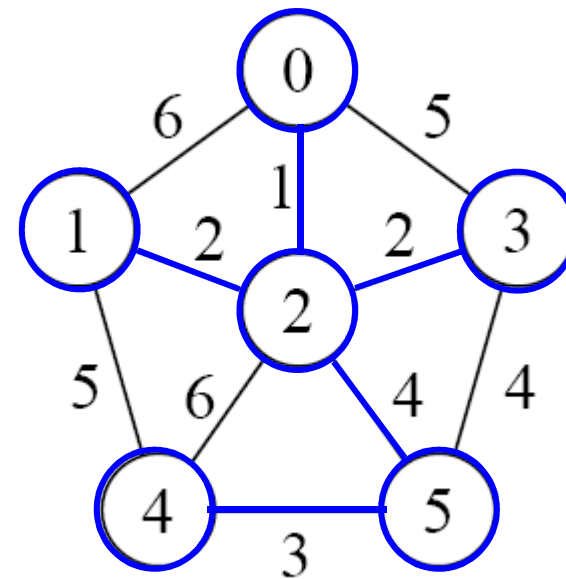


# Algoritmo de Prim

- Há mais de uma árvore geradora mínima para um mesmo grafo



ou



---

# Algoritmo de Prim

- Maneira mais eficiente de determinar a aresta segura
  - Manter todas as arestas que ainda não estão na árvore em fila(s) de prioridade (heaps)
    - Prioridade é dada à aresta de menor peso adjacente a um vértice na árvore e outro fora dela

Complexidade de temp:  $O(|A| \log(|V|))$

---

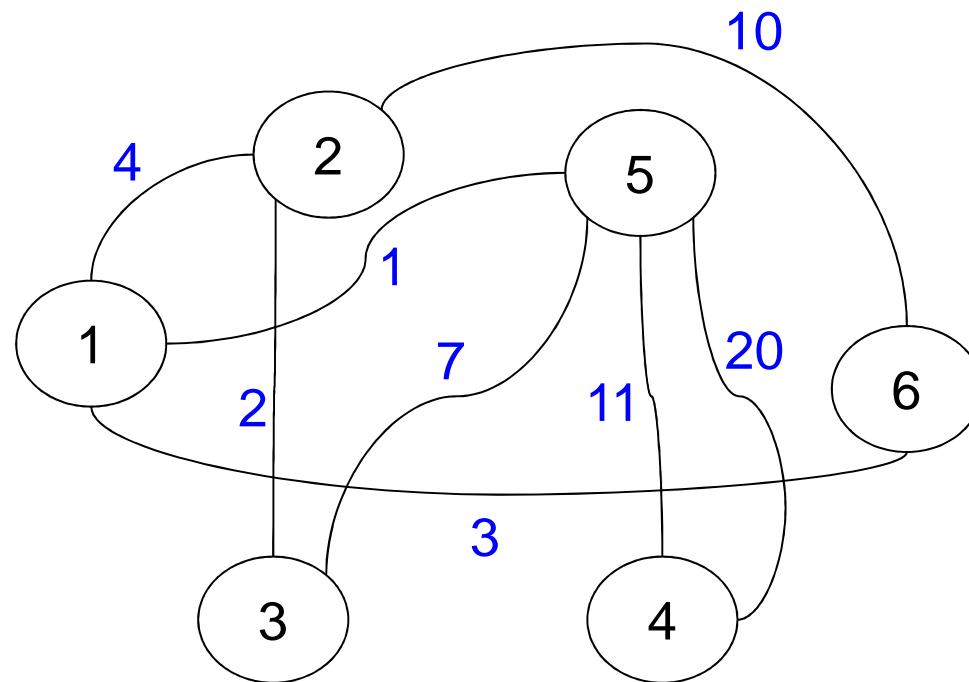
# Algoritmo de Kruskal

- Também encontra árvore geradora mínima em  $O(|A| \log |V|)$
- Pode ser usado para florestas;
  - Constrói a árvore acrescentando arestas;
  - Não parte de um nó específico

Verifique na literatura!

# Árvore geradora mínima

- Exercício: encontre uma árvore geradora mínima para o grafo abaixo utilizando o algoritmo de Prim (e o de Kruskal)



---

# Árvore geradora mínima

- Exercício

- Implementação em C de sub-rotina para encontrar uma árvore geradora mínima para um grafo (algoritmo de Prim)

- Decida como buscar as arestas seguras

- Lembre-se: essa busca deve ser eficiente!