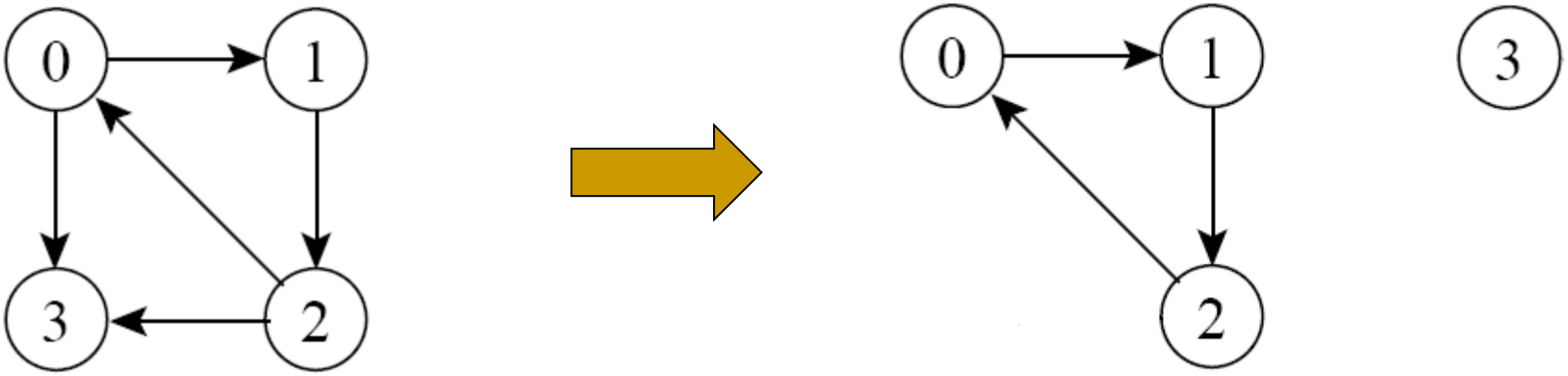
Grafos: componentes fortemente conexos, árvores geradoras mínimas

SCE-183 Algoritmos e Estruturas de Dados 2

Thiago A. S. Pardo
Maria Cristina

Componentes fortemente conexos

- Um componente fortemente conectado de um grafo direcionado é um subconjunto máximo de vértices tal que, para todo par de vértices u e v, u e v são mutuamente alcançáveis



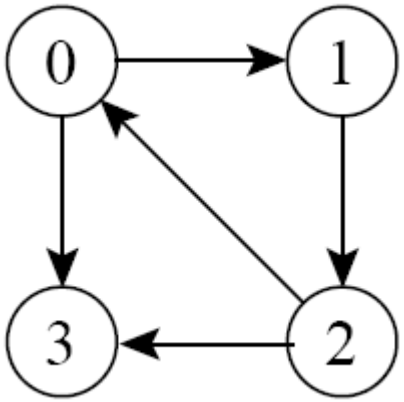
Componentes fortemente conexos

- Idéia básica
  - Um grafo direcionado G e sua versão transposta T devem possuir os mesmos componentes fortemente conectados
  - Grafo transposto
    - Mesmos vértices
    - Se há a aresta (u,v) em G, há (v,u) em T

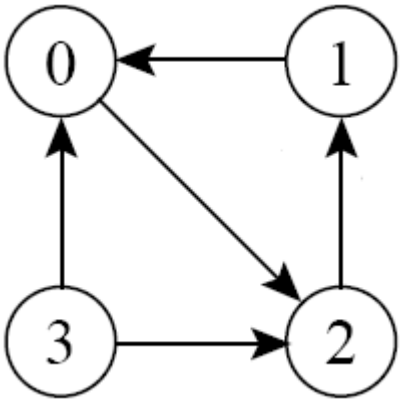
Componentes fortemente conexos

- Exemplo: transpondo um grafo
  - Os componentes fortemente conectados são os mesmos

G



T



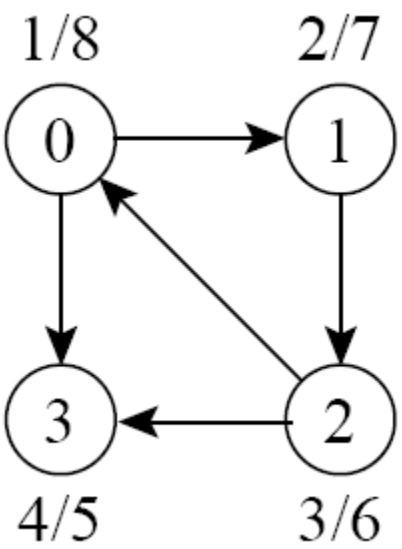
Componentes fortemente conexos

- Algoritmo: uso da busca em profundidade

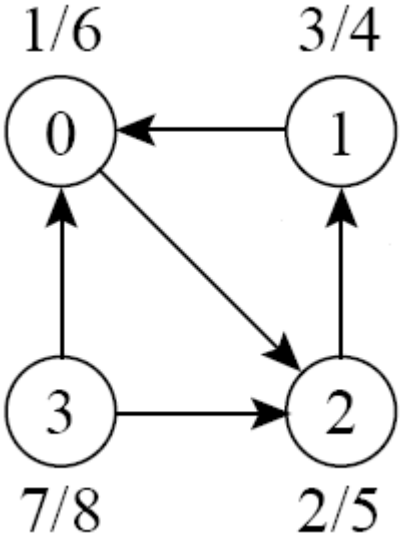
1. Busca em profundidade para determinar tempo de término de cada vértice no grafo original
2. Transposição do grafo
3. Busca em profundidade no grafo transposto a partir do vértice de maior tempo de término; se necessário, reinicia-se esse processo a partir do próximo vértice de maior tempo de término
4. Os vértices de cada árvore de busca em profundidade do grafo transposto formam os componentes fortemente conectados

Componentes fortemente conexos

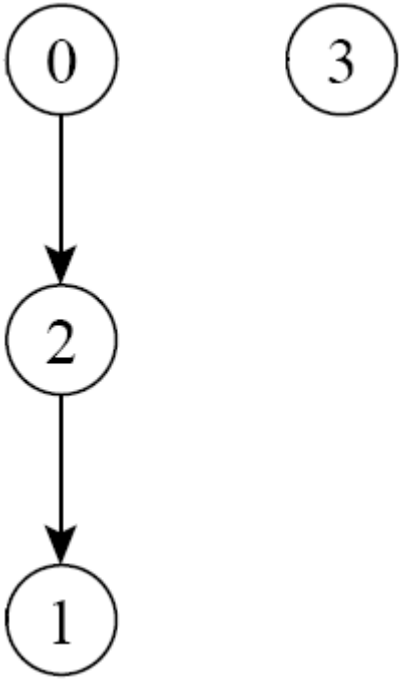
Busca em profundidade



Busca em profundidade no grafo transposto

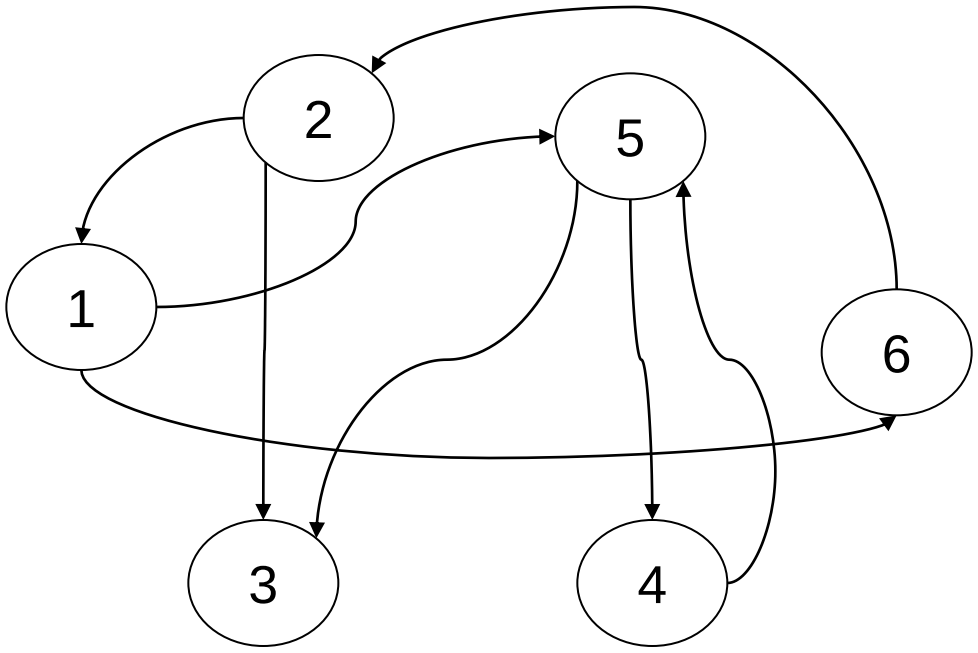


Árvores com componentes



Componentes fortemente conexos

- Exercício: encontre os componentes fortemente conectados do grafo abaixo



Componentes fortemente conexos

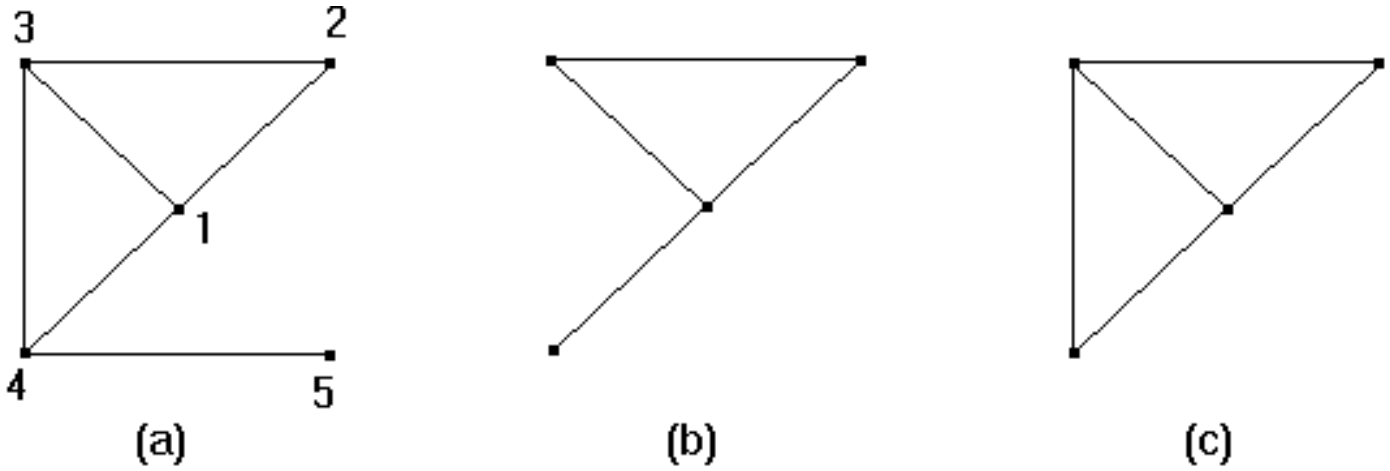
- Implementação da sub-rotina para transpor um grafo

Componentes fortemente conexos

- Como alterar o algoritmo da busca em profundidade para identificar componentes fortemente conectados?
- Qual a complexidade de tempo desse algoritmo?

Sub-grafo

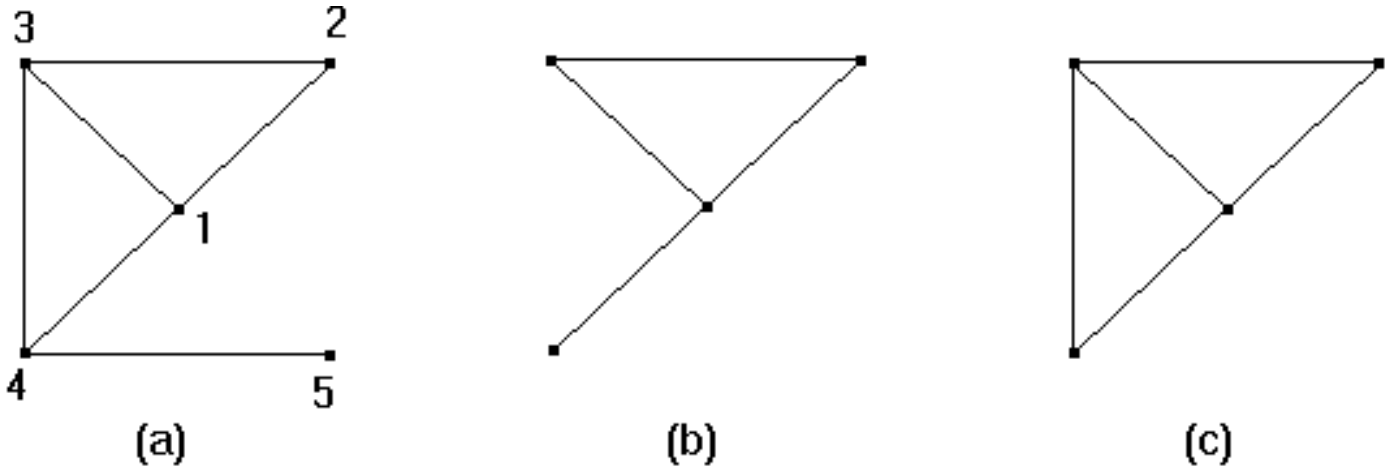
Um sub-grafo **G2(V2,E2)** de um grafo **G1(V1,E1)** é um grafo tal que **V2** está contido em **V1** e **E2** está contido em **E1**



b e c são sub-grafos de a

Sub-grafo induzido

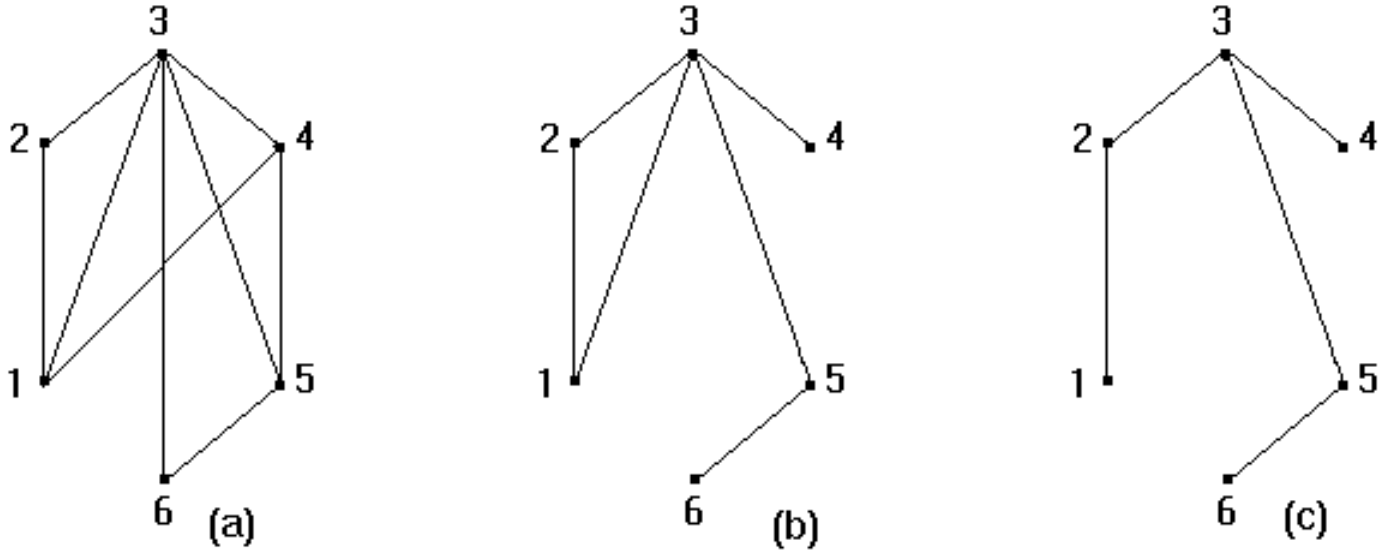
Se o sub-grafo **G2** de **G1** satisfaz: para quaisquer **v**, **w** pertencentes a **V2**, se **(v,w)** pertence a **E1**, então **(v,w)** também pertence a **E2**. Dessa forma, **G2** é dito sub-grafo induzido pelo conjunto de vértices **V2**



b e c são sub-grafos de a, mas apenas c é sub-grafo induzido

Sub-grafo gerador

Sub-grafo Gerador **o**u sub-grafo de espalhamento de um grafo **G1(V1,E1)** é um sub-grafo **G2(V2,E2)** de **G1** tal que **V1=V2**. Quando o sub-grafo gerador é uma árvore, ele recebe o nome de árvore geradora (ou de espalhamento).



b e c são sub-grafos geradores de a

c é árvore geradora de a e b

Sub-grafo gerador de custo mínimo

- Formalmente
  - Dado um grafo não orientado G(V,E)
    - onde *w*: E$\rightarrow\Re^{+}$ define os custos das arestas
    - queremos encontrar um sub-grafo gerador conexo T de G tal que, para todo sub-grafo gerador conexo T´ de G

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

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!

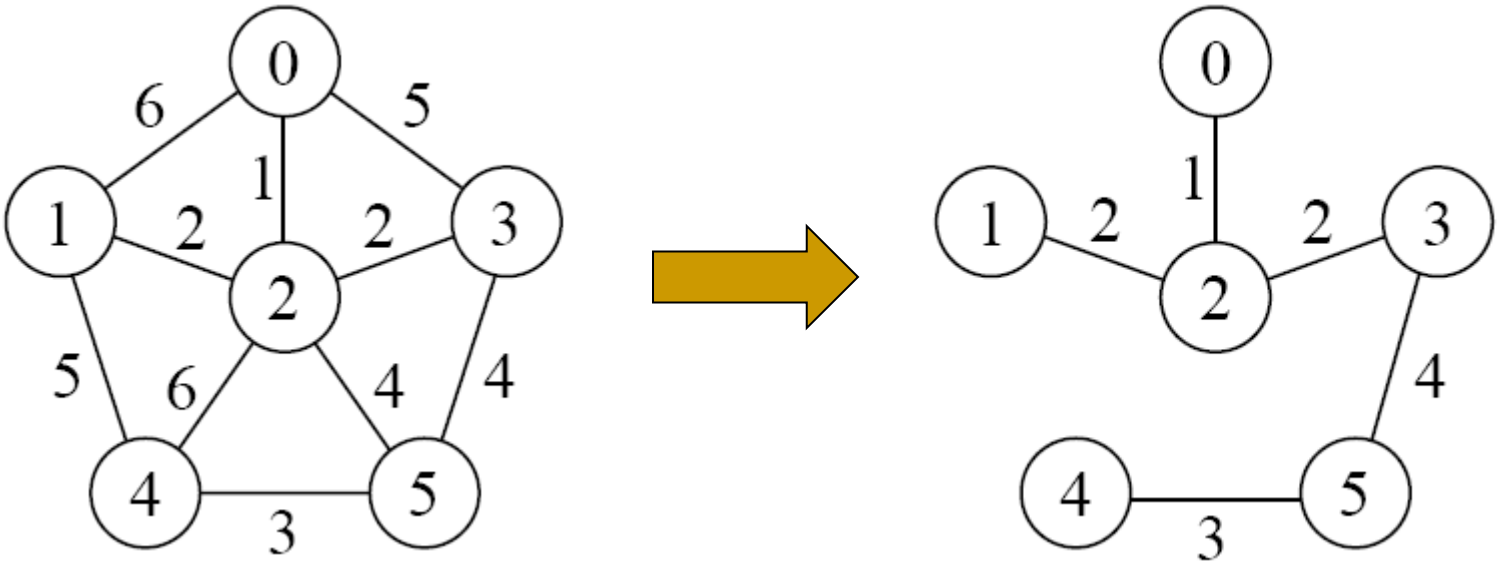
Árvore geradora mínima

- Árvore Geradora (*Spanning Tree*) de um grafo G é um sub-grafo de G que contém todos os seus vértices e, ainda, é uma árvore

- Arvore 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



Porque é um problema interessante?

- 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?

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

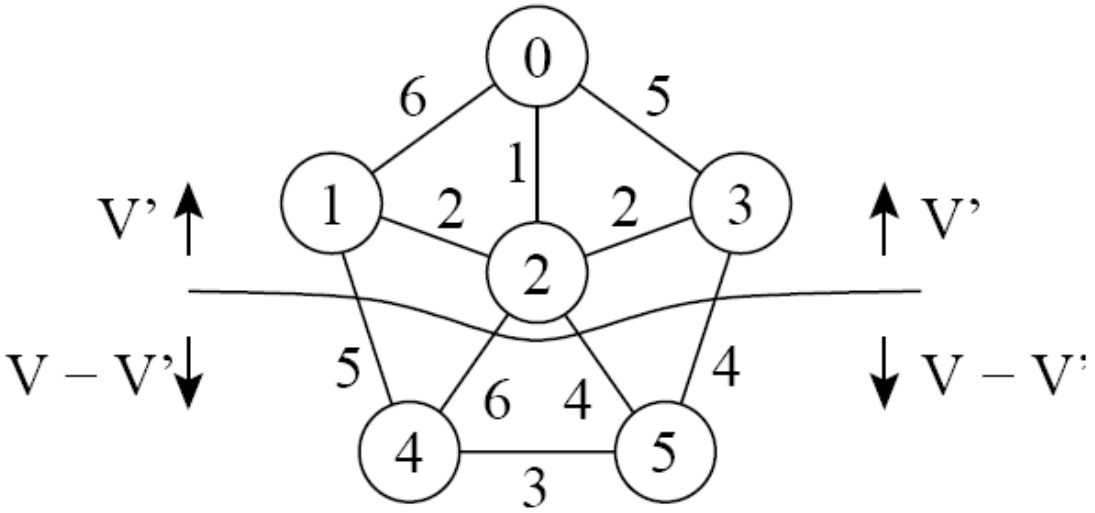
A – conjunto de arestas

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 sub-conjunto de vértices

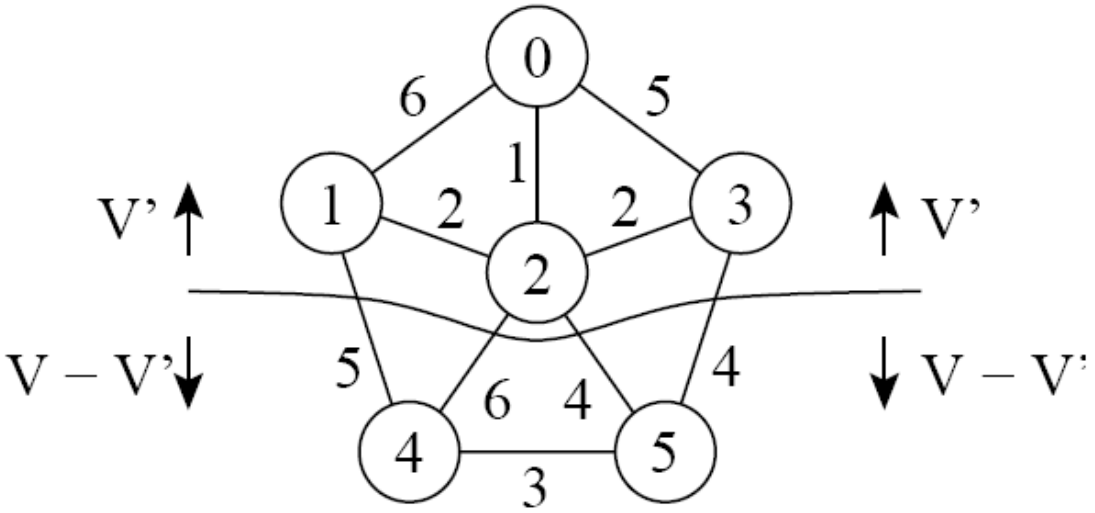
Árvore geradora mínima

- 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'



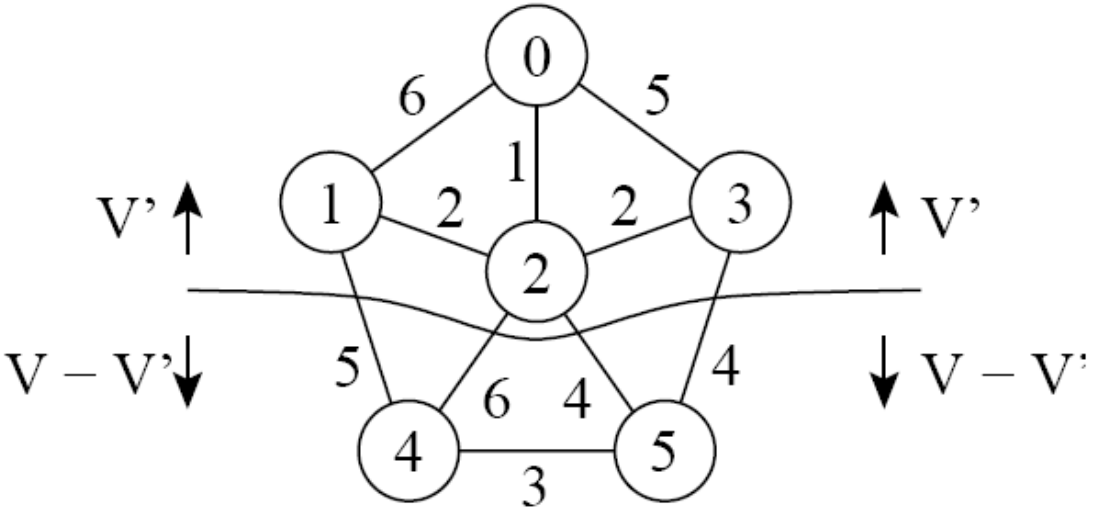
Árvore geradora mínima

- 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

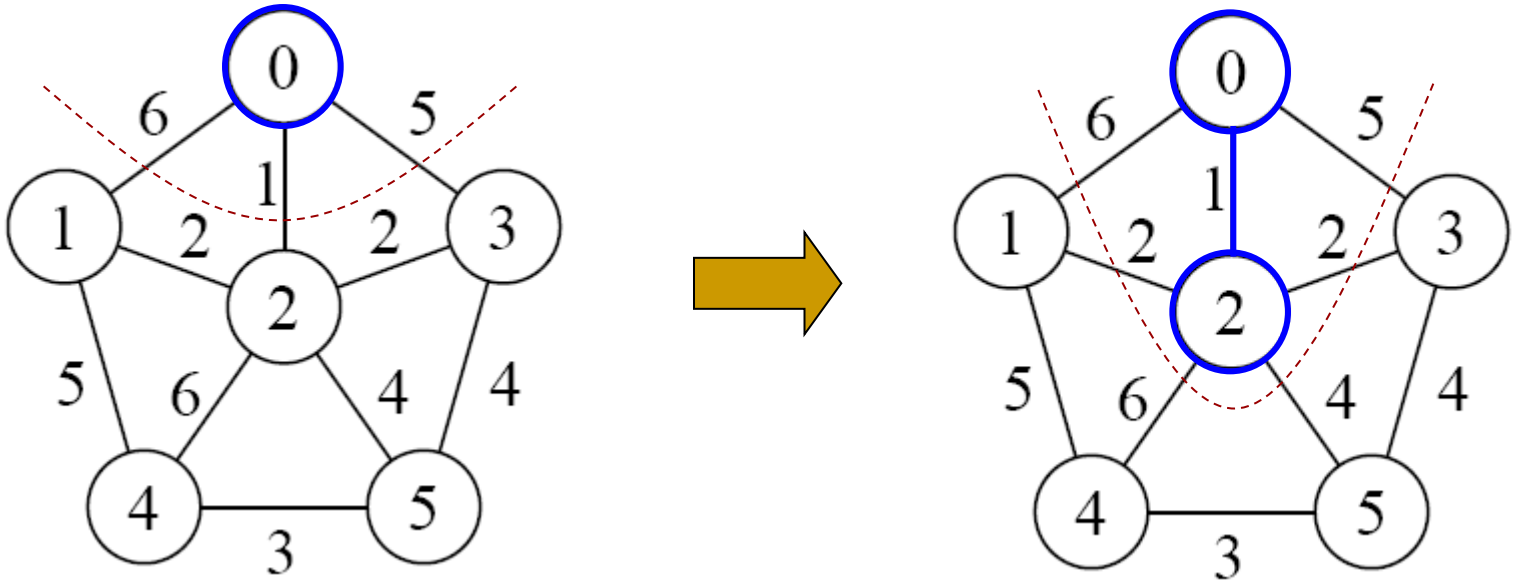
Algoritmo de Prim

```
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
```

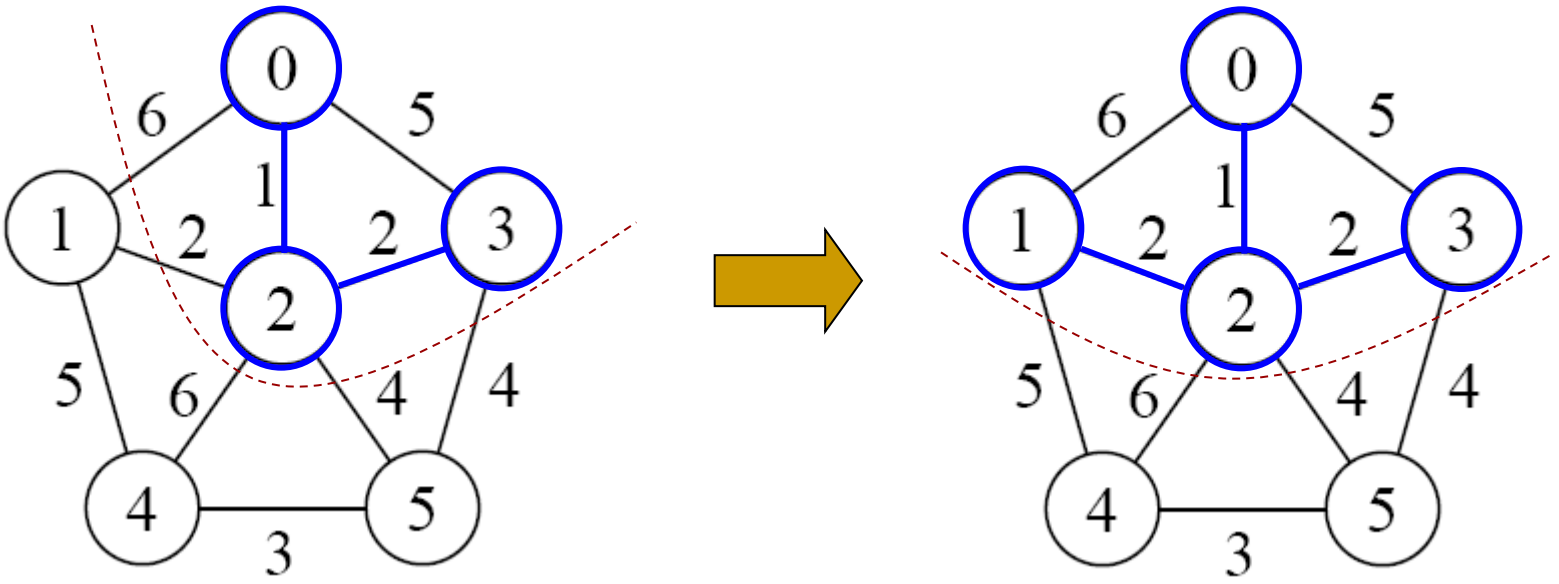
Ponto importante do algoritmo: seleção de uma aresta segura

Algoritmo de Prim

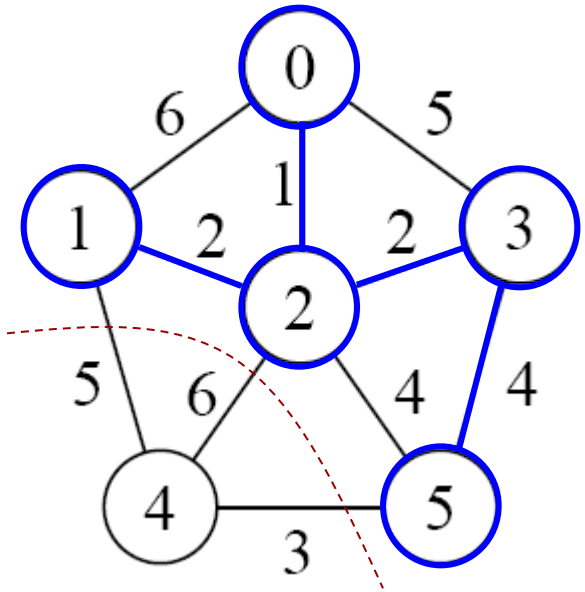
- Exemplo: iniciando o algoritmo pelo vértice 0

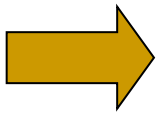


Algoritmo de Prim

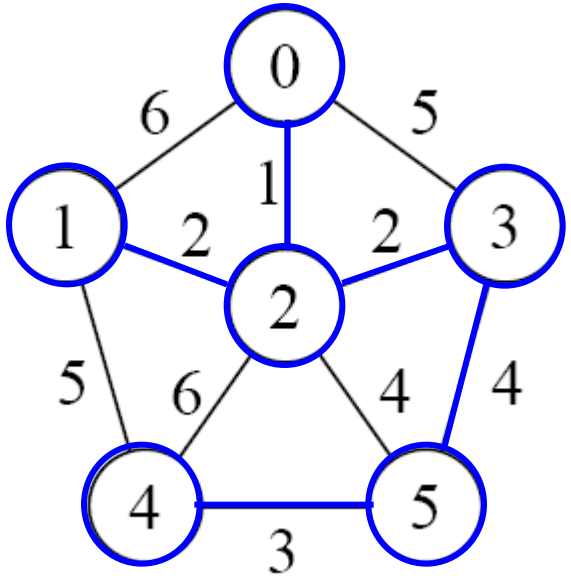


Algoritmo de Prim



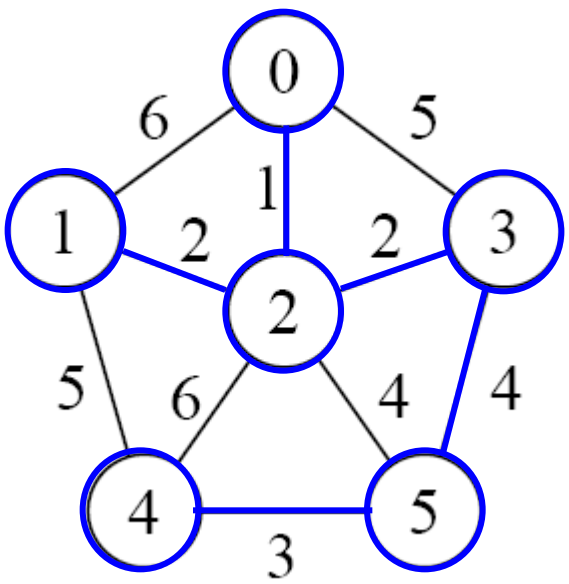


Árvore geradora mínima

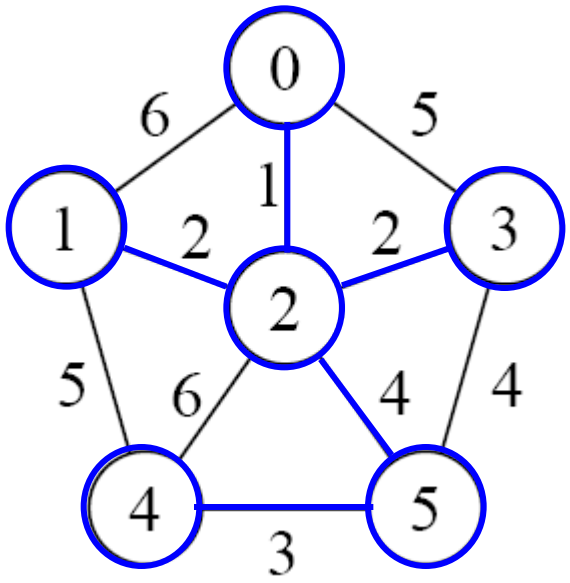


Algoritmo de Prim

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



ou



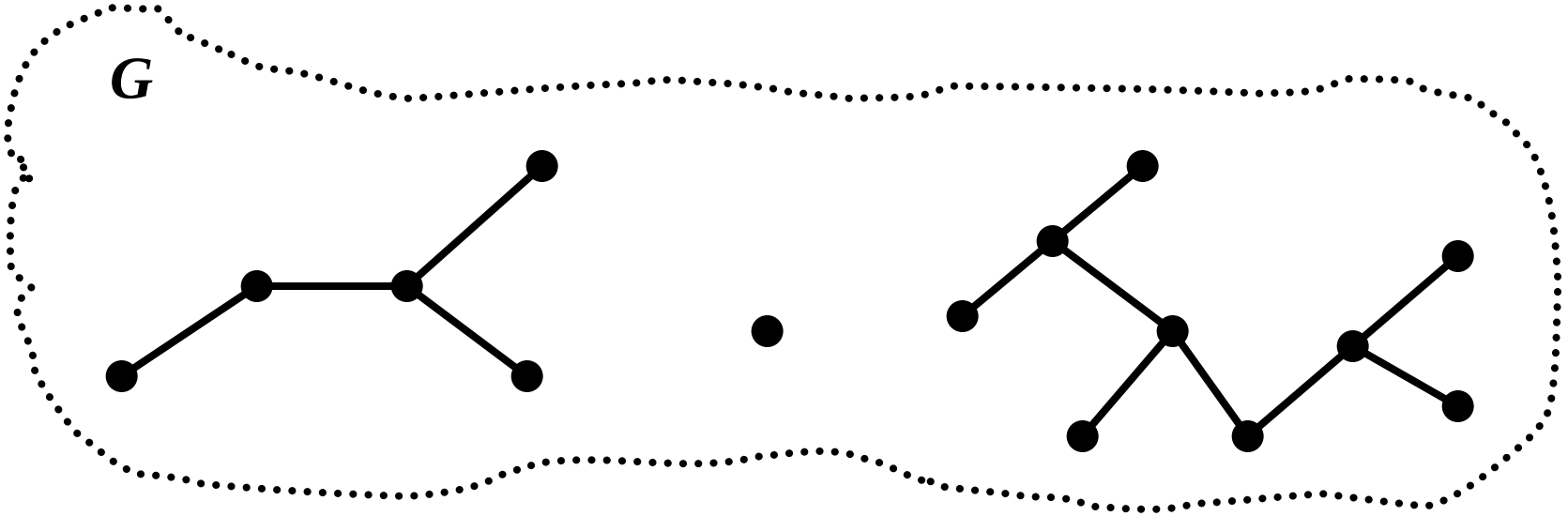
Algoritmo de Prim

- Maneira mais eficiente de determinar a aresta segura
  - Manter todas as arestas que ainda não estão na árvore em fila(s) de prioridade
    - 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

- Relembrando: uma floresta é um conjunto de árvores



Algoritmo de Kruskal

- Mais eficiente que Prim em grafos esparsos

- Não inicia em nenhum vértice em particular
  - Verifica se as arestas podem ou não pertencer à árvore, analisando-as em ordem crescente de custo
    - Diferencial de Prim: podem haver conjuntos disjuntos
  - As árvores que compõem a floresta são identificadas pelos conjuntos S_i, que contém os vértices que a compõem

  - Complexidade: O(|A| log(|V|))
    - Se o teste $S_p \cap S_q = \varnothing$ for bem implementado
      - Evitam-se ciclos

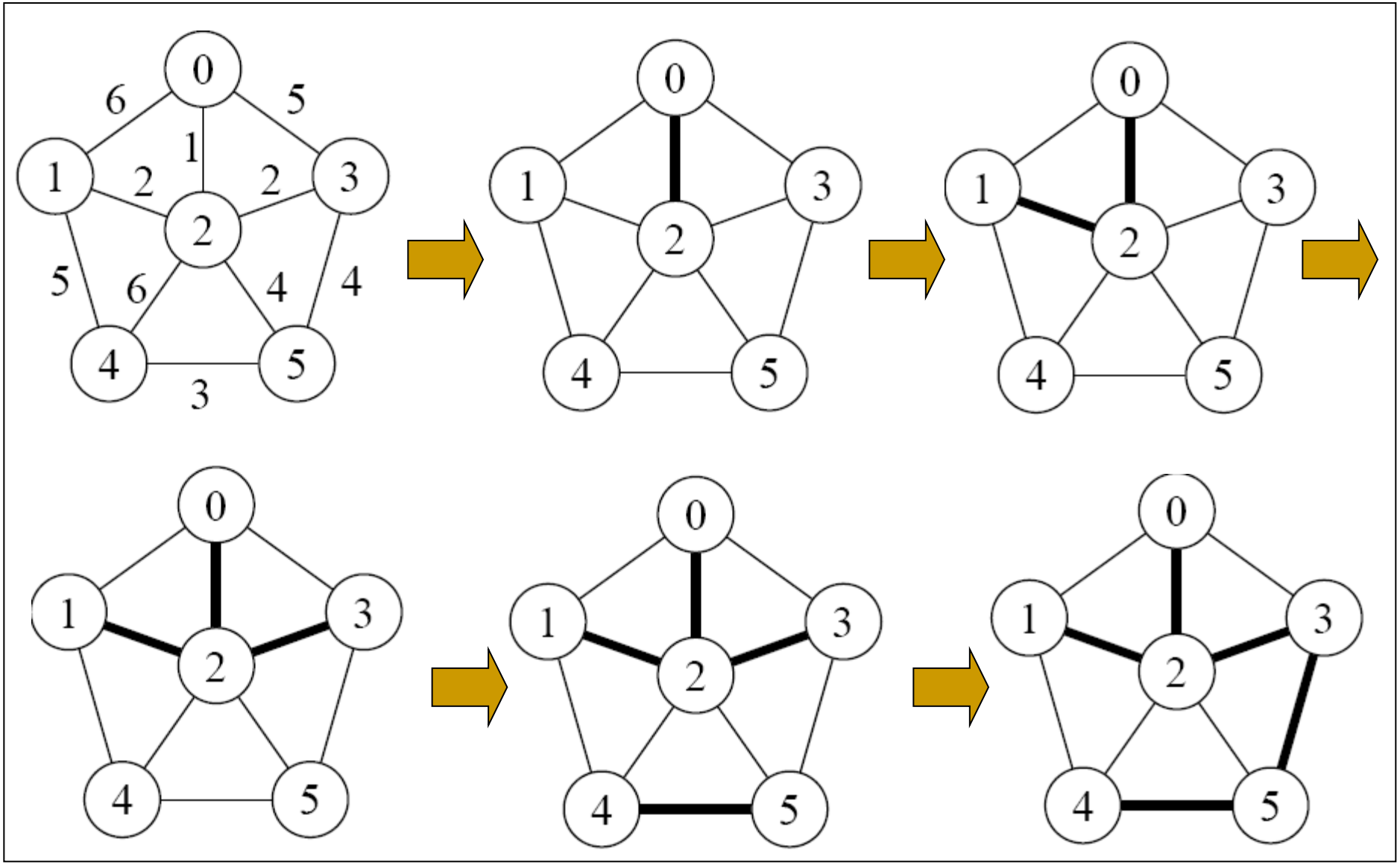
Algoritmo de Kruskal

- Basicamente, o algoritmo consiste na construção de uma árvore geradora a partir de uma floresta
  - Estado inicial: corresponde à floresta formada por |V| árvores triviais (i.e., um só vértice cada)

Algoritmo de Kruskal

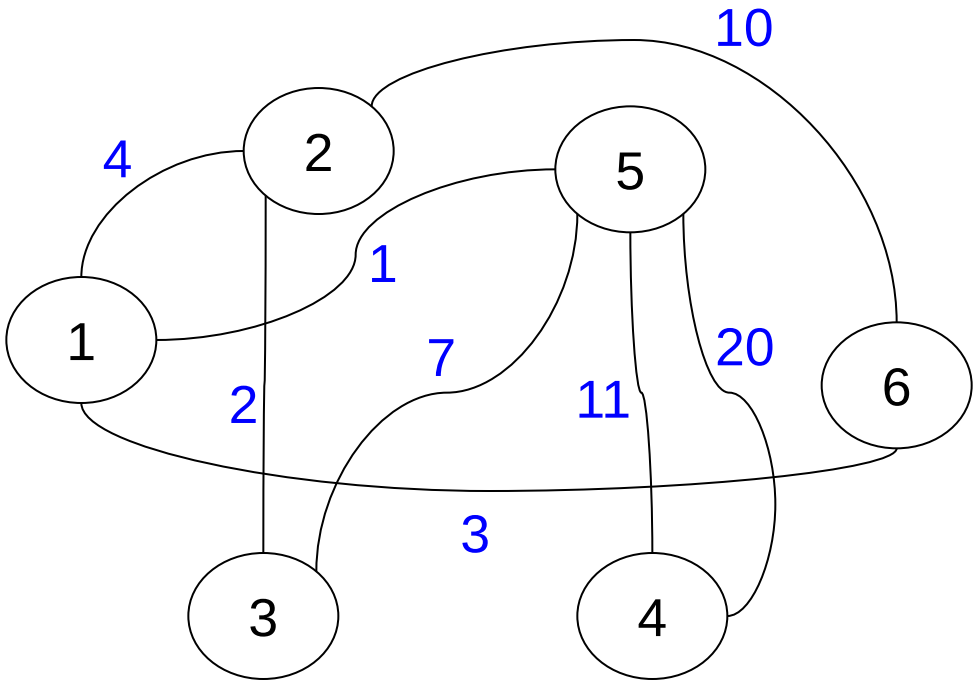
```
procedimento Kruskal(G)
  definir conjuntos S_j={v_j}, 1≤j≤|V|
  inserir as arestas de G em uma fila de prioridade
      F segundo o peso (ordem crescente)
  enquanto houver arestas na fila faça
      (v,w)=desenfileira(F)
se v∈S_p e w∈S_q, S_p∩S_q=∅ então
        S_p=S_p∪S_q
        eliminar S_q
```

Algoritmo de Kruskal: exemplo



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

- Implementação pode usar o campo Antecessor
  - Por exemplo, no algoritmo de Prim
    - No início do procedimento, a árvore geradora mínima é o conjunto {v}: v=Raiz
    - Ao final, a árvore é o conjunto {v, Antecessor[v]}: $v \in V-\{Raiz\}$

Árvore geradora mínima

- Exercício em duplas para entregar
  - Implementação em C de sub-rotina para encontrar uma árvore geradora mínima para um grafo (algoritmo de Prim ou de Kruskal)
    - Decida como buscar as arestas seguras
      - Lembre-se: essa busca deve ser eficiente!