Árvores Geradoras de Peso Mínimo

Prof. Andrei Braga

UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

Conteúdo

- Outra estratégia geral para encontrar árvores geradoras de peso mínimo
- Algoritmo de Kruskal
- Exercícios
- Referências

Como encontrar uma árvore geradora de peso mínimo

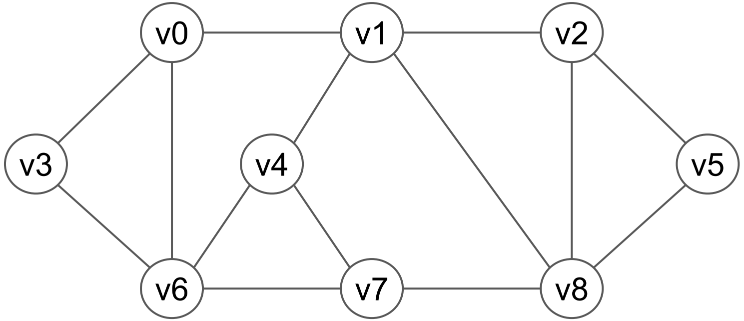
- Vimos o Algoritmo de Prim para encontrar uma árvore geradora de peso mínimo
- Este algoritmo usa a estratégia de adicionar arestas (e vértices) a uma árvore até que seja formada uma árvore geradora de peso mínimo
- Agora, veremos outra estratégia para obter uma árvore geradora de peso mínimo

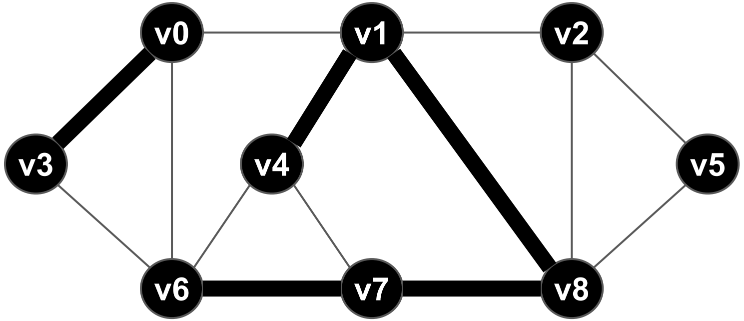
Floresta geradora

- Um subgrafo de um grafo *G* que é gerador e acíclico é denominado uma **floresta geradora** de *G*

- Exemplo:

G:





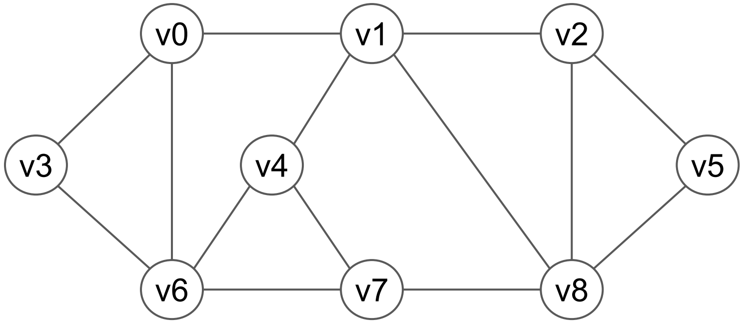
Floresta geradora de *G*

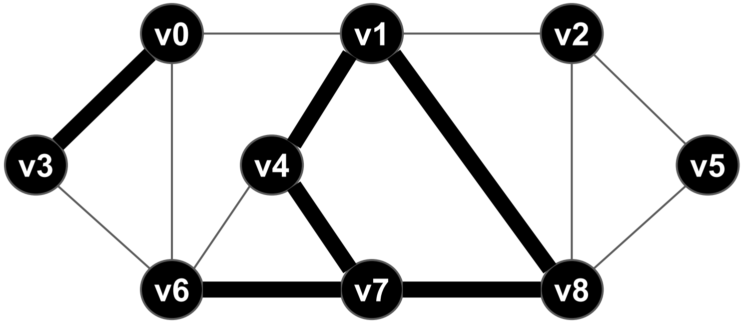
Floresta geradora

- Um subgrafo de um grafo *G* que é gerador e acíclico é denominado uma **floresta geradora** de *G*

- Exemplo:

G:





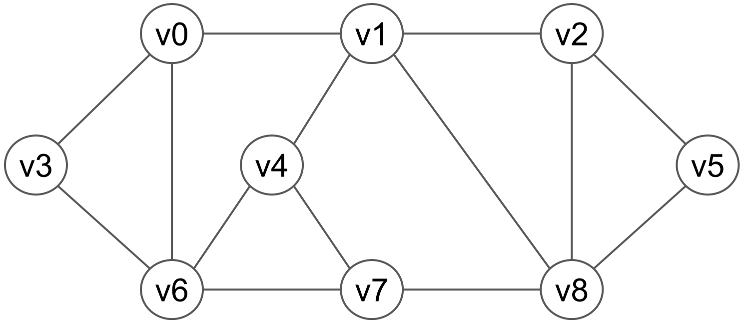
Floresta geradora de *G*

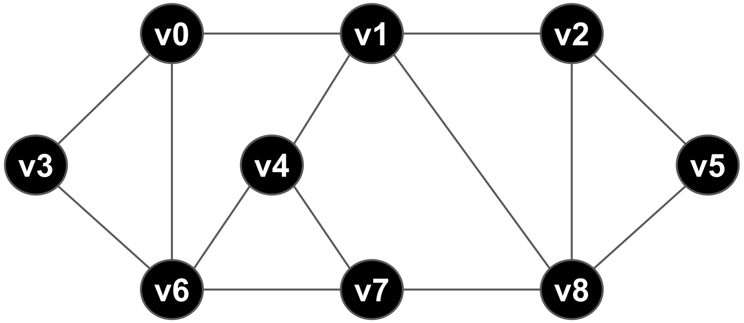
Floresta geradora

- Um subgrafo de um grafo *G* que é gerador e acíclico é denominado uma **floresta geradora** de *G*

- Exemplo:

G:





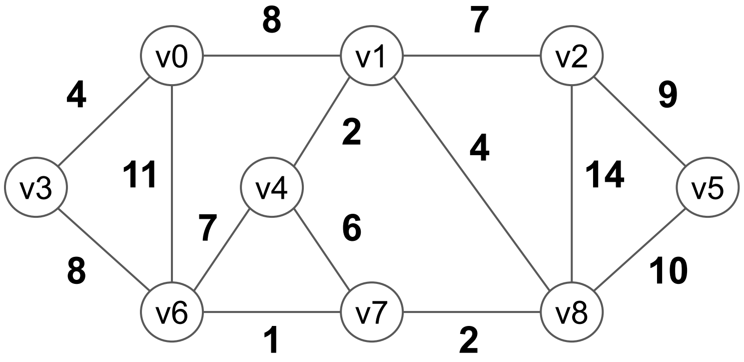
Floresta geradora de *G*

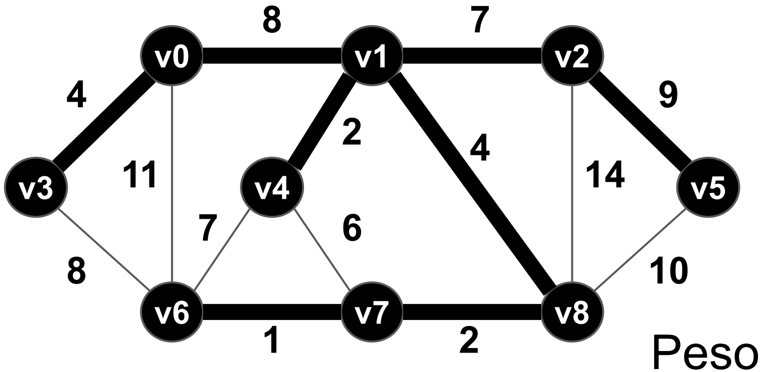
Aresta segura

- Seja G um grafo conexo com pesos nas arestas e F uma floresta geradora de G com a seguinte propriedade: F está contida em uma árvore geradora de peso mínimo de G. Uma aresta uv de G é **segura para *F*** se F continua tendo a mesma propriedade depois de uv ser adicionada a F

Aresta segura

- Exemplo:





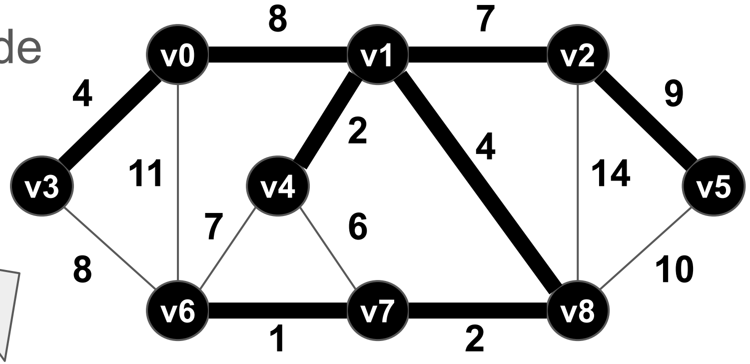
Peso: 37

Árvore geradora de peso mínimo

Aresta segura

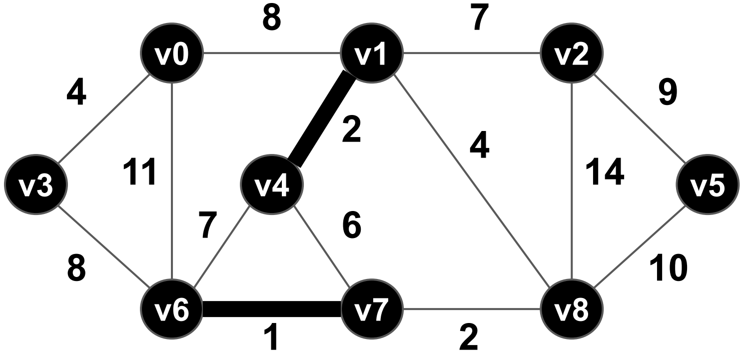
- Exemplo:

Árvore geradora de peso mínimo



contida

F:

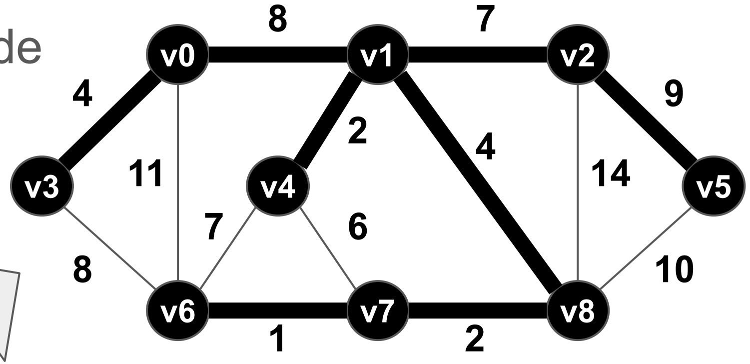


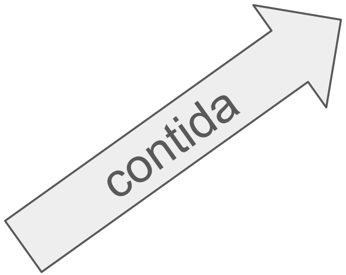
Floresta geradora contida em uma árvore geradora de peso mínimo

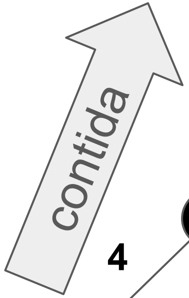
Aresta segura

- Exemplo:

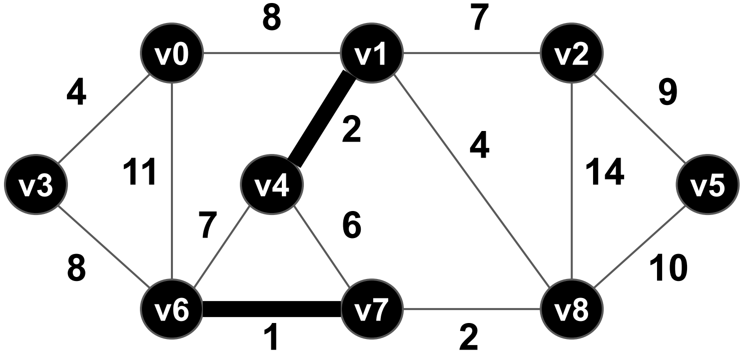
Árvore geradora de peso mínimo





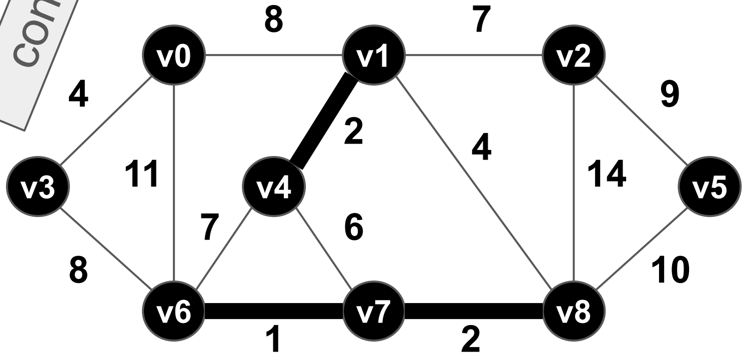


F:



Floresta geradora contida em uma árvore geradora de peso mínimo

F:

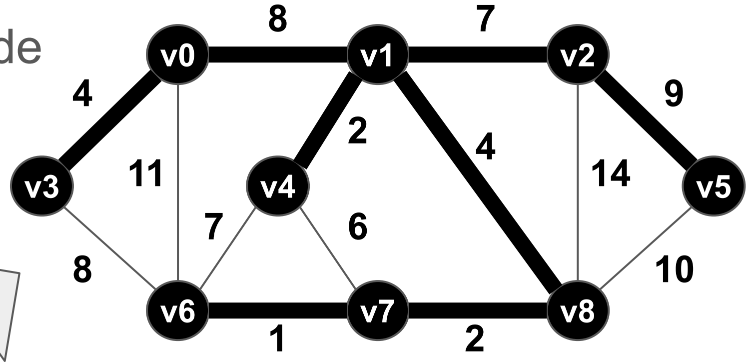


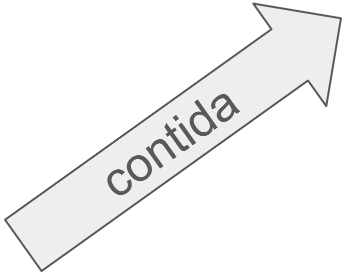
v7v8 é uma aresta segura para *F*

Aresta segura

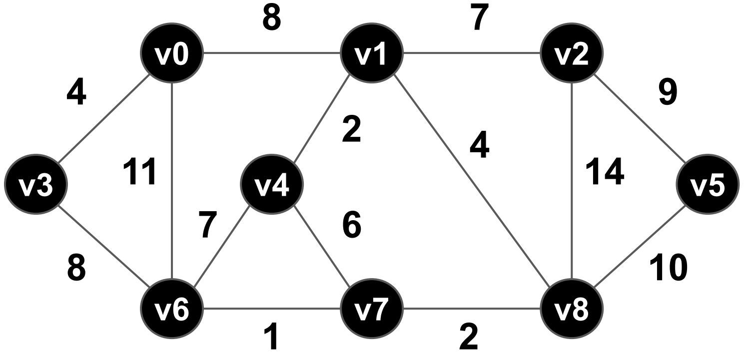
- Exemplo:

Árvore geradora de peso mínimo





F:

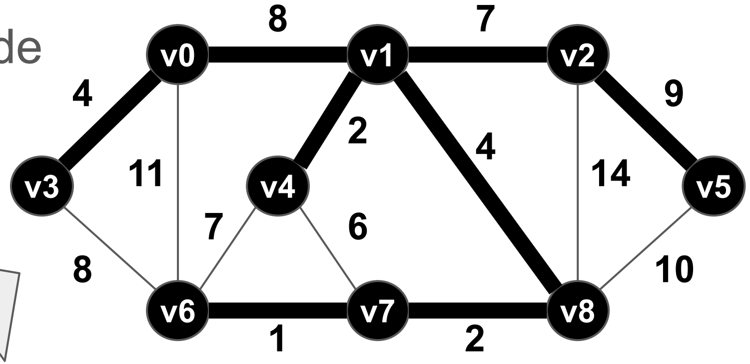


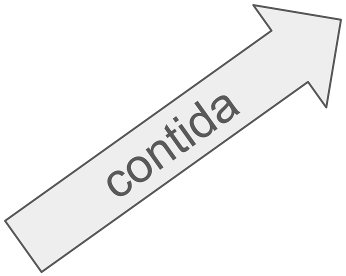
Floresta geradora contida em uma árvore geradora de peso mínimo

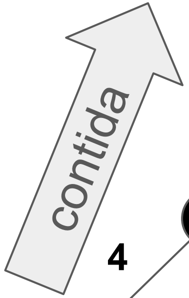
Aresta segura

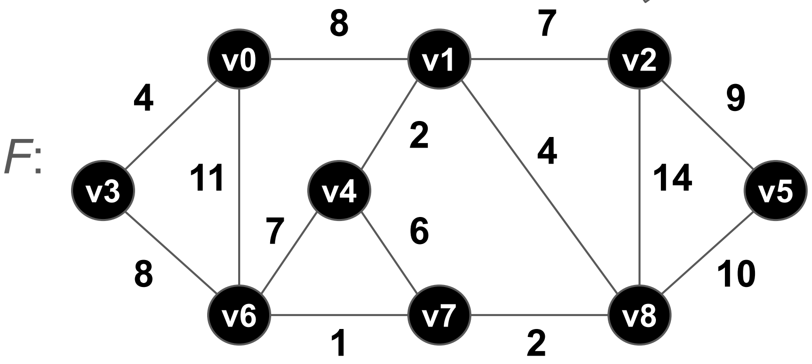
- Exemplo:

Árvore geradora de peso mínimo

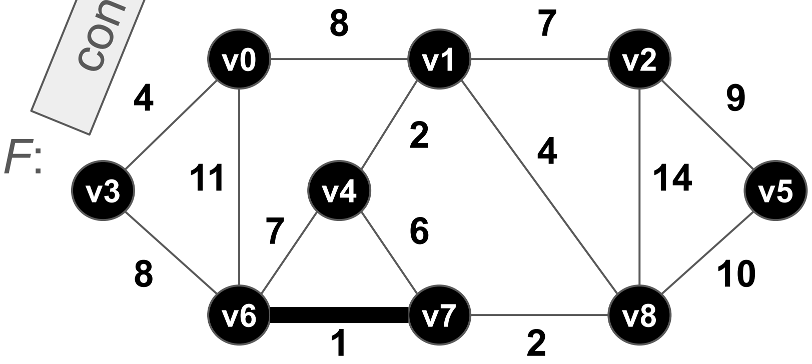








Floresta geradora contida em uma árvore geradora de peso mínimo

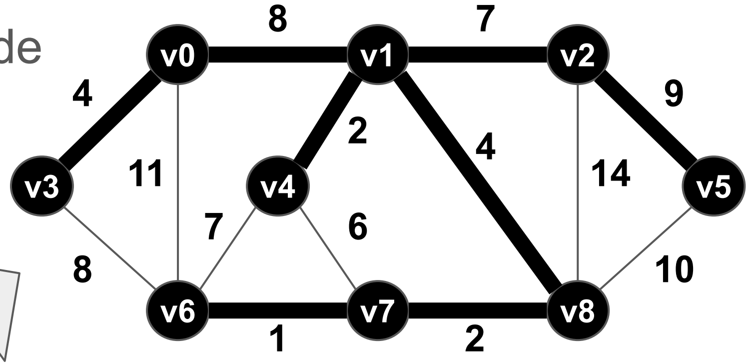


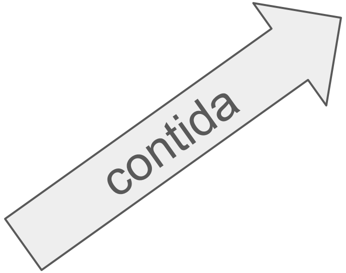
v6v7 é uma aresta segura para *F*

Aresta segura

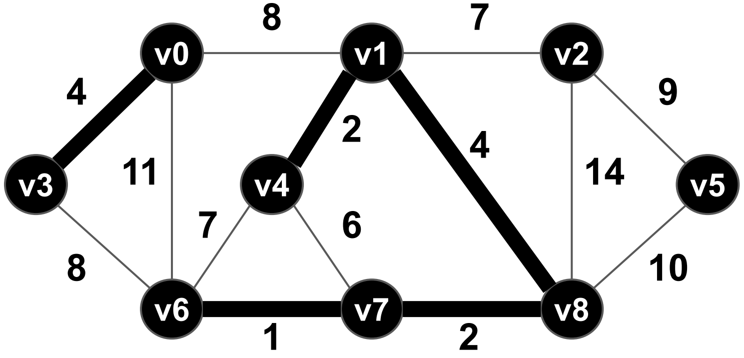
- Exemplo:

Árvore geradora de peso mínimo





F:

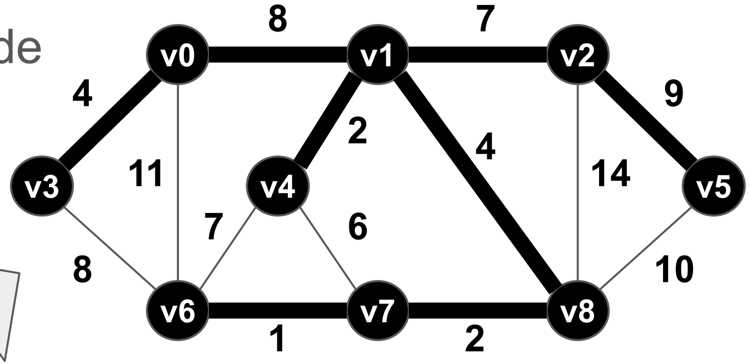


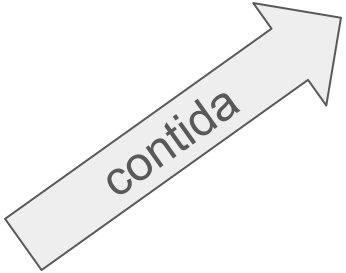
Floresta geradora contida em uma árvore geradora de peso mínimo

Aresta segura

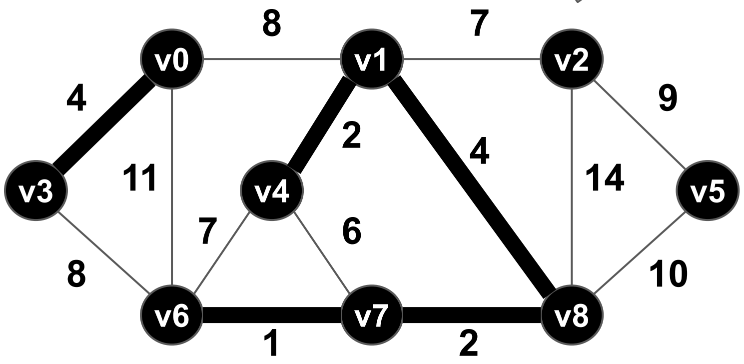
- Exemplo:

Árvore geradora de peso mínimo



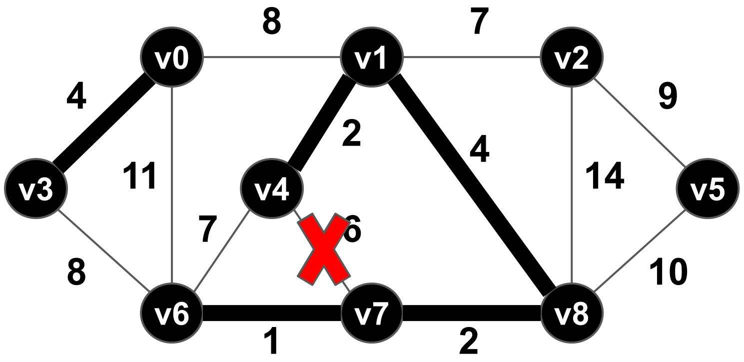


F:



Floresta geradora contida em uma árvore geradora de peso mínimo

F:



v4v7 **não** é uma aresta segura para *F*

Como encontrar uma árvore geradora de peso mínimo

EncontraArvGerPesMin(G conexo)

1. $F = (V(G), \varnothing)$
2. Enquanto F não é uma árvore geradora de G:
3. Encontre uma aresta uv de G que é segura para F
4. Adicione uv a F
5. Retorne F

Inicialmente, F é a floresta formada por todos os vértices de G e por nenhuma aresta. Em outras palavras, F é formada por $V(G)$ árvores que contêm apenas um vértice

Ao fim de cada iteração do laço das linhas 2-4, temos uma floresta geradora F

- que tem 1 aresta a mais e
- que está contida em uma árvore geradora de peso mínimo de G

Ao fim do laço das linhas 2-4, temos uma floresta geradora F

- que tem $V(G)$ - 1 arestas e
- que está contida em uma árvore geradora de peso mínimo de G

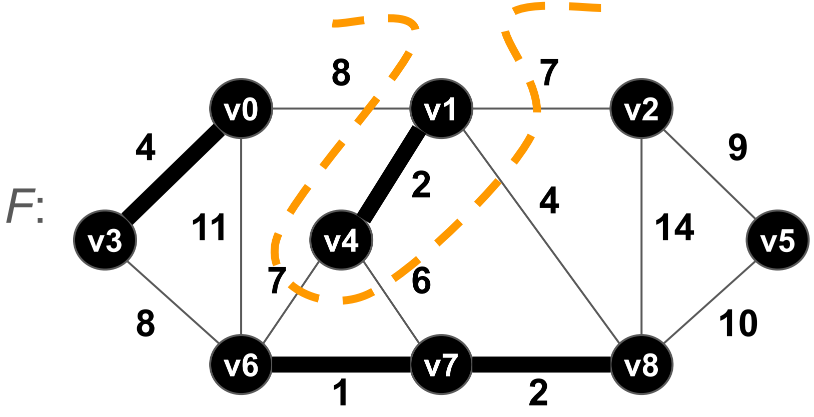
Então, temos uma árvore geradora de peso mínimo de G!

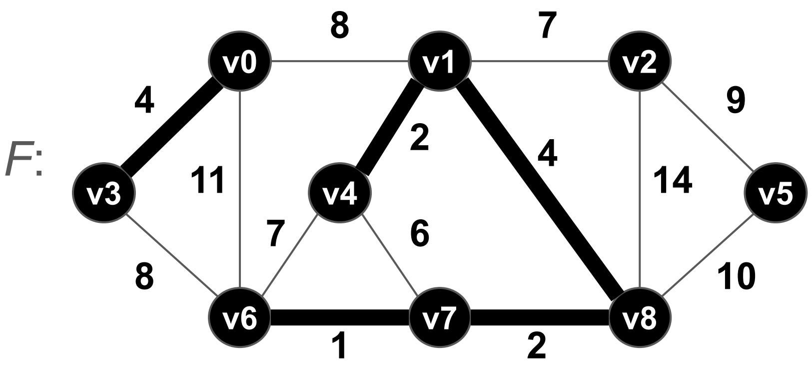
Algoritmo de Kruskal

- Assim como o Algoritmo de Prim, o Algoritmo de Kruskal se baseia em um resultado que indica como seguir escolhendo arestas seguras para formar uma árvore geradora de peso mínimo
- Este resultado é semelhante ao resultado visto anteriormente

Aresta segura em um corte

- **Teorema:** Seja *G* um grafo conexo com pesos nas arestas e *F* uma floresta geradora de *G* com a seguinte propriedade: *F* está contida em uma árvore geradora de peso mínimo de *G*. Seja *T* uma componente conexa de *F* (*T* é uma árvore de *F*). Se *uv* é uma aresta de peso mínimo do corte (*V*(*T*), *V*(*G*) \ *V*(*T*)), então *uv* é uma aresta segura para *F*





Aresta segura em um corte

- **Prova:**
  - Seja *T'* uma árvore geradora de peso mínimo de *G* que contém *F*
  - Vamos construir uma árvore geradora *T''* de peso mínimo de *G* que contém *F* adicionada de *uv*
  - Com isso, o teorema estará provado

  - Se *T'* contém *uv*, então fazemos simplesmente *T''* = *T'*
  - Suponha, então, que *T'* não contém *uv*

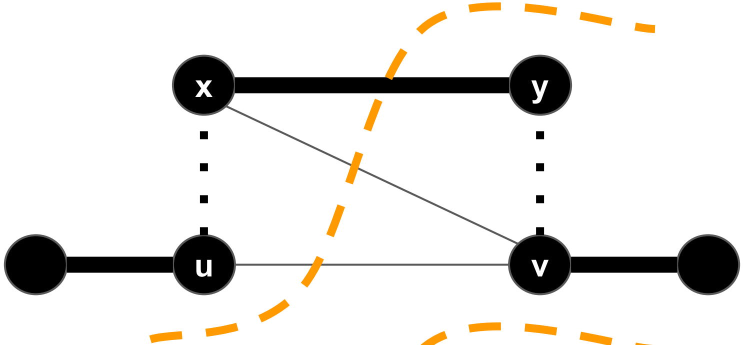
Aresta segura em um corte

- **Prova:**
  - Como *T’* é uma árvore geradora de *G*, existe um único caminho *P* entre *u* e *v* em *T’*
  - Defina *T”* como *T’* adicionada de *uv*. *T”* contém um único ciclo, que é formado pelo caminho *P* adicionado de *uv*
  - Já que *u* e *v* estão em subconjuntos diferentes do corte (*V*(*T*), *V*(*G*) \ *V*(*T*)), existe, no caminho *P*, uma aresta *xy* do corte
  - Note que a aresta *xy* não está contida em *F*, pois nenhuma aresta do corte (*V*(*T*), *V*(*G*) \ *V*(*T*)) está contida em *F*
  - Remova *xy* de *T”*. Agora, *T”* é uma árvore geradora de *G* que contém *F* adicionada de *uv*

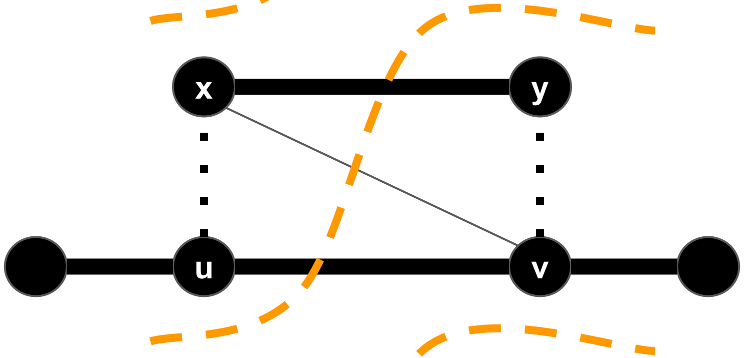
Aresta segura em um corte

- **Prova:**

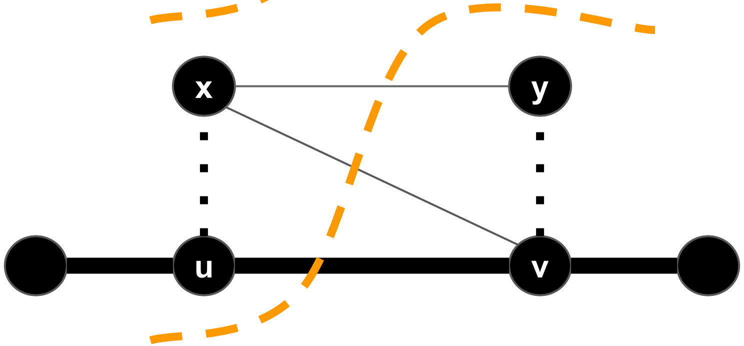
T’:



T” = *T’* adicionada de *uv*:



T” = *T”* com a remoção de *xy*:



Aresta segura em um corte

- **Prova:**
  - Sendo *uv* uma aresta de peso mínimo do corte ($V(T)$, $V(G) \setminus V(T)$), o peso de *uv* é menor ou igual ao peso de *xy*
  - O peso de *T''* é igual a
    o peso de *T'* + o peso de *uv* - o peso de *xy*
  - Então, o peso de *T''* é menor ou igual ao peso de *T'*
  - Como *T'* é uma árvore geradora de peso mínimo de *G*, *T''* também é uma árvore geradora de peso mínimo de *G*
  - Portanto, *T''* é uma árvore geradora de peso mínimo de *G* que contém *F* adicionada de *uv* □

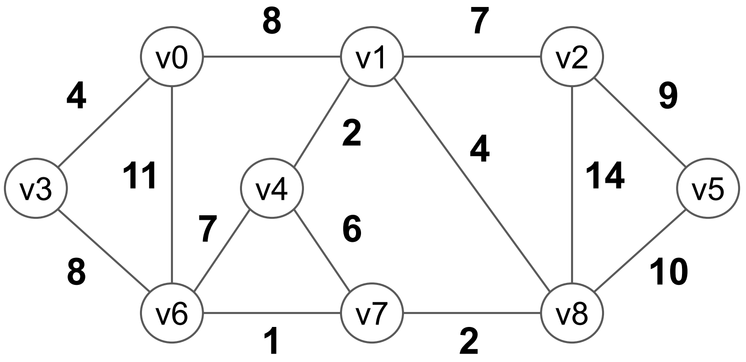
Algoritmo de Kruskal

Inicialmente, F é a floresta formada por todos os vértices de G e por nenhuma aresta. Em outras palavras, F é formada por $V(G)$ árvores que contêm apenas um vértice

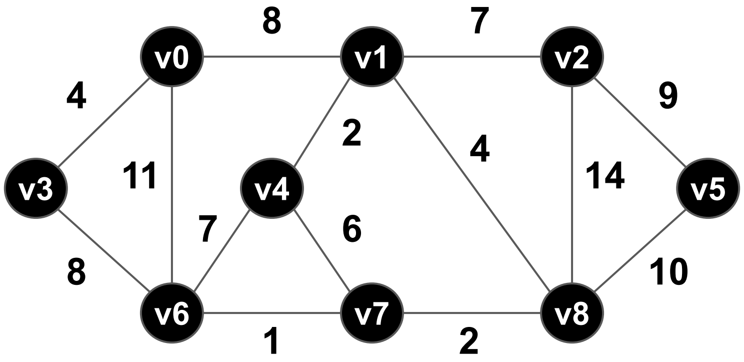
Kruskal(G conexo)

1. $F = (V(G), \varnothing)$
2. Enquanto F não é uma árvore geradora de G:
3. Encontre uma aresta uv de peso mínimo do corte $(V(T), V(G) \setminus V(T))$, sendo T uma componente conexa de F (T é uma árvore de F)
4. Adicione uv a F
5. Retorne F

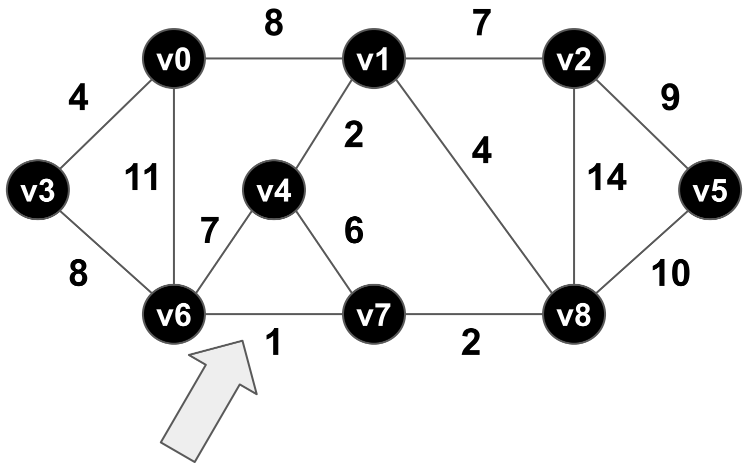
Algoritmo de Kruskal



Algoritmo de Kruskal



Algoritmo de Kruskal

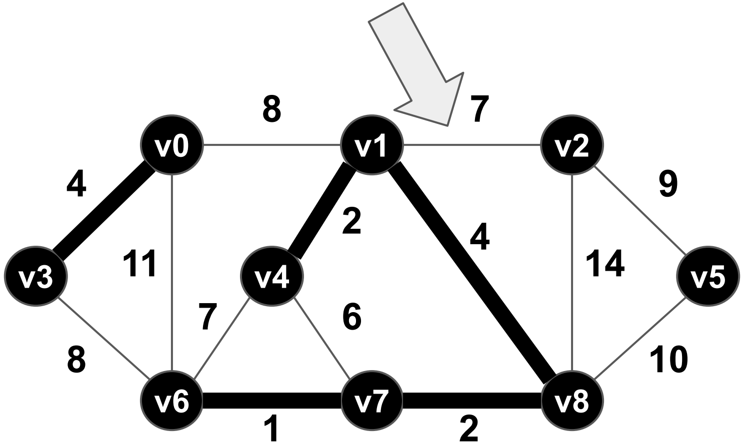


v6v7 é uma aresta que conecta duas árvores T_a e T_b diferentes da floresta geradora

Então, v6v7 é uma aresta do corte $(V(T_a), V(G) \setminus V(T_a))$ – podemos dizer a mesma coisa para T_b

v6v7 é uma aresta de peso mínimo do corte $(V(T_a), V(G) \setminus V(T_a))$?

Algoritmo de Kruskal



v1v2 é uma aresta que conecta duas árvores T_a e T_b diferentes da floresta geradora

Então, v1v2 é uma aresta do corte $(V(T_a), V(G) \setminus V(T_a))$ – podemos dizer a mesma coisa para T_b

v1v2 é uma aresta de peso mínimo do corte $(V(T_a), V(G) \setminus V(T_a))$?

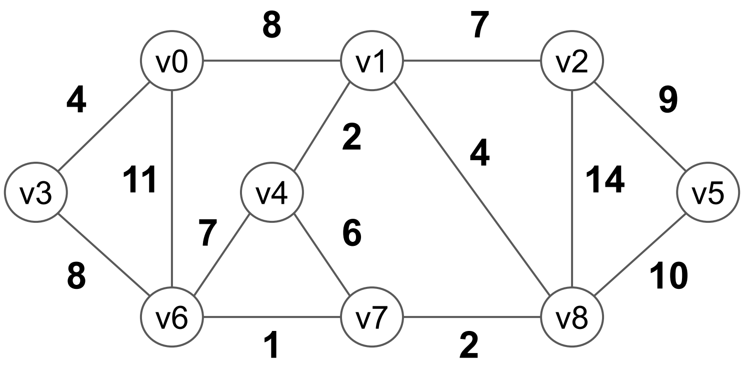
Algoritmo de Kruskal

Inicialmente, F é a floresta formada por todos os vértices de G e por nenhuma aresta. Em outras palavras, F é formada por $V(G)$ árvores que contêm apenas um vértice

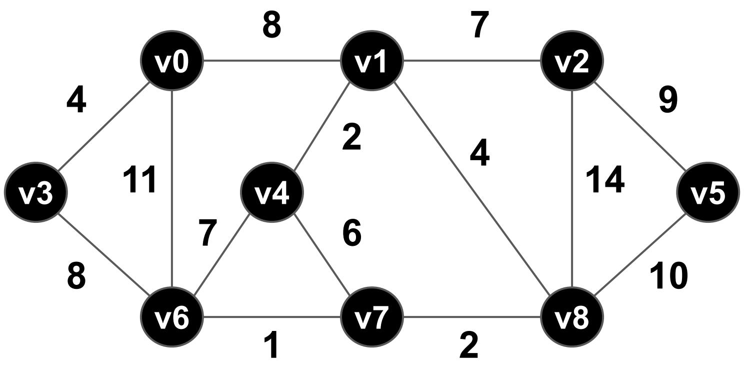
Kruskal(G conexo)

1. $F = (V(G), \varnothing)$
2. Ordene as arestas de G em ordem crescente de peso
3. Para cada aresta uv de G considerando as arestas de G em ordem crescente de peso:
4. Se uv conecta duas componentes conexas de F (duas árvores de F) diferentes:
5. Adicione uv a F
6. Retorne F

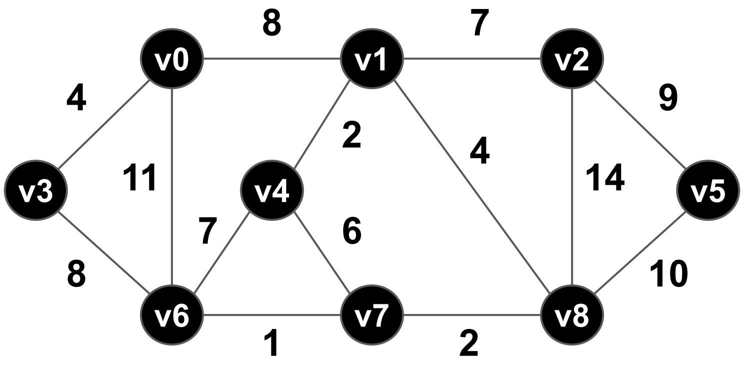
Algoritmo de Kruskal



Algoritmo de Kruskal



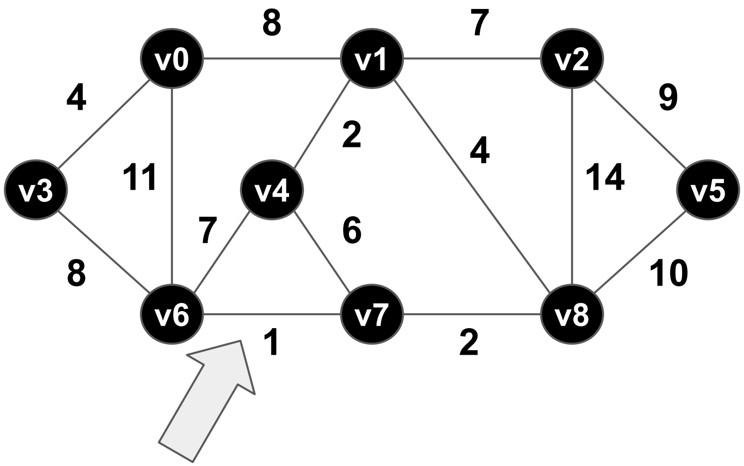
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

v6 v7
v1 v4
v7 v8
v0 v3
v1 v8
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

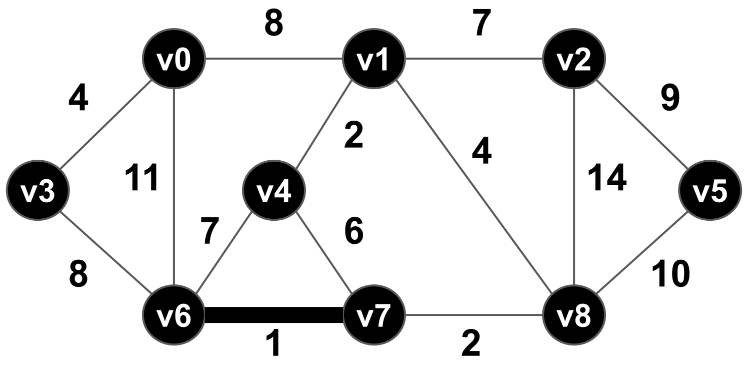
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

v6 v7
v1 v4
v7 v8
v0 v3
v1 v8
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

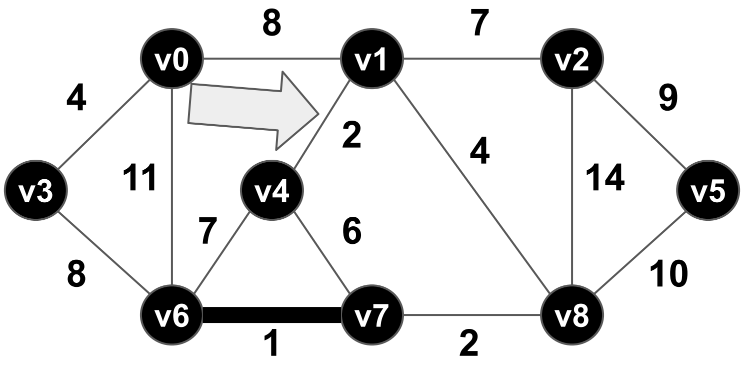
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
v1 v4
v7 v8
v0 v3
v1 v8
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

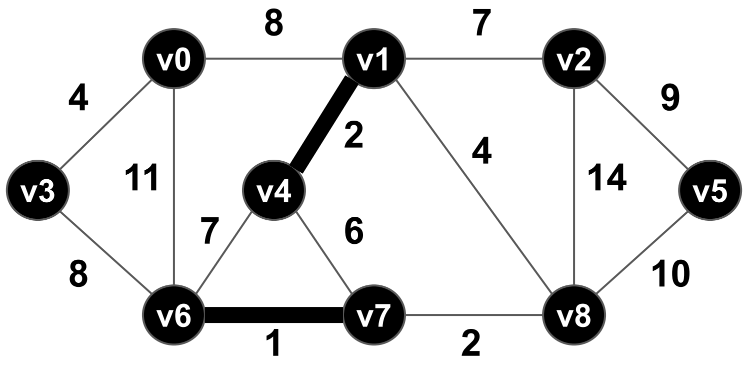
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
v1 v4
v7 v8
v0 v3
v1 v8
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

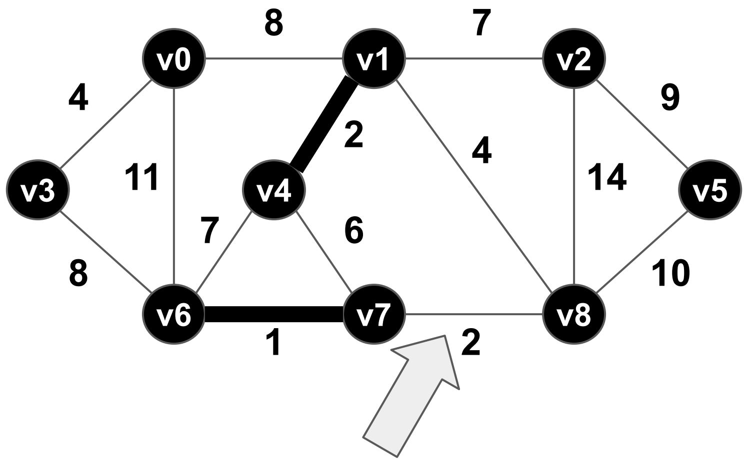
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
v7 v8
v0 v3
v1 v8
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

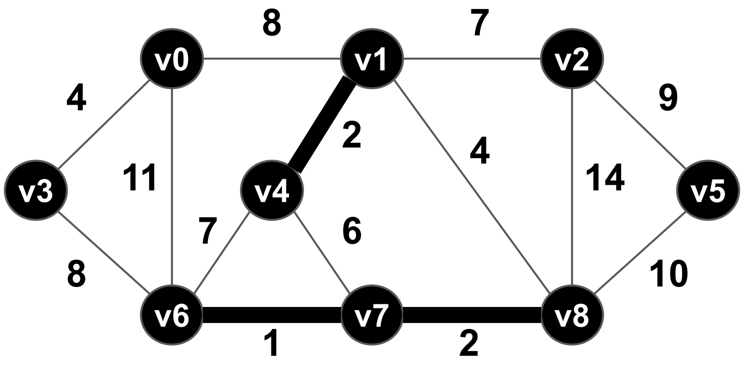
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
v7 v8
v0 v3
v1 v8
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

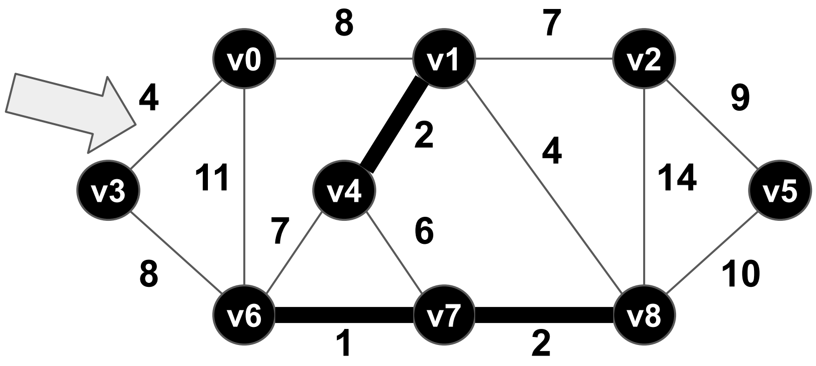
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
v0 v3
v1 v8
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

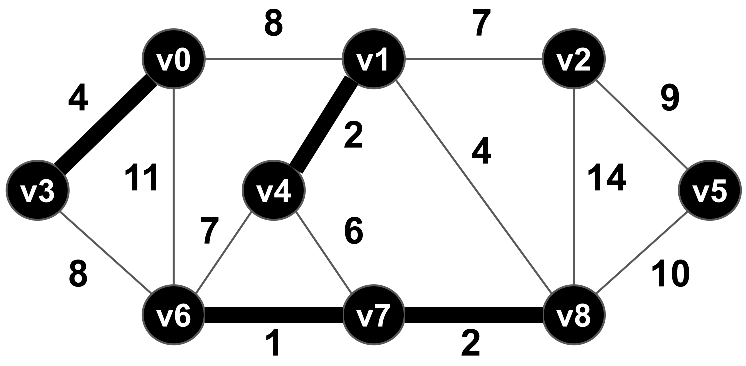
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
v0 v3
v1 v8
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

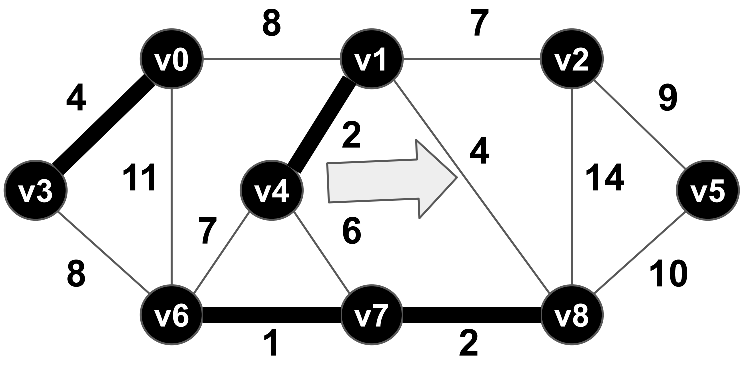
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
v1 v8
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

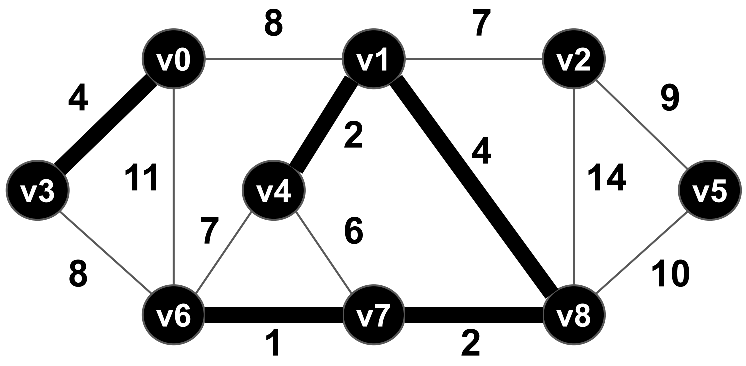
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
v1 v8
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

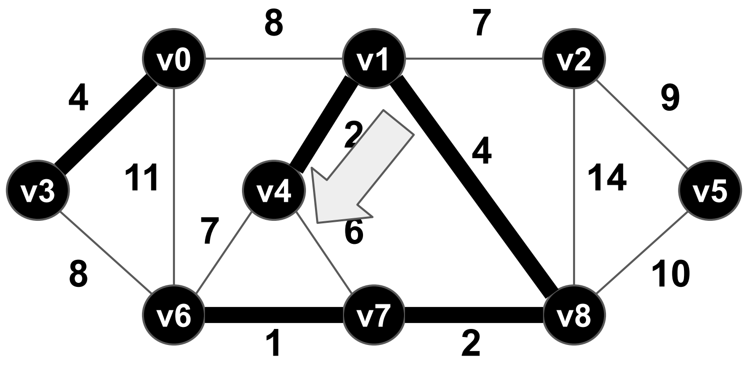
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

Algoritmo de Kruskal

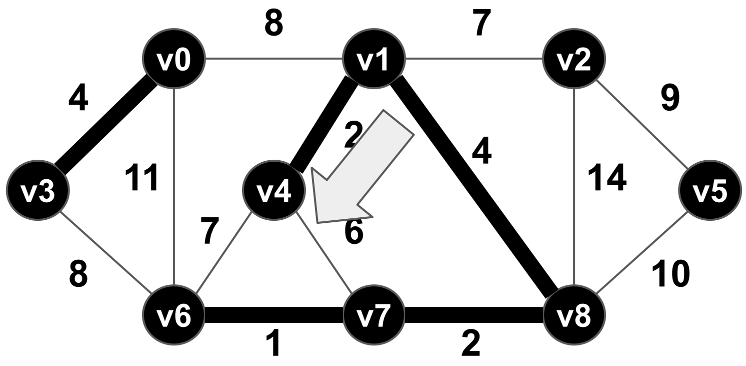


Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

Algoritmo de Kruskal

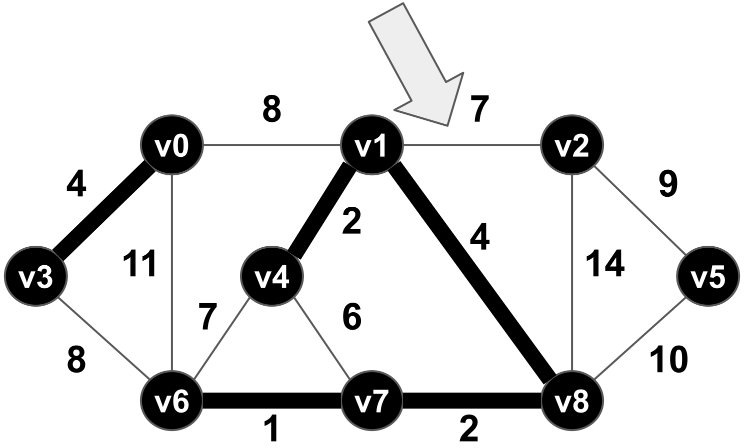
Arestas em ordem crescente de peso:



~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
v4 v7
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

v4 v7 **não conecta duas árvores diferentes**.
Portanto, não é adicionada à floresta

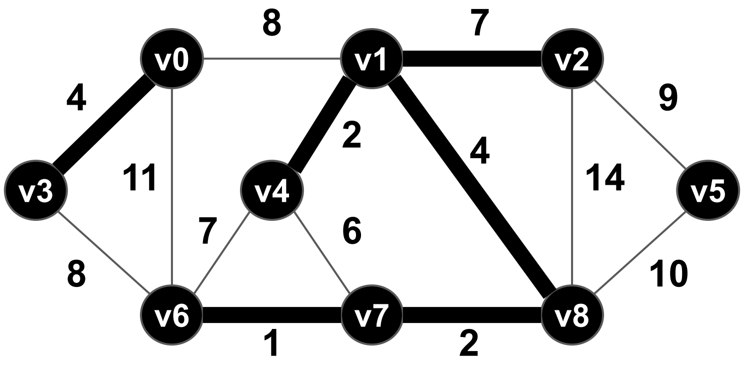
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
v1 v2
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

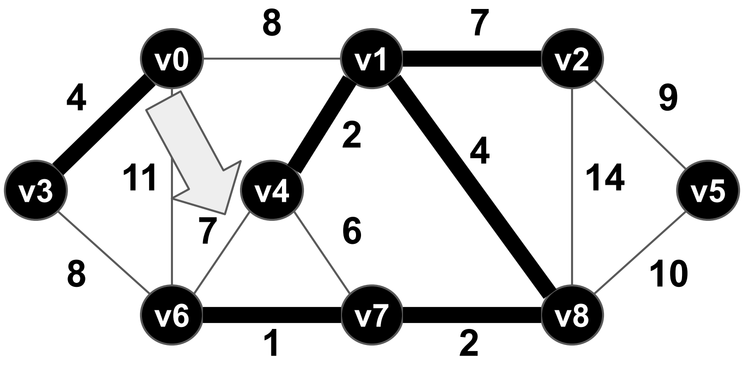
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

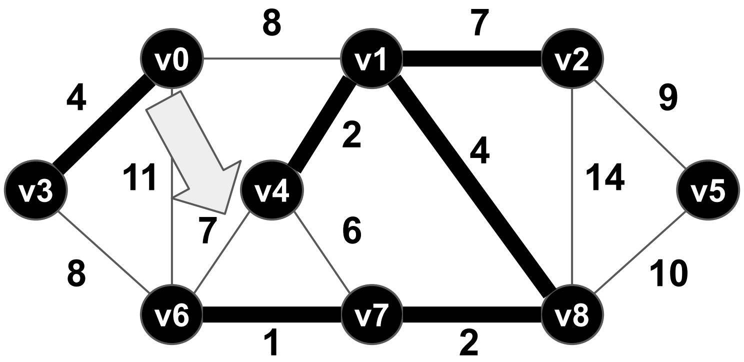
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

Algoritmo de Kruskal

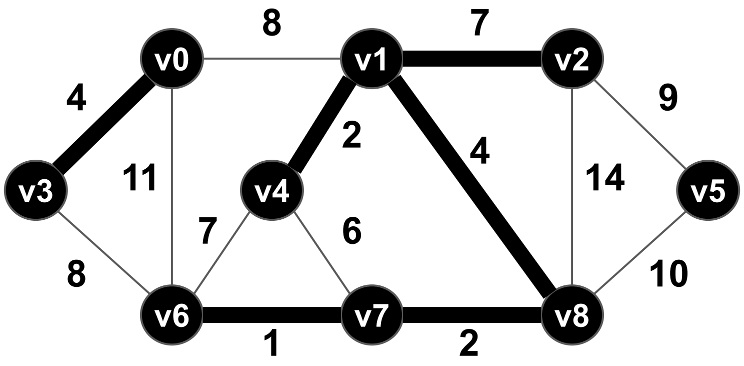


v4 v6 **não conecta duas árvores diferentes**.
Portanto, não é adicionada à floresta

Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
v4 v6
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

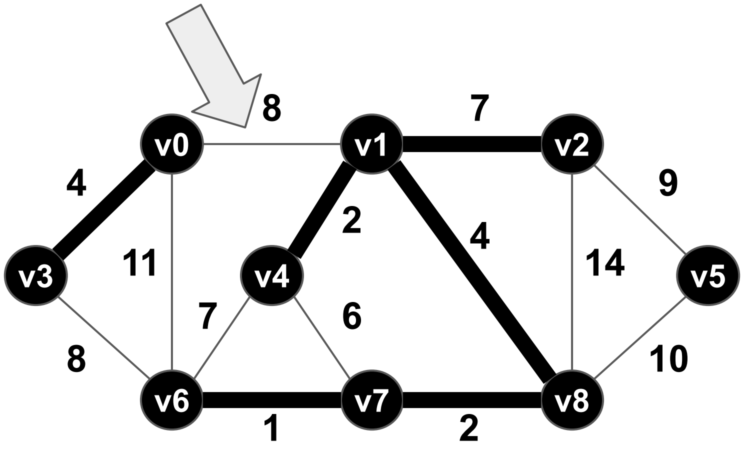
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

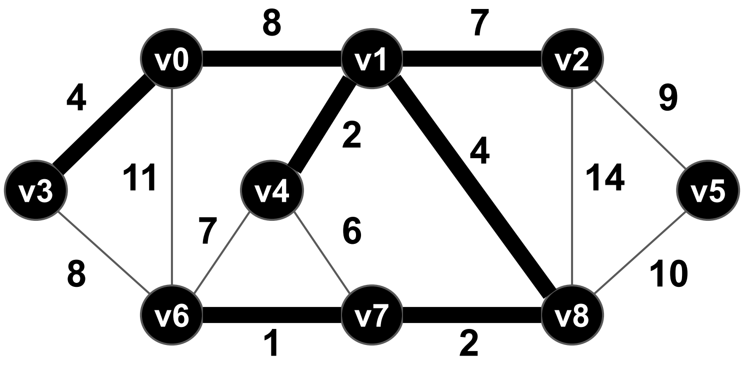
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
v0 v1
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

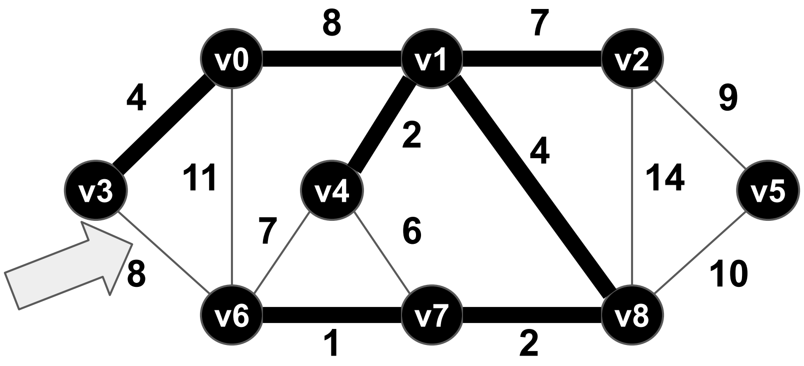
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

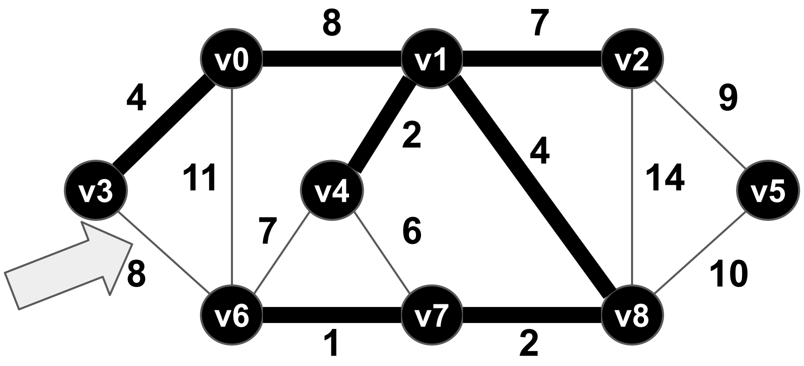
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

Algoritmo de Kruskal

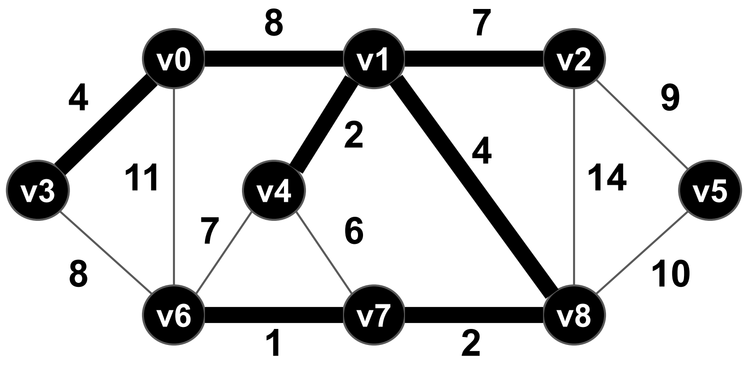


v3 v6 **não conecta duas árvores diferentes**.
Portanto, não é adicionada à floresta

Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
v3 v6
v2 v5
v5 v8
v0 v6
v2 v8

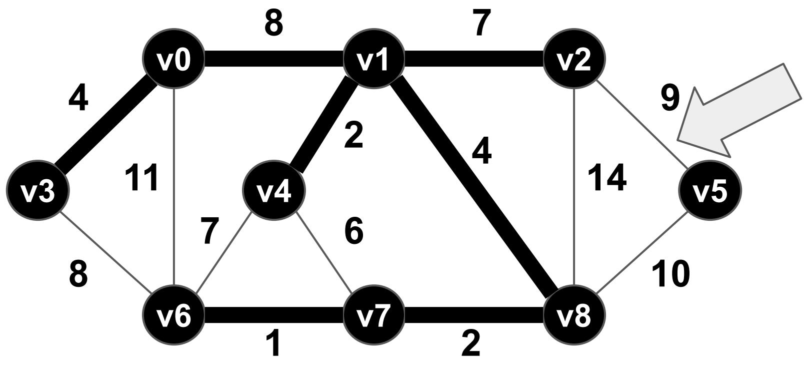
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
v2 v5
v5 v8
v0 v6
v2 v8

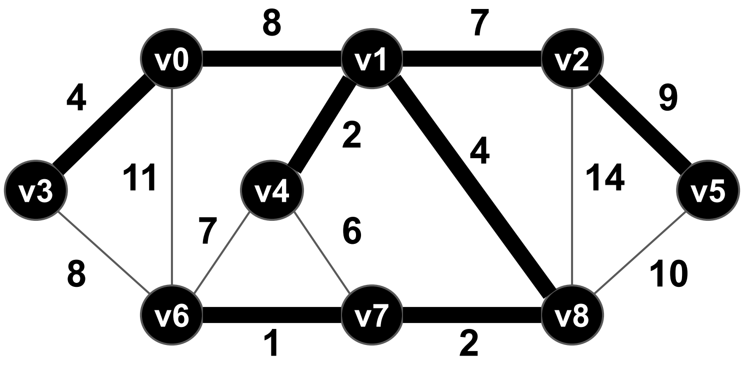
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
v2 v5
v5 v8
v0 v6
v2 v8

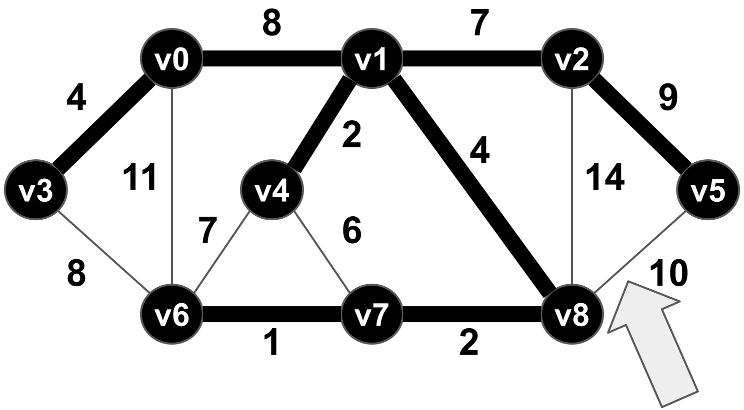
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
~~v2 v5~~
v5 v8
v0 v6
v2 v8

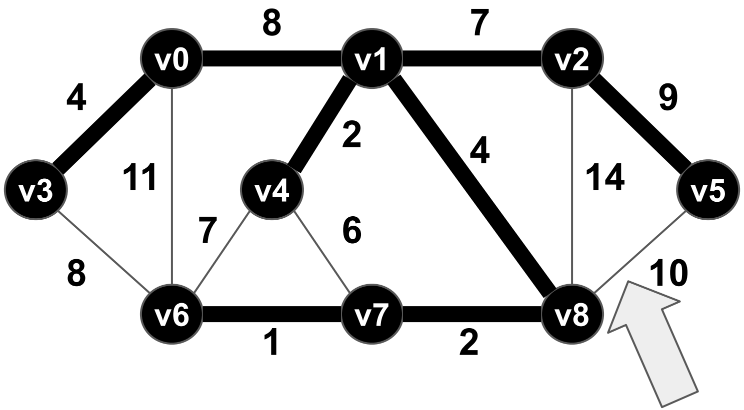
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
~~v2 v5~~
v5 v8
v0 v6
v2 v8

Algoritmo de Kruskal

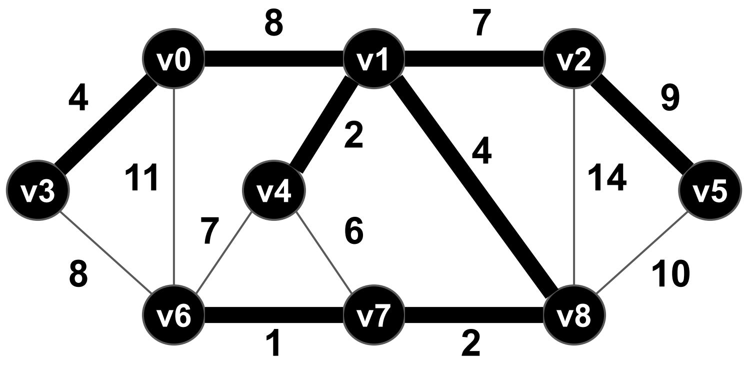


v5 v8 **não conecta duas árvores diferentes**.
Portanto, não é adicionada à floresta

Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
~~v2 v5~~
v5 v8
v0 v6
v2 v8

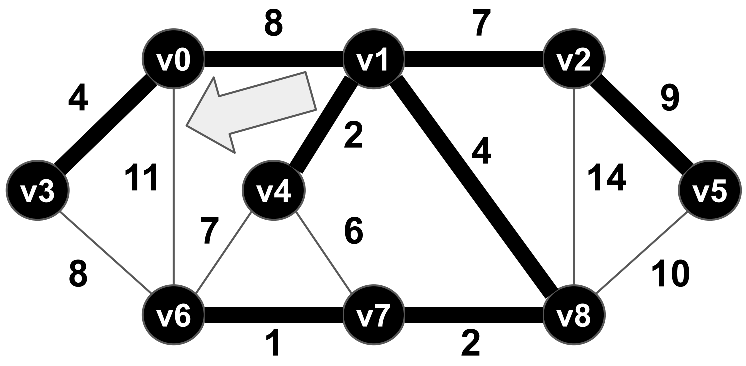
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
~~v2 v5~~
~~v5 v8~~
v0 v6
v2 v8

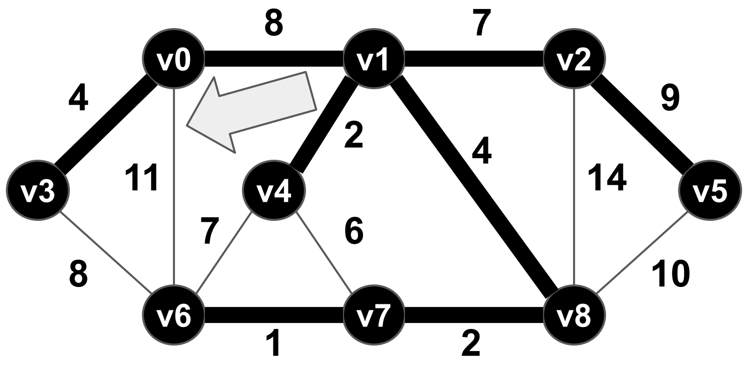
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
~~v2 v5~~
~~v5 v8~~
v0 v6
v2 v8

Algoritmo de Kruskal

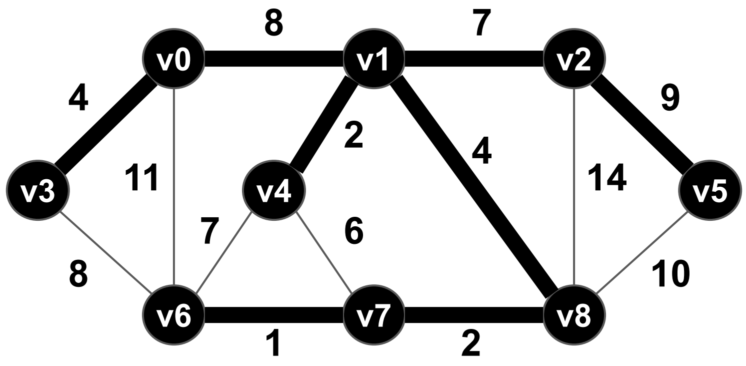


Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
~~v2 v5~~
~~v5 v8~~
v0 v6
v2 v8

v0 v6 **não conecta duas árvores diferentes**.
Portanto, não é adicionada à floresta

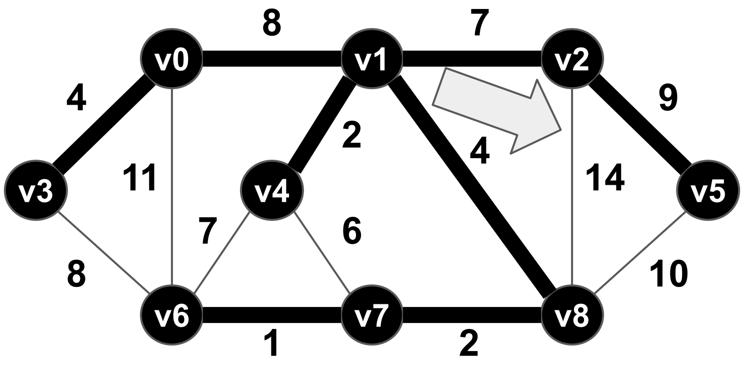
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
~~v2 v5~~
~~v5 v8~~
~~v0 v6~~
v2 v8

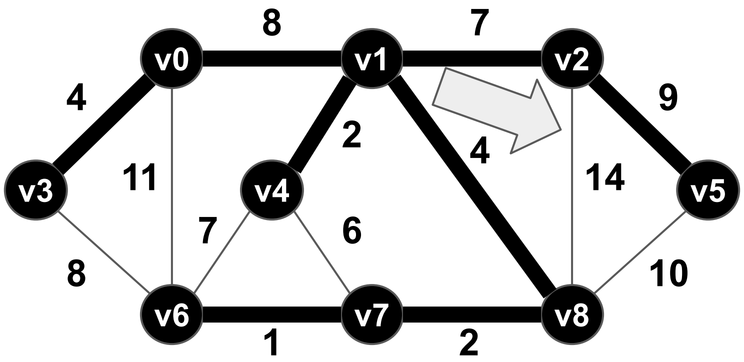
Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
~~v2 v5~~
~~v5 v8~~
~~v0 v6~~
v2 v8

Algoritmo de Kruskal

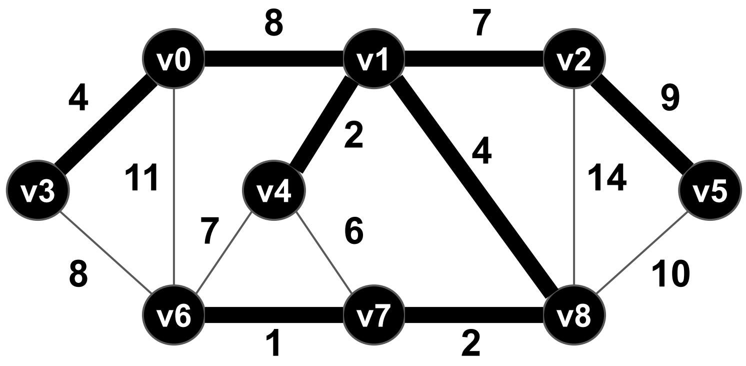


v2 v8 **não conecta duas árvores diferentes**.
Portanto, não é adicionada à floresta

Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
~~v2 v5~~
~~v5 v8~~
~~v0 v6~~
v2 v8

Algoritmo de Kruskal



Arestas em ordem crescente de peso:

~~v6 v7~~
~~v1 v4~~
~~v7 v8~~
~~v0 v3~~
~~v1 v8~~
~~v4 v7~~
~~v1 v2~~
~~v4 v6~~
~~v0 v1~~
~~v3 v6~~
~~v2 v5~~
~~v5 v8~~
~~v0 v6~~
~~v2 v8~~

Algoritmo de Kruskal

- Para implementar o Algoritmo de Kruskal, podemos usar uma estrutura de dados conhecida como **conjuntos-disjuntos** (*disjoint-sets* ou *union-find*)
- Uma estrutura de dados deste tipo mantém uma coleção de conjuntos disjuntos, associando a cada conjunto um **representante**, que é um dos elementos do conjunto
- Tipicamente, podemos realizar três operações em uma estrutura de dados de conjuntos disjuntos:
  - CriaConjuntos(n): Para $i = 0, 1, \ldots, n - 1$, cria um conjunto que contém apenas i
  - UneConjuntos(x, y): Faz a união do conjunto que contém x com o conjunto que contém y
  - EncontraConjunto(x): Retorna o representante do conjunto que contém x

Algoritmo de Kruskal

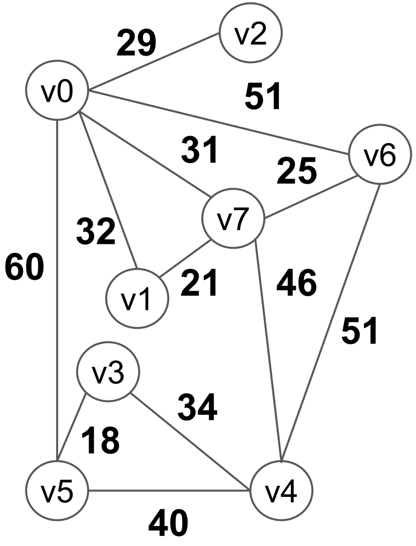
O laço dos Passos 4 a 7 pode ser encerrado após *n* - 1 arestas terem sido adicionadas à floresta que estamos construindo

Kruskal(*G* conexo)

1. *num_arestas_F* = 0
2. CriaConjuntos(*n*); sendo *n* o número de vértices de *G*
3. Ordene as arestas de *G* em ordem crescente de peso
4. Para cada aresta *uv* de *G* considerando as arestas de *G* em ordem crescente de peso:
5. Se EncontraConjunto(u) ≠ EncontraConjunto(v):
6. *arestas_F*[*num_arestas_F*] = *uv*
7. *num_arestas_F = num_arestas_F + 1*
8. UneConjuntos(*u*, *v*)
9. Retorne *arestas_F*

Exercícios

1. Indique a árvore geradora de peso mínimo retornada pelo Algoritmo de Kruskal para o grafo abaixo.



Exercícios

2. Um grafo desconexo não tem árvores geradoras, mas tem florestas geradoras. O peso de uma floresta geradora é igual à soma dos pesos das suas arestas. Em relação aos Algoritmos de Prim e Kruskal, responda às seguintes questões justificando a sua resposta:
   a. Aplicado a um grafo desconexo G, o Algoritmo de Prim encontra uma floresta geradora de peso mínimo de G?
   b. Aplicado a um grafo desconexo G, o Algoritmo de Kruskal encontra uma floresta geradora de peso mínimo de G?

Referências

- Esta apresentação é baseada nos seguintes materiais:

1. Capítulo 23 do livro
   Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. Introduction to Algorithms. 3rd. ed. MIT Press, 2009.
2. Capítulo 20 do livro
   Sedgewick, R. Algorithms in C++ – Part 5. Graph Algorithms. 3rd. ed. Addison-Wesley, 2002.