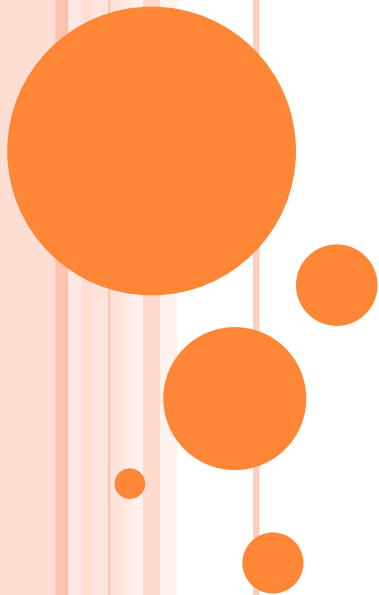


ESTRUTURAS DE DADOS III

Grafos – Árvores Geradoras Mínimas

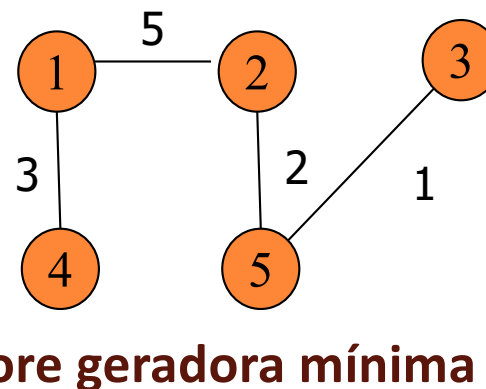
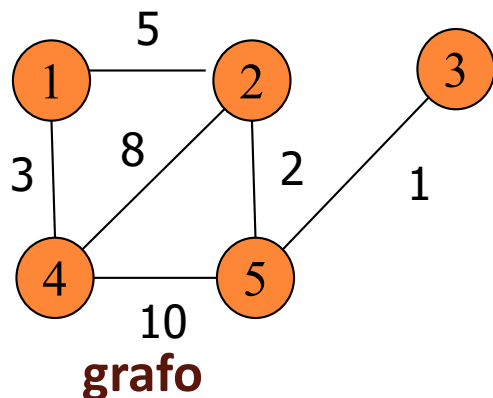
Profa. Elaine Parros Machado de Sousa
adaptações: Cristina Dutra de Aguiar Ciferri

Material baseado em aulas dos professores:
Gustavo Batista, Robson Cordeiro, Moacir Ponti Jr. e
Maria Cristina Oliveira, Thiago A. S. Pardo



ÁRVORE GERADORA MÍNIMA: RELEMBRANDO DEFINIÇÕES

- **Árvore** (ou árvore livre):
 - um **grafo conexo acíclico**.
- **Árvore geradora** (*spanning tree*) de um grafo conexo:
 - um **subgrafo gerador** que é uma **árvore** => contém todos os vértices
- **Árvore geradora mínima** (*minimum spanning tree*):
 - uma árvore geradora com a **menor soma de pesos** de arestas



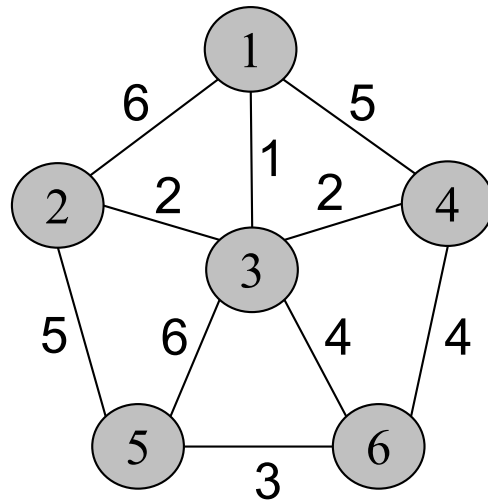
ÁRVORE GERADORA MÍNIMA: ALGORITMOS

- Dois algoritmos bastante conhecidos
 - Algoritmo de **Prim**
 - Algoritmo de **Kruskal**
- Características
 - algoritmos “gulosos”
 - encontram a árvore geradora mínima de um grafo não direcionado

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM

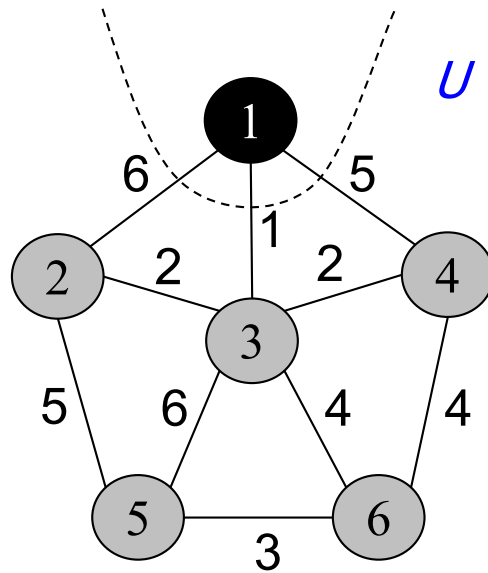
- Ideia geral
 - 1) Começar com um vértice v qualquer, e adicioná-lo a um conjunto U ;
 - 2) Escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$;
 - 3) Incluir o vértice da aresta escolhida em U ;
 - 4) Incluir a aresta escolhida em um conjunto T ;
 - 5) Voltar ao passo 2 enquanto $U \neq V$.

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO



vértice de início: 1

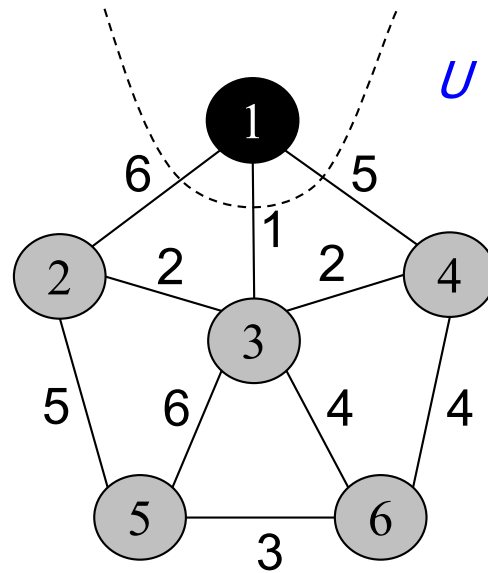
ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO



algoritmo: adicionar vértice ao conjunto U

$U = \{1\}$
 $T =$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

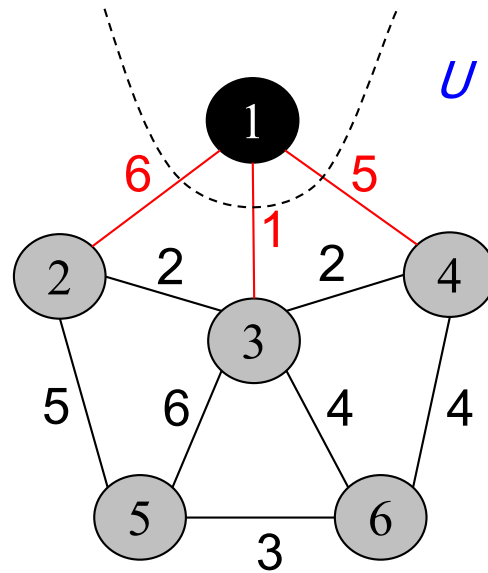


algoritmo: escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$

$U = \{1\}$

$T =$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

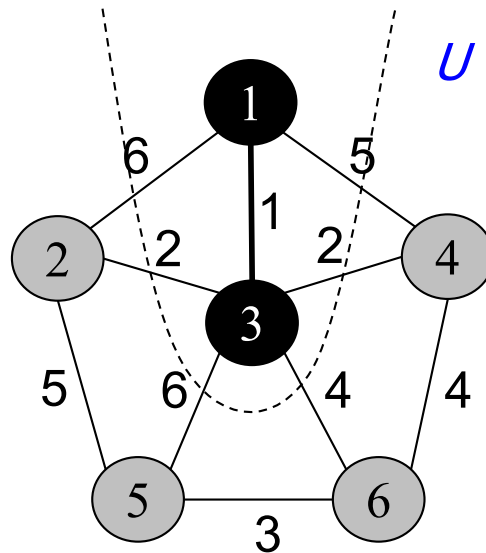


algoritmo: escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$

$U = \{1\}$

$T =$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

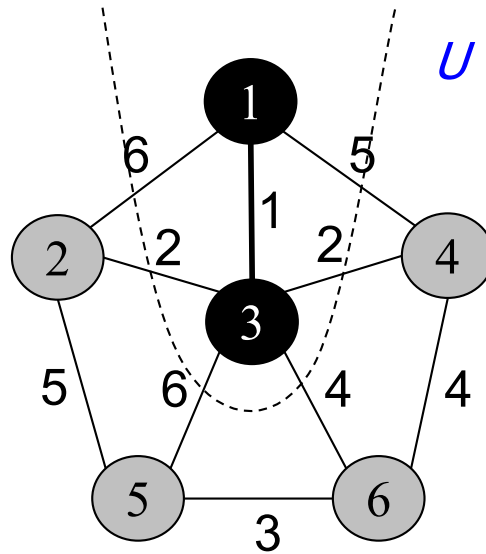


algoritmo: incluir o vértice da aresta escolhida em U
incluir a aresta escolhida em T

$$U = \{1,3\}$$

$$T = \{(1,3)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

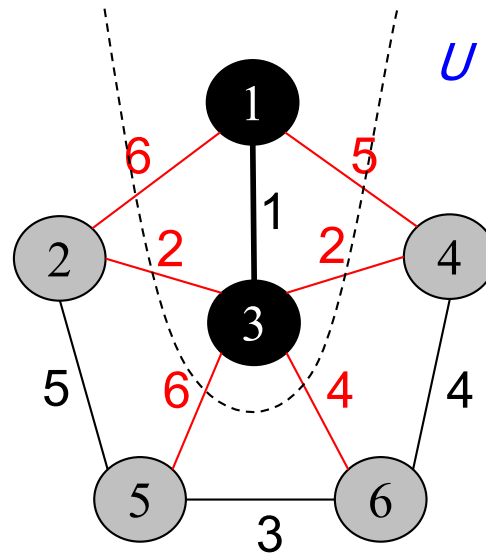


algoritmo: escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$

$$U = \{1,3\}$$

$$T = \{(1,3)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

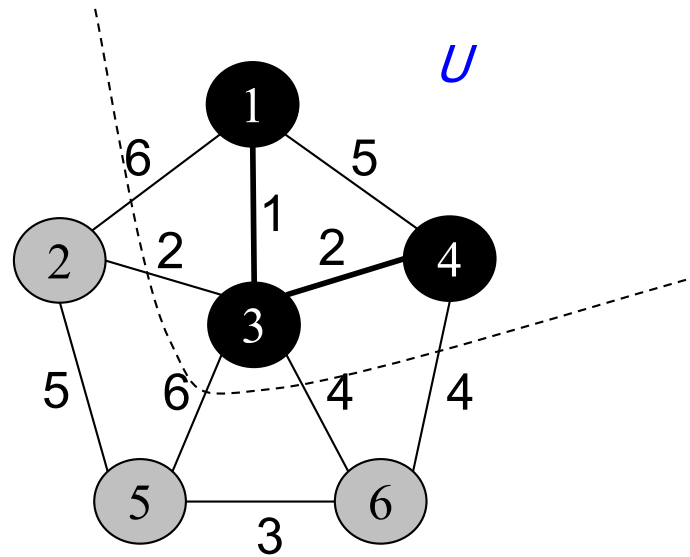


algoritmo: escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$

$$U = \{1,3\}$$

$$T = \{(1,3)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

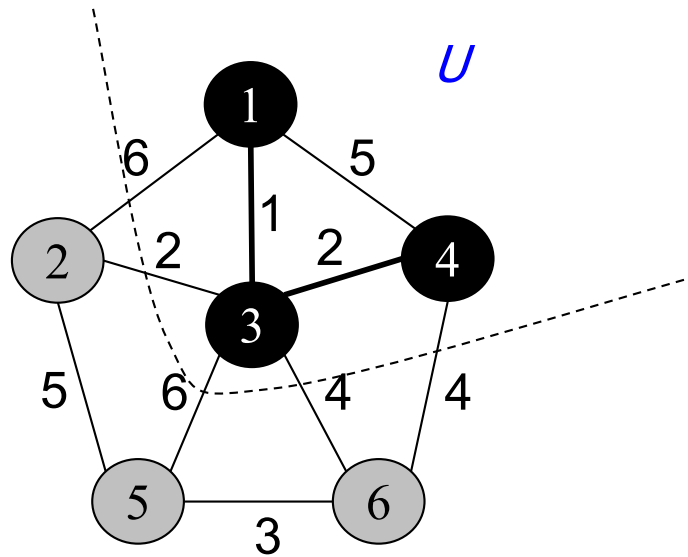


algoritmo: incluir o vértice da aresta escolhida em U
incluir a aresta escolhida em T

$$U = \{1,3,4\}$$

$$T = \{(1,3),(3,4)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

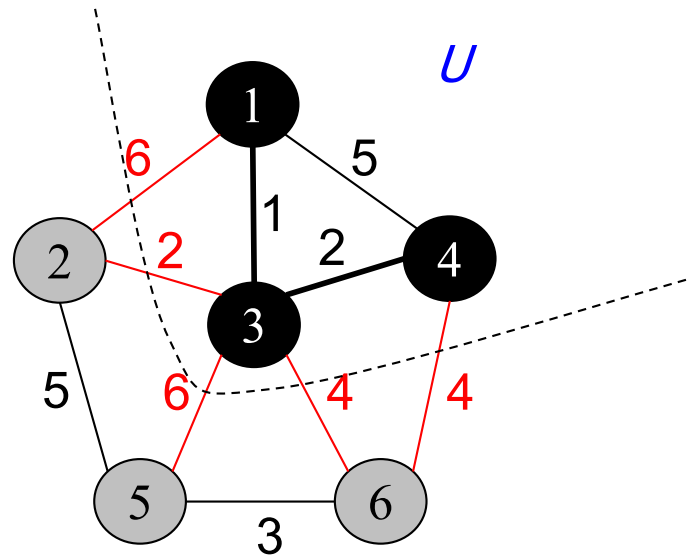


algoritmo: escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$

$$U = \{1, 3, 4\}$$

$$T = \{(1, 3), (3, 4)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

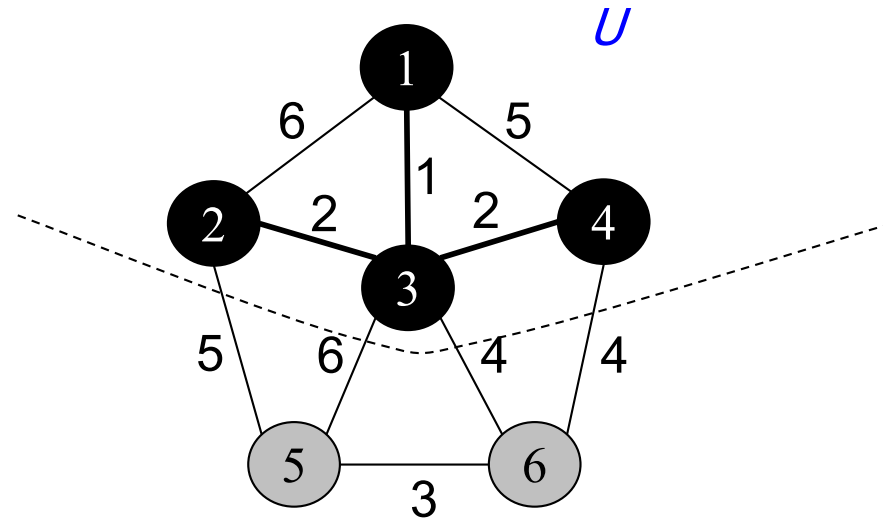


algoritmo: escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$

$$U = \{1, 3, 4\}$$

$$T = \{(1, 3), (3, 4)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

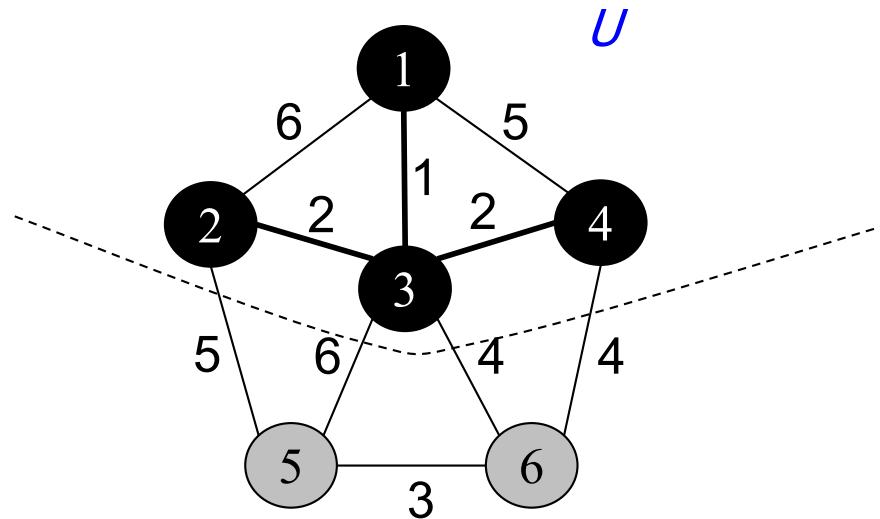


algoritmo: incluir o vértice da aresta escolhida em U
incluir a aresta escolhida em T

$$U = \{1,3,4,2\}$$

$$T = \{(1,3),(3,4),(2,3)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

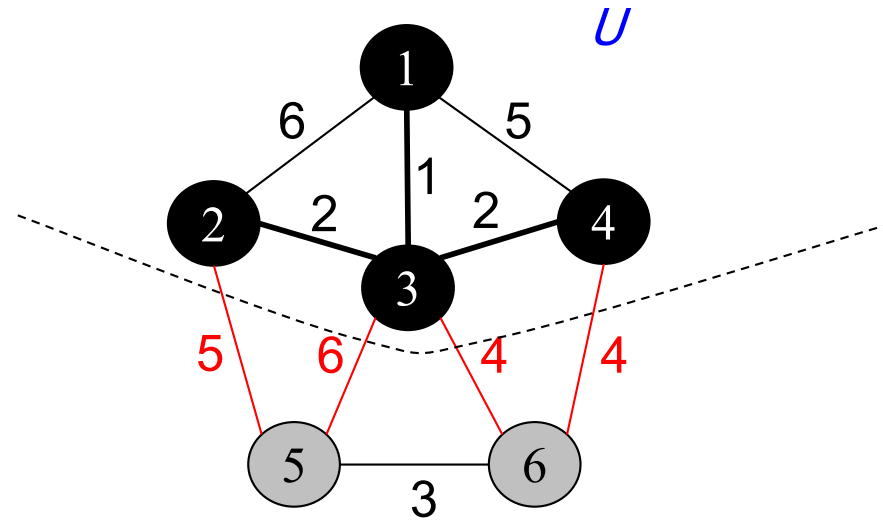


algoritmo: escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$

$$U = \{1,3,4,2\}$$

$$T = \{(1,3),(3,4),(2,3)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

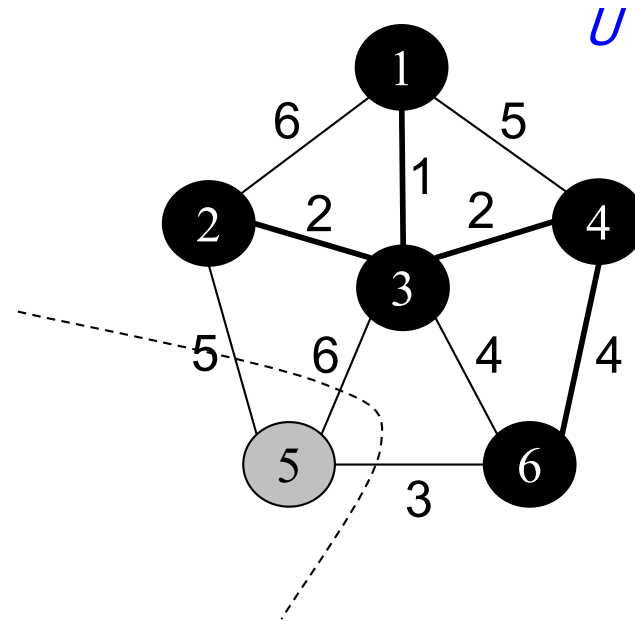


algoritmo: escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$

$$U = \{1,3,4,2\}$$

$$T = \{(1,3),(3,4),(2,3)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

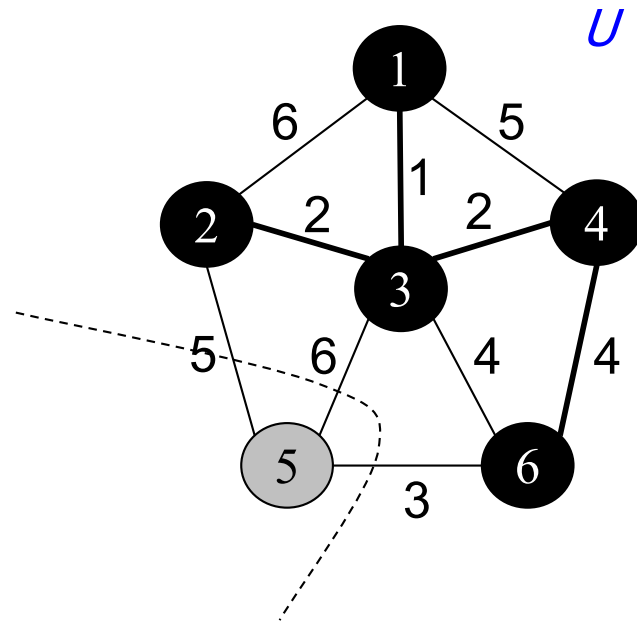


algoritmo: incluir o vértice da aresta escolhida em U
incluir a aresta escolhida em T

$$U = \{1,3,4,2,6\}$$

$$T = \{(1,3),(3,4),(2,3),(4,6)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

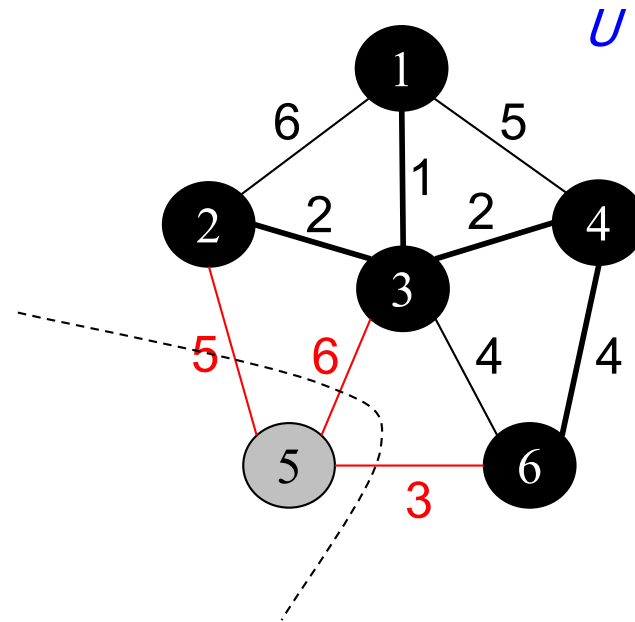


algoritmo: escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$

$$U = \{1, 3, 4, 2, 6\}$$

$$T = \{(1, 3), (3, 4), (2, 3), (4, 6)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

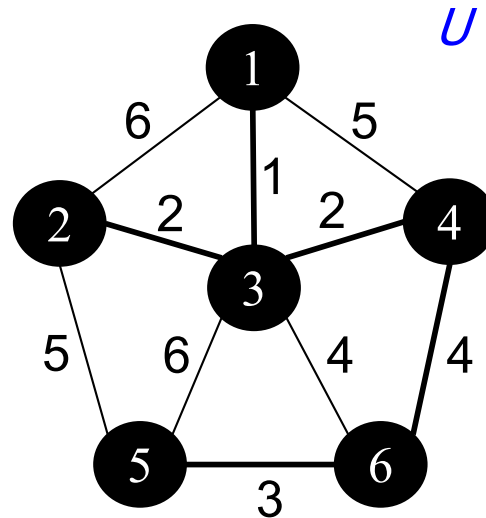


algoritmo: escolher a aresta de menor peso que conecta um vértice em U a um vértice em $V-U$

$$U = \{1, 3, 4, 2, 6\}$$

$$T = \{(1, 3), (3, 4), (2, 3), (4, 6)\}$$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

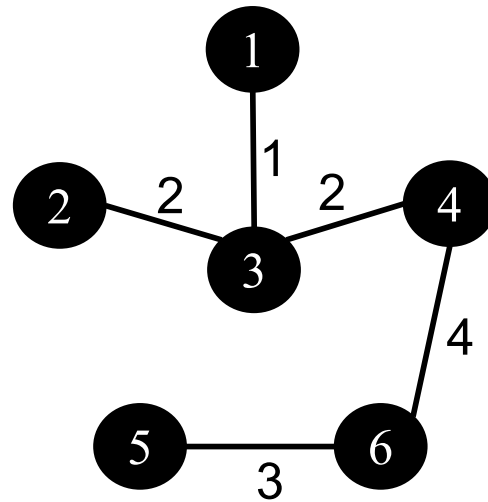


algoritmo: incluir o vértice da aresta escolhida em U
incluir a aresta escolhida em T

$$U = \{1,3,4,2,6,5\}$$

$$T = \{(1,3),(3,4),(2,3),(4,6),(5,6)\}$$

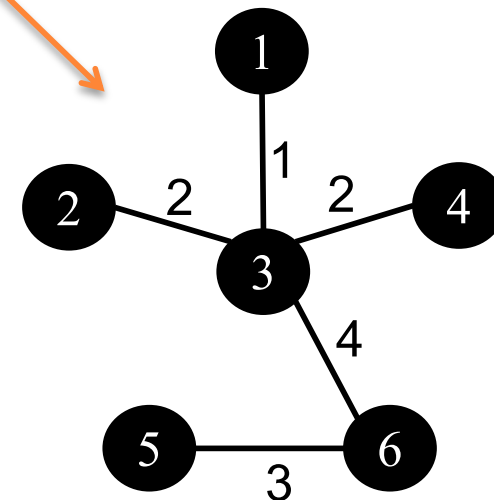
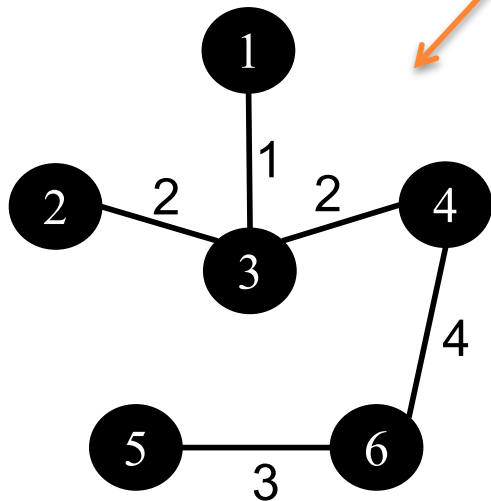
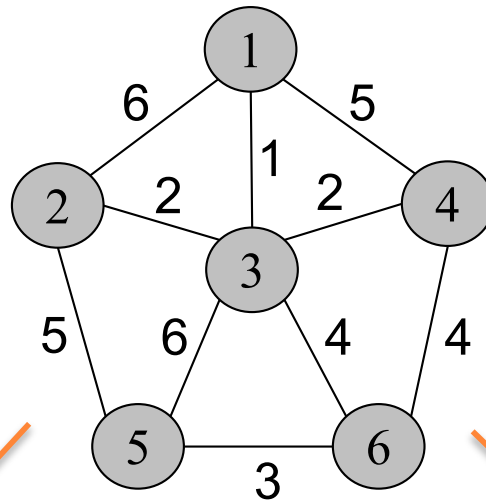
ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO




FIM DO ALGORITMO

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM - EXEMPLO

Dado um grafo G ,
pode existir mais
de uma árvore
geradora mínima
para G



ÁRVORE GERADORA MÍNIMA: ALGORITMO DE PRIM

```
procedimento Prim(var Grafo: TGrafo;  
                  var T: conjunto de arestas)  
variáveis  
    u, v: TVertice;  
    U: conjunto de TVertice;  
início  
    T :=  $\emptyset$ ;  
    U := {1};  
    enquanto U  $\neq$  V faça  
        início  
             seja (u, v) a aresta de menor peso  
                tal que (u  $\in$  U) e (v  $\in$  V-U)  
                T := T  $\cup$  {(u, v)};  
                U := U  $\cup$  {v};  
        fim  
fim
```


ALGORITMO DE PRIM:

COMPLEXIDADE

Eficiência do algoritmo de Prim depende de como será feita a seleção da aresta (u, v)

- Implementação simples com dois vetores:
 - **prox[i]** fornece o vértice em **U** atualmente mais próximo ao vértice **i** em **V-U**.
 - **mc[i]** fornece o custo da aresta **(i, prox[i])**.
 - operação de encontrar **(u, v)** => percorrer o vetor **mc**
=> **$O(|V|)$**
 - necessário atualizar os vetores **prox** e **mc** a cada novo vértice em U
- Complexidade dessa implementação
 - **$O(|V|^2)$** .

ALGORITMO DE PRIM:

COMPLEXIDADE

Eficiência do algoritmo de Prim depende de como será feita a seleção da aresta (u, v)

- Implementação mais sofisticada:
 - fila de prioridade para manter os vértices em $V-U$.
 - chave da fila de prioridade de um vértice $v \in V-U$ é o peso da aresta mais leve que liga v a um vértice de U .
 - se a fila de prioridade for implementada com um *heap*
- Complexidade dessa implementação
 - $O(|A| \log |V|)$.

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL

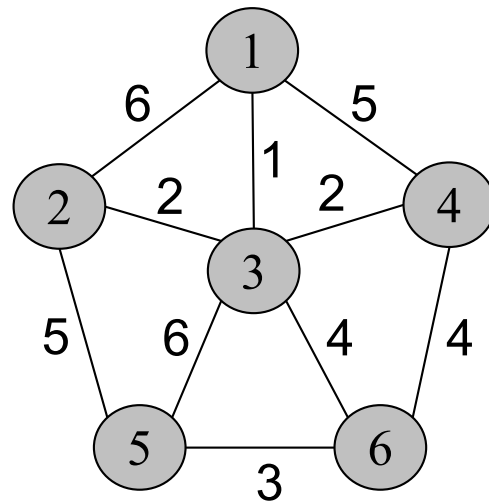
○ Ideia geral:

- inicia-se com um grafo $G' = (V, \emptyset)$
- cada vértice é um componente conexo de si mesmo
- a cada iteração, são construídos componentes conexos cada vez maiores

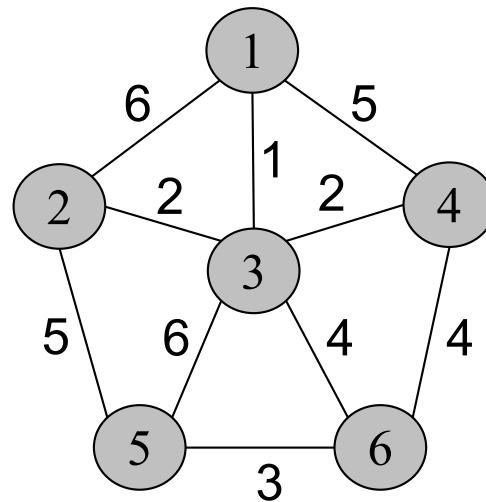
ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL

- Ideia geral (cont.):
 - para “aumentar” os componentes conexos \Rightarrow arestas em **A** são analisadas por **ordem ascendente** de peso.
 - **Q**: contém as arestas de G ordenadas pelo peso
 - se a aresta conecta dois vértices em dois componentes separados \Rightarrow a aresta é adicionada a **T**.
 - se a aresta conecta dois vértices do mesmo componente \Rightarrow ela é descartada, pois criaria um ciclo.

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



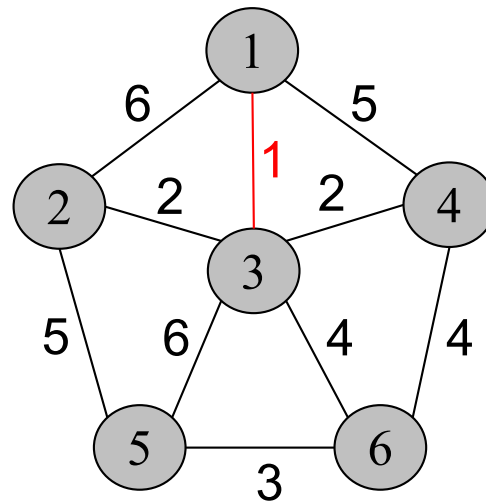
no início,
cada vértice
é um
componente
distinto

Componentes conexos = {1}, {2}, {3}, {4}, {5}, {6}

$Q = \{(1,3), (2,3), (3,4), (5,6), (3,6), (4,6), (1,4), (2,5), (1,2), (3,5)\}$

T =

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



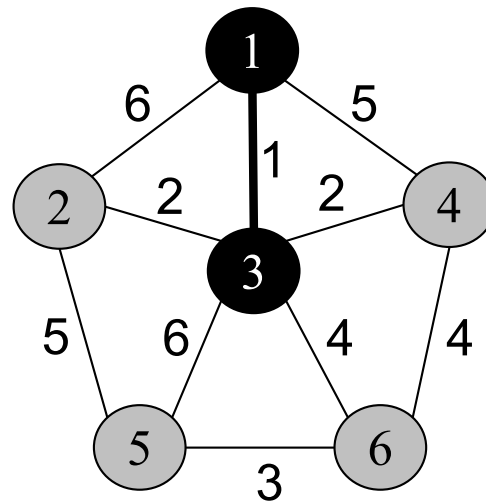
algoritmo: escolher a aresta de menor peso que conecta dois componentes distintos

Componentes conexos = {1}, {2}, {3}, {4}, {5}, {6}

$Q = \{(1,3), (2,3), (3,4), (5,6), (3,6), (4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T =$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



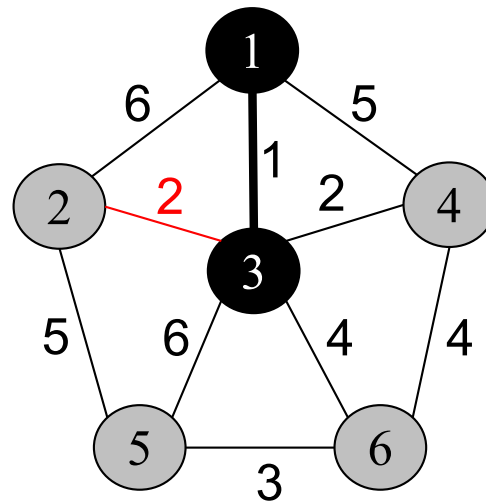
algoritmo: adicionar aresta a T

Componentes conexos = $\{1,3\}, \{2\}, \{4\}, \{5\}, \{6\}$

$Q = \{(2,3), (3,4), (5,6), (3,6), (4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



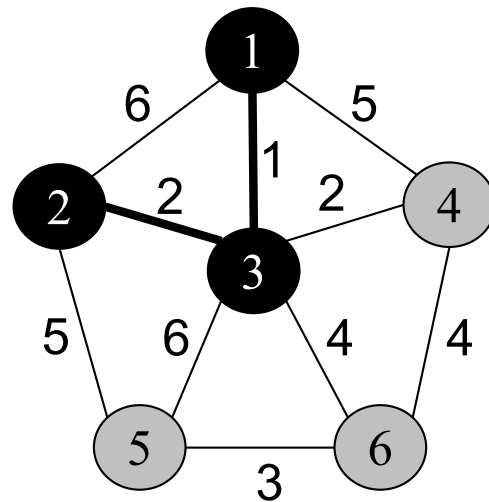
algoritmo: escolher a aresta de menor peso que conecta dois componentes distintos

Componentes conexos = $\{1,3\}, \{2\}, \{4\}, \{5\}, \{6\}$

$Q = \{(2,3), (3,4), (5,6), (3,6), (4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



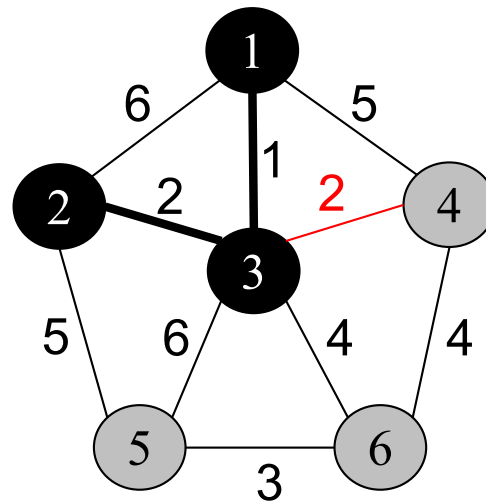
algoritmo: adicionar aresta a T

Componentes conexos = $\{1, 2, 3\}, \{4\}, \{5\}, \{6\}$

$Q = \{(3,4), (5,6), (3,6), (4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3), (2,3)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



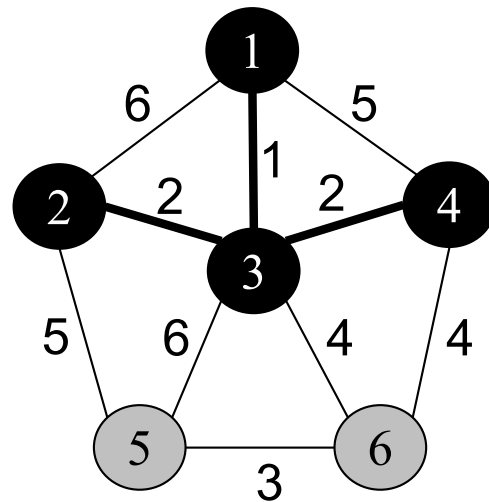
algoritmo: escolher a aresta de menor peso que conecta dois componentes distintos

Componentes conexos = $\{1, 2, 3\}, \{4\}, \{5\}, \{6\}$

$Q = \{(3,4), (5,6), (3,6), (4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3), (2,3)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



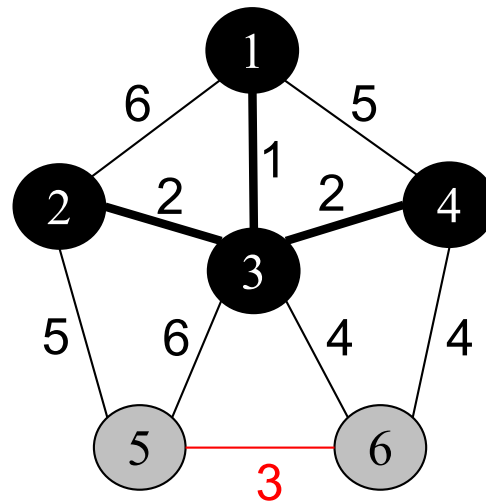
algoritmo: adicionar aresta a T

Componentes conexos = $\{1, 2, 3, 4\}, \{5\}, \{6\}$

$Q = \{(5,6), (3,6), (4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3), (2,3), (3,4)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



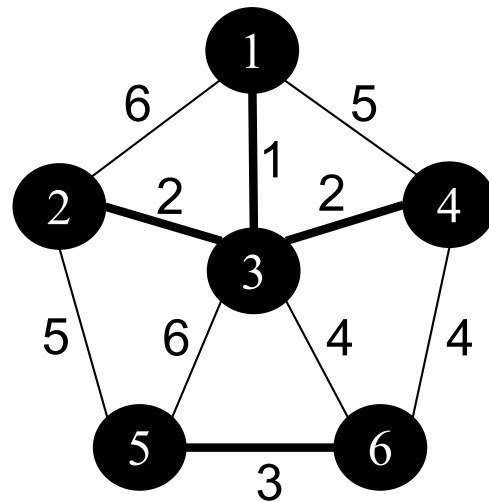
algoritmo: escolher a aresta de menor peso que conecta dois componentes distintos

Componentes conexos = $\{1, 2, 3, 4\}, \{5\}, \{6\}$

$Q = \{(5,6), (3,6), (4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3), (2,3), (3,4)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



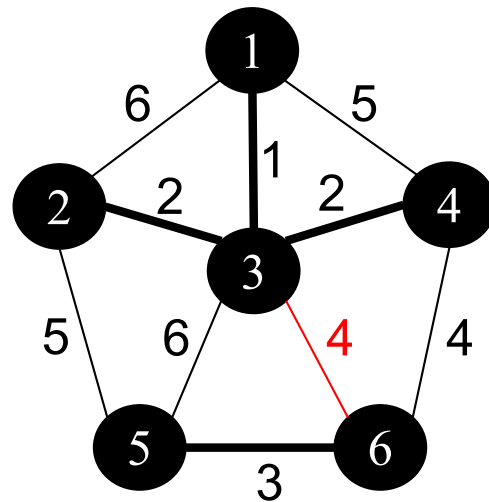
algoritmo: adicionar aresta a T

Componentes conexos = $\{1, 2, 3, 4\}, \{5, 6\}$

$Q = \{(3,6), (4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3), (2,3), (3,4), (5,6)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



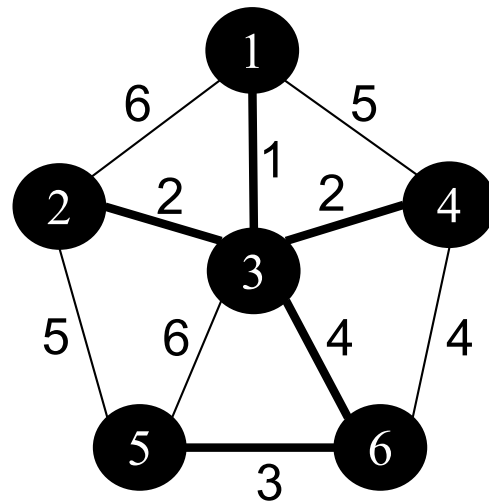
algoritmo: escolher a aresta de menor peso que conecta dois componentes distintos

Componentes conexos = $\{1, 2, 3, 4\}, \{5, 6\}$

$Q = \{(3,6), (4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3), (2,3), (3,4), (5,6)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



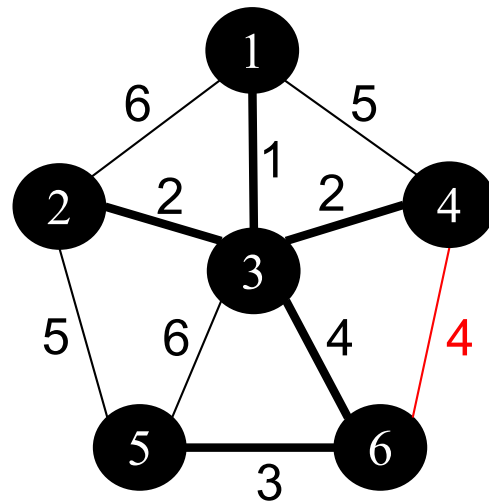
algoritmo: adicionar aresta a T

Componentes conexos = {1, 2, 3, 4, 5, 6}

$Q = \{(4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3), (2,3), (3,4), (5,6), (3,6)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



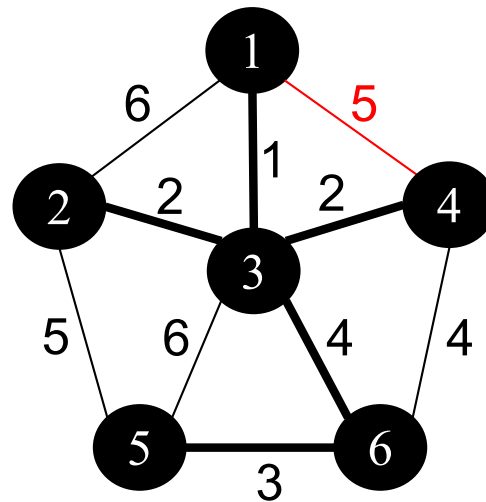
algoritmo: descartar a aresta de menor peso, desde que ela conecta dois vértices do mesmo componente

Componentes conexos = {1, 2, 3, 4, 5, 6}

$Q = \{(4,6), (1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3), (2,3), (3,4), (5,6), (3,6)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



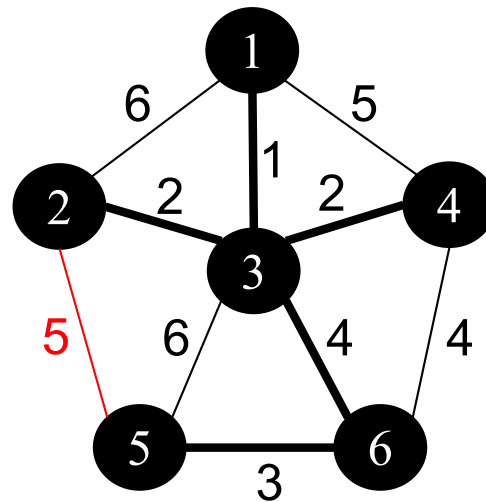
algoritmo: descartar a aresta de menor peso, desde que ela conecta dois vértices do mesmo componente

Componentes conexos = {1, 2, 3, 4, 5, 6}

$Q = \{(1,4), (2,5), (1,2), (3,5)\}$

$T = \{(1,3), (2,3), (3,4), (5,6), (3,6)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



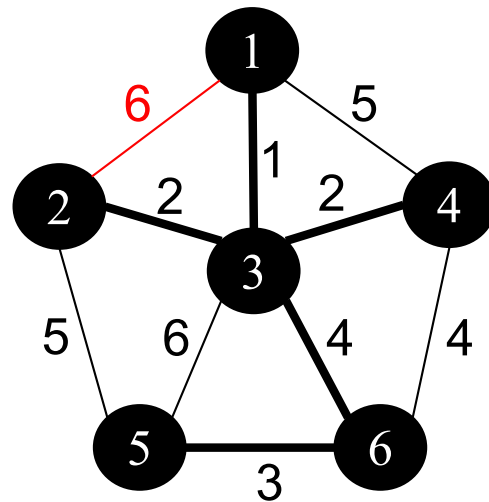
algoritmo: descartar a aresta de menor peso, desde que ela conecta dois vértices do mesmo componente

Componentes conexos = {1, 2, 3, 4, 5, 6}

$Q = \{(2,5), (1,2), (3,5)\}$

$T = \{(1,3), (2,3), (3,4), (5,6), (3,6)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



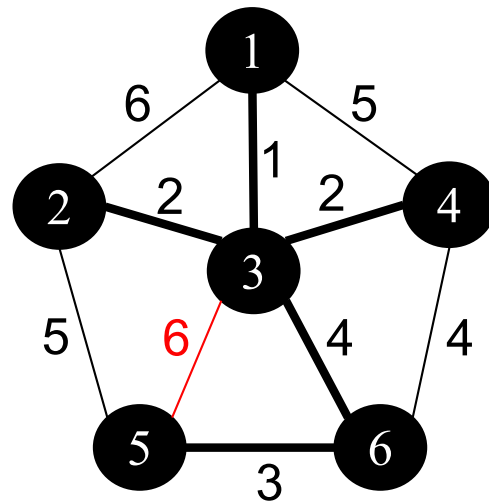
algoritmo: descartar a aresta de menor peso, desde que ela conecta dois vértices do mesmo componente

Componentes conexos = {1, 2, 3, 4, 5, 6}

$Q = \{(1,2), (3,5)\}$

$T = \{(1,3), (2,3), (3,4), (5,6), (3,6)\}$

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



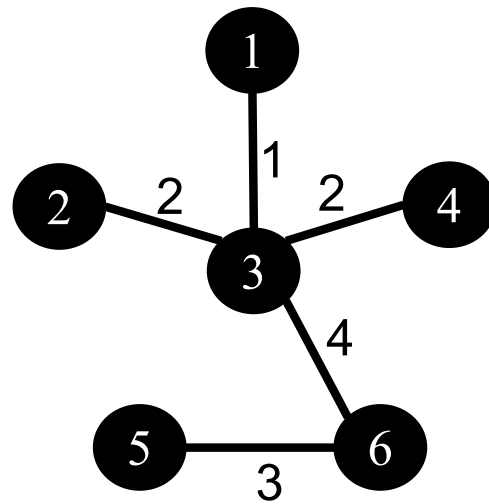
algoritmo: descartar a aresta de menor peso, desde que ela conecta dois vértices do mesmo componente

Componentes conexos = {1, 2, 3, 4, 5, 6}

$Q = \{(3,5)\}$

$T = \{(1,3), (2,3), (3,4), (5,6), (3,6)\}$


ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL - EXEMPLO



FIM DO ALGORITMO

ÁRVORE GERADORA MÍNIMA: ALGORITMO DE KRUSKAL

```
procedimento Kruskal(var Grafo: TGrafo;  
                    var T: conjunto de arestas)  
variáveis  
    u, v: TVertice;  
    U1, ..., Un: conjunto de TVertice;  
    Q: fila de prioridade;  
início  
    T := ∅;  
    Q := as arestas de G ordenadas pelo seu peso;  
    para i:=1 até Grafo.NumVertices faça  
        Ui := {i};  
        enquanto houver arestas em Q faça  
            início  
                seja (u, v) a aresta de menor peso de Q tal que  
                    (u ∈ Up) e (v ∈ Uq) e (Up ∩ Uq) = ∅  
                T := T ∪ {(u, v)};  
                Up := Up ∪ Uq;  
                eliminar Uq;  
            fim  
    fim
```



ALGORITMO DE KRUSKAL:

COMPLEXIDADE

Eficiência do algoritmo de Kruskal depende de dois fatores principais:

- encontrar a aresta de menor peso
- verificar se a aresta conecta dois componentes distintos ($U_p \cap U_q$)

○ Implementação mais sofisticada:

- **Q** implementada como uma fila de prioridade com um *heap*
- operação de conjuntos eficiente

○ Complexidade dessa implementação

- $O(|A| \log |A|)$.

mais eficiente do que o algoritmo de Prim para grafos esparsos

BIBLIOGRAFIA

- N. Ziviani. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.
- T. H. Cormen, C. E. Leiserson and R. L. Rivest. Introduction to Algorithms, MIT Press, 2nd Edition, 2001.