SCC 503 - ALGORITMOS E ESTRUTURAS DE DADOS II

Grafos - Árvores Geradoras Mínimas

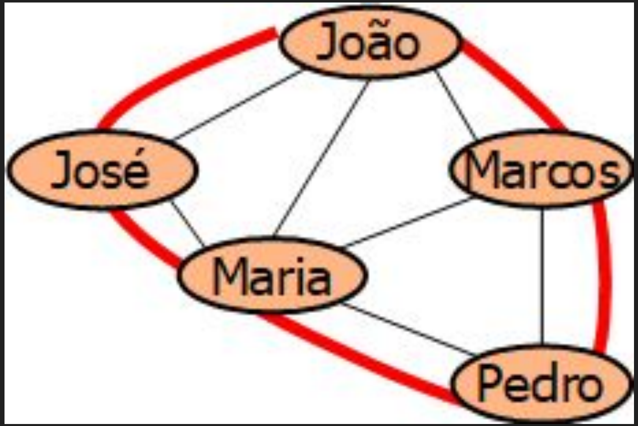
Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

*Material baseado em aulas dos professores: Elaine Parros Machado de Souza, Gustavo Batista, Robson Cordeiro, Moacir Ponti Jr., Maria Cristina Oliveira e Cristina Ciferri.

Ciclos

Ciclos

- Quanto tempo para eu receber uma notícia que eu mesmo postei?
- Ciclo
  - Caminho no qual o primeiro e o último vértices são iguais.

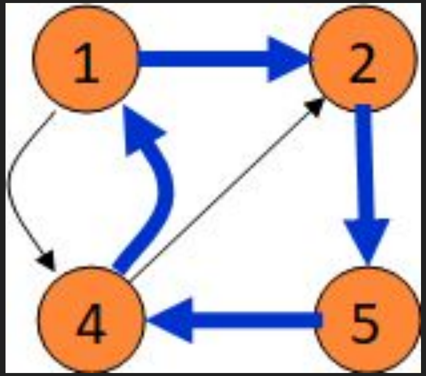


Ciclos

- Ciclo simples
  - Ciclo em que nenhum vértice se repete (exceto primeiro e último)
  - O ciclo é simples se os vértices v1, v2,...,vk são distintos.
- Grafo acíclico
  - Grafo sem ciclos.

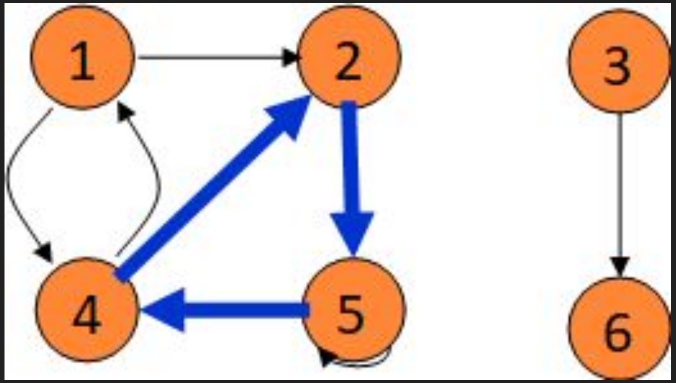
Ciclos

- Em um grafo direcionado:
  - Um caminho (v0, v1,...,vk) forma um ciclo se v0 = vk e o caminho contém pelo menos uma aresta.
    - Ex: (1,2,5,4,1)
- O self-loop é um ciclo de tamanho 1.
    - Ex: (5,5)



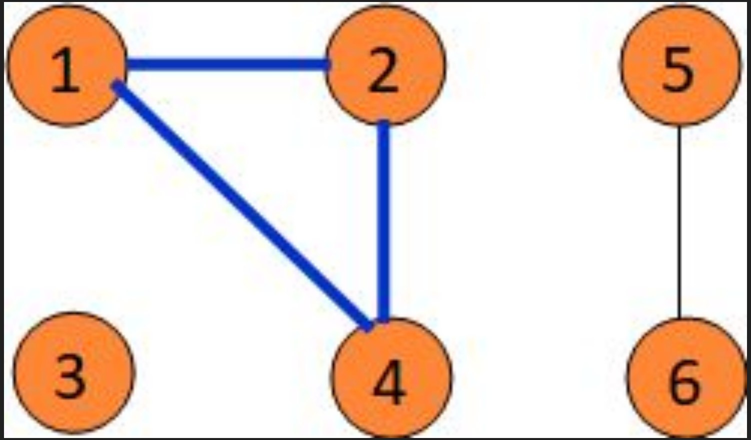
Ciclos

- Em um grafo direcionado:
  - Dois caminhos ($v0$, $v1$,..., vk) e ($v'0$,$v'1$,...,$v'k$) formam o mesmo ciclo se existir um inteiro j tal que $v'i = v(i+j) \bmod k$ para $i = 0,1,...,k-1$.
  - Ex.: caminhos que formam o mesmo ciclo:
    - (2,5,4,2)
    - (5,4,2,5)
    - (4,2,5,4).



Ciclos

- Em um grafo não direcionado:
  - Um caminho (v0,v1,...,vk) forma um ciclo se v0 = vk e o caminho contém pelo menos três arestas.
    - ex: (1,2,4,1)



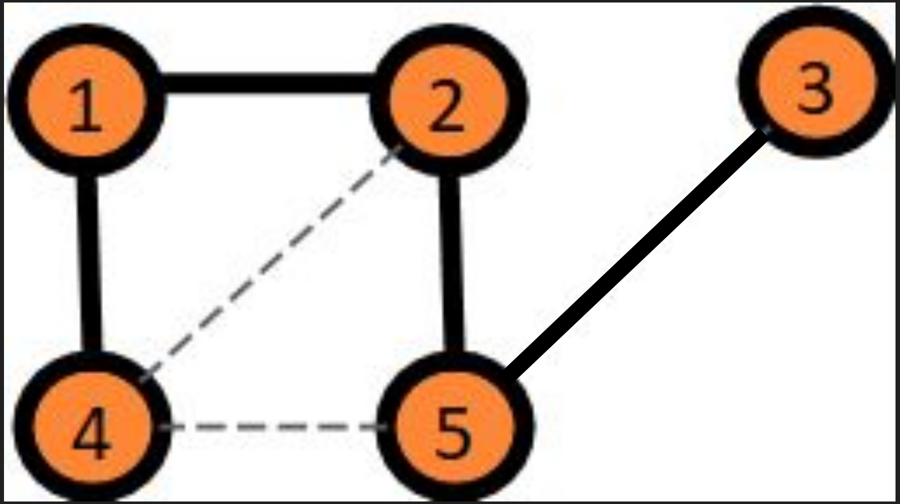
Subgrafo

Subgrafo

- Um subgrafo S de um grafo G é um grafo tal que:
  - Os vértices de S são um subconjunto dos vértices de G
  - As arestas de S são um subconjunto das arestas de G

Subgrafo

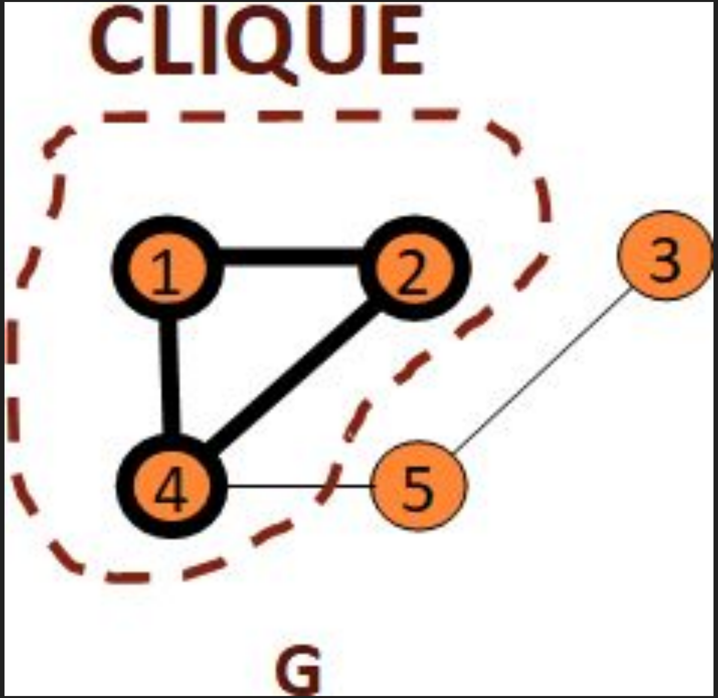
→ Um subgrafo gerador (spanning subgraph) de G é um subgrafo que contém todos os vértices de G.



Clique

Clique

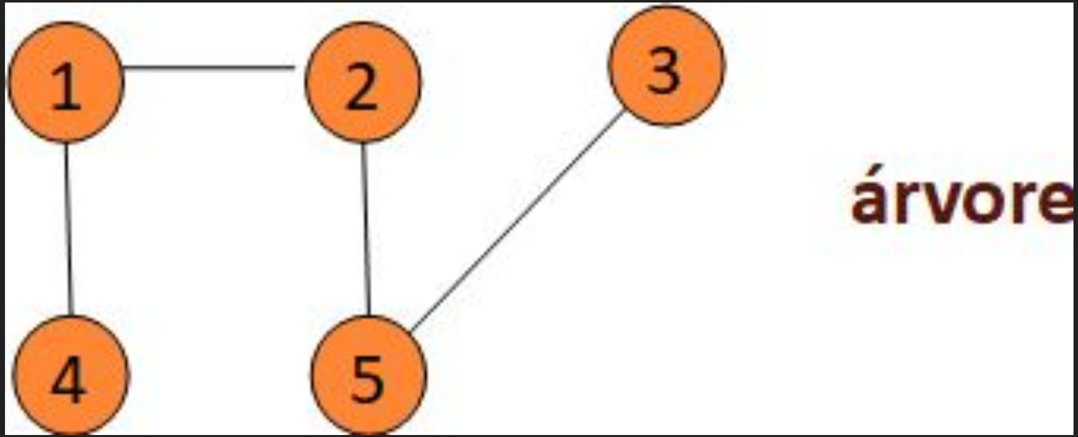
→ Um clique de um grafo G é um subgrafo completo de G



Árvore e Floresta

Árvore e Floresta

→ Uma árvore é um grafo conexo acíclico.



Árvore e Floresta

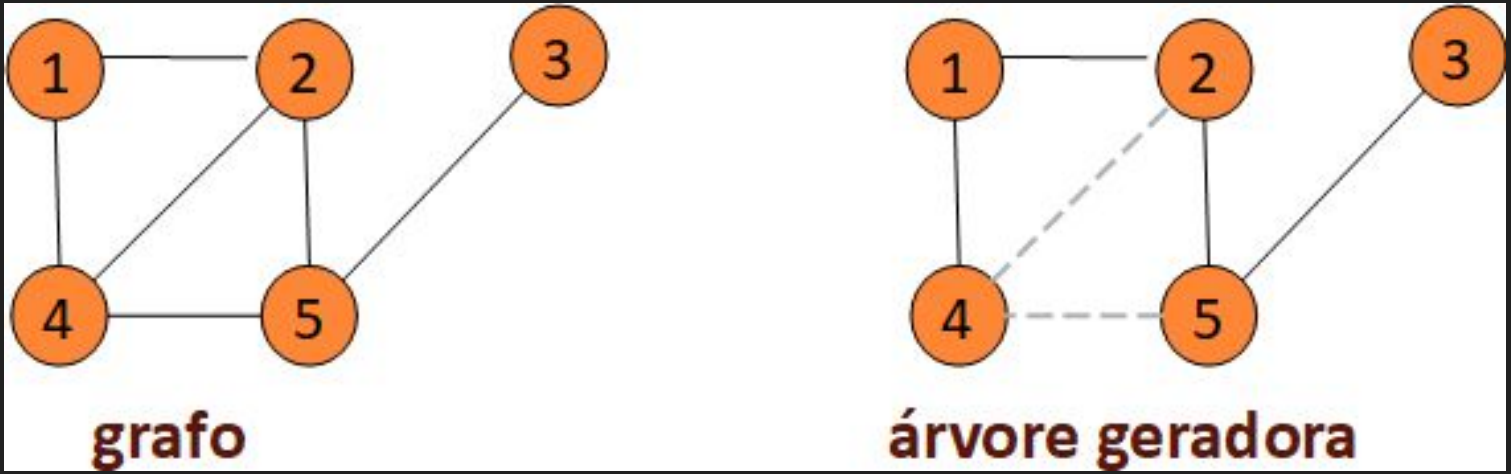
- Uma floresta é um grafo acíclico.
  - Toda árvore é uma floresta
  - Os componentes conexos de uma floresta são árvores.

Árvore e Floresta

- Uma árvore geradora (spanning tree) de um grafo conexo é um subgrafo gerador que é uma árvore.
  - Pode haver mais de uma árvore geradora (a menos que o grafo seja uma árvore)
  - A árvore geradora mínima (minimum spanning tree) é a árvore geradora com menor soma de pesos de arestas

Árvore e Floresta

→ Uma floresta geradora de um grafo é um subgrafo gerador que é uma floresta



Árvore Geradora Mínima (MST)

Árvore Geradora Mínima (MST)

- Exemplos de aplicações em que o objetivo seja encontrar a MST?
- Dois algoritmos ("gulosos") bastante conhecidos para encontrar uma MST de um grafo não direcionado conexo:
  - Algoritmo de Prim;
  - Algoritmo de Kruskal.

Algoritmo de Prim

Algoritmo de Prim

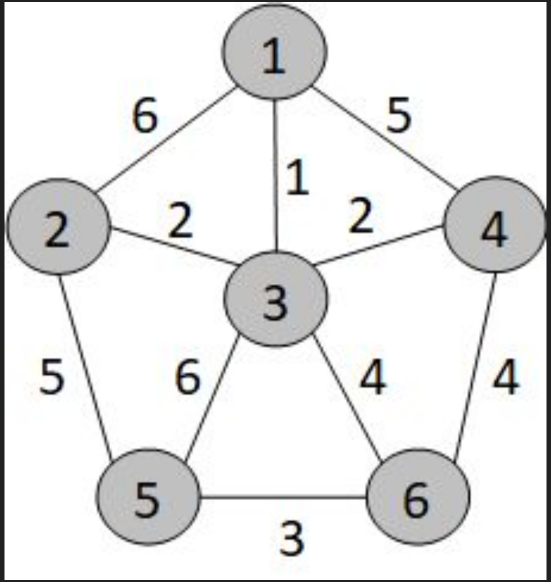
- Ideia geral para o algoritmo de Prim:
  1. Começar com um vértice v qualquer, e adicioná-lo a um conjunto U;
  2. Escolher a aresta de menor peso que conecta um vértice em U a um vértice em V-U;
  3. Incluir o vértice da aresta escolhida em U;

Algoritmo de Prim

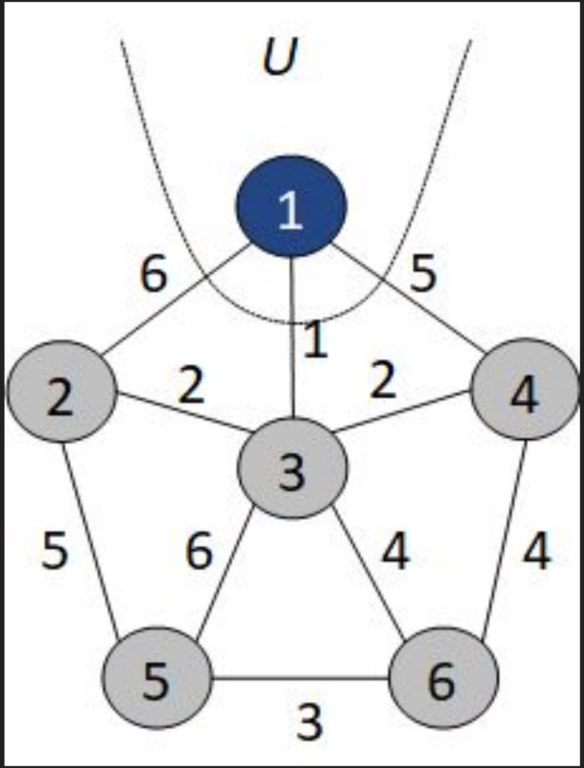
- Ideia geral para o algoritmo de Prim:

4. Incluir a aresta escolhida em um conjunto T;
    - Os vértices em U e as arestas em T sempre formam uma única árvore
5. Voltar para 2 enquanto U != V.

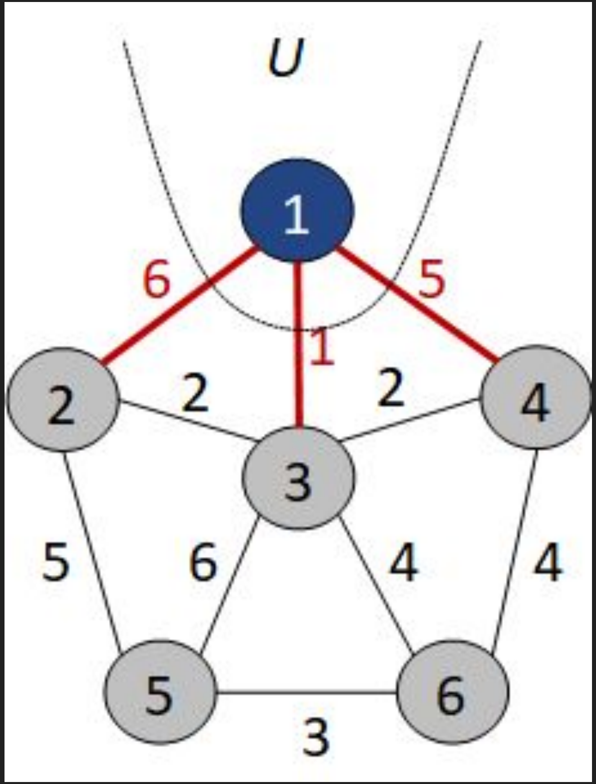
Algoritmo de Prim



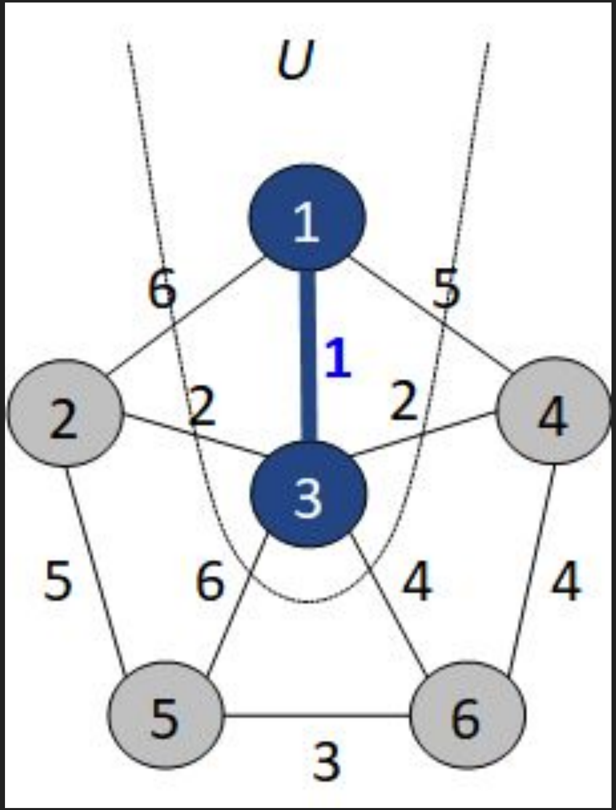
Algoritmo de Prim



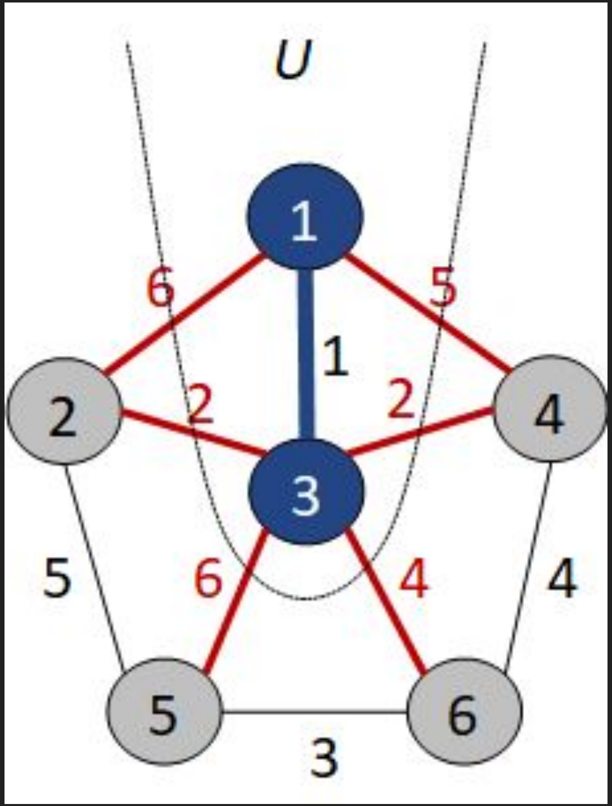
Algoritmo de Prim



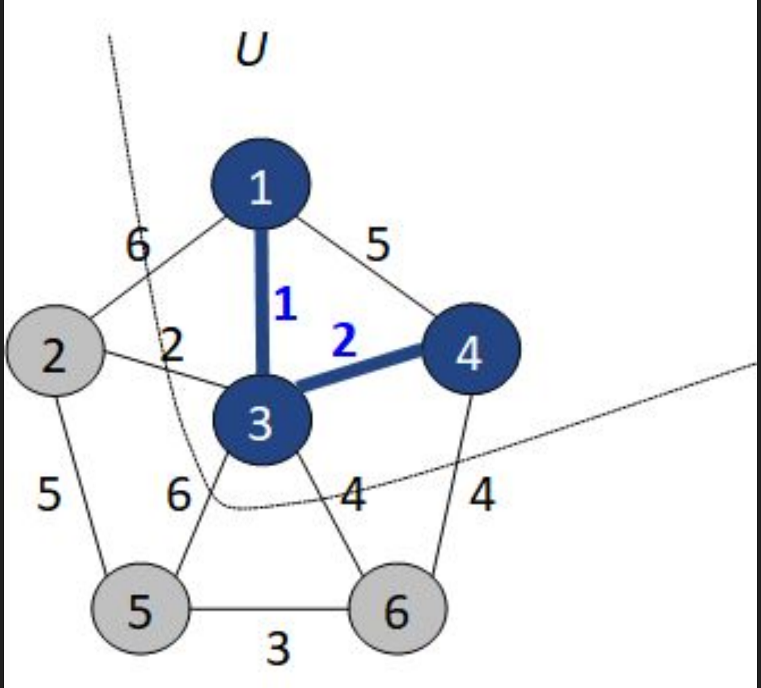
Algoritmo de Prim



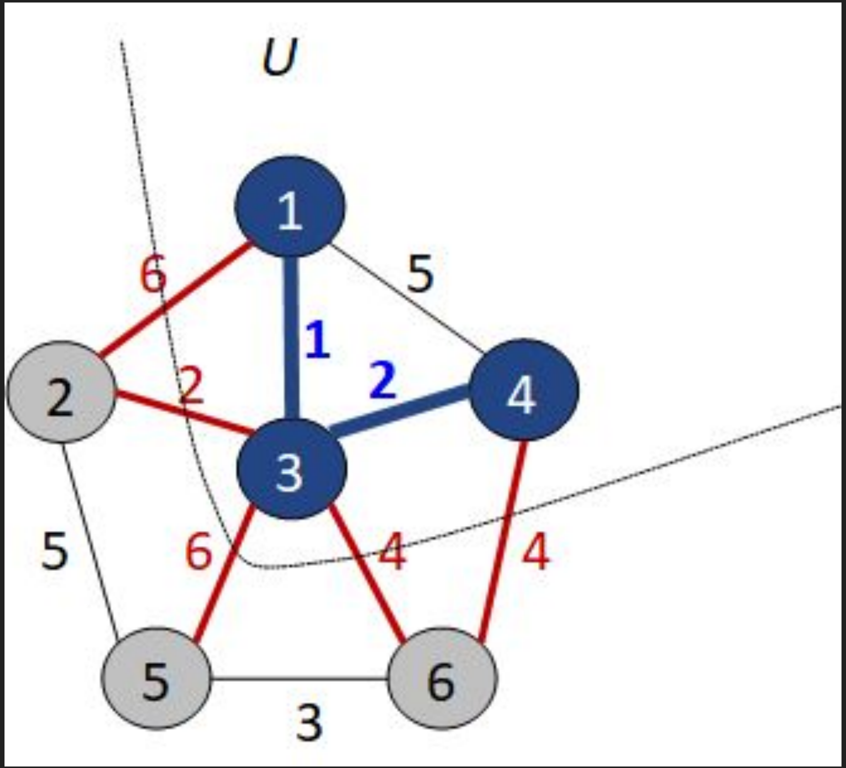
Algoritmo de Prim



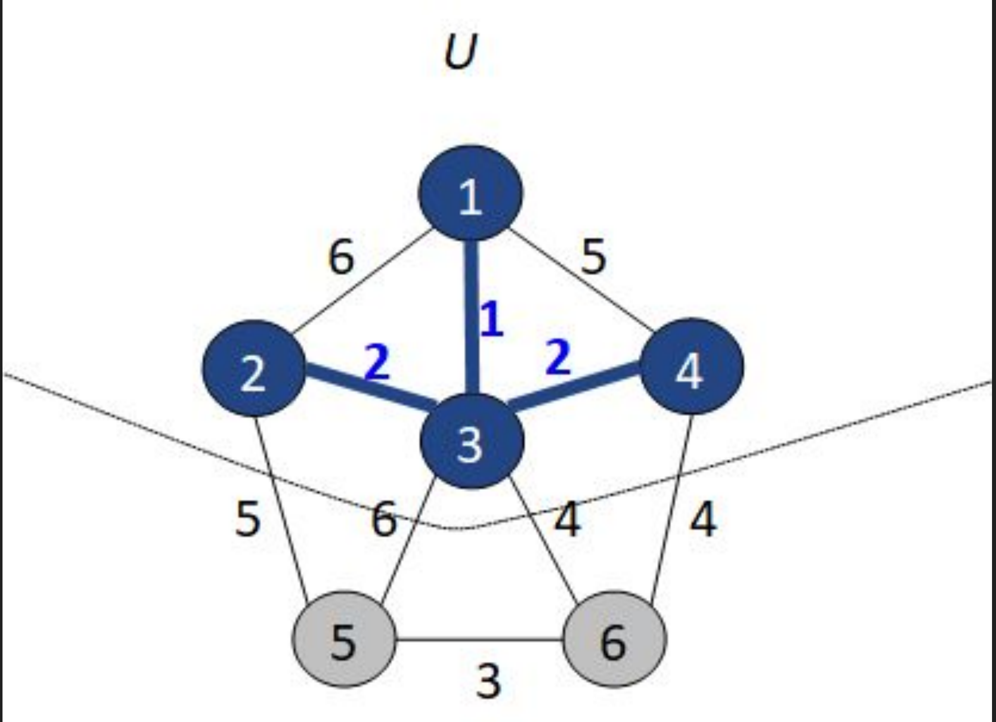
Algoritmo de Prim



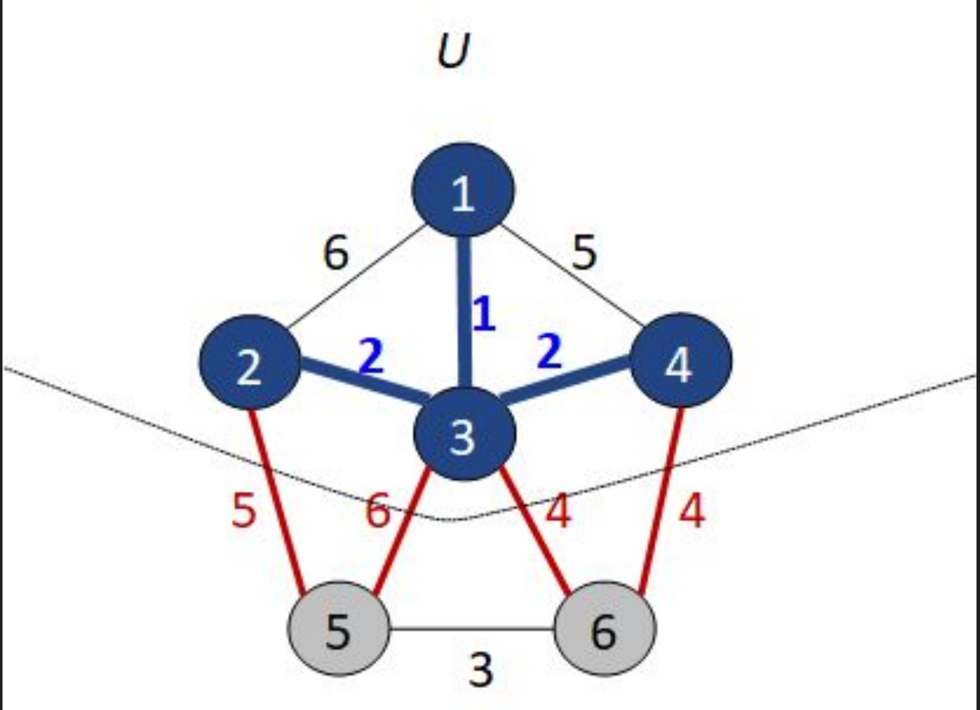
Algoritmo de Prim



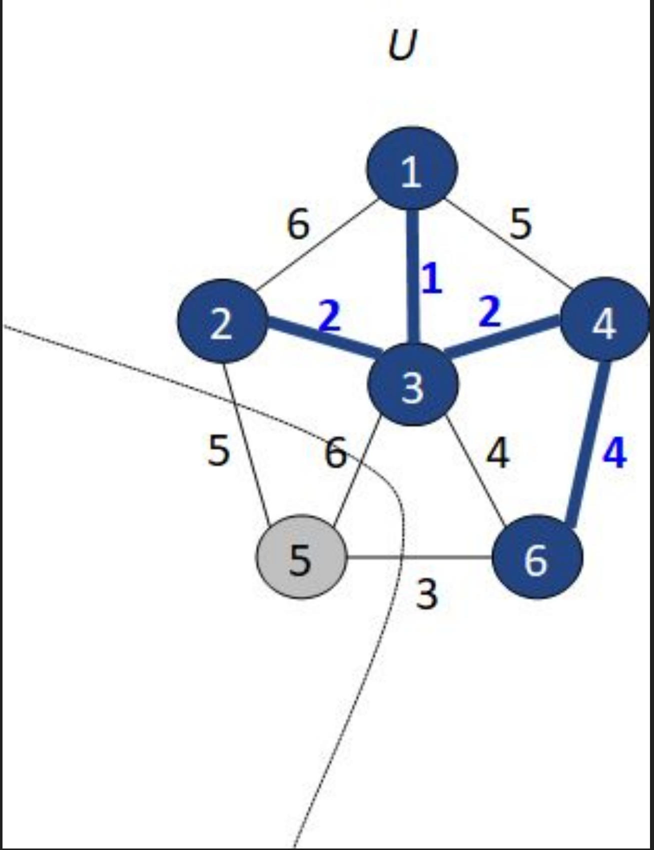
Algoritmo de Prim



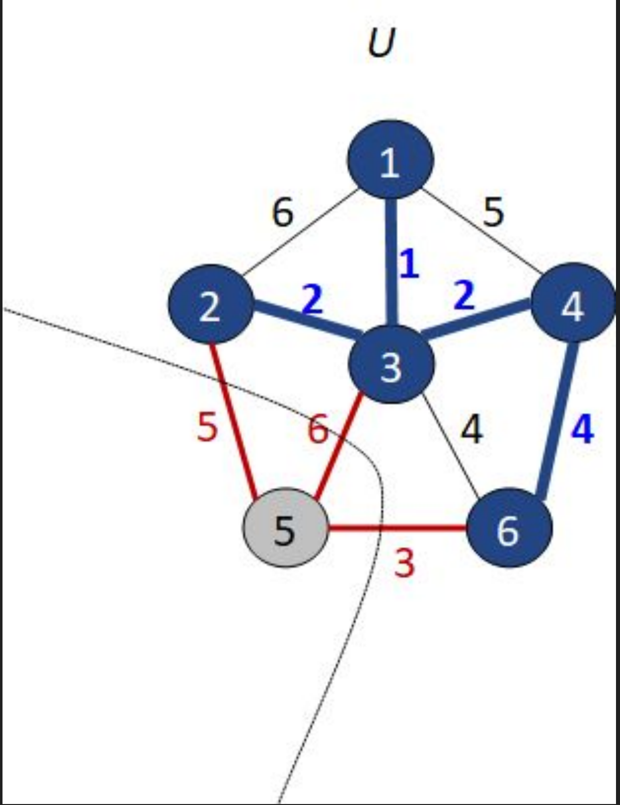
Algoritmo de Prim



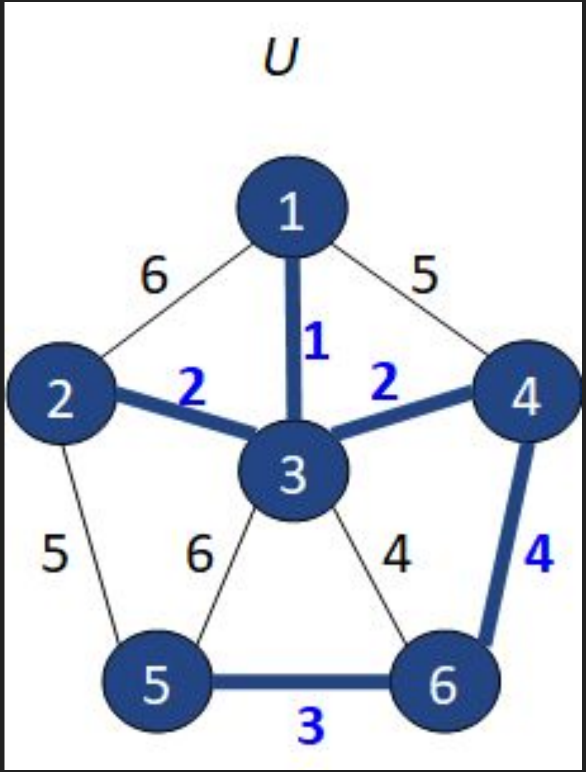
Algoritmo de Prim



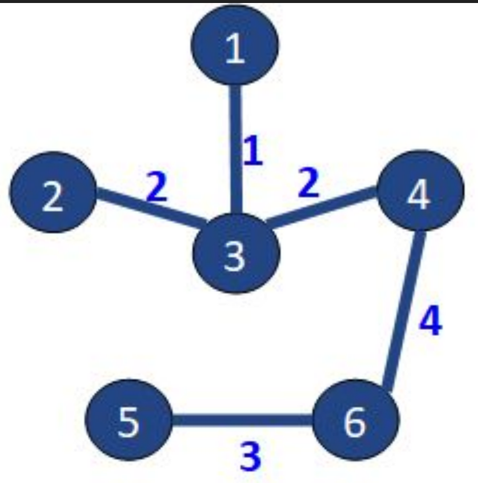
Algoritmo de Prim



Algoritmo de Prim



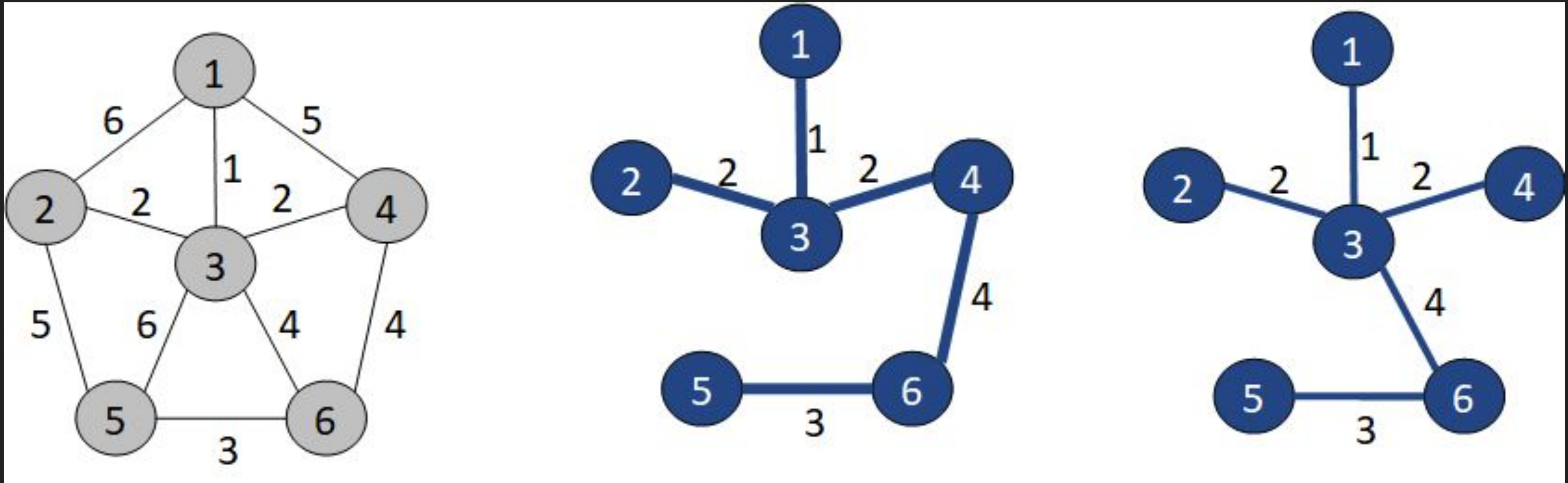
Algoritmo de Prim



pode haver mais de uma MST para um mesmo grafo?

Algoritmo de Prim

→ Sim, pode haver mais de uma 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 := ∅;
      U := {u};
      enquanto U ≠ V faça
      início
          seja (u, v) a aresta de menor peso
                 tal que (u ∈ U) e (v ∈ V−U)
          T := T ∪ {(u, v)};
          U := U ∪ {v};
      fim
fim
```

Algoritmo de Prim

- O desempenho do algoritmo de Prim depende de como será feita a seleção da aresta (u, v) de menor peso.
- Exemplo de implementação:
  - Fila de prioridade para manter os vértices em V-U.
  - Chave da fila de prioridade de um vértice $v \in V\text{-}U$ é o peso da aresta mais leve que liga v a um vértice de U.
  - Se a fila de prioridade for implementada com heap
    - Complexidade do algoritmo é $O(|A| \log |V|)$.

Algoritmo de Kruskal

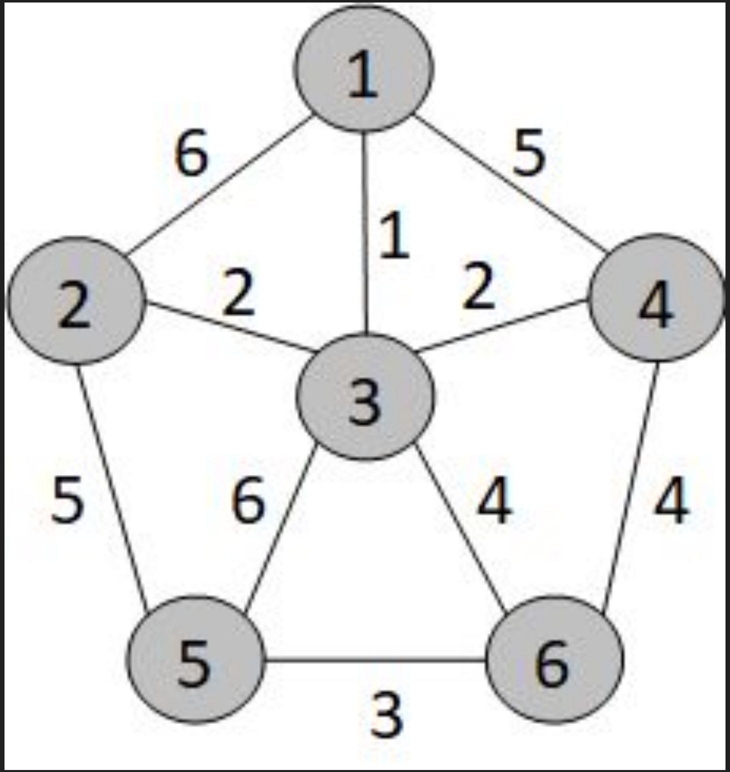
Algoritmo de Kruskal

- Ideia geral para o algoritmo de Kruskal :
  - Inicia-se com um grafo G′ = (V,)
  - Cada vértice é um componente conexo de si mesmo
  - A cada iteração, são construídos componentes conexos cada vez maiores, ou seja: árvores em uma floresta
  - Para “aumentar” os componentes conexos (árvores)
    - Arestas em A são analisadas por ordem ascendente de peso.

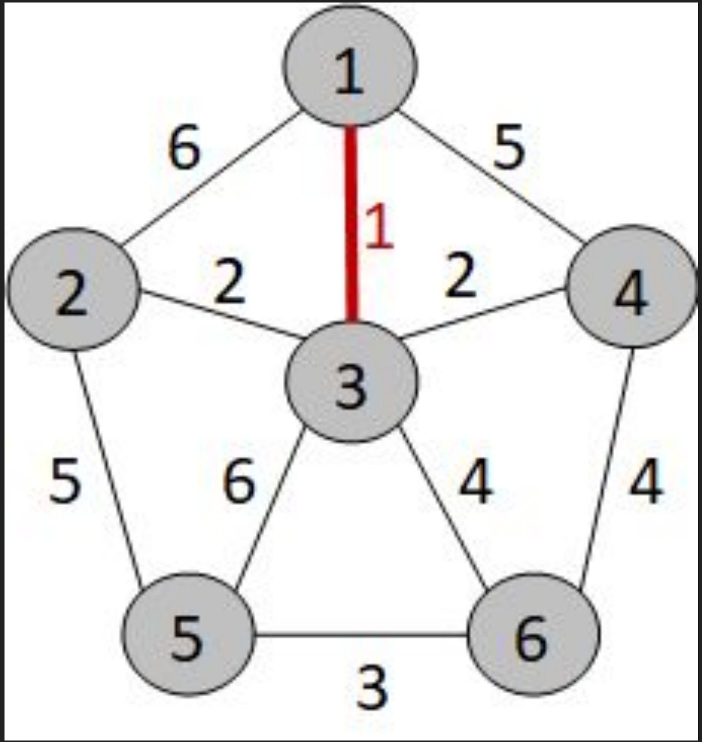
Algoritmo de Kruskal

- Ideia geral (cont.):
  - Se a aresta conecta dois vértices em dois componentes (árvores) separados
    - A aresta é adicionada a T.
  - Se a aresta conecta dois vértices do mesmo componente (árvore)
    - Ela é descartada, pois criaria um ciclo.

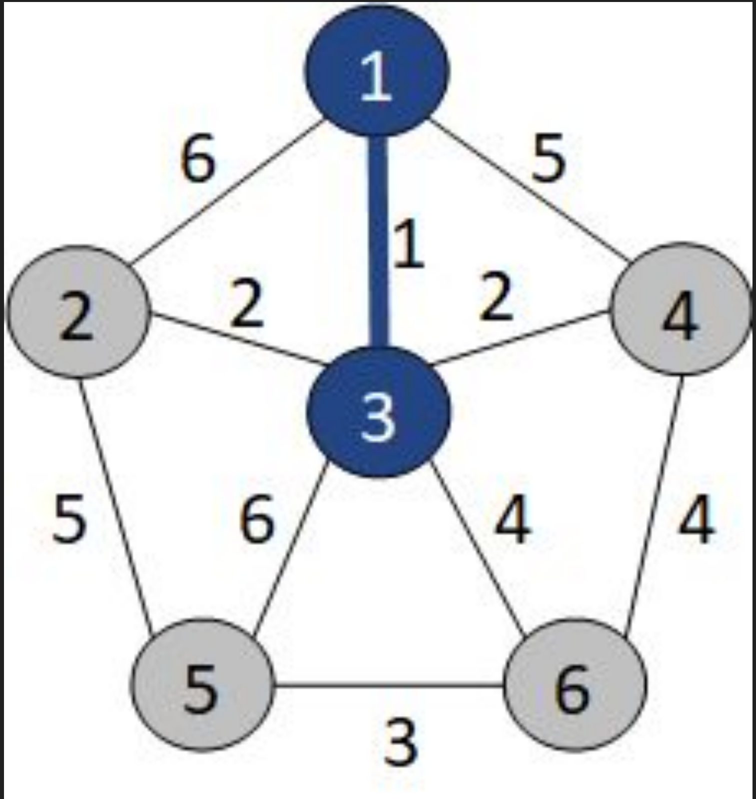
Algoritmo de Kruskal



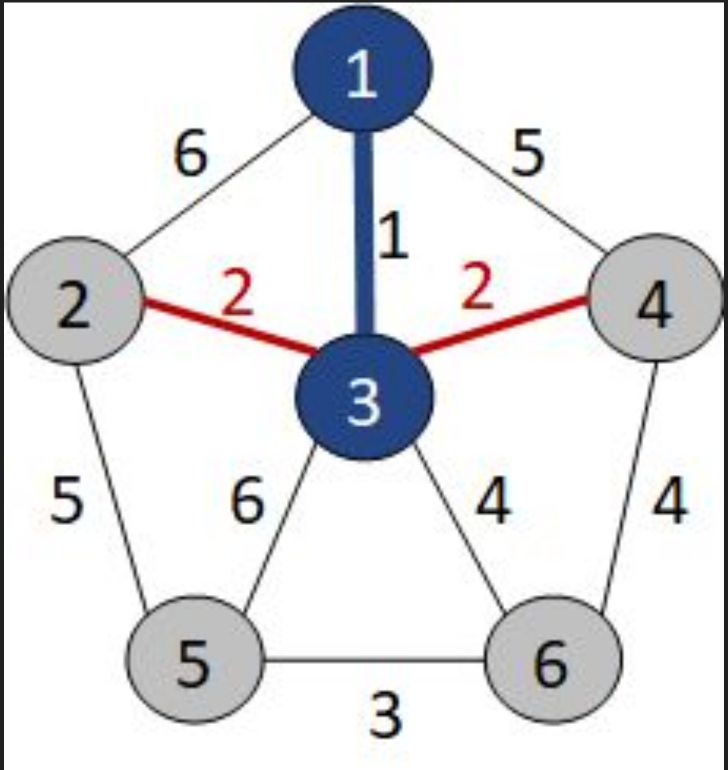
Algoritmo de Kruskal



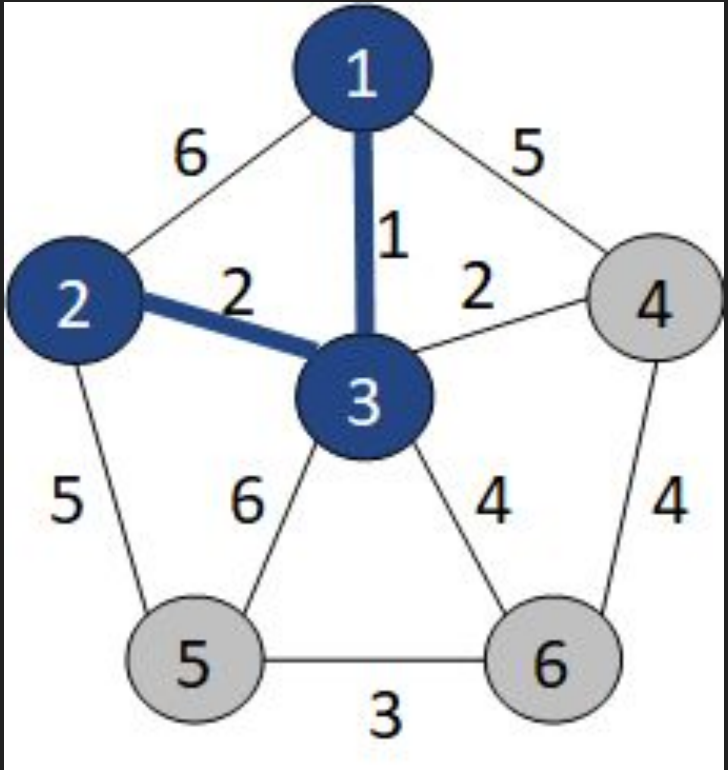
Algoritmo de Kruskal



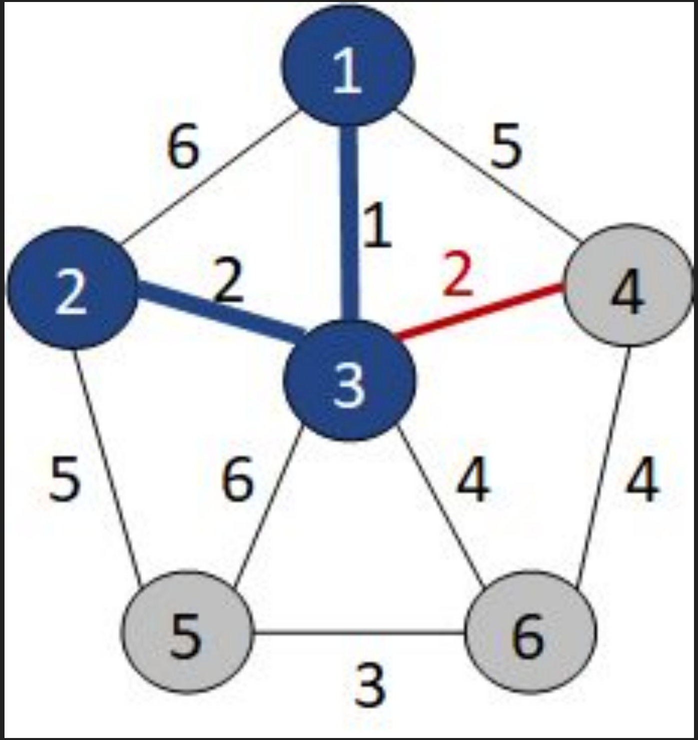
Algoritmo de Kruskal



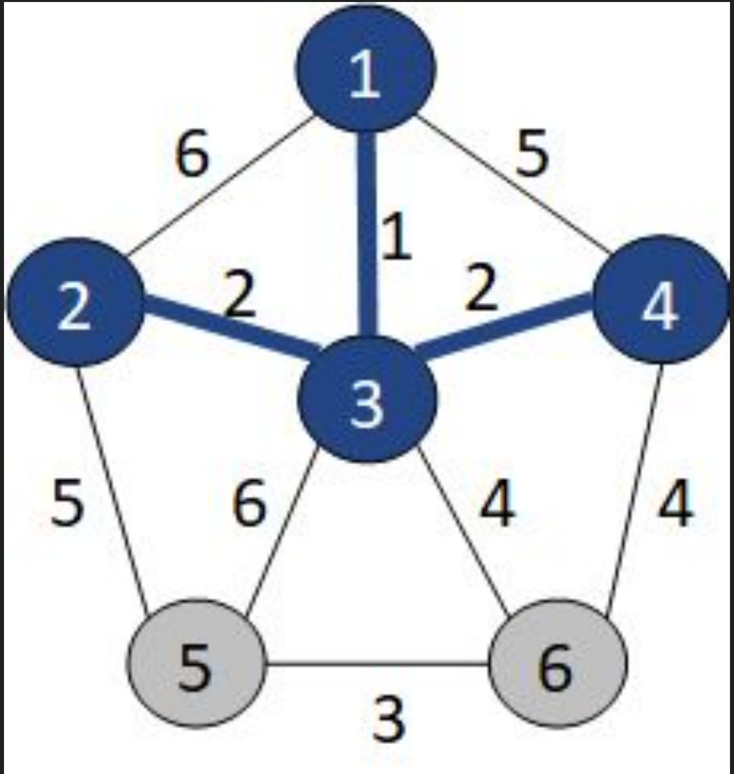
Algoritmo de Kruskal



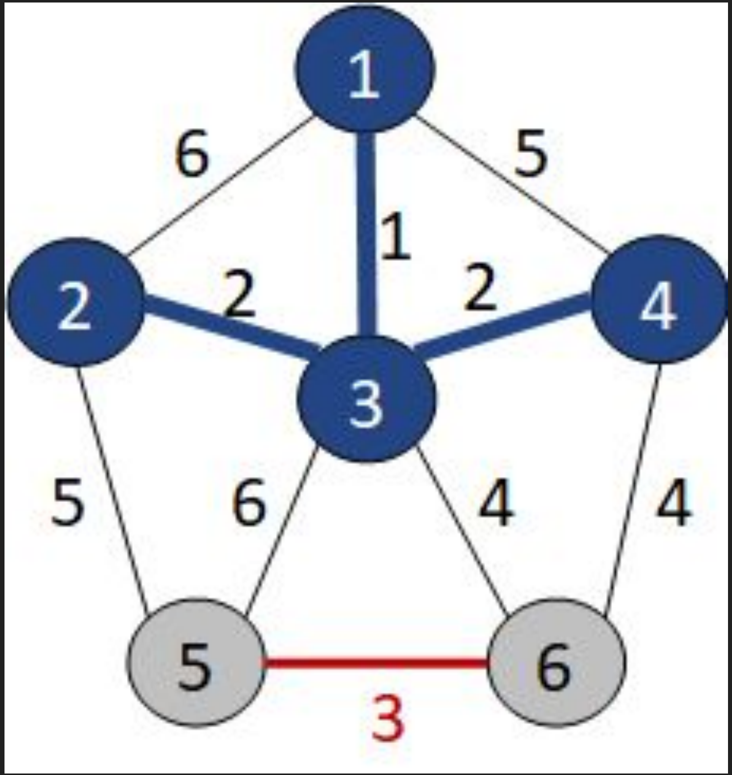
Algoritmo de Kruskal



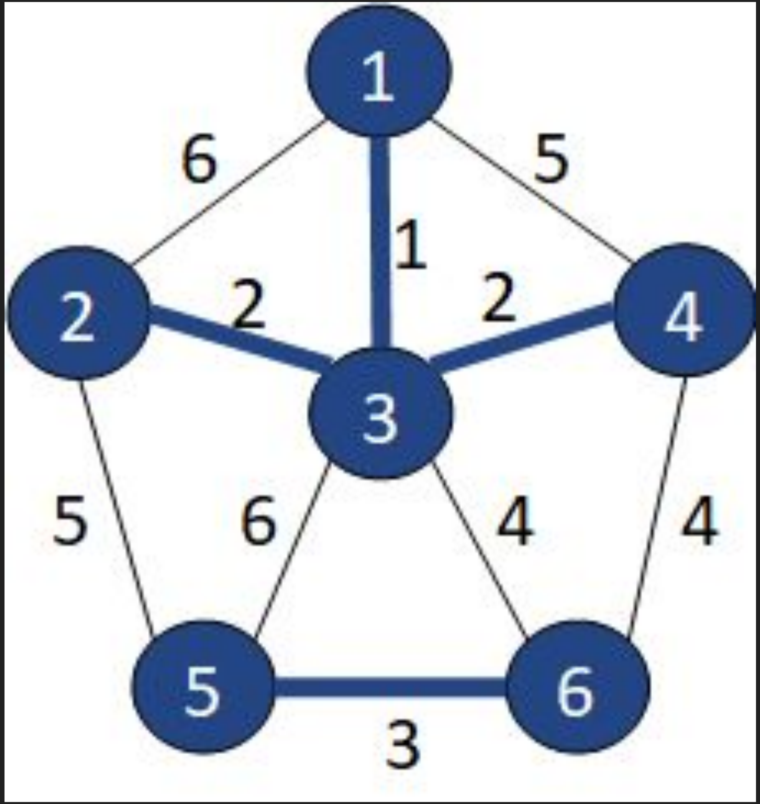
Algoritmo de Kruskal



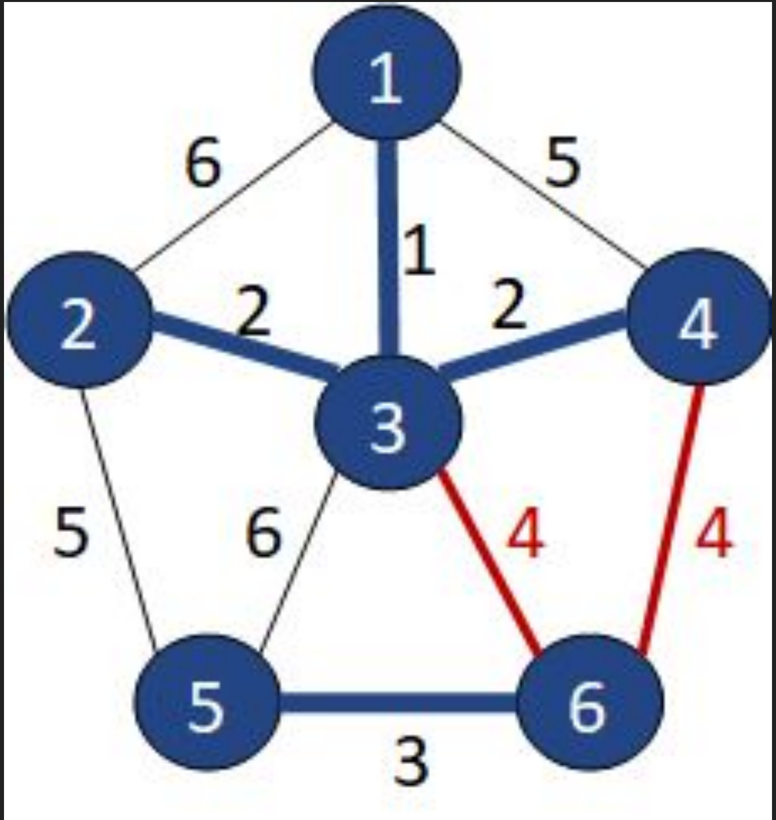
Algoritmo de Kruskal



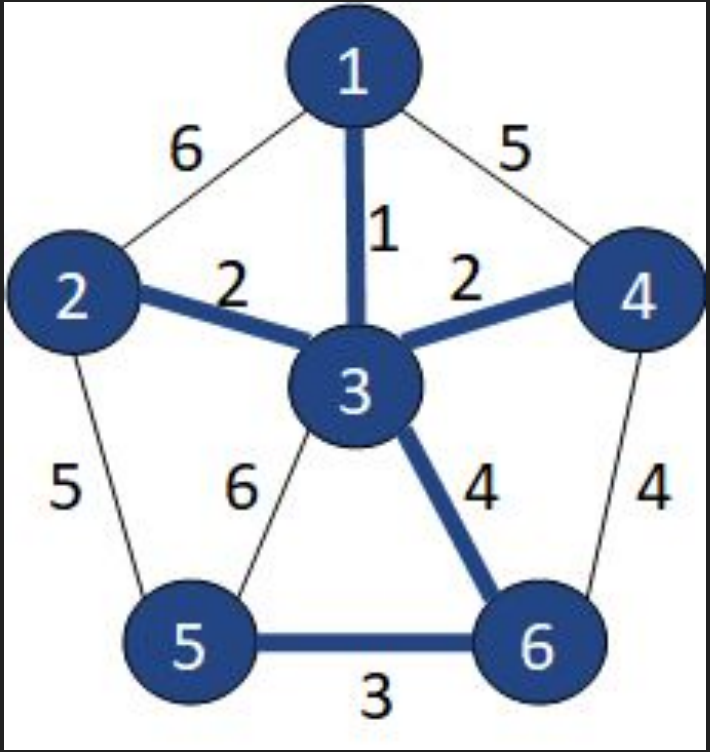
Algoritmo de Kruskal



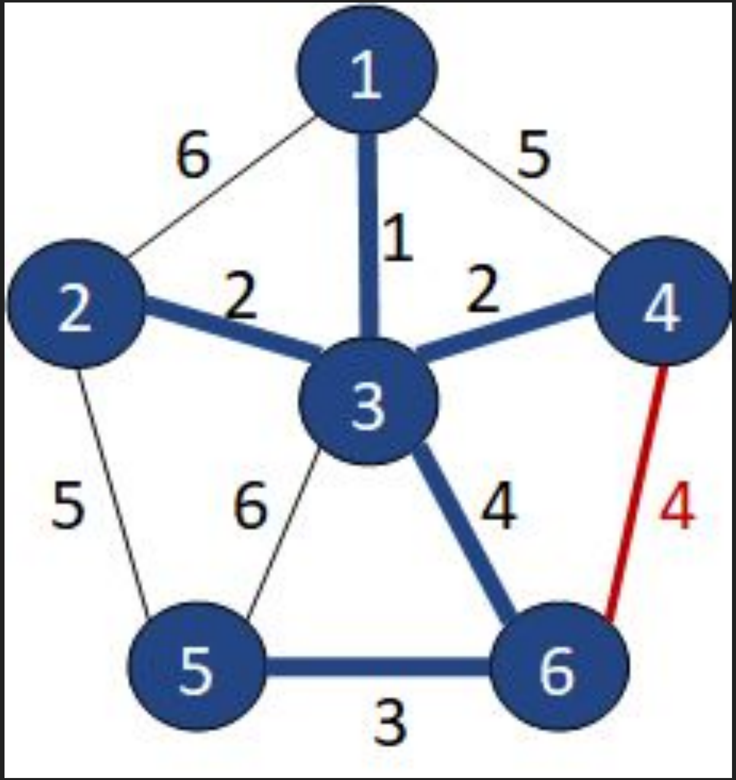
Algoritmo de Kruskal



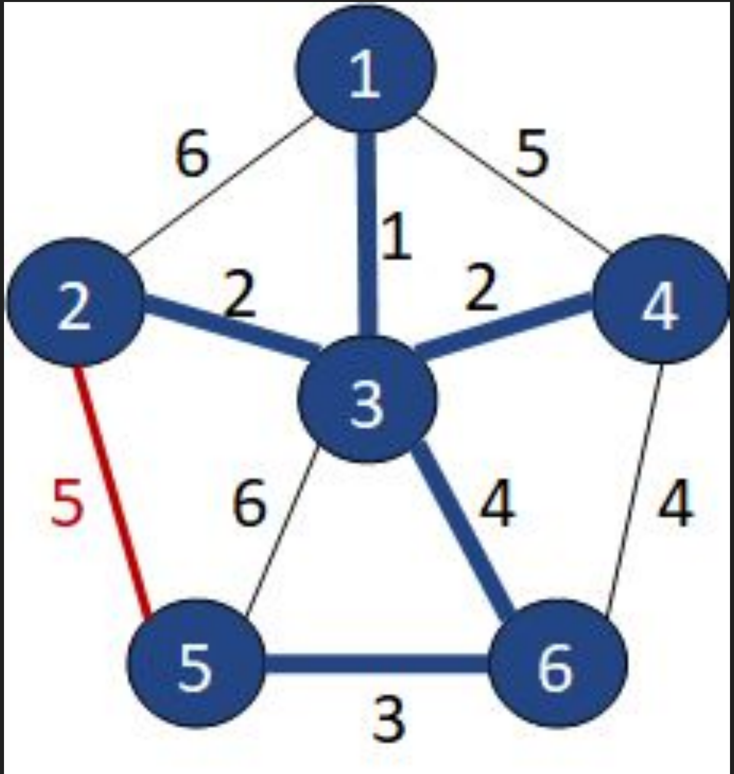
Algoritmo de Kruskal



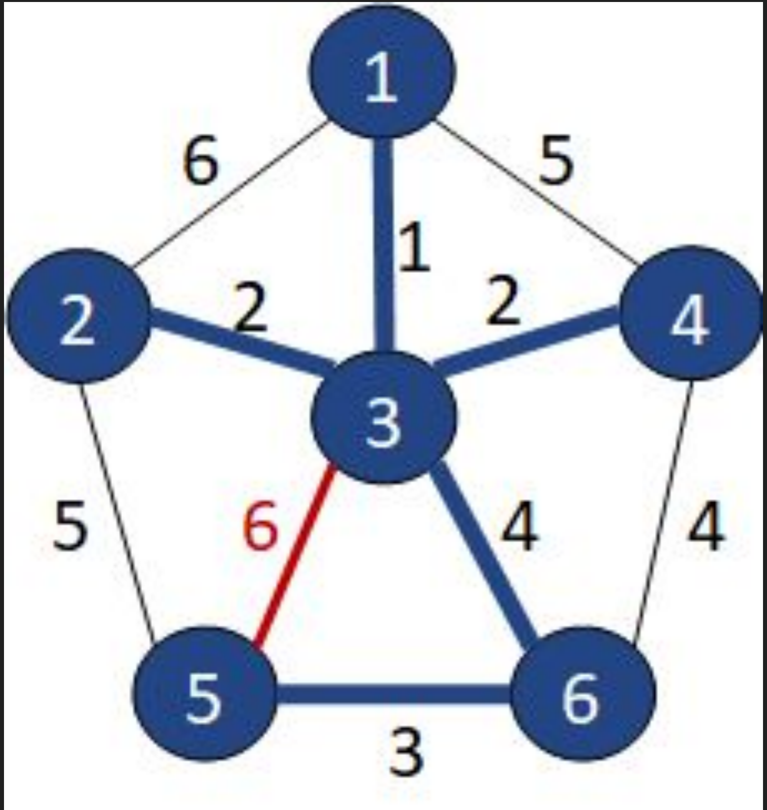
Algoritmo de Kruskal



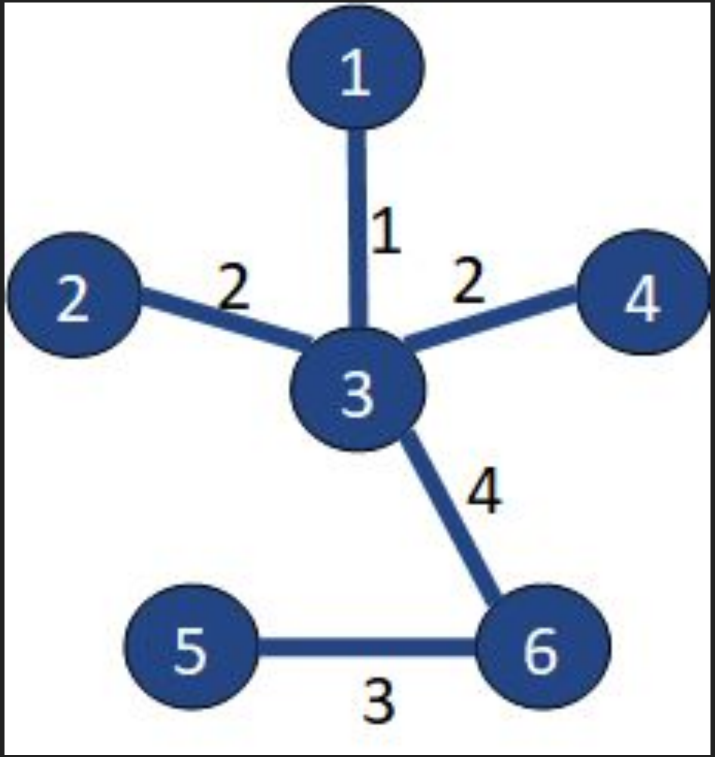
Algoritmo de Kruskal



Algoritmo de Kruskal



Algoritmo de Kruskal



Algoritmo de Kruskal

```
procedimento Kruskal(var Grafo: TGrafo;
        var T: conjunto de arestas)
variáveis
  u, v: TVertice;
  U_1,...,U_n: 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
  U_i := {i};
  enquanto houver arestas em Q faça
  início
  seja (u, v) a aresta de menor peso de Q tal que
  (u ∈ U_p) e (v ∈ U_q) e (U_p ∩ U_q)= ∅
  T := T ∪ {(u, v)};
  U_p := U_p ∪ U_q;
  eliminar U_q;
  fim
fim
```

Algoritmo de Kruskal

- O desempenho algoritmo de Kruskal depende de dois fatores principais:
  - Encontrar a aresta de menor peso;
  - Verificar se a aresta conecta dois componentes distintos.

Algoritmo de Kruskal

- Soluções eficientes...
  - Q implementada como uma fila de prioridade com heap
  - Operações eficientes de conjuntos distintos
    - Estruturas de dados para conjuntos distintos
    - Ex: florestas de conjuntos distintos
      - Cada árvore representando um conjunto
  - Algoritmo é O(|A| log |A|).

Referências

➔ WIRTH,N. Algorithms and Data Structures, Englewood Cliffs, Prentice-Hall, 1986.

➔ CORMEN, H.T.; LEISERSON, C.E.; RIVEST, R.L. Introduction to Algorithms, MIT Press, McGraw-Hill, 1999.

➔ ZIVIANI, N. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.

➔ SZWARCFITER,J.L. Grafos e Algoritmos Computacionais. Editora Campus, 1983.

➔ Van Steen, Maarten. "Graph theory and complex networks." An introduction 144 (2010).

➔ Gross, Jonathan L., and Jay Yellen. Graph theory and its applications. CRC press, 2005.

➔ Barabási, A.-L., Pósfai, M. (2016). Network science. Cambridge: Cambridge University Press. ISBN: 9781107076266 1107076269