Projeto e Análise de Algoritmos
2024.2

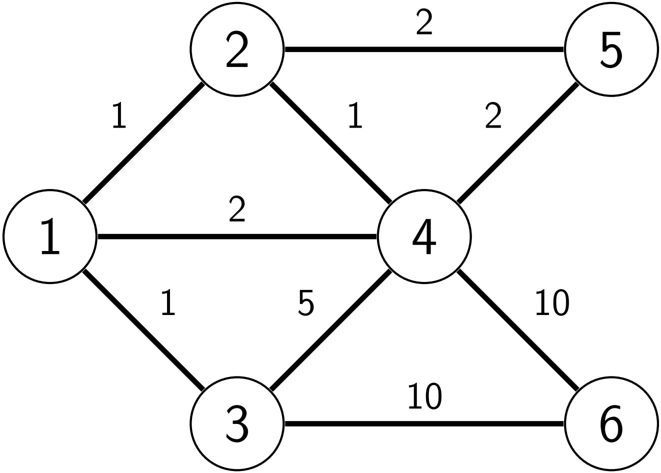
Árvore Geradora Mínima

Prof. Marcio Costa Santos DCC/UFMG

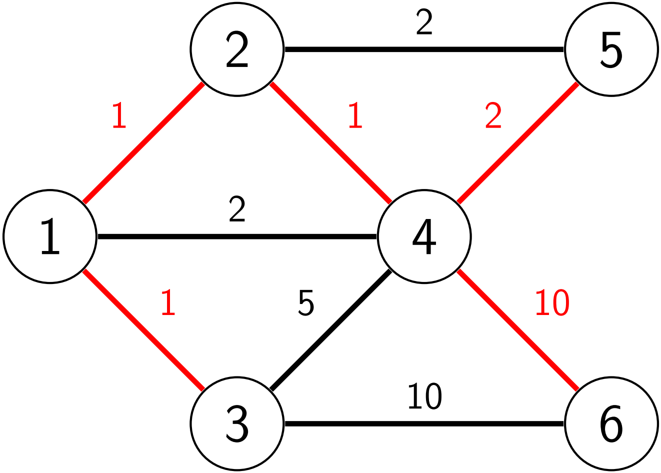
Árvore Geradora Mínima

- Dado um grafo $G = (V, E)$ não direcionado com pesos w_{uv} em suas arestas.
- Defina o peso de um subgrafo de G como sendo a soma do pesos de suas arestas.
- Queremos encontrar o subgrafo de G mais leve que é uma árvore e que contém todos os vértices de G.

Árvore Geradora Mínima - Exemplo



Árvore Geradora Mínima - Exemplo



Árvore Geradora Mínima - Kruskal

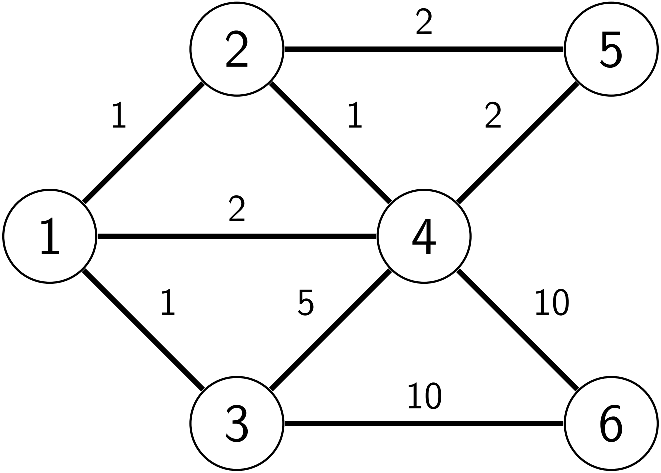
- Uma das aresta mais leves de um grafo sempre está na sua árvore geradora mínima.
- Podemos usar essa observação para incluir arestas na árvore de forma simples.
- Vamos tentar colocar todas as arestas de menor peso na árvore.

Kruskal - Algoritmo

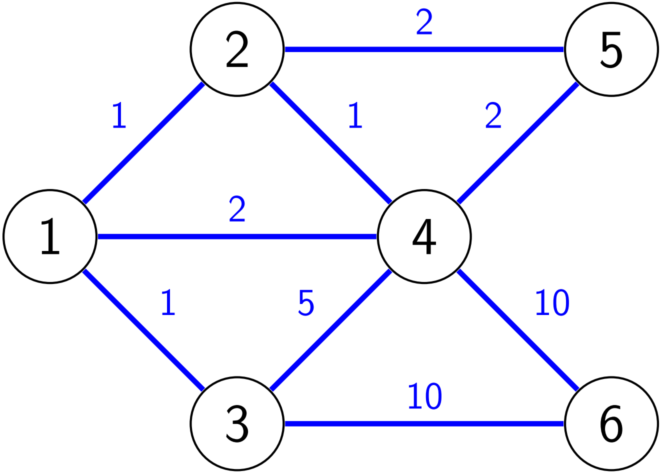
$A \leftarrow \emptyset$;
$E \leftarrow E(G)$ ordenadas;
para todo $uv \in E$ *em ordem* **faça**
    se $uv + A$ *não tem um ciclo* **então**
        $A \leftarrow A \cup \{uv\}$;
    fim
fim

Algoritmo 1: KRUSKAL(G)

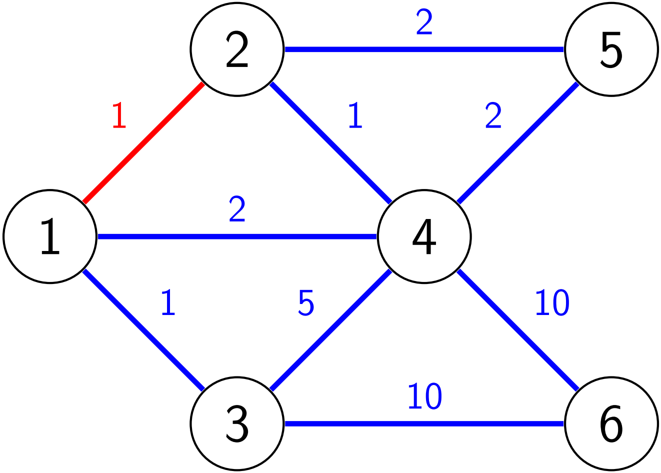
Kruskal - Exemplo



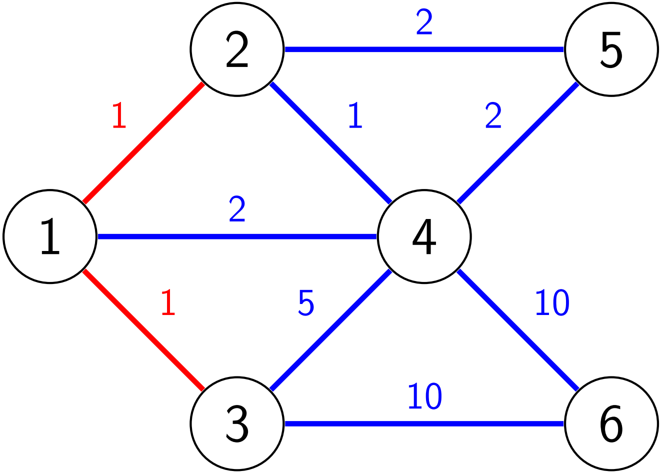
Kruskal - Exemplo



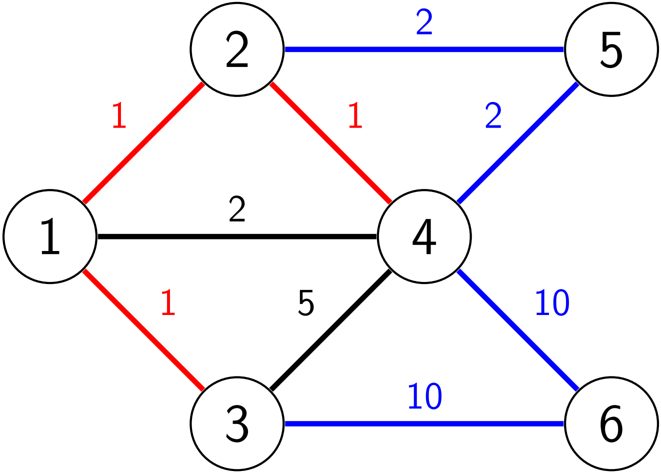
Kruskal - Exemplo



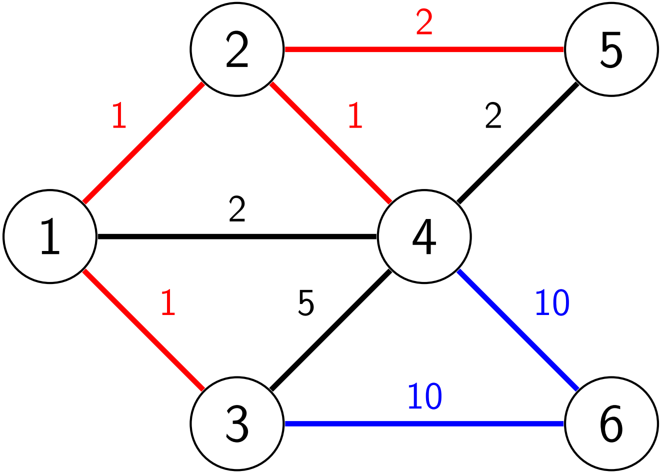
Kruskal - Exemplo



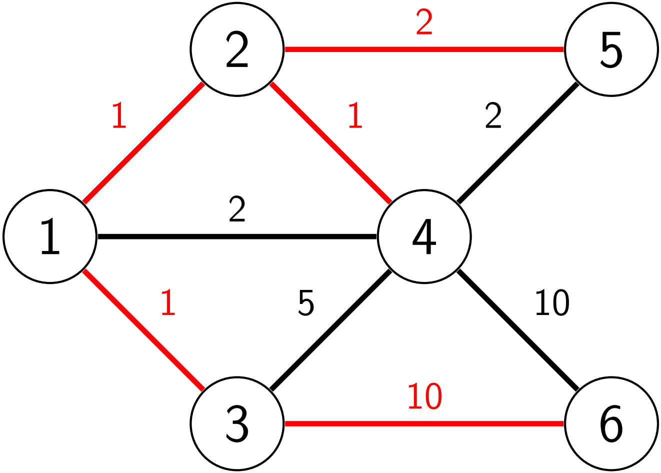
Kruskal - Exemplo



Kruskal - Exemplo



Kruskal - Exemplo



Complexidade

- Assuma que verificar se $uv + A$ contém um ciclo tem complexidade de $O(X)$.
- Incluir uv em A tem complexidade de $O(Y)$.
- O algoritmo tem complexidade de
  - Matriz de Adjacências: $O(n^2 \log n + n^2.X + n.Y)$.
  - Lista de Adjacências: $O(m \log n + m.X + n.Y)$.

Complexidade - Solução Simples

- Verificar se $uv + A$ contém um ciclo tem complexidade de $O(n)$ (DFS).
- Incluir uv em A tem complexidade de $O(1)$.
- O algoritmo tem complexidade de
    - Matriz de Adjacências: $O(n^2 \log n + n^2.n + n.1) = O(n^3)$.
    - Lista de Adjacências: $O(m \log n + m.n + n.1) = O(mn)$.

Complexidade - Solução Complicada

- Usando *UNION-FIND*
- Verificar se $uv + A$ contém um ciclo tem complexidade de $O(1)$(-sh, na verdade menor que $\log n$).
- Incluir uv em A tem complexidade de $O(1)$.
- O algoritmo tem complexidade de
  - Matriz de Adjacências: $O(n^2 \log n + n^2.1 + n.1) = O(n^2 \log n)$.
  - Lista de Adjacências: $O(m \log n + m.1 + n.1) = O(m \log n)$.

Árvore Geradora Mínima - Prim

- Uma das aresta mais leves de um grafo sempre está na sua árvore geradora mínima.
- As demais arestas devem formar uma árvore.
- Vamos tentar construir essa árvore.

Prim - Algoritmo

para todo $u \in V(G)$ **faça**

  $c[u] \leftarrow \infty$;

  $\pi[u] \leftarrow u$;

fim

Selecione a menor aresta uv;

$c[u] \leftarrow 0$;

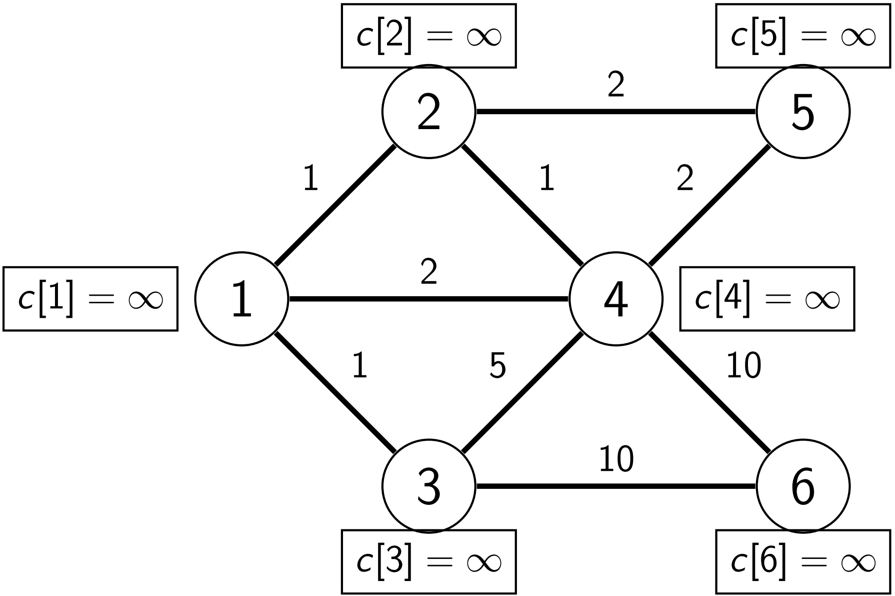
$Q \leftarrow V(G)$;

Algoritmo 2: PRIM(G)

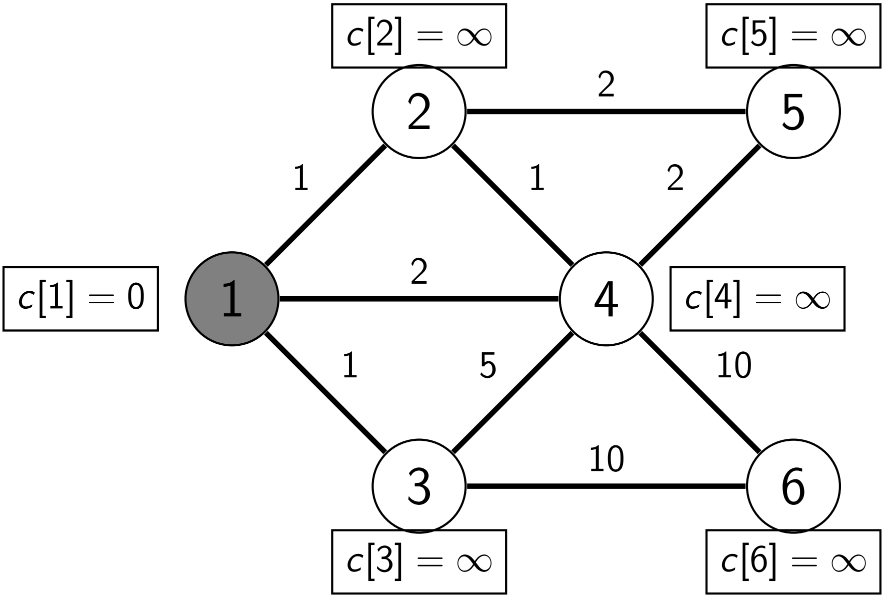
Prim - Algoritmo

enquanto $Q \neq \emptyset$ **faça**
    $u \leftarrow$ `mira_minimo(Q)`;
    para todo $v \in N(u)$ **faça**
        se $v \in Q$ e $w_{uv} < c[v]$ **então**
            $\pi[v] \leftarrow u$;
            $c[v] \leftarrow w_{uv}$;
        fim
    fim
fim
´

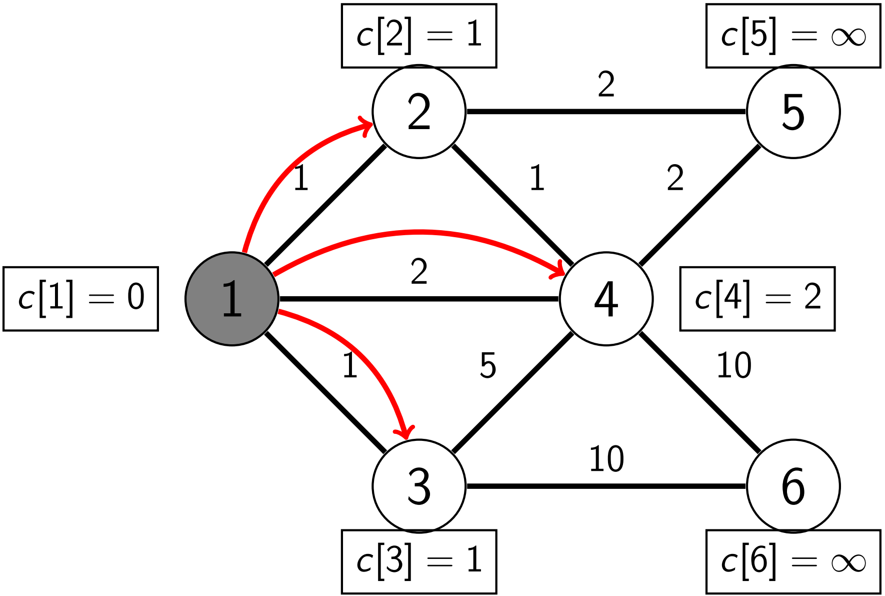
Prim - Exemplo



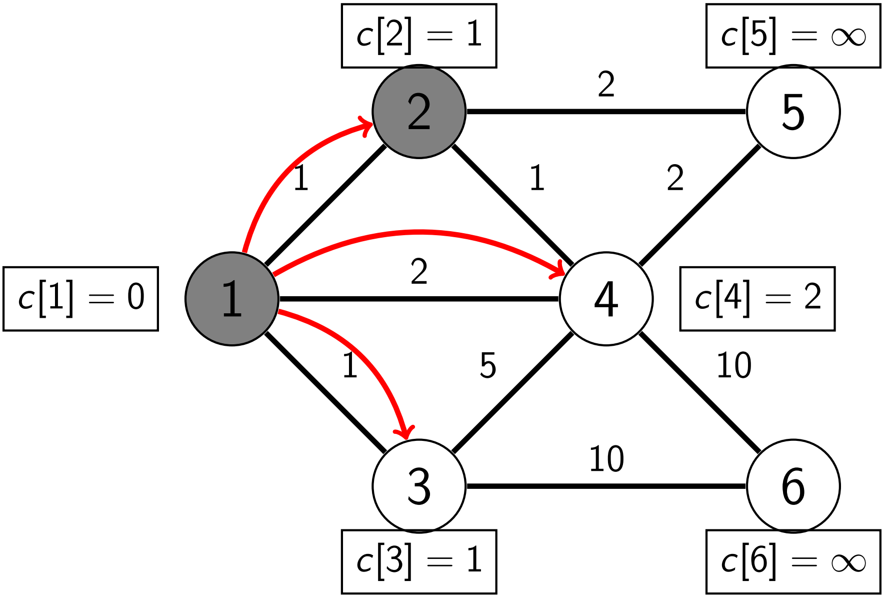
Prim - Exemplo



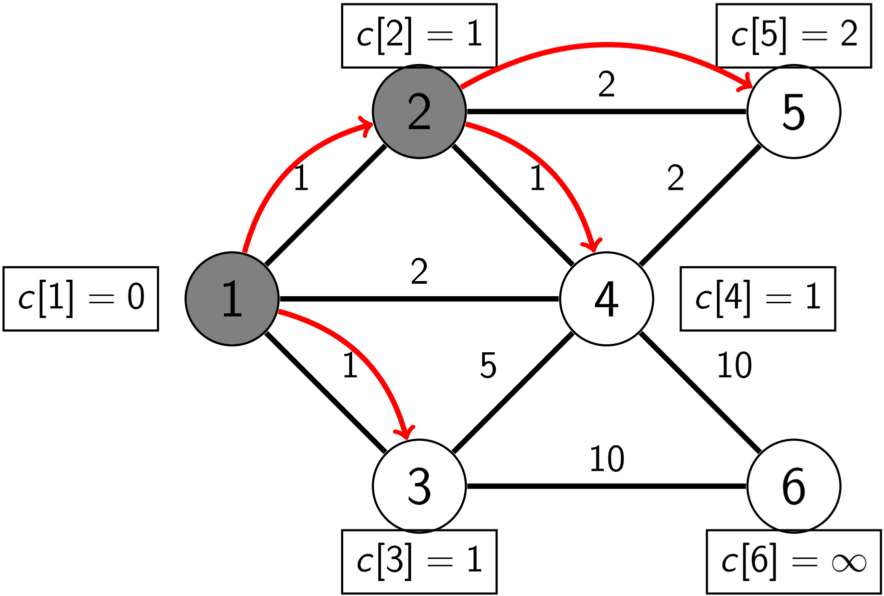
Prim - Exemplo



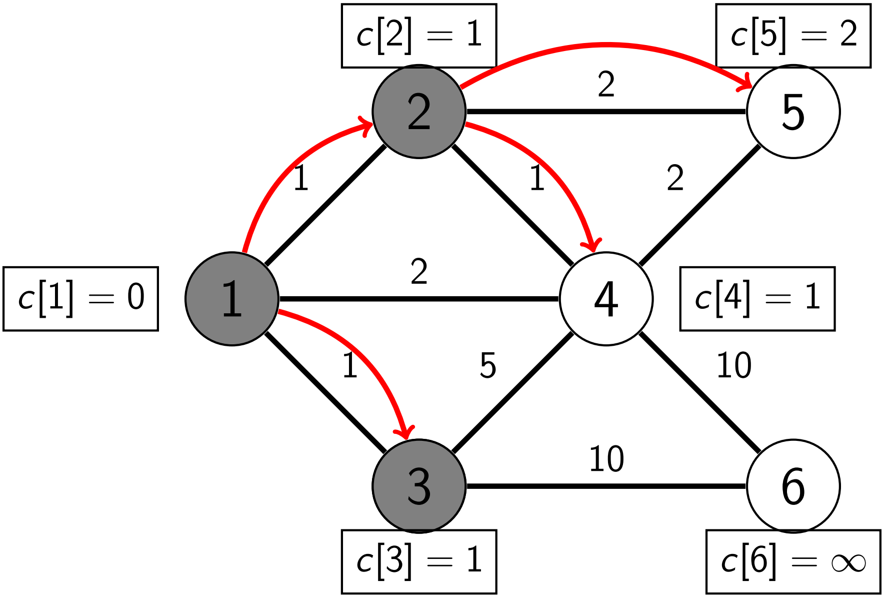
Prim - Exemplo



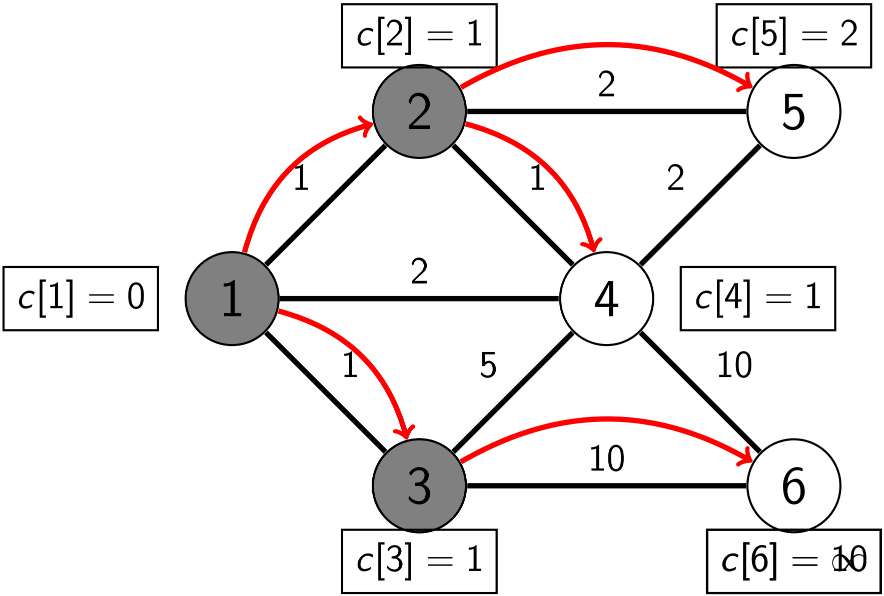
Prim - Exemplo



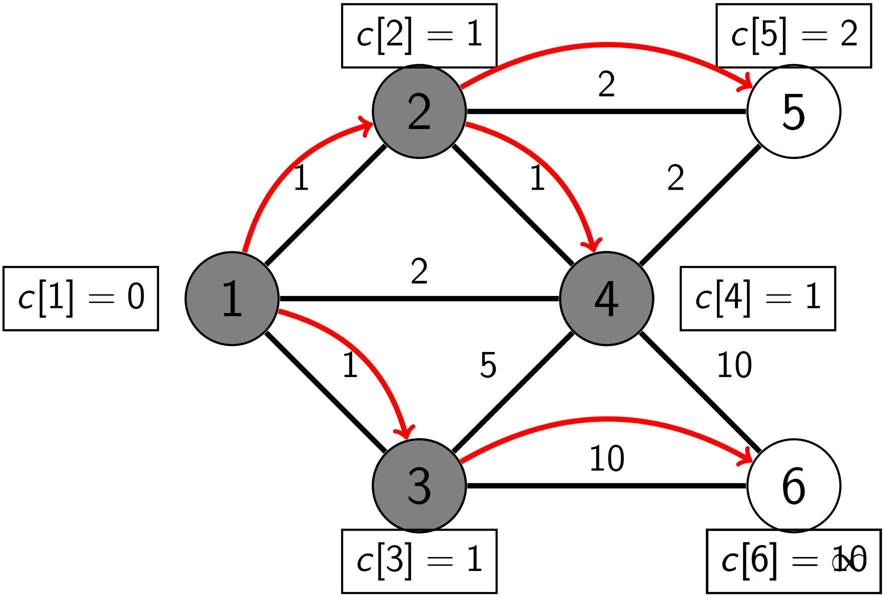
Prim - Exemplo



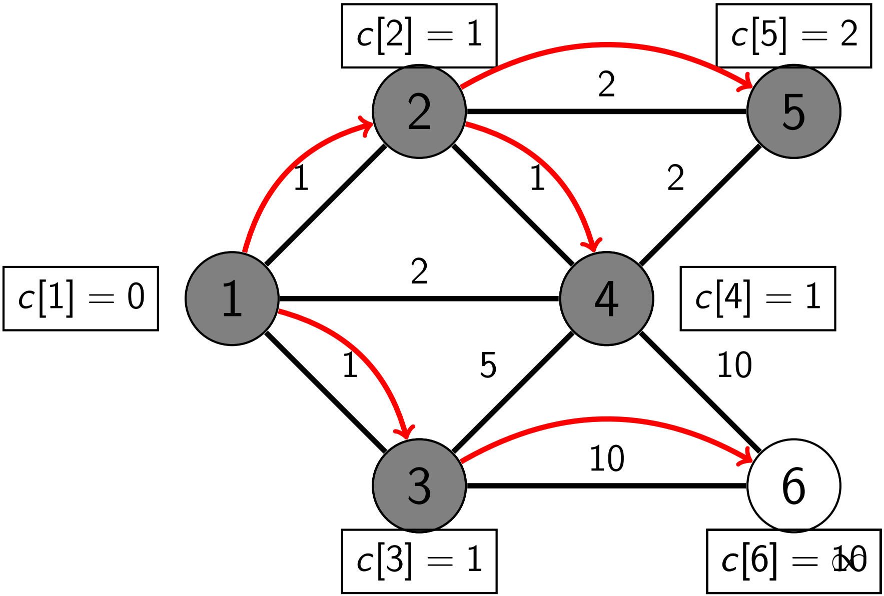
Prim - Exemplo



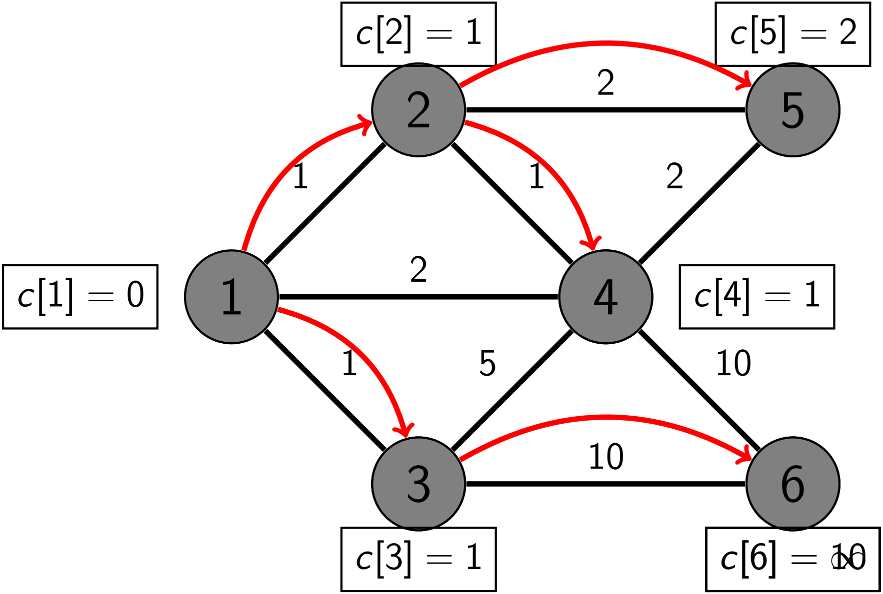
Prim - Exemplo



Prim - Exemplo



Prim - Exemplo



Complexidade - Lista Ordenada

- Ordenar os vértices por $c[v] = O(n \log n)$.
- Extrair o mínimo $= O(1)$.
- Re-ordenar após mudança $= O(n)$.
- Buscar um elemento $= O(n)$
- O algoritmo tem complexidade de
  - Matriz de Adjacências:
    $O(n + n \log n + n.1 + n^2.n) = O(n^3)$.
  - Lista de Adjacências:
    $O(n + n \log n + n.1 + m.n) = O(mn)$.

Complexidade - Heap

- Ordenar os vértices por $c[v] = O(n)$.
- Extrair o mínimo $= O(1)$.
- Re-ordenar após mudança $= O(\log n)$.
- Buscar um elemento $= O(\log n)$
- O algoritmo tem complexidade de
  - Matriz de Adjacências: $O(n + n + n.1 + n^2. \log n) = O(n^2 \log n)$.
  - Lista de Adjacências: $O(n + n + n.1 + m. \log n) = O(m \log n)$.

Complexidade - Heap de Fibonacci

- Ordenar os vértices por $c[v] = O(n)$.
- Extrair o mínimo $= O(\log n)$.
- Re-ordenar após mudança $= O(1)$.
- Buscar um elemento $= O(1)$
- O algoritmo tem complexidade de
  - Matriz de Adjacências:
    $O(n + n + n.\log n + n^2.1) = O(n^2)$.
  - Lista de Adjacências:
    $O(n + n + n.\log n + m.1) = O(m + n \log n)$.