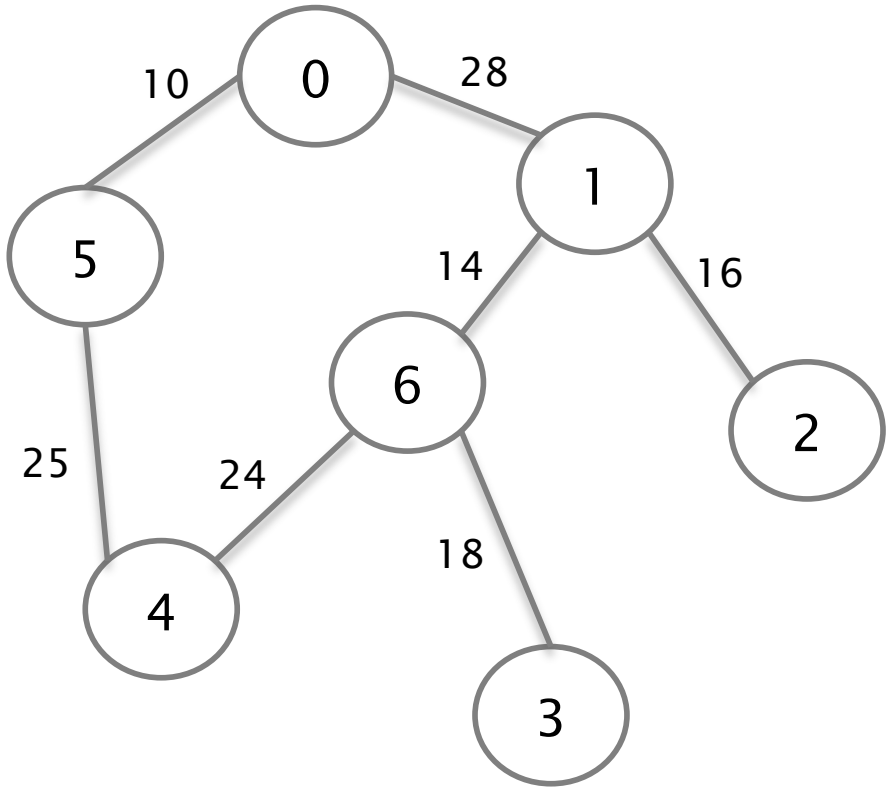
INF 1010
Estruturas de Dados Avançadas

Grafos

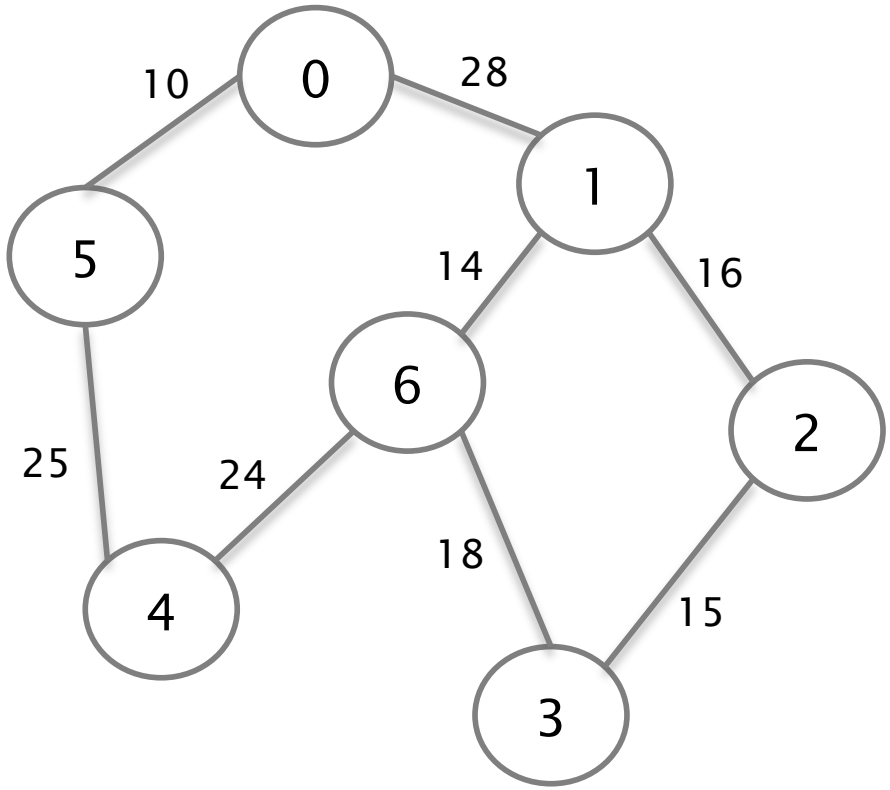
problemas comuns

- caminhos mais curtos
- árvores geradoras



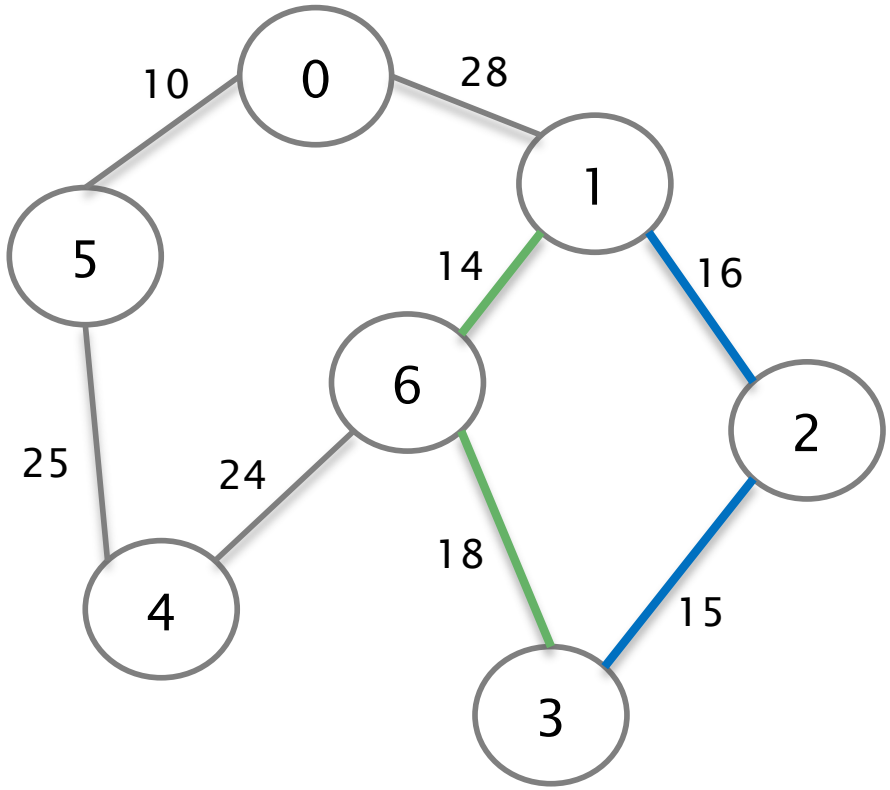
problemas comuns

- caminho mais curto
  - caminho entre nós I e j com menor peso total de arcos



problemas comuns

- caminho mais curto
  - caminho entre nós I e j com menor peso total de arcos



Algoritmo de Dijkstra

Entradas:

Um grafo ponderado G = (V,E,p)

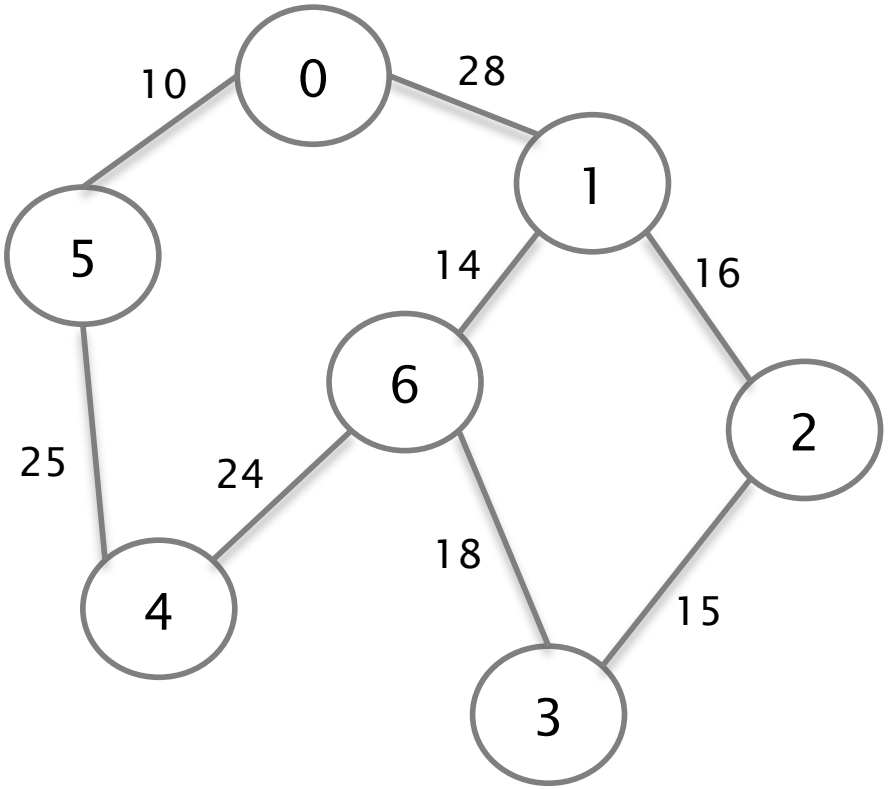
Um vértice V do grafo

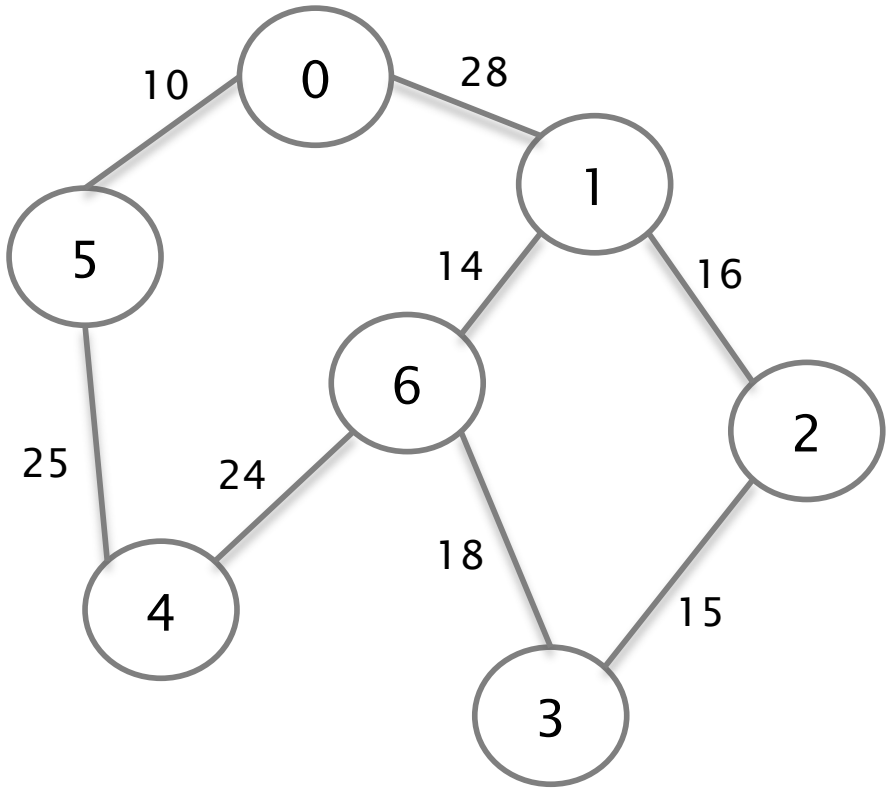
Saída:

Menor caminho entre V e
cada um dos nós do grafo

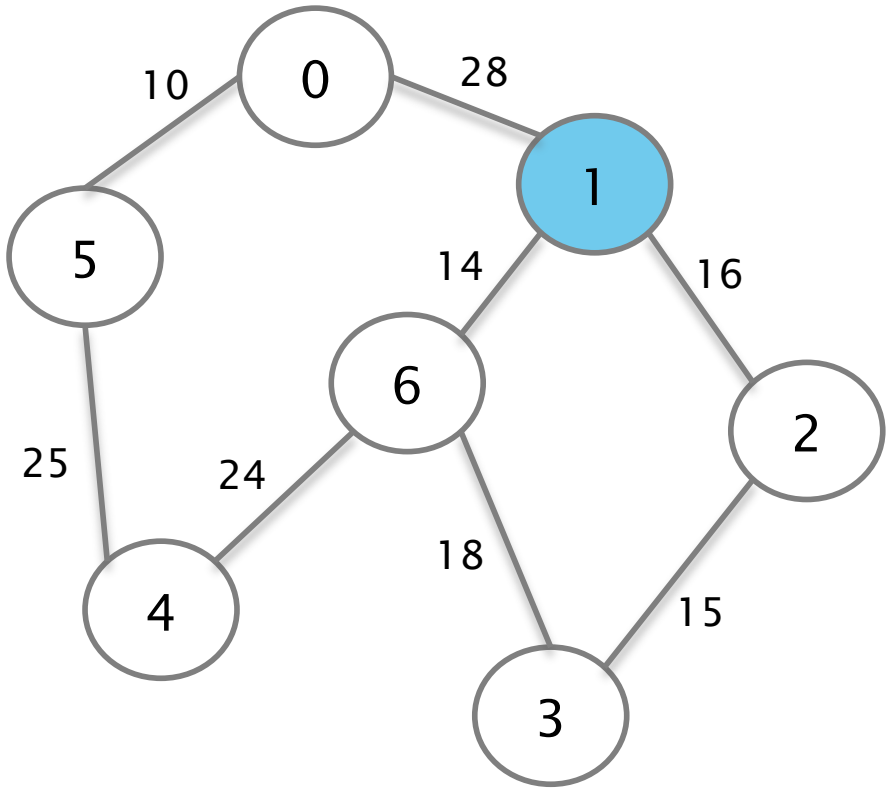
- roteamento em redes
- deslocamento de caminhões em trânsito pesado
- desenho de chips
- roteamento de mensagens em telecomunicações
- ...

Dijkstra: caminhos mais curtos

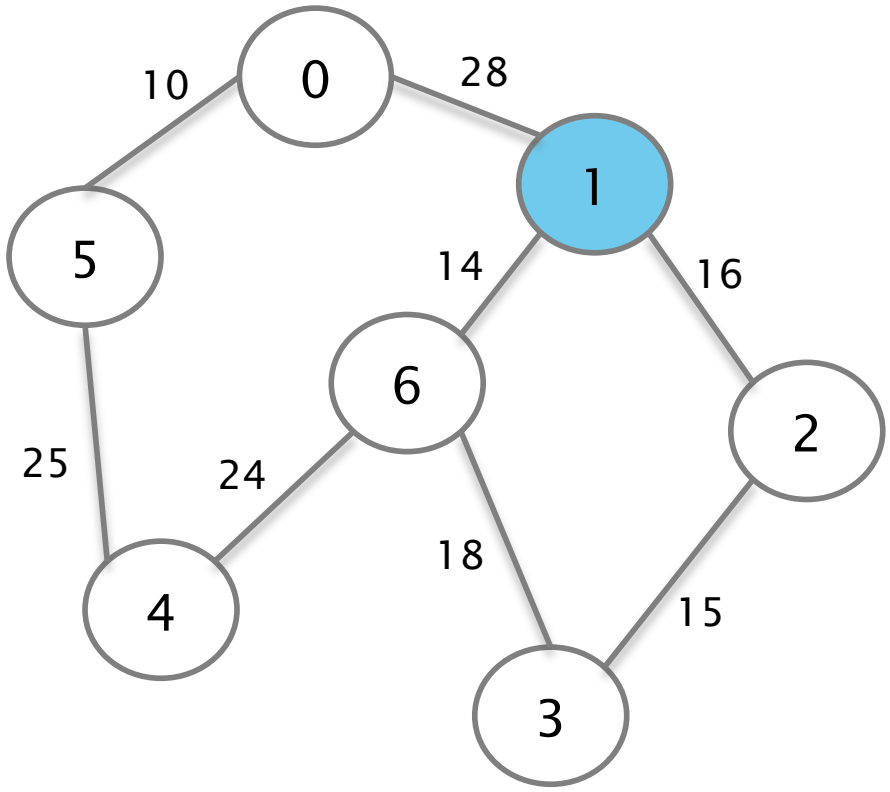




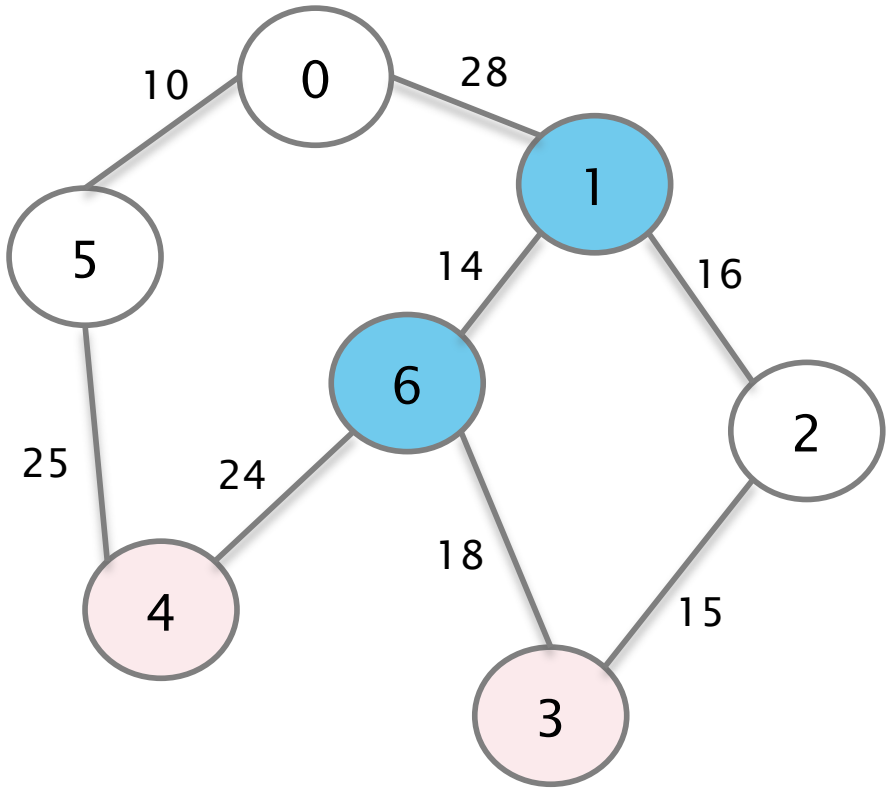
	dist
0	∞
1	0
2	∞
3	∞
4	∞
5	∞
6	∞



0	28
1	0
2	16
3	∞
4	∞
5	∞
6	14

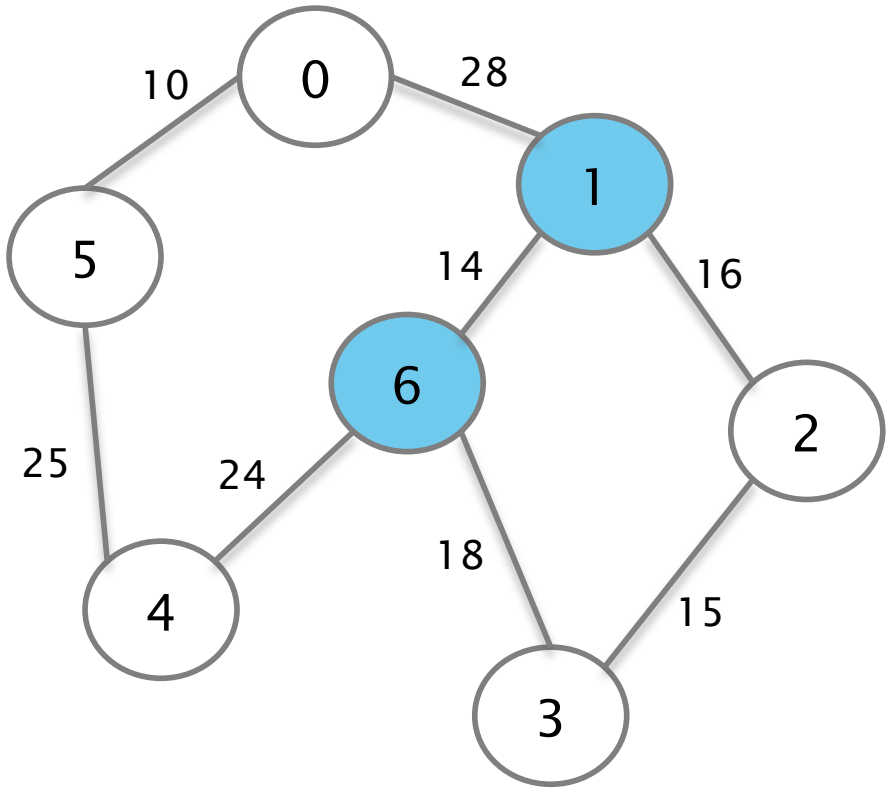


0	28
1	0
2	16
3	∞
4	∞
5	∞
6	14

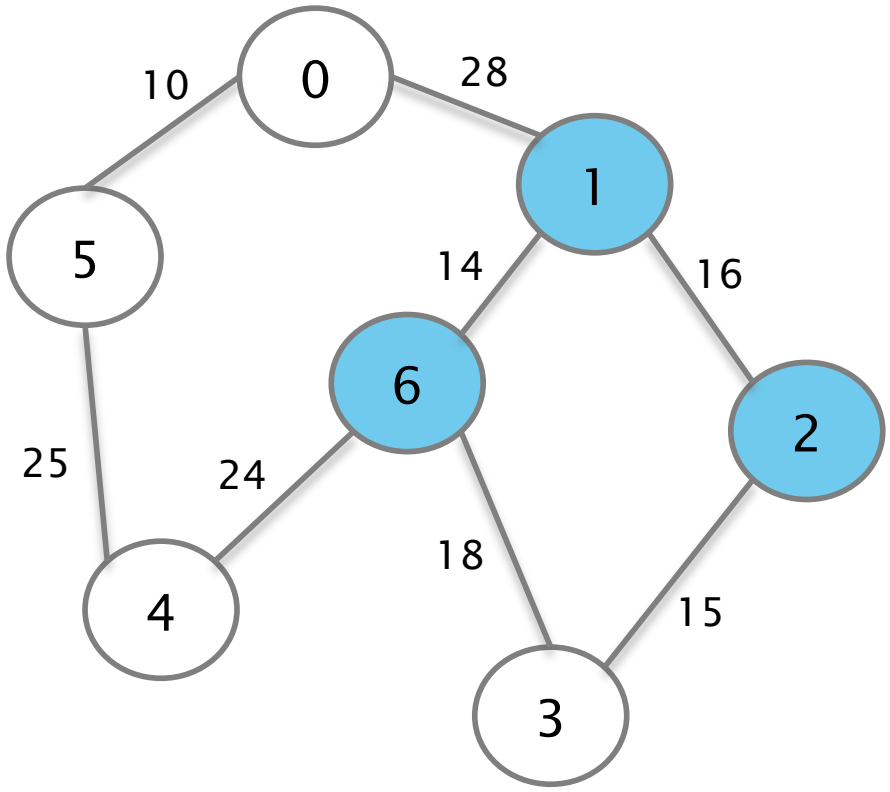


0	28
1	0
2	16
3	∞
4	∞
5	∞
6	14

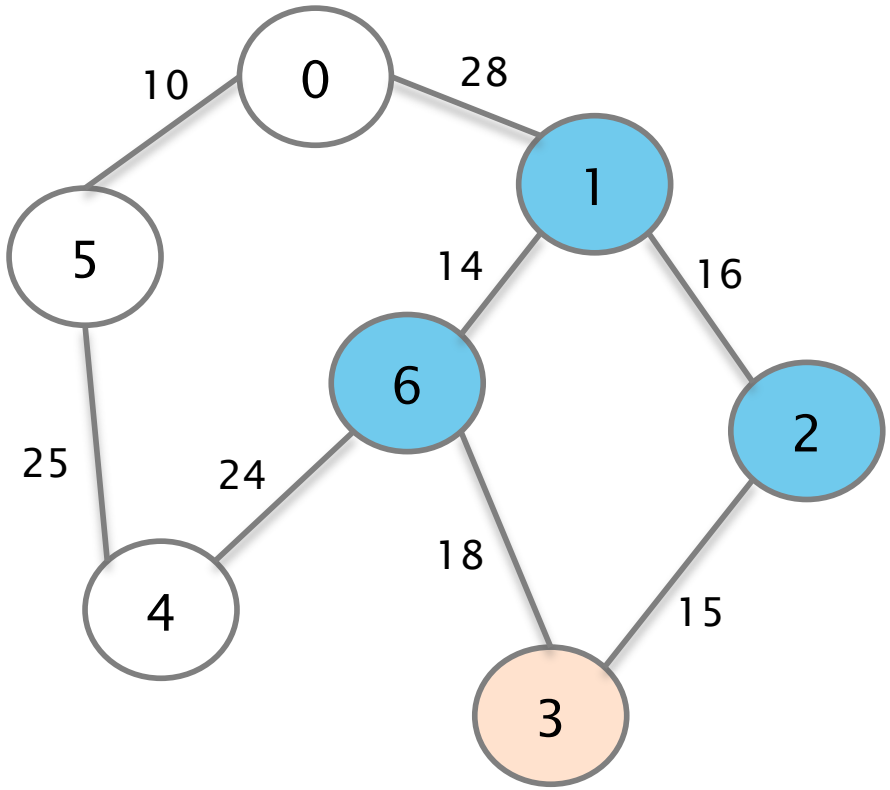
0	28
1	0
2	16
3	32
4	38
5	∞
6	14



0	28
1	0
2	16
3	32
4	38
5	∞
6	14

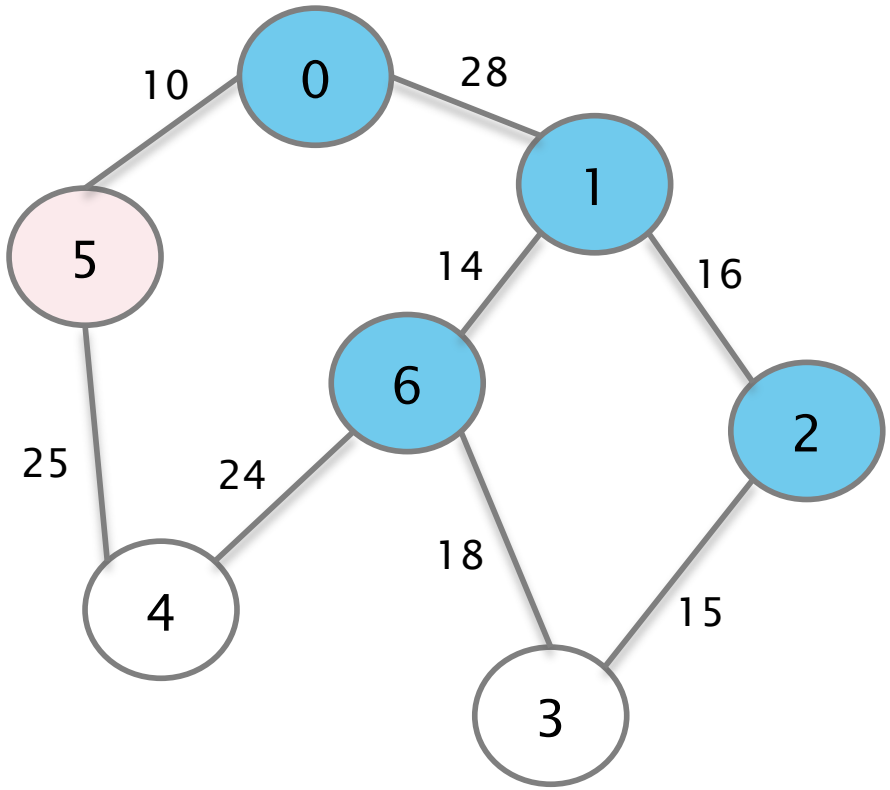


0	28
1	0
2	16
3	32
4	38
5	∞
6	14



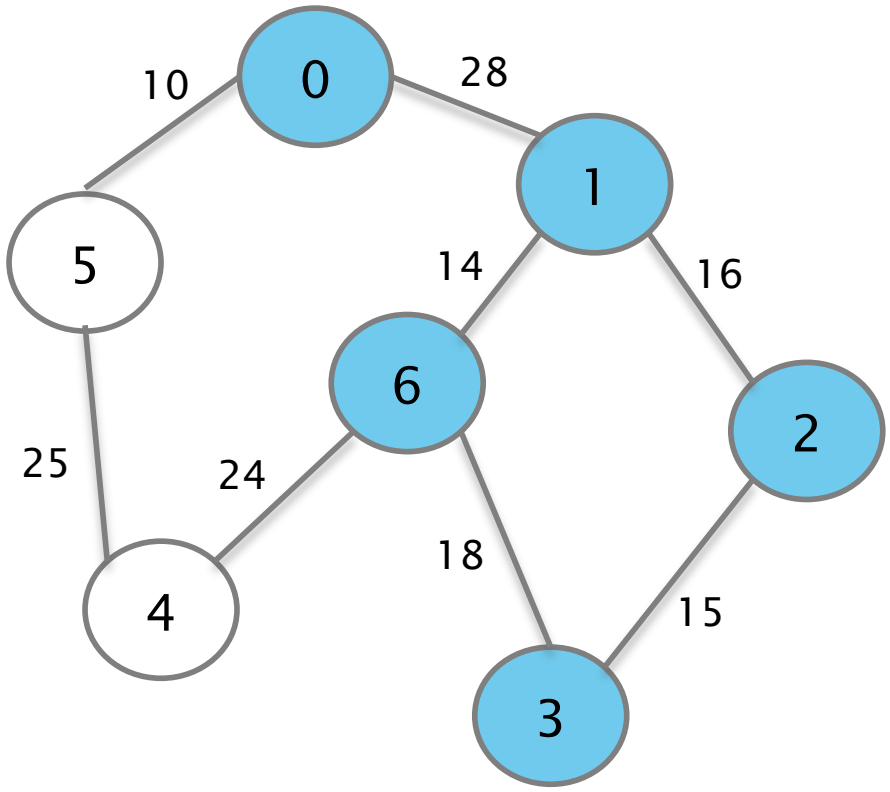
0	28
1	0
2	16
3	32
4	38
5	∞
6	14

0	28
1	0
2	16
3	31
4	38
5	∞
6	14

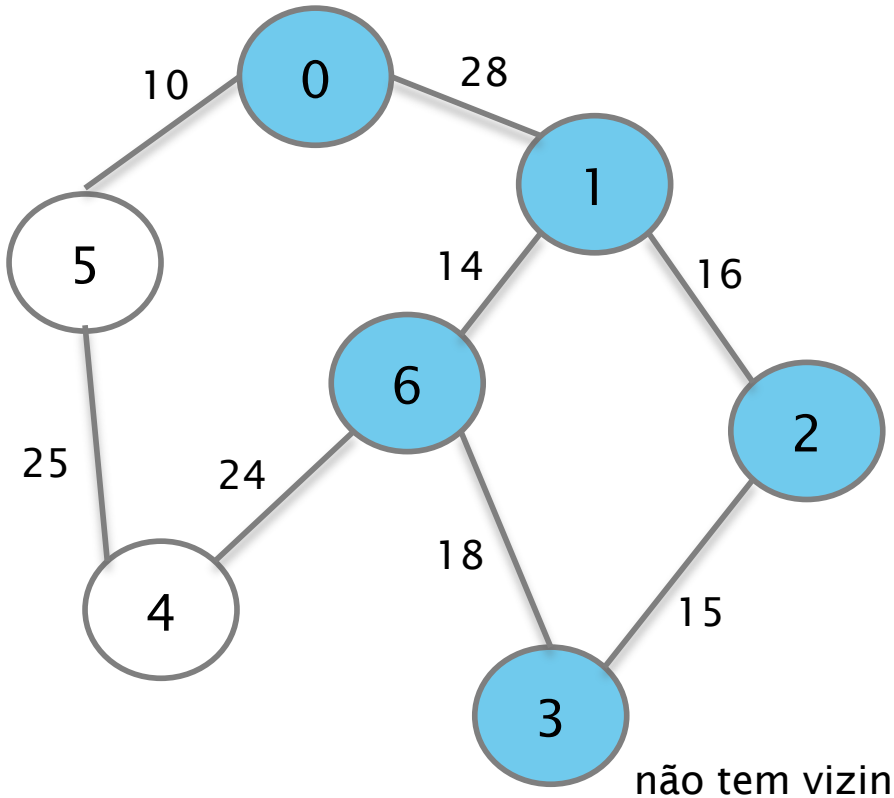


0	28
1	0
2	16
3	31
4	38
5	∞
6	14

0	28
1	0
2	16
3	31
4	38
5	38
6	14

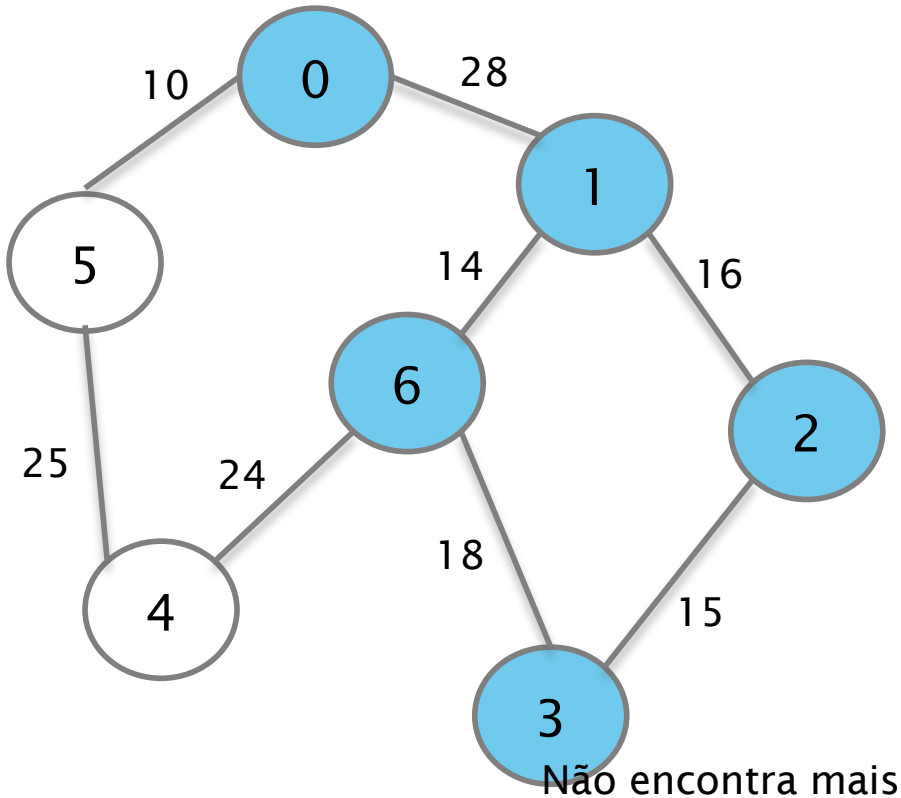


0	28
1	0
2	16
3	31
4	38
5	38
6	14



0	28
1	0
2	16
3	31
4	38
5	38
6	14

não tem vizinhos não visitados!



0	28
1	0
2	16
3	31
4	38
5	38
6	14

Não encontra mais caminhos mais curtos

Algoritmo de Dijkstra

Algoritmo de Dijkstra /* Menor caminho entre um nó de origem e um de destino */

1. Defina o nó de origem
2. Atribua a todos os nós um valor de distância ao nó de origem: valor zero ao nó de origem, e *infinito* para todos os outros nós.
3. Marque todos os demais nós como não visitados e o nó origem como corrente (A).
4. Considere a distância de todos os nós vizinhos não visitados ao nó corrente e calcule uma distância deles ao nó origem através do nó corrente. Se essa distância for menor do que a distância registrada anteriormente, sobrescreva a distância de B.

Por exemplo, se o nó atual A tiver distância 6 e houver uma aresta de peso 2 conectando-o com um outro nó B, a distância de B através de A será 8.

Ao final, tem-se a menor distância entre o nó de origem e cada um dos nós do grafo.

PUC RIO

Algoritmo de Dijkstra

Algoritmo de Dijkstra /* Menor caminho entre um nó de origem e um de destino */

1. Defina o nó de origem
2. Atribua a todos os nós um valor de distância ao nó de origem:
valor zero ao nó de origem, e *infinito* para todos os outros nós.
3. Marque todos os demais nós como não visitados e o nó origem como corrente (A).
4. Considere a distância de todos os nós vizinhos não visitados ao nó corrente e
calcule uma distância deles ao nó origem através do nó corrente. Se essa distância for menor
do que a distância registrada anteriormente, sobrescreva a distância de B.
5. Ao terminar de considerar todos os vizinhos do nó atual A, marque-o como visitado.
Um nó visitado não será mais verificado; sua distância registrada agora é final e mínima.
6. Se todos os nós tiverem sido visitados, termine.
Caso contrário, marque o nó não visitado com a menor distância (ao nó de origem)
como o próximo "nó corrente", e repita a partir do passo 4.

Ao final, tem-se a menor distância entre o nó de origem e cada um dos nós do grafo.

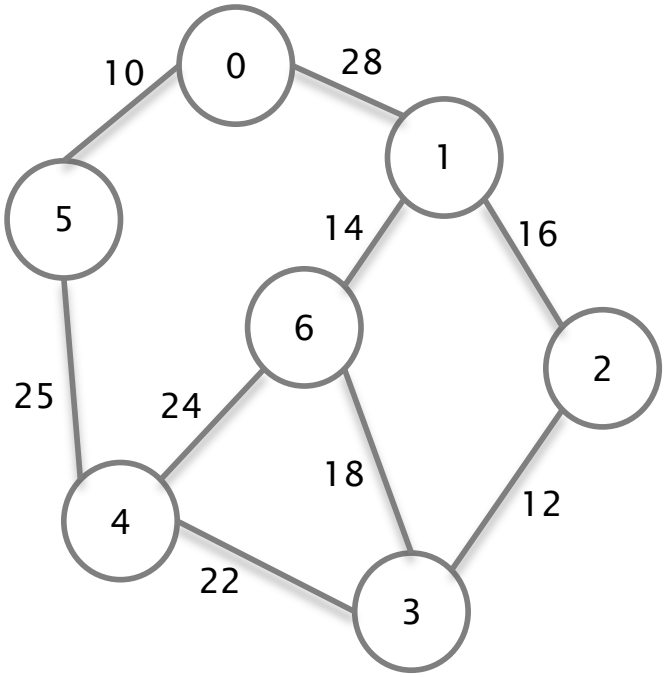
Algoritmo de Dijkstra

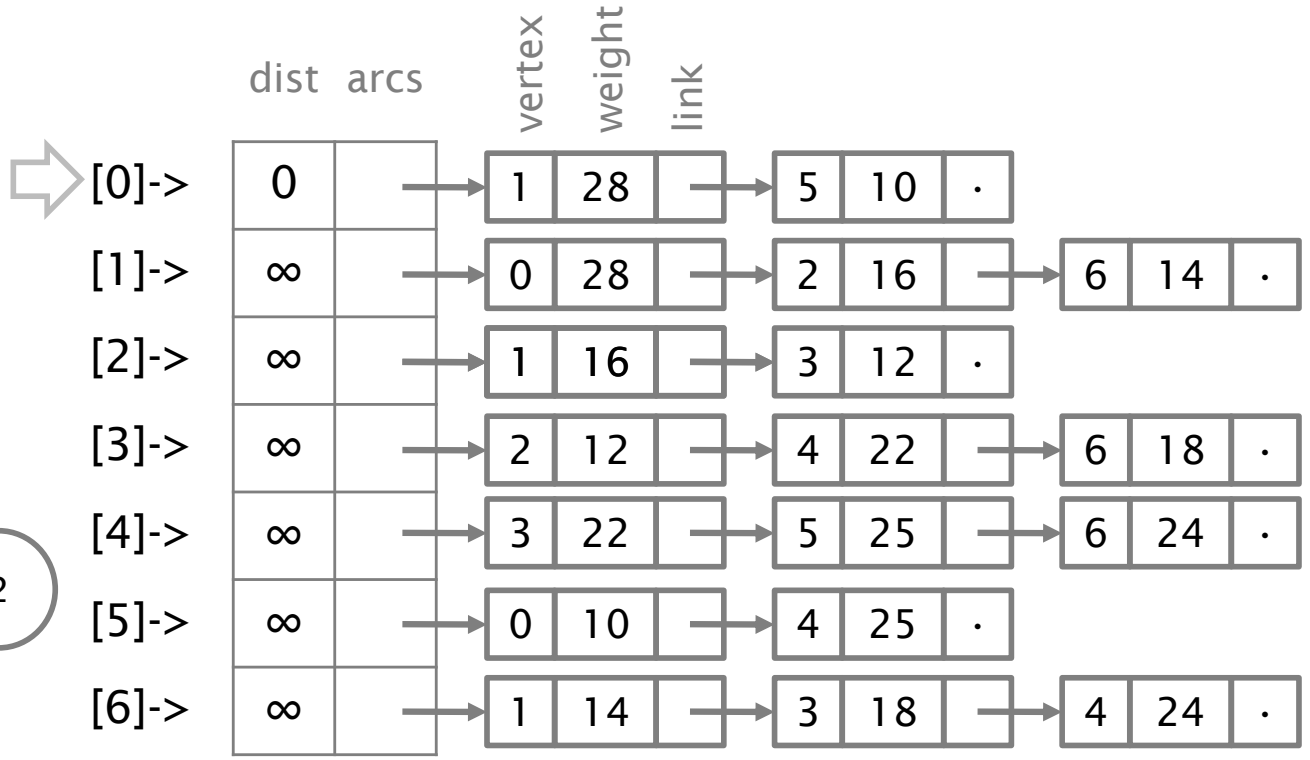
Complexidade temporal:

Depende da implementação do conjunto de vértices Q

Dijkstra - exemplo

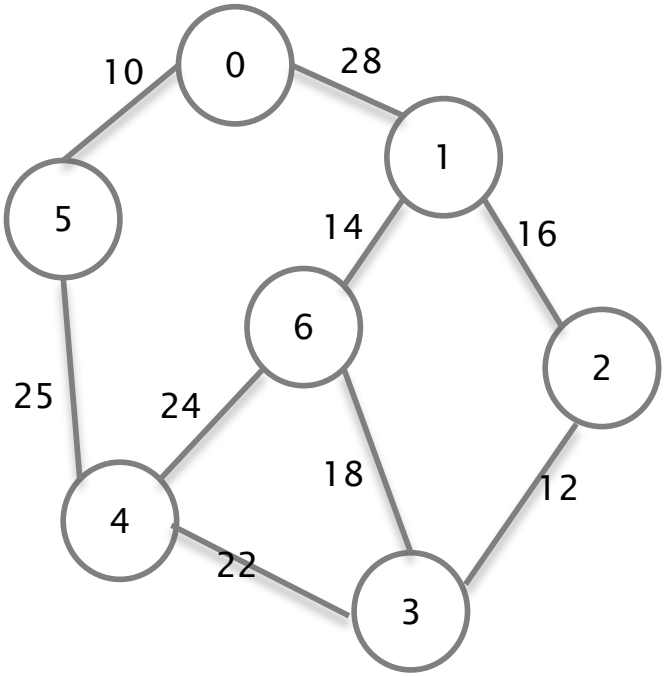
Vértice inicial: 0

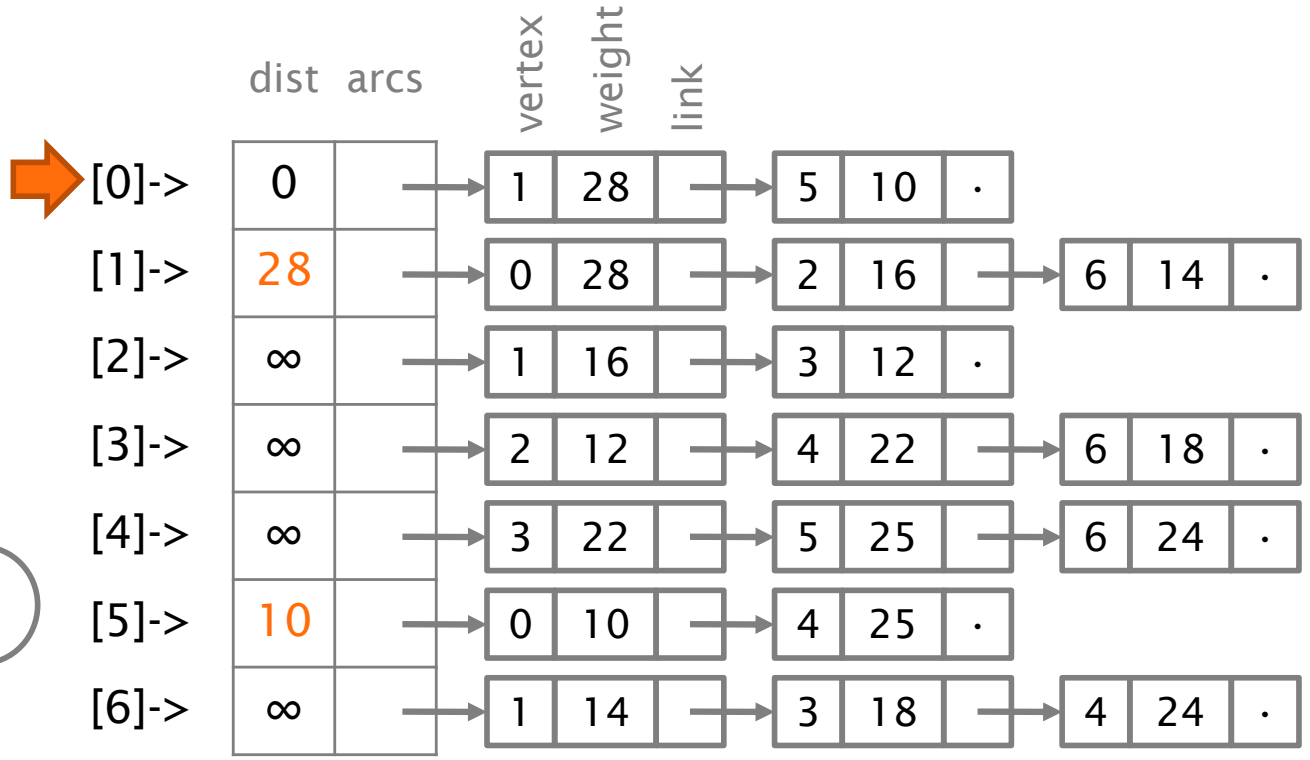




Dijkstra - exemplo

Modifica distâncias dos vértices 1 e 5



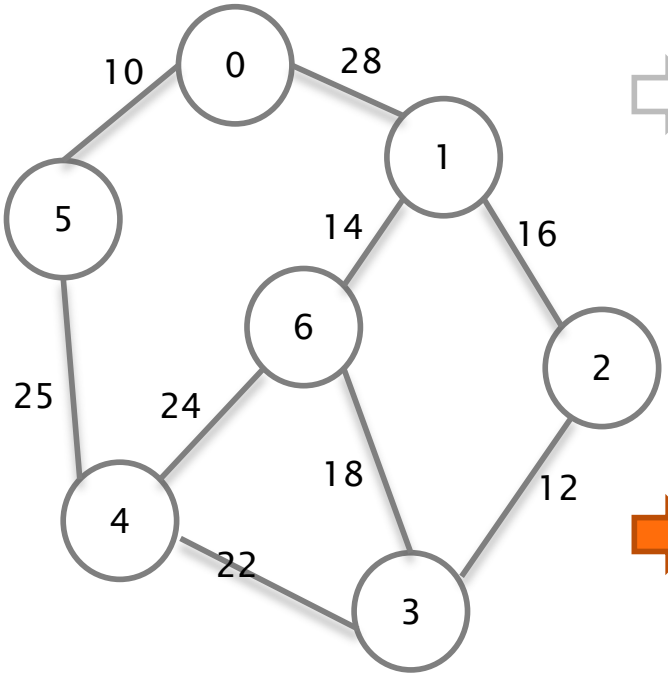


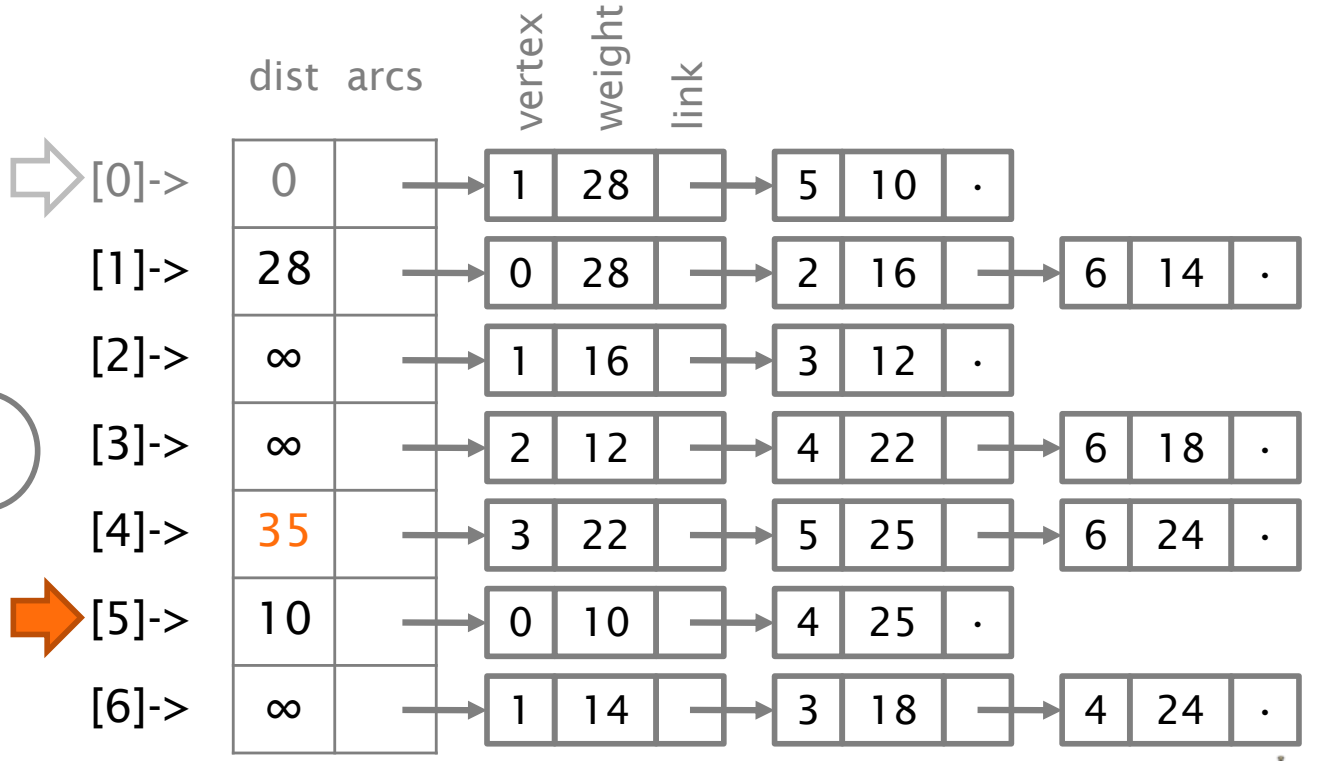
Dijkstra - exemplo

Marca o vértice 0 como visitado

Seleciona o vértice 5
(vértice não visitado de menor distância à origem)

Ignora vértice 0 (já visitado) e modifica distância do vértice 4



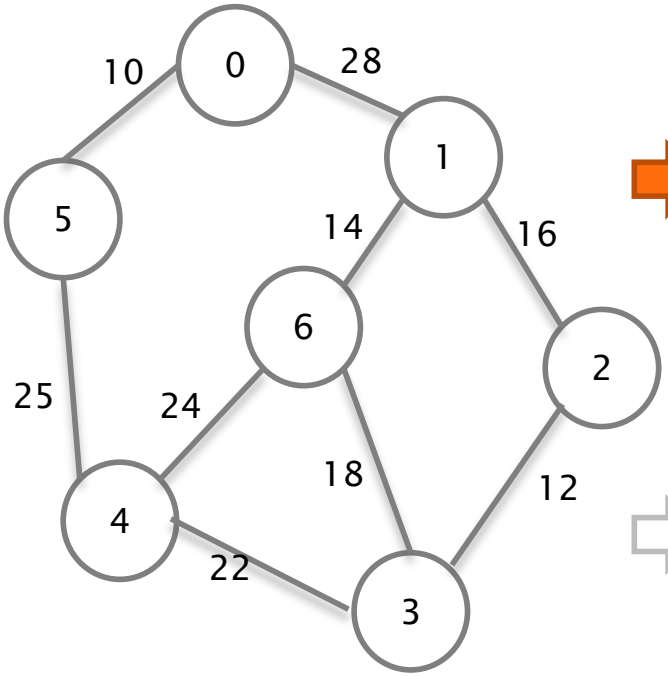


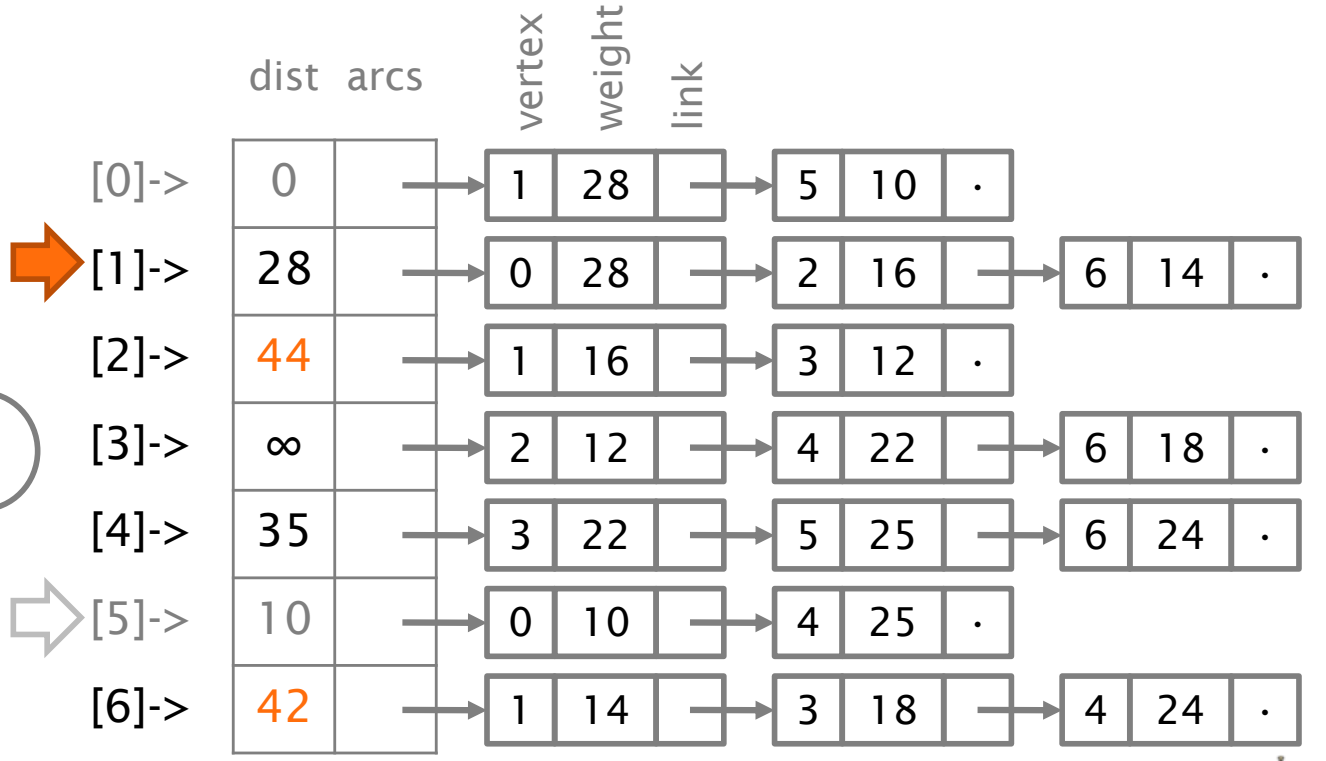
Dijkstra - exemplo

Marca o vértice 5 como visitado

Seleciona o vértice 1
(vértice não visitado de menor distância à origem)

Ignora vértice 0 e modifica distâncias dos vértices 2 e 6



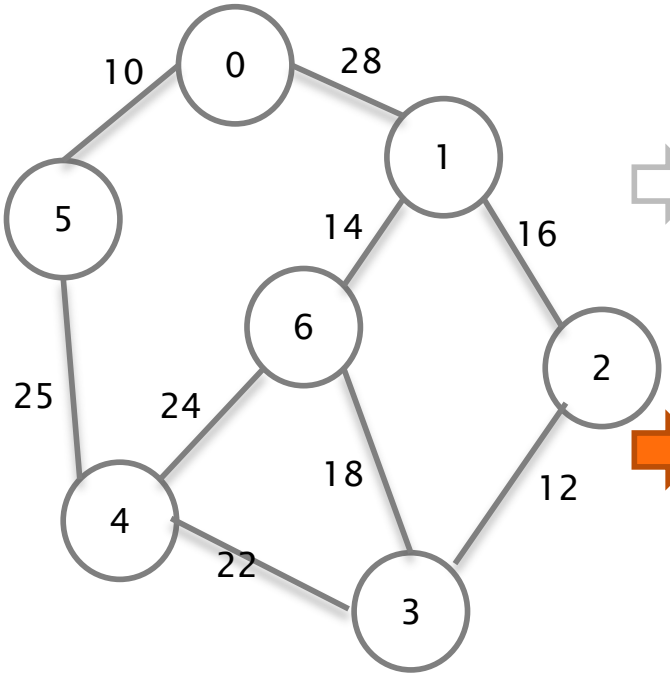


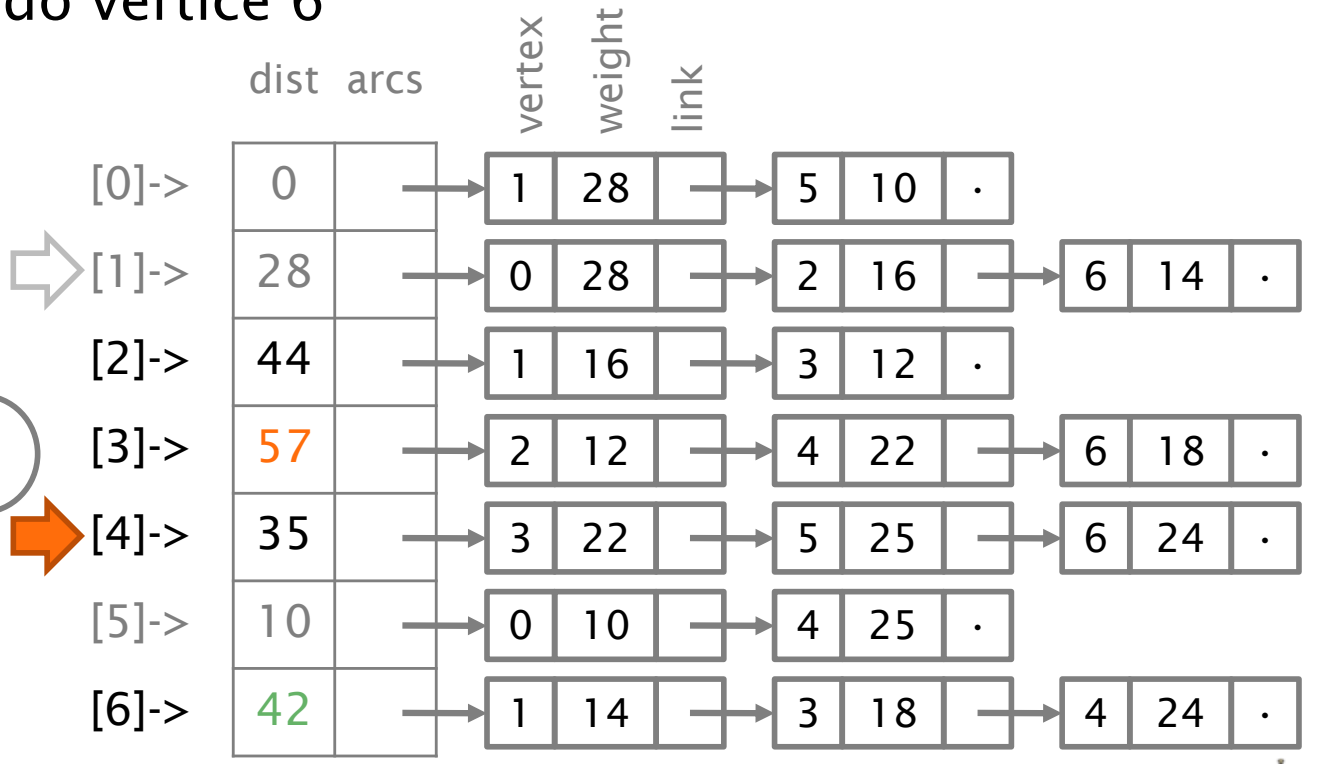
Dijkstra - exemplo

Marca o vértice 1 como visitado

Seleciona o vértice 4
(vértice não visitado de menor distância à origem)

Modifica distância do vértice 3; ignora vértice 5;
mantém a distância do vértice 6



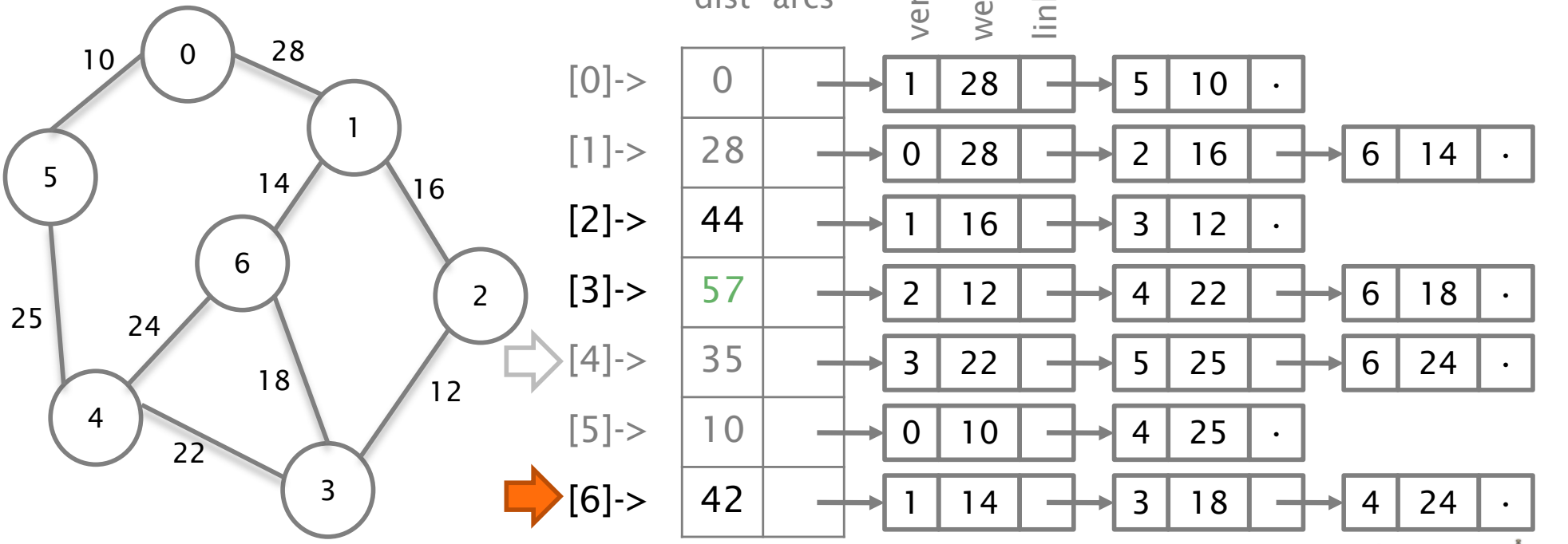


Dijkstra - exemplo

Marca o vértice 4 como visitado

Seleciona o vértice 6
(vértice não visitado de menor distância à origem)

Ignora vértice 1; mantém distância do vértice 3; ignora vértice 4

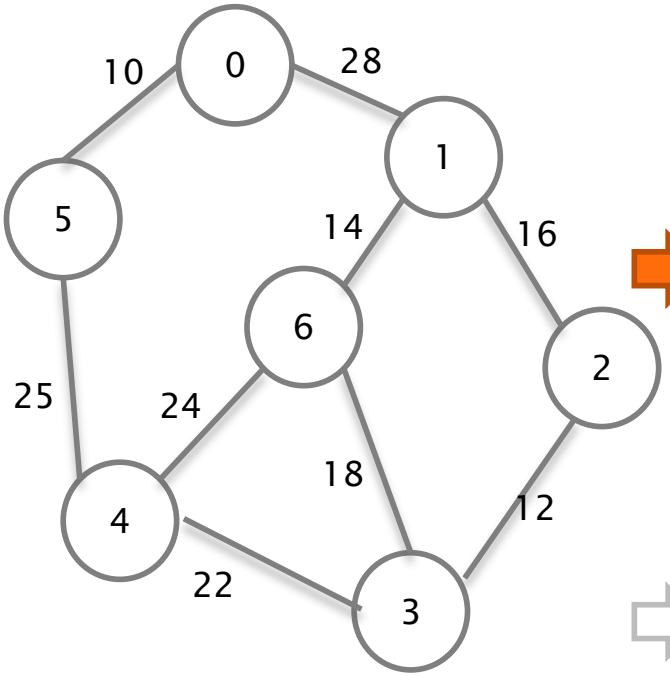


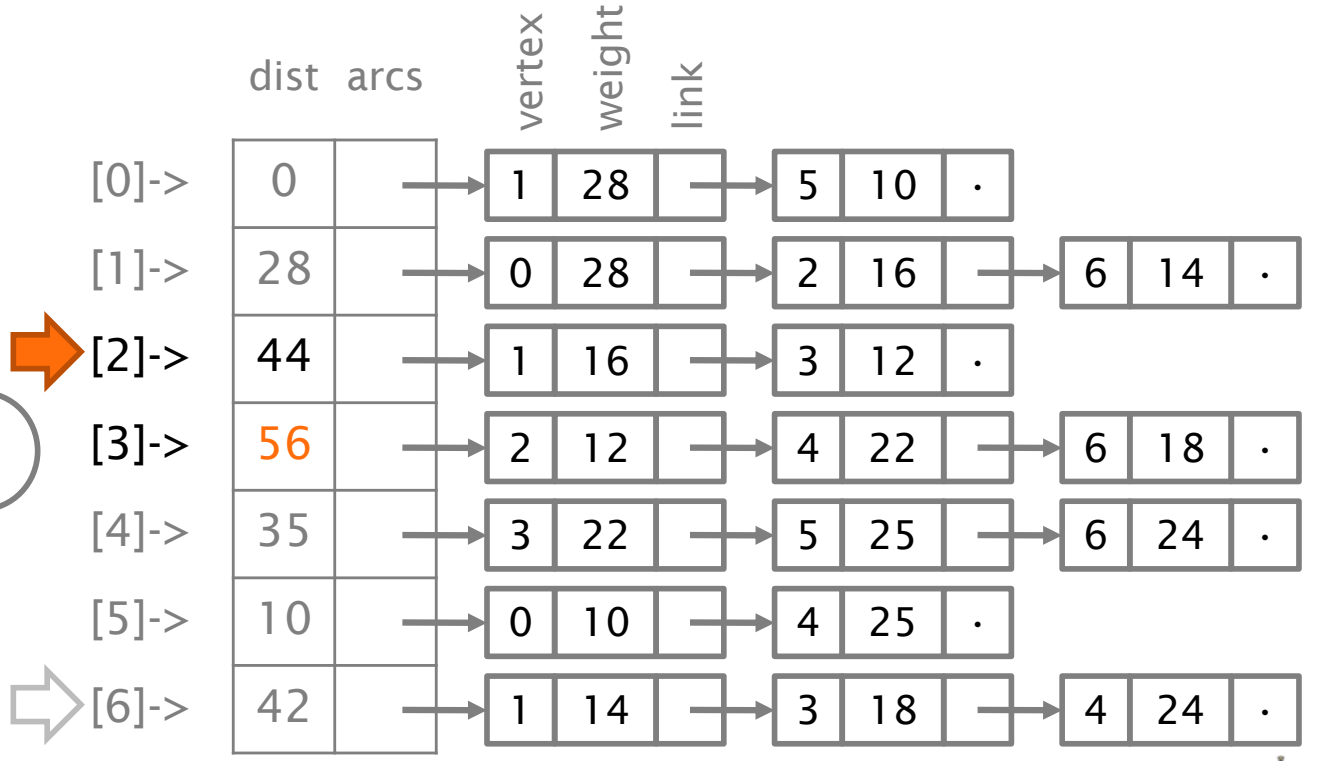
Dijkstra - exemplo

Marca o vértice 6 como visitado

Seleciona o vértice 2
(vértice não visitado de menor distância à origem)

ignora vértice 1; modifica distância do vértice 3



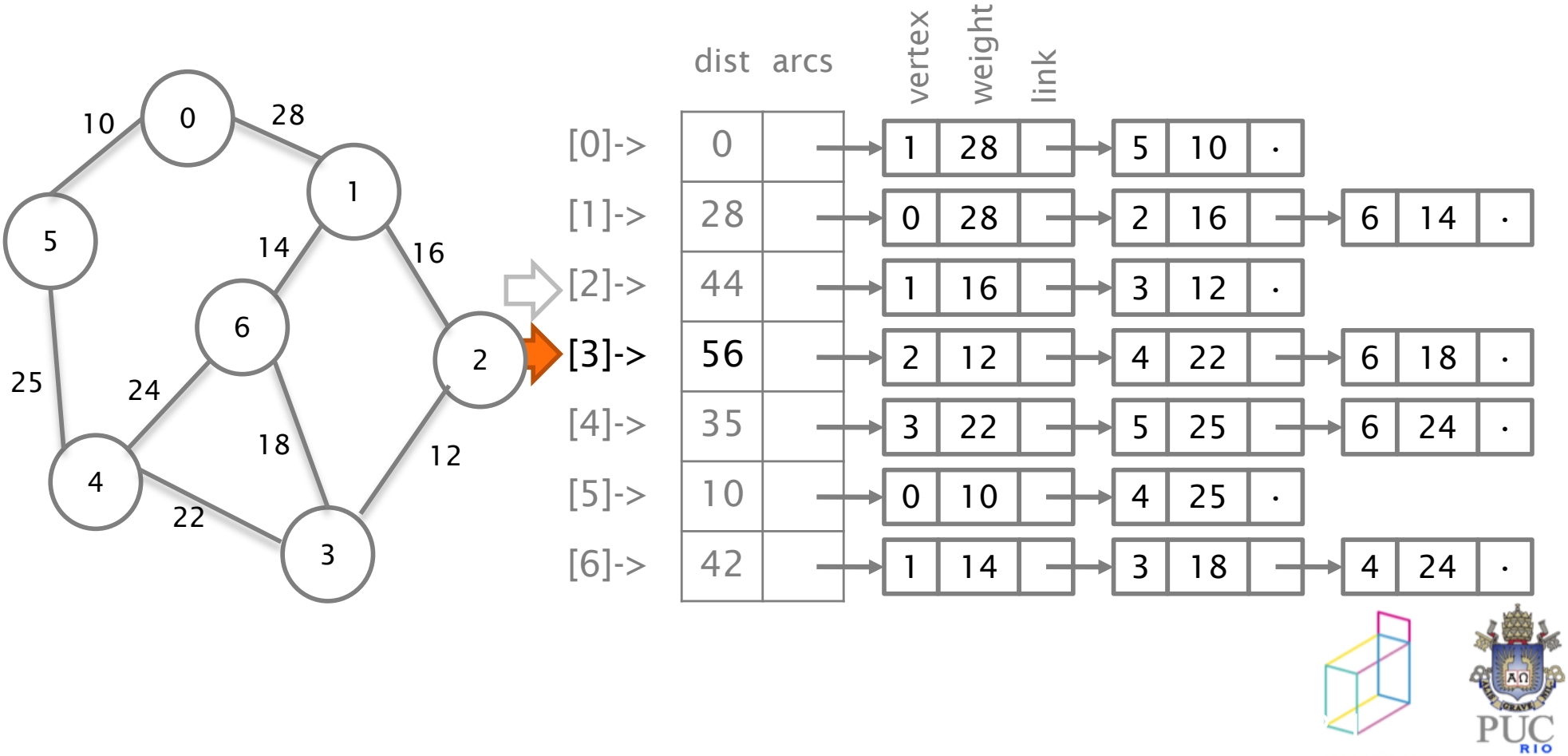


Dijkstra - exemplo

Marca o vértice 2 como visitado

seleciona o vértice 3
(vértice não visitado de menor distância à origem)

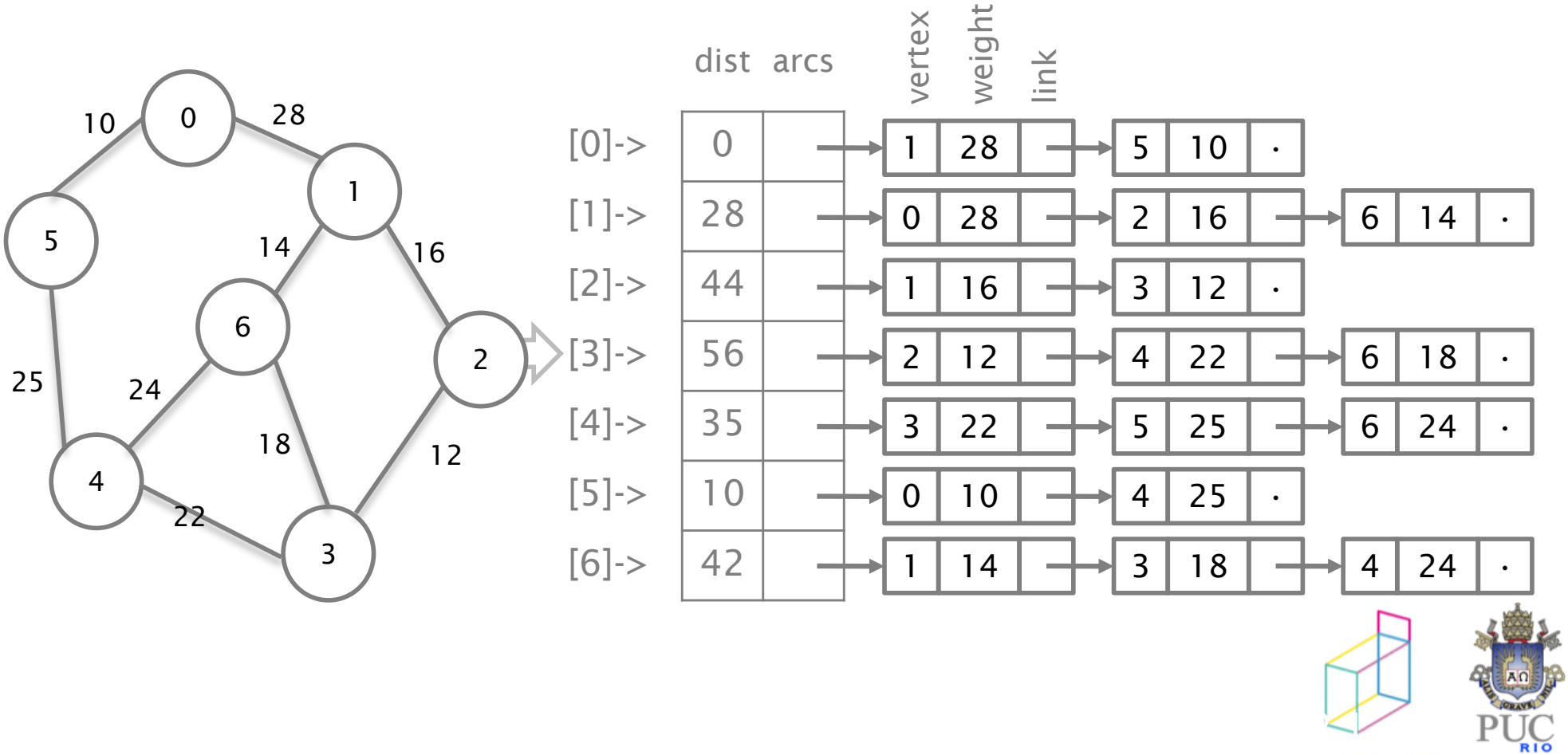
Ignora vértices 2, 4 e 6



Dijkstra - exemplo

Marca o vértice 3 como visitado

Não há mais vértices não visitados – FIM!

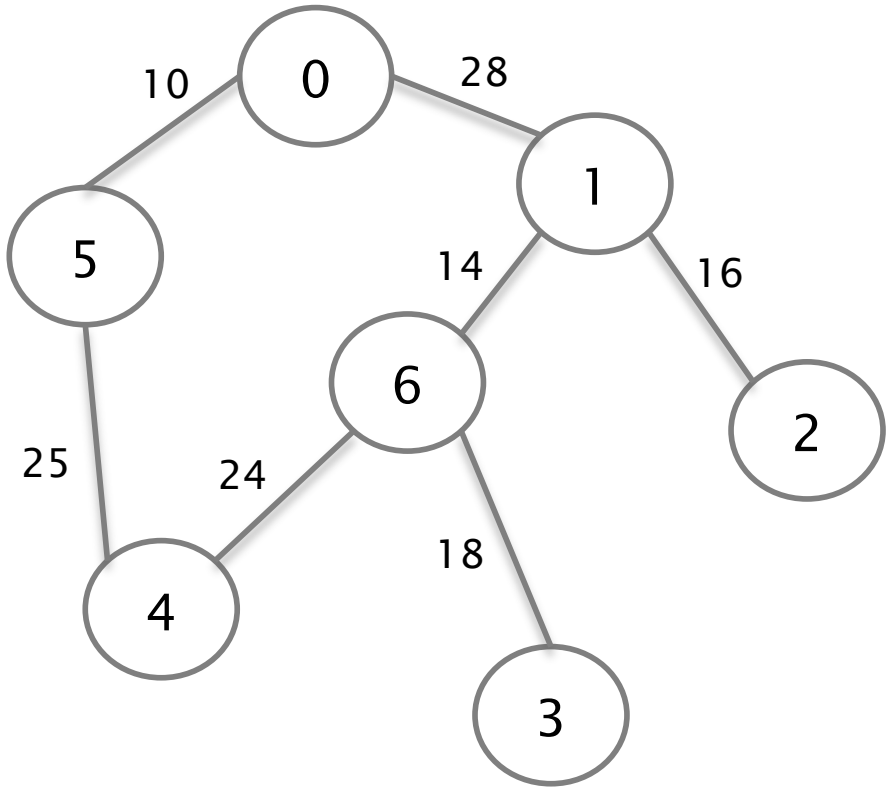


Como manter o custo dos nós não visitados?

- Podemos usar um min-heap
  - Implementação:
    - Armazenar pares [prioridade, nó]
    - Criar função corrige-prioridade

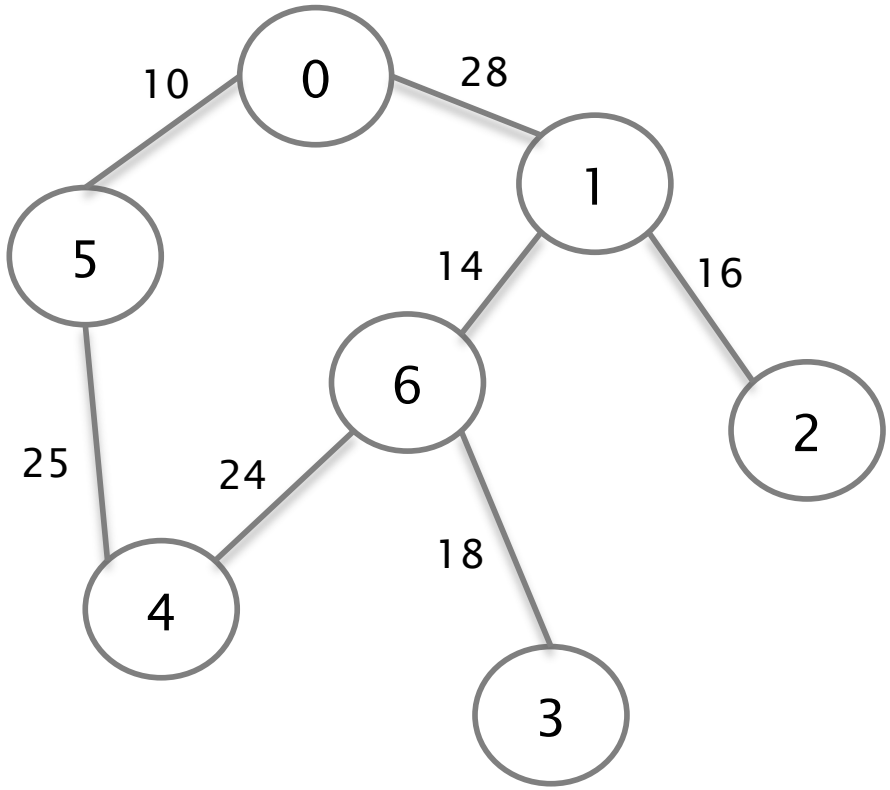
problemas comuns

- árvores geradoras
  - já vimos um exemplo em lab
  - mas agora queremos a de custo mínimo



problemas comuns

- árvores geradora:
  - subgrafo que é árvore e que contém todos os vértices
    - grafo conexo sem circuitos



Algoritmo de Kruskal

Árvore geradora de custo mínimo

Dado um grafo ponderado G = (V,E,p),

uma *árvore geradora de custo mínimo* para G

é uma árvore tal que:

V é o conjunto de nós da árvore

A soma dos pesos das arestas é minima
(entre as árvores geradoras)

Algoritmo de Kruskal

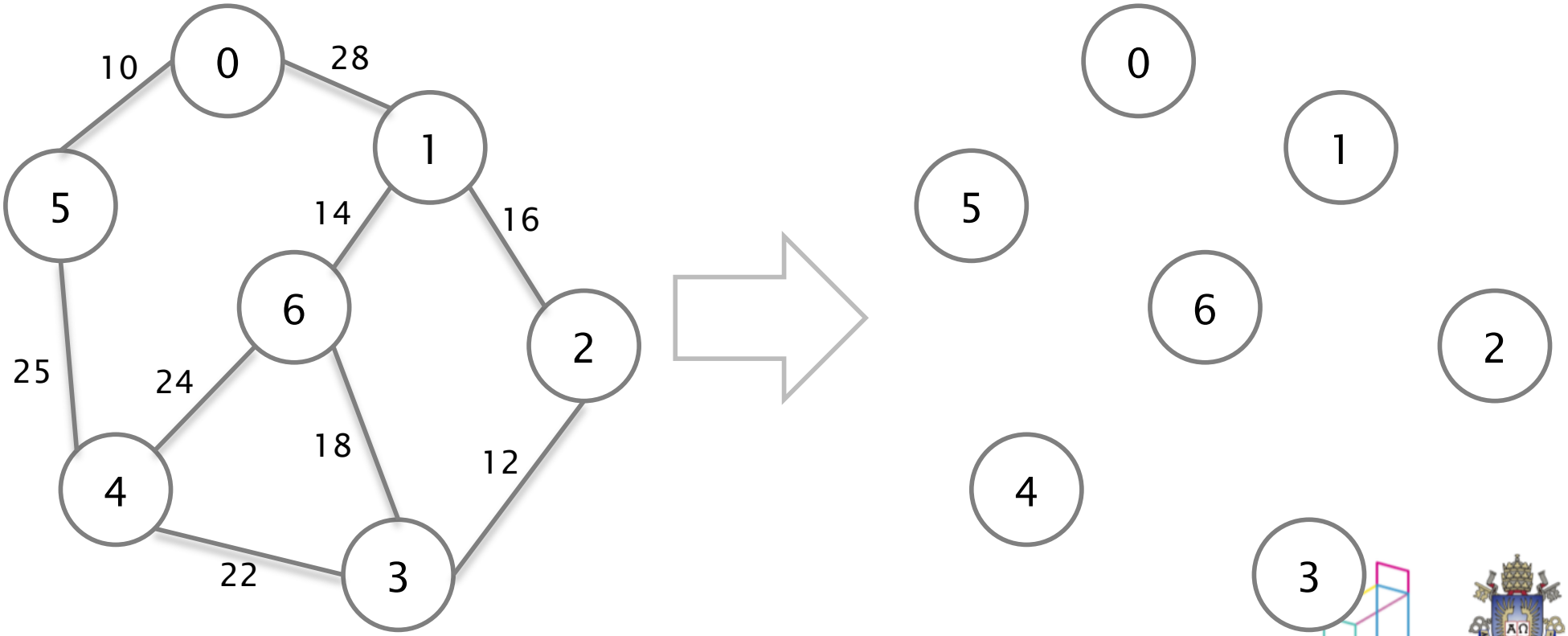
Entrada: Um grafo ponderado G = (V,E,p)

Saída: Árvore geradora de custo mínimo

1. Considere cada nó em V como
   uma árvore separada (formando uma floresta)
2. Examine a aresta de menor custo.
   Se ela unir duas árvores na floresta, inclua-a.
3. Repita o Passo (2) até todos os nós
   estarem conectados.

Algoritmo de Kruskal

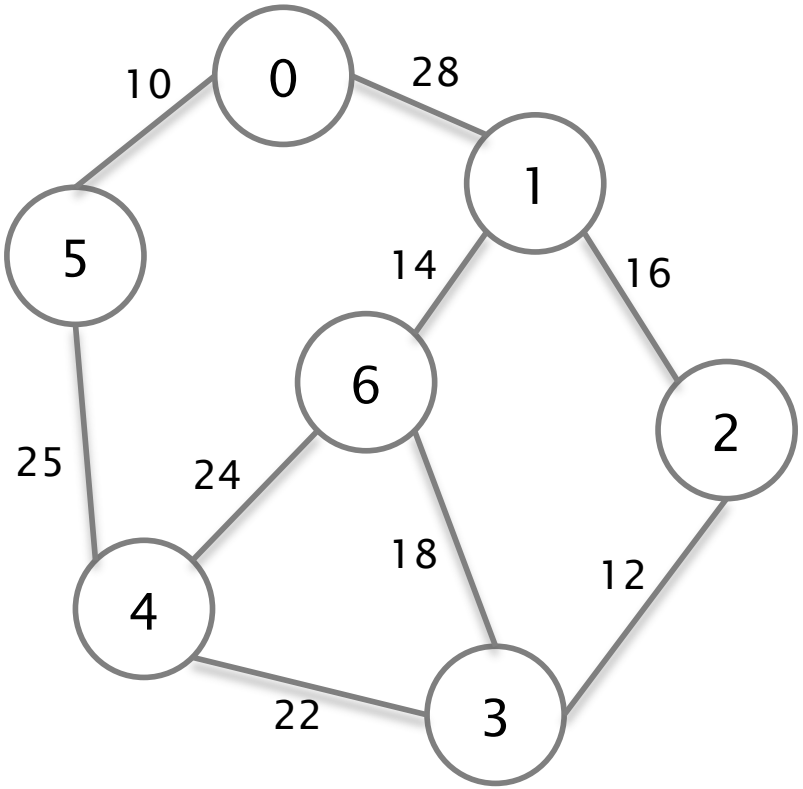
1. Considere cada nó como uma árvore separada (formando uma floresta)

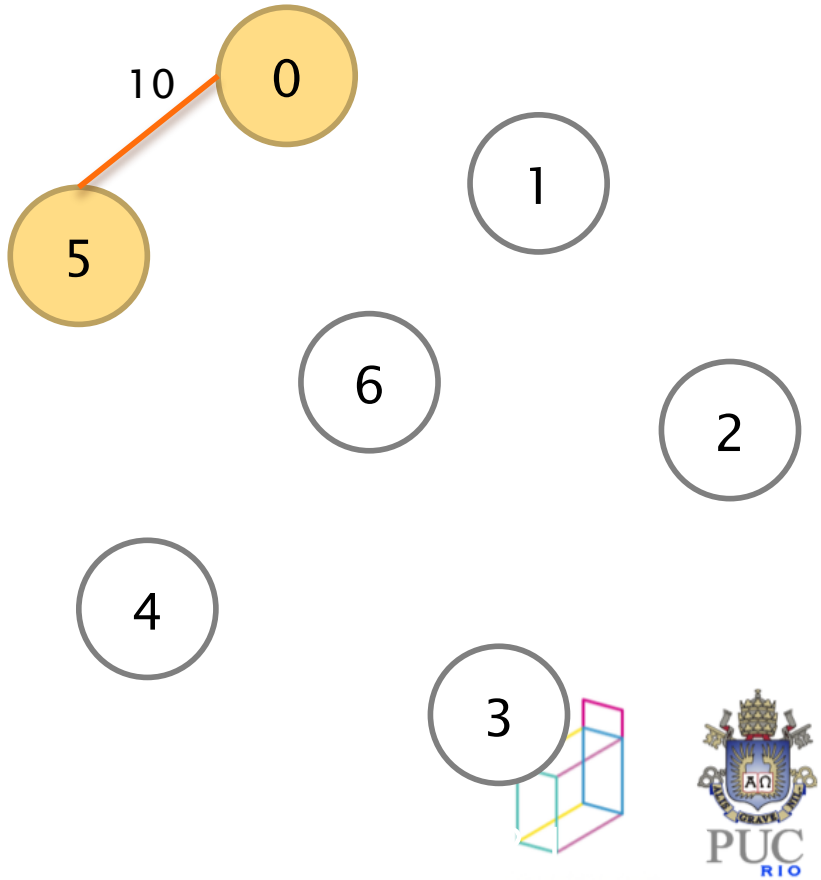


Algoritmo de Kruskal

2. Examine a aresta de menor custo.
Se ela unir duas árvores na floresta, inclua-a.

3. Repita o Passo (2) até todos os nós estarem conectados.

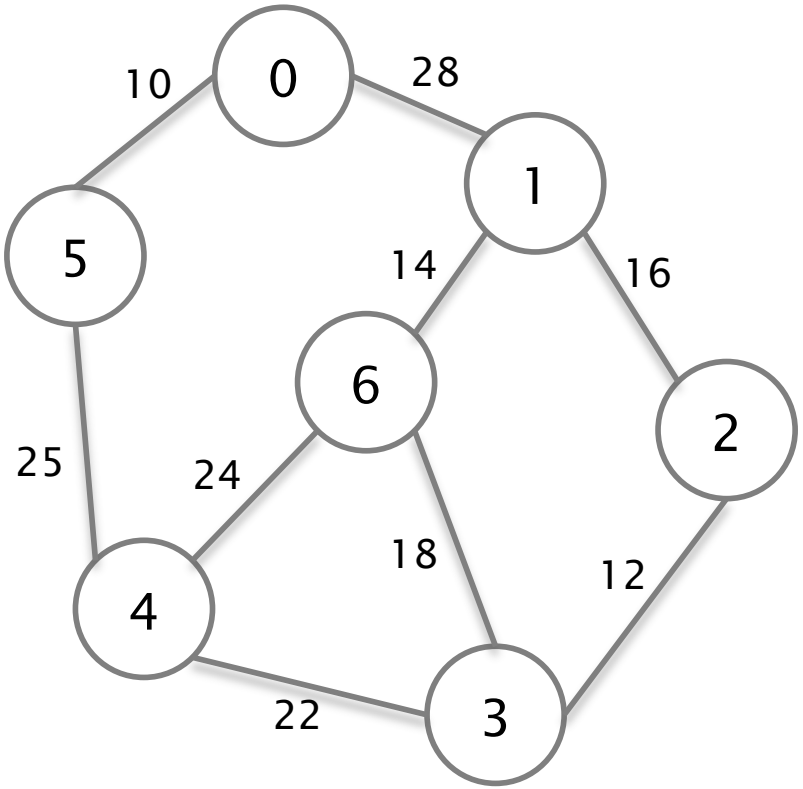


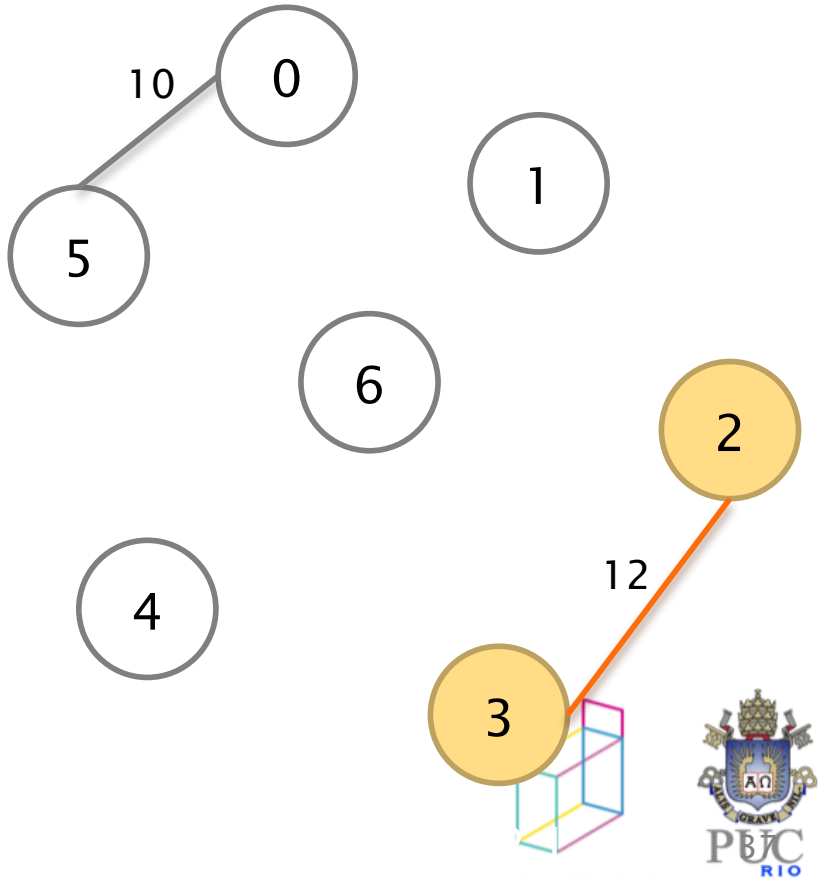


Algoritmo de Kruskal

2. Examine a aresta de menor custo.
Se ela unir duas árvores na floresta, inclua-a.

3. Repita o Passo (2) até todos os nós estarem conectados.

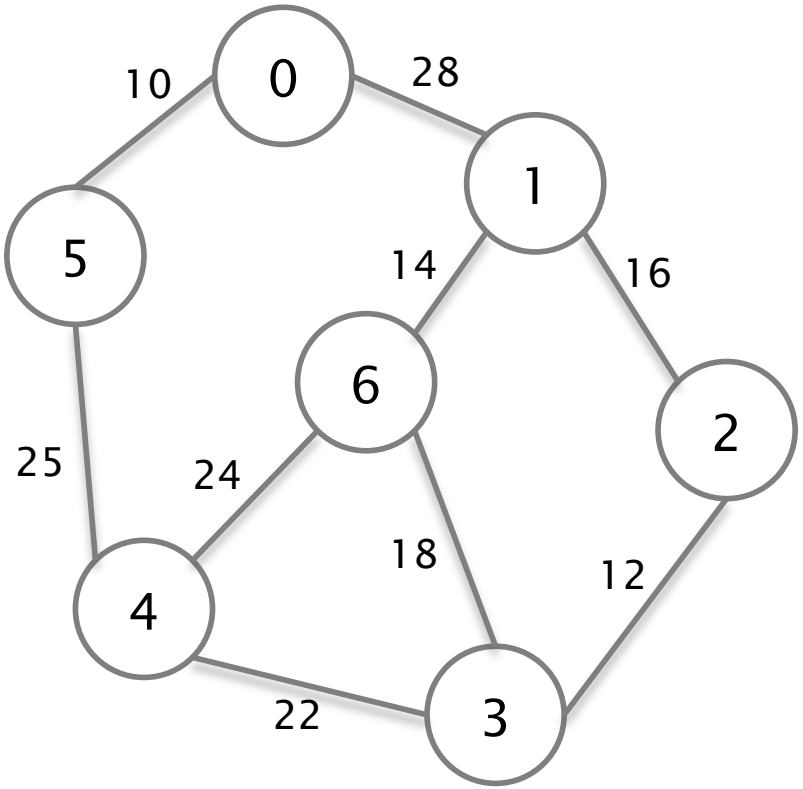


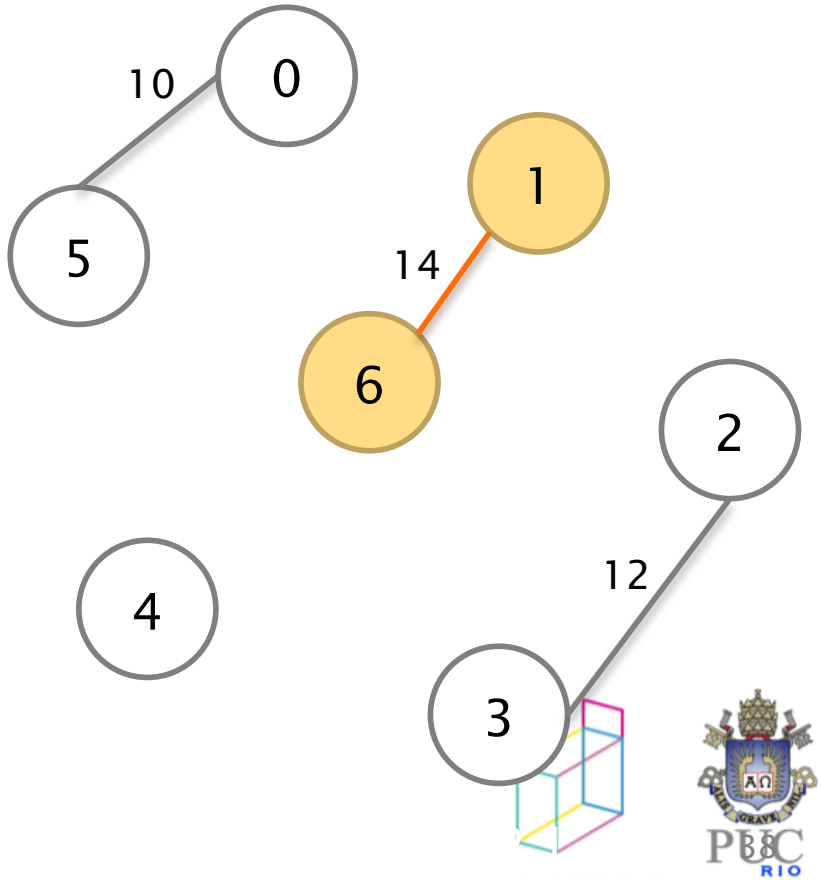


Algoritmo de Kruskal

2. Examine a aresta de menor custo.
Se ela unir duas árvores na floresta, inclua-a.

3. Repita o Passo (2) até todos os nós estarem conectados.



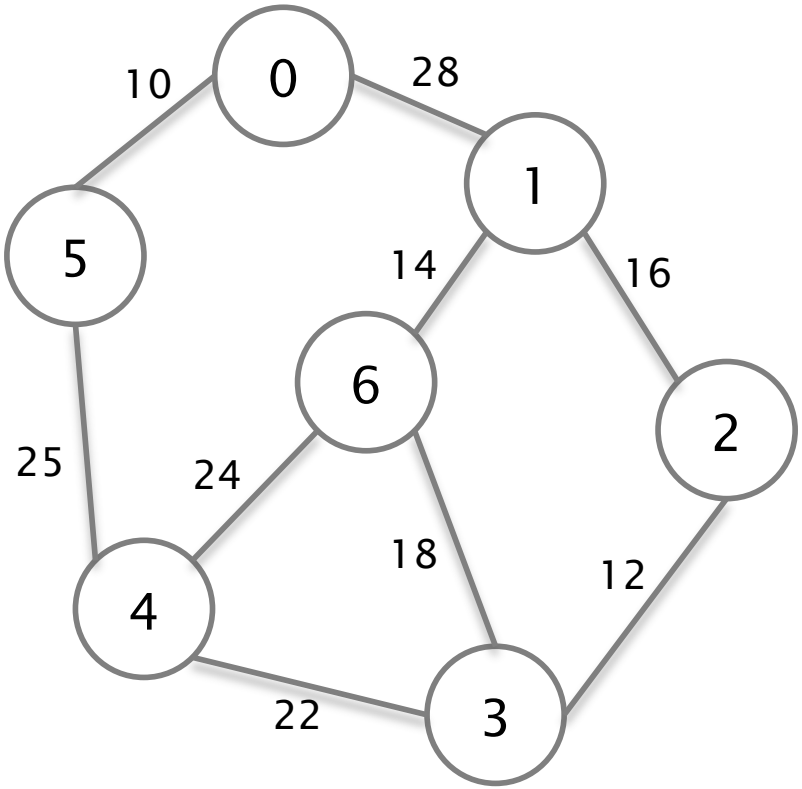


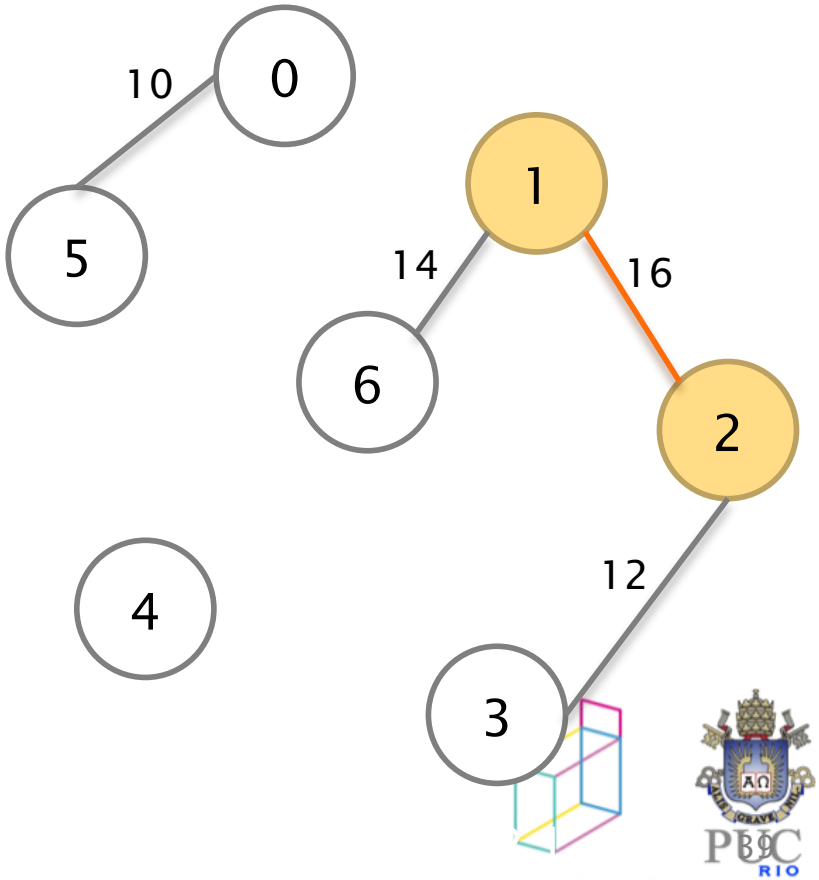
Algoritmo de Kruskal

2. Examine a aresta de menor custo.

Se ela unir duas árvores na floresta, inclua-a.

3. Repita o Passo (2) até todos os nós estarem conectados.



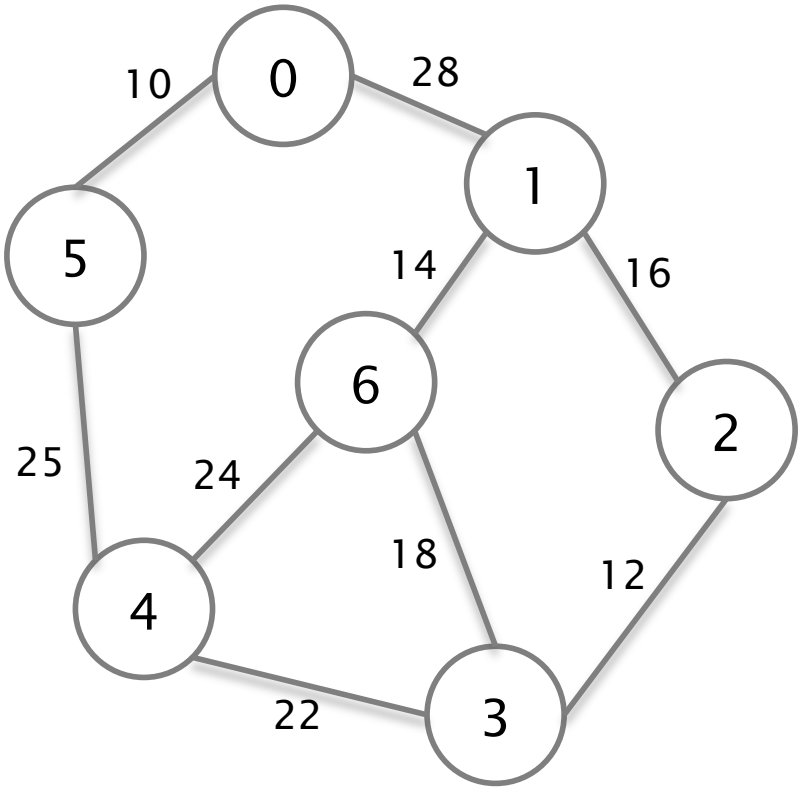


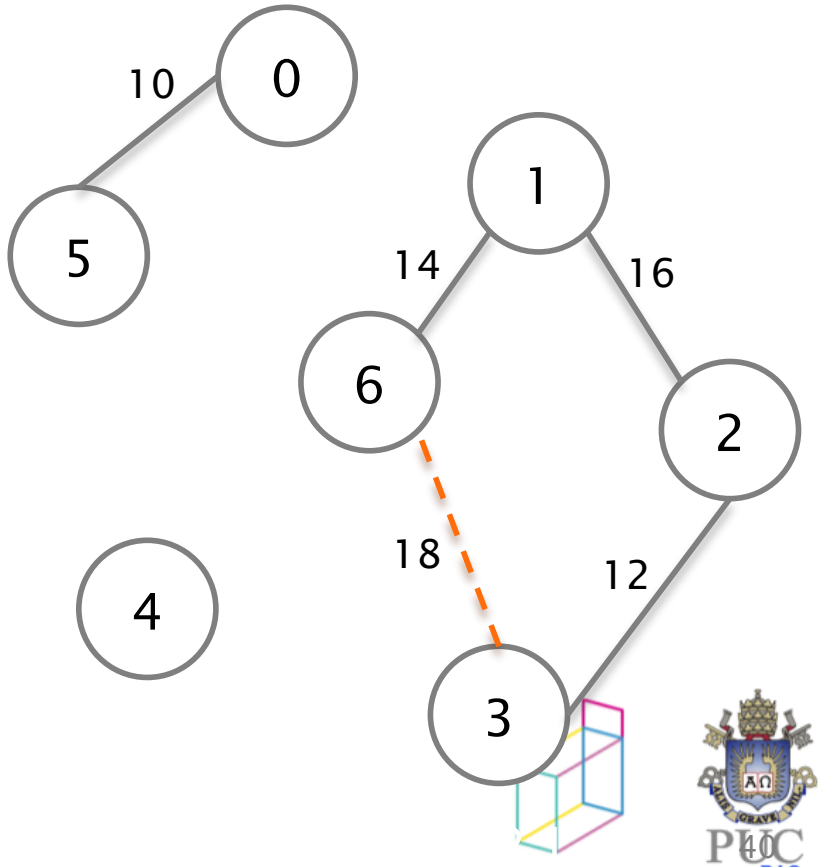
Algoritmo de Kruskal

2. Examine a aresta de menor custo.

Se ela unir duas árvores na floresta, inclua-a.

3. Repita o Passo (2) até todos os nós estarem conectados.

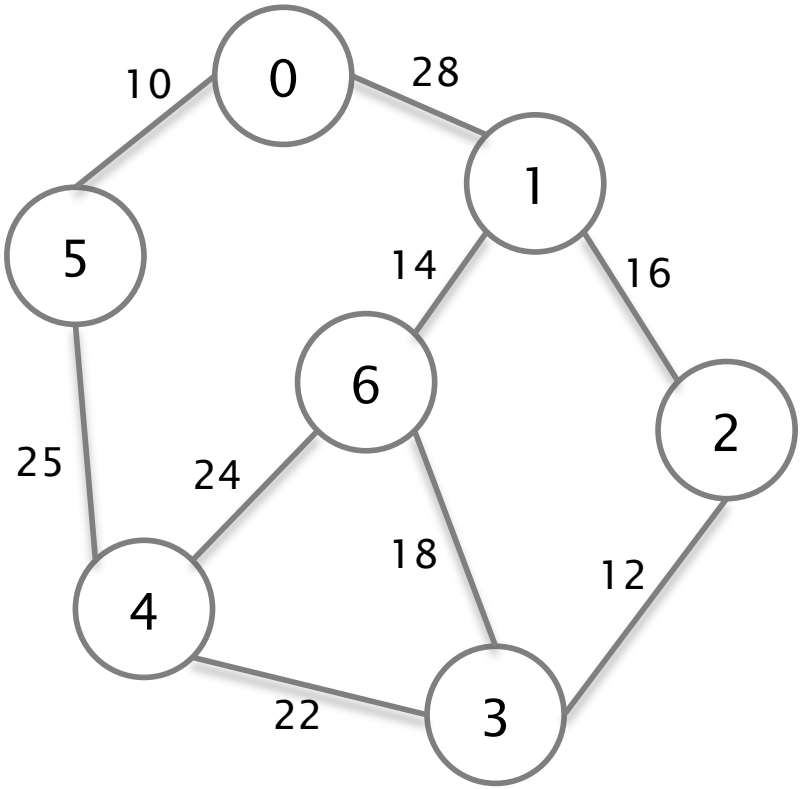


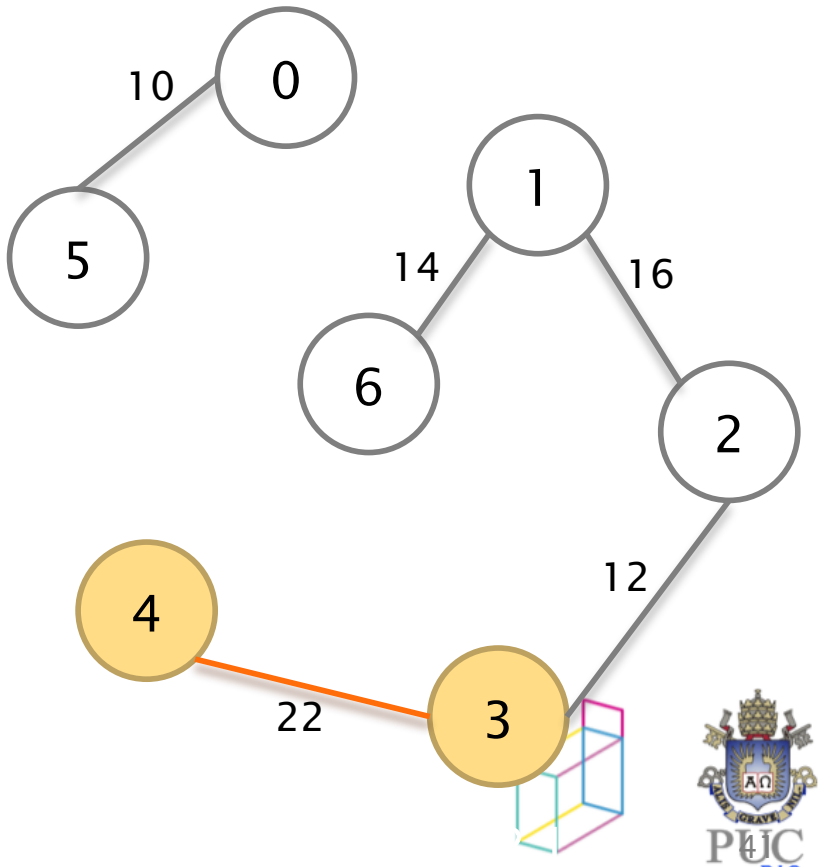


Algoritmo de Kruskal

2. Examine a aresta de menor custo.
Se ela unir duas árvores na floresta, inclua-a.

3. Repita o Passo (2) até todos os nós estarem conectados.



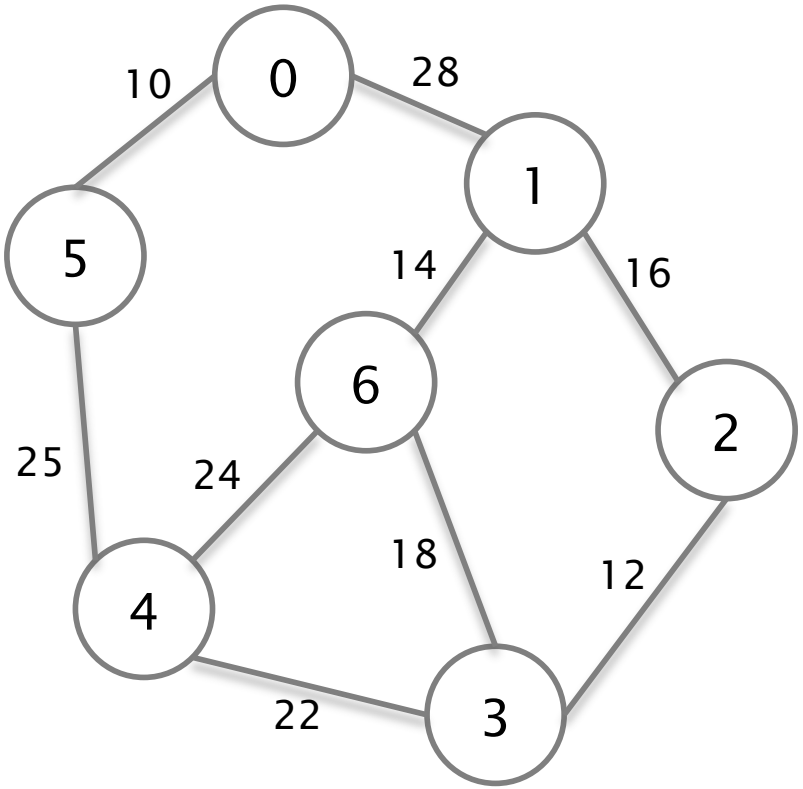


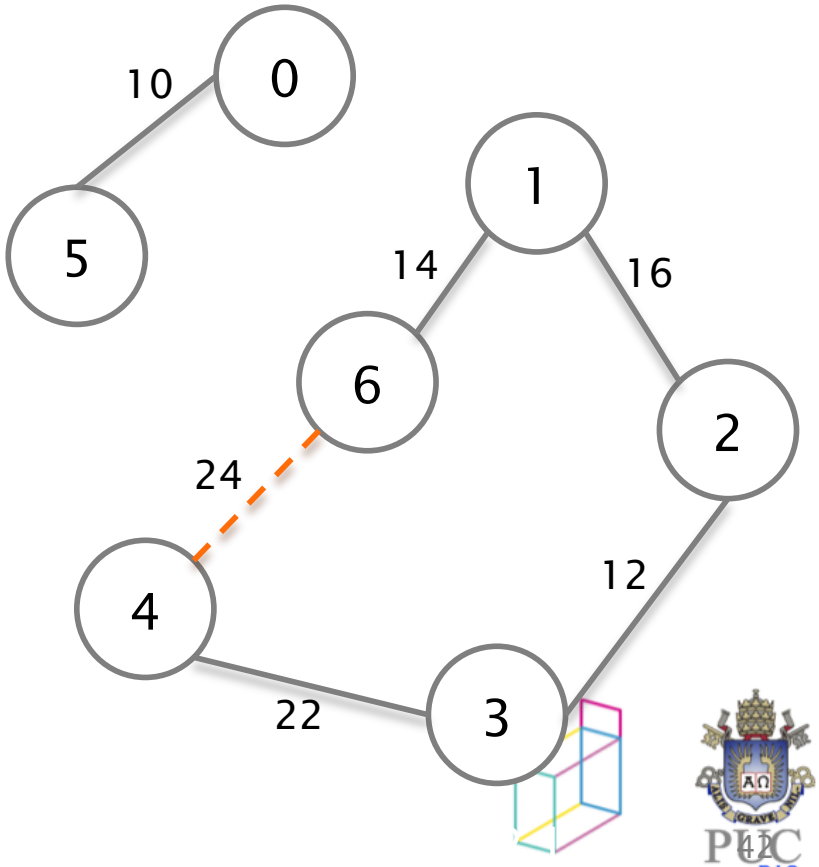
Algoritmo de Kruskal

2. Examine a aresta de menor custo.

Se ela unir duas árvores na floresta, inclua-a.

3. Repita o Passo (2) até todos os nós estarem conectados.





Algoritmo de Kruskal

2. Examine a aresta de menor custo.
Se ela unir duas árvores na floresta, inclua-a.

3. Repita o Passo (2) até todos os nós estarem conectados.