INF 1010
Estruturas de Dados Avançadas

Grafos

Aplicações de grafos

grafo	vértices	arestas
Cronograma	tarefas	restrições de preferência
Malha viária	interseções de ruas	ruas
Rede de água (telefônica,…)	Edificações (telefones,…)	Canos (cabos,…)
Redes de computadores	computadores	linhas
Software	funções	chamadas de função
Web	páginas Web	links
Redes Sociais	pessoas	relacionamentos
…		

Grafos

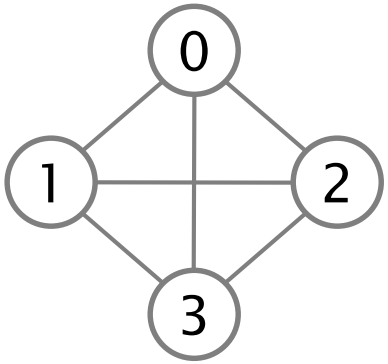
... não são estruturas de dados, e sim estruturas matemáticas que implementamos com estruturas de dados...

Grafo não dirigido

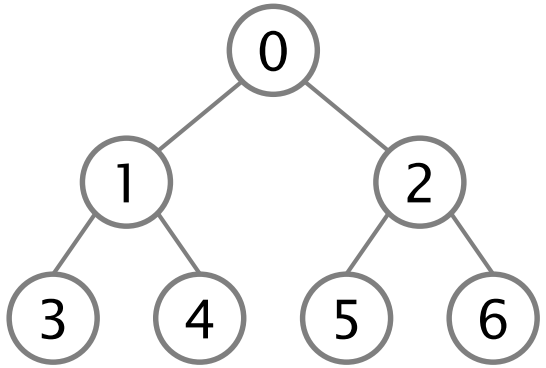
Um *grafo não dirigido* é um par G = (V,E), onde
V é um conjunto de *nós* ou *vértices* e
E é um conjunto de *arestas*

uma *aresta* é um conjunto de 2 vértices

Exemplos



vértices: V = {0,1,2,3}
arestas: E = {{0,1},{0,2},{0,3},{1,2},{1,3},{2,3}}



vértices : V = {0,1,2,3,4,5,6}
arestas: E = {{0,1},{0,2},{1,3},{1,4},{2,5},{2,6}}

Grafo dirigido (orientado, ou Digrafo)

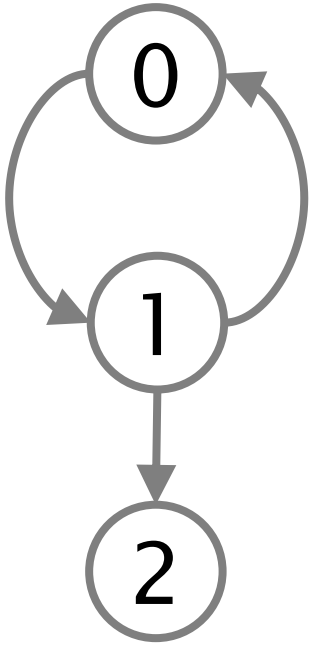
Um *grafo dirigido* é um par G = (V,E), onde
V é um conjunto de *n nós* ou *vértices* e
E é um conjunto de *m arcos*

um *arco* é um par ordenado de vértices

Exemplo

vértices: V = {0,1,2}

arcos: E = {(0,1), (1,0), (1,2)}

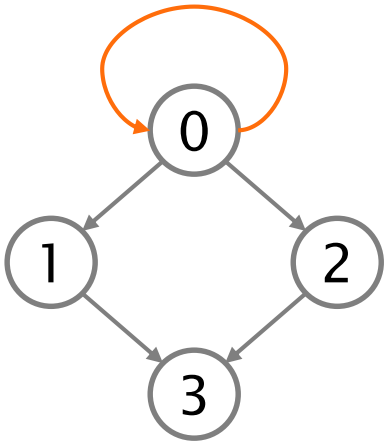


Grafo dirigido (orientado, ou digrafo)

Exemplo - Digrafo com auto-arco

vértices: V = {0,1,2,3}

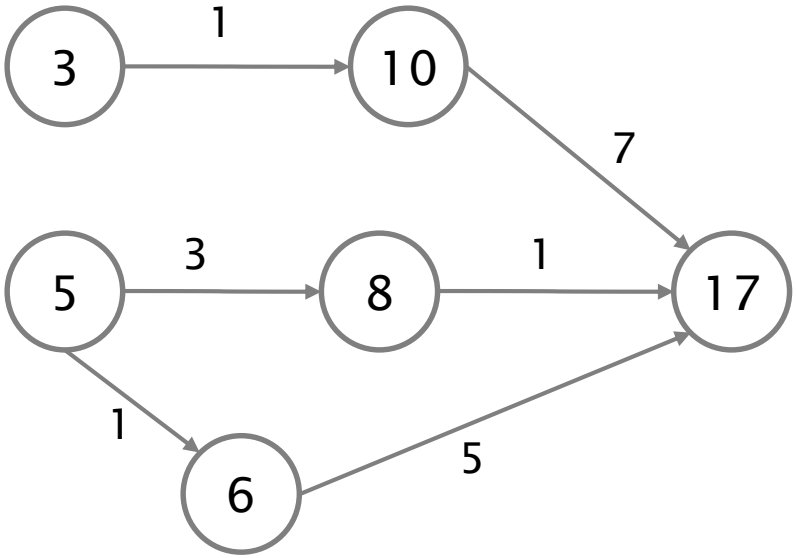
arcos: E = {(0,0), (0,1), (0,2), (1,3), (2,3)}



Grafo ponderado

Um *grafo ponderado* é uma tripla G = (V,E,p), onde
V é um conjunto de *n nós* ou *vértices* e
E é um conjunto de *m arcos*
p é uma função que atribui a cada arco um *peso*

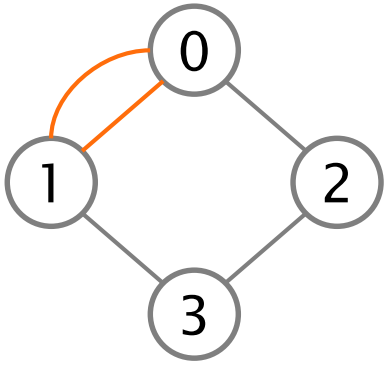
Exemplo



Multigrafo

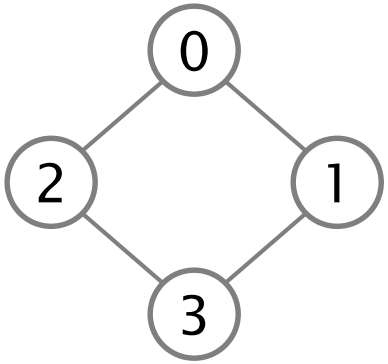
Um *multigrafo* é um grafo onde
dois nós podem estar conectados por mais de uma aresta

Exemplo

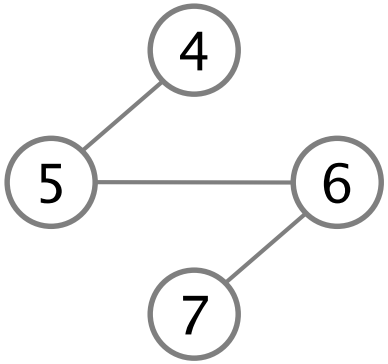


Vértices adjacentes

Vértices conectados por arestas



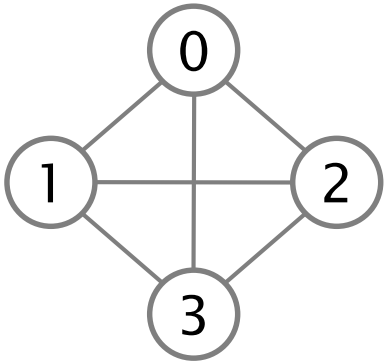
0 e 1
0 e 2
1 e 3
2 e 3



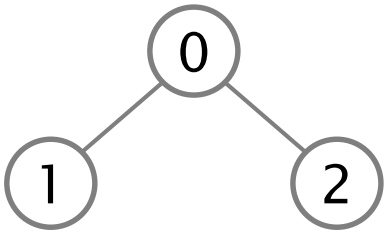
4 e 5
5 e 6
6 e 7

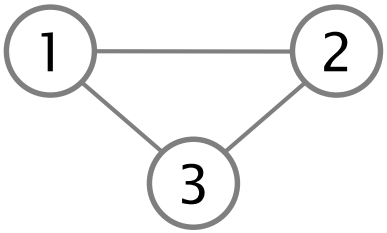
Subgrafo

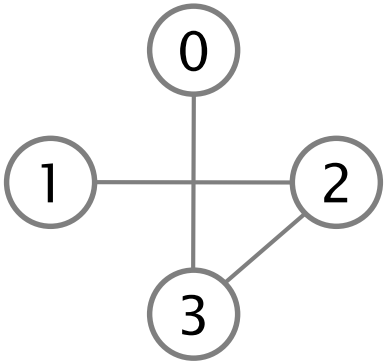
G_1



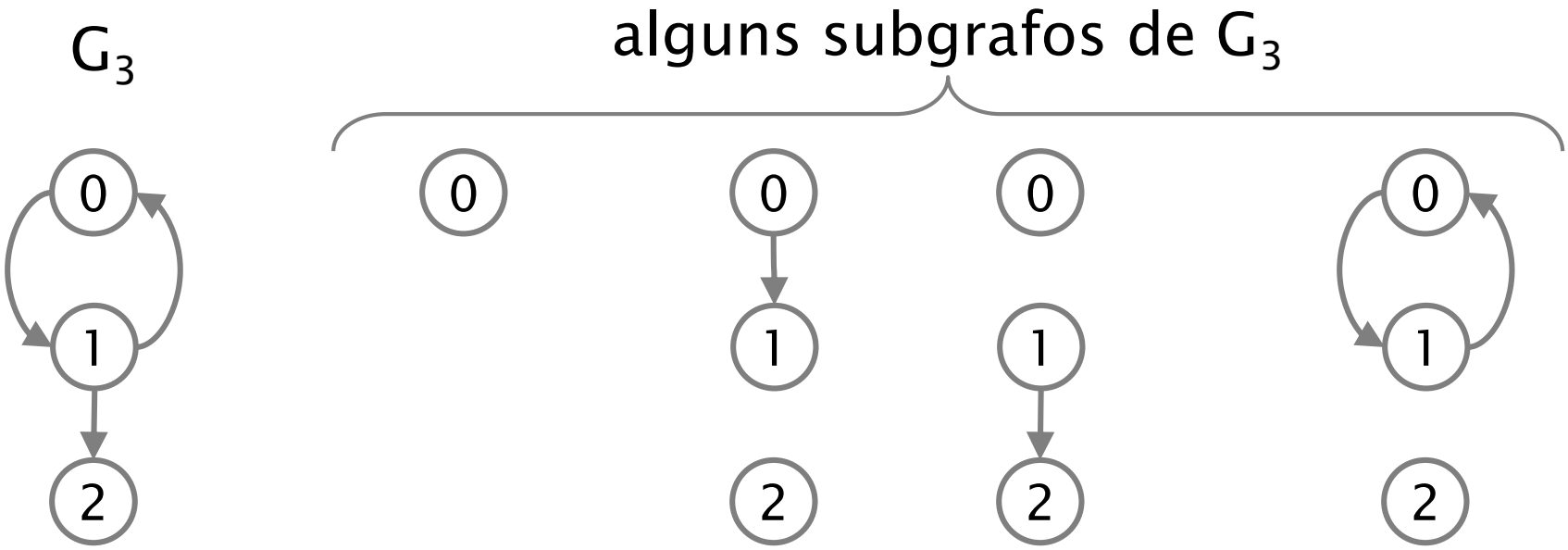
alguns subgrafos de G_1





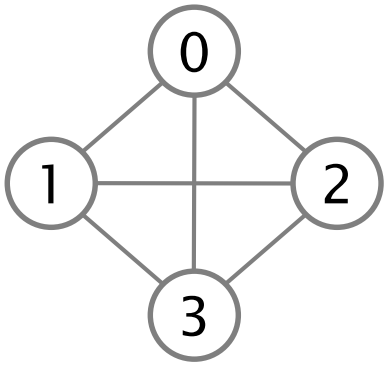


Subgrafo



Grafo completo

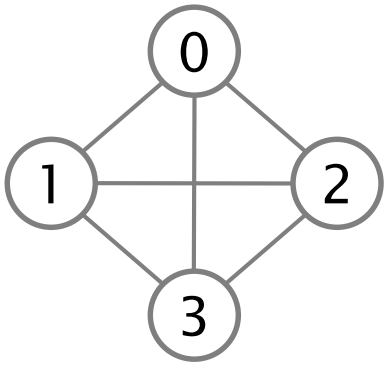
Um grafo não direcionado é *completo* sse cada vértice está conectado a cada um dos outros vértices por uma aresta



Quantas arestas há em um grafo completo de n vértices?

Grafo completo

Um grafo não direcionado é *completo* sse cada vértice está conectado a cada um dos outros vértices por uma aresta



Quantas arestas há em um grafo completo de n vértices?

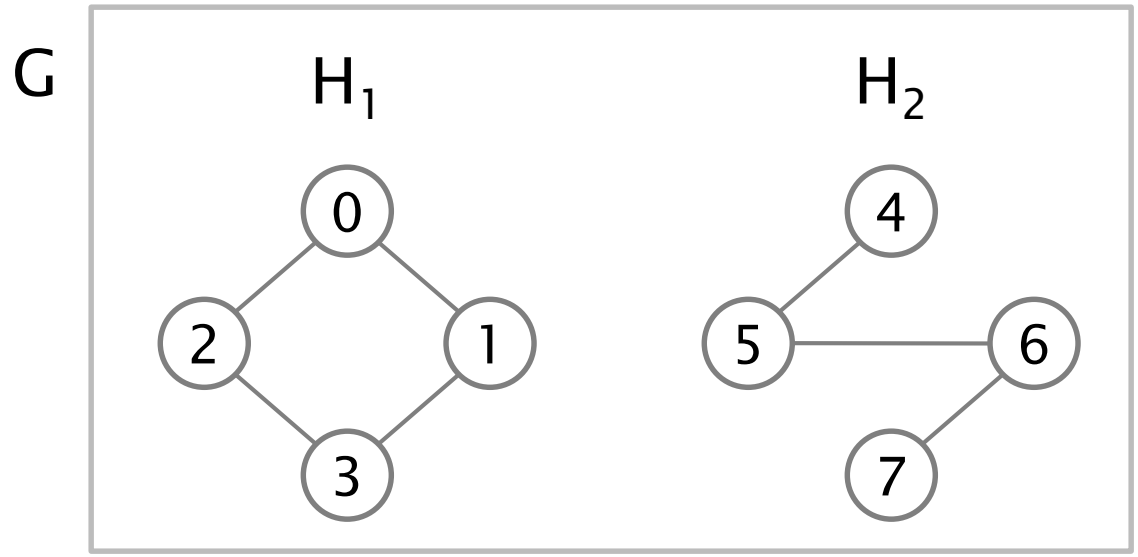
n (n-1)/2

PUC RIO

Grafo conectado (ou conexo)

Um grafo não direcionado é *conectado* ou *conexo* sse existe um caminho entre quaisquer dois vértices

Componente conexa de um grafo



Grau

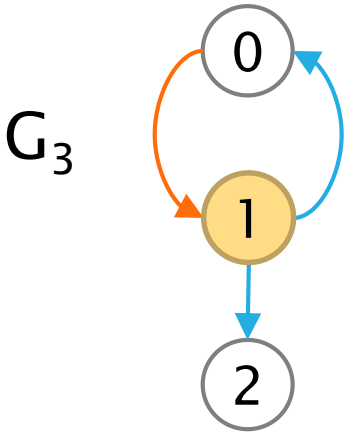
Um vértice possui *grau n* sse
há exatamente *n* arestas incidentes ao vértice

Exemplo:

grau do vértice 1: 3

grau de entrada do vértice 1: 1

grau de saída do vértice 1: 2



Caminhos e ciclos

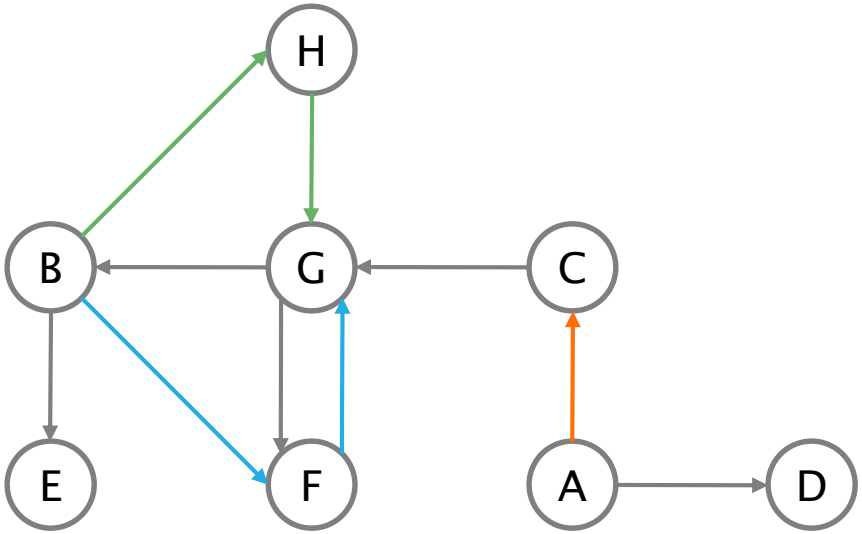
Caminho

de comprimento 1 entre A e C

de comprimento 2 entre B e G, passando por H

de comprimento 2 entre B e G, passando por F

de comprimento 3 de A a F



Ciclos

Um *ciclo* é um caminho de um nó para ele mesmo

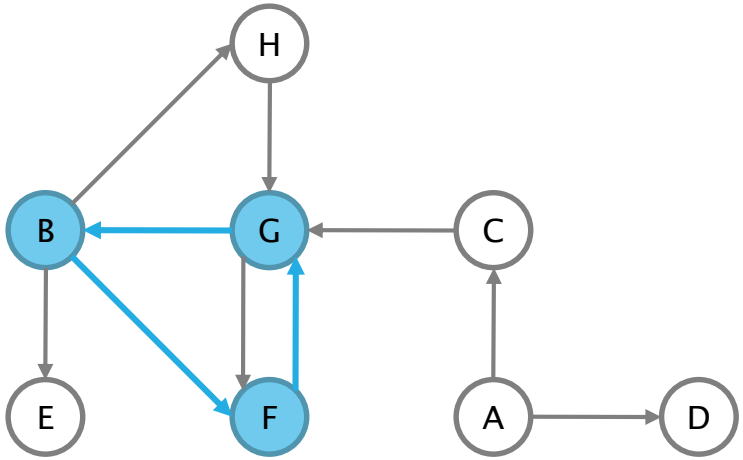
exemplo: B-F-G-B

Grafo cíclico

contém um ou mais ciclos

Grafo acíclico

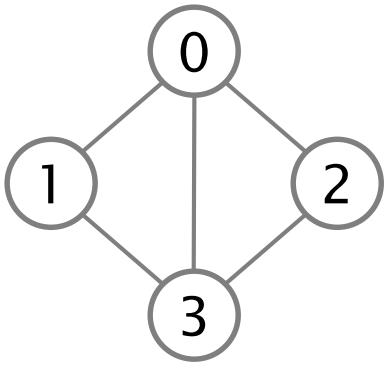
não contém ciclos

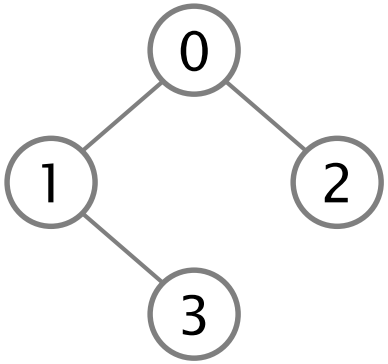


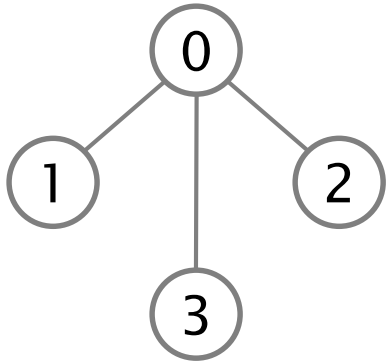
árvore geradora

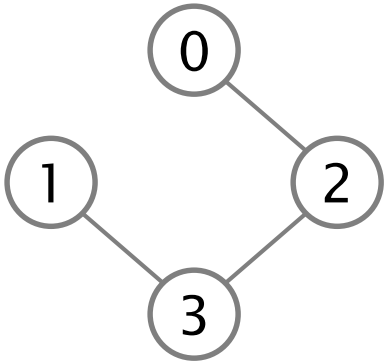
subgrafo acíclico contendo todos os vértices com caminhos entre quaisquer 2 vértices

Grafo G







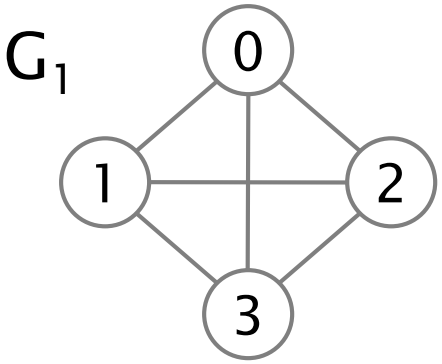


Representações de grafo

- Matriz de adjacências
- Listas de adjacências (incidências)

Matriz de adjacências

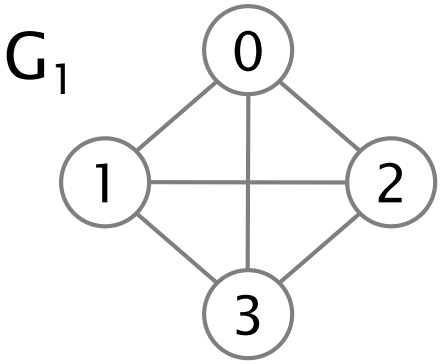
$$mat[i][j] = \begin{cases} 1, \text{ se houver uma aresta do nó i para o nó j} \\ 0, \text{ caso contrário} \end{cases}$$



	0	1	2	3
0	0	1	1	1
1	1	0	1	1
2	1	1	0	1
3	1	1	1	0

Matriz de adjacências

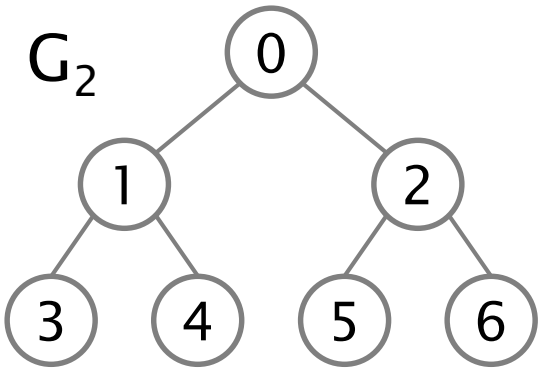
$$mat[i][j] = \begin{cases} 1, \text{ se houver uma aresta do nó i para o nó j} \\ 0, \text{ caso contrário} \end{cases}$$



	0	1	2	3
0	0	1	1	1
1	1	0	1	1
2	1	1	0	1
3	1	1	1	0

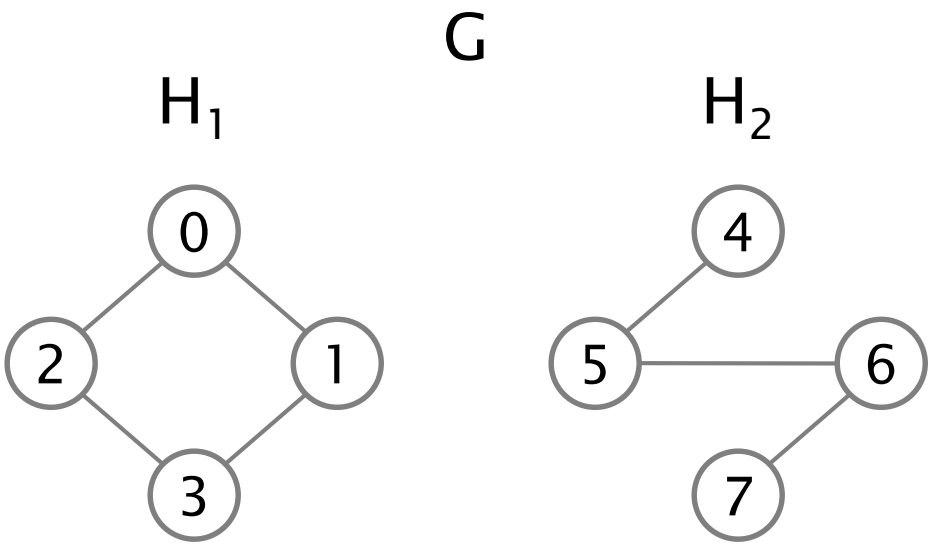
matrizes simétricas para grafos não direcionados

Matriz de adjacências



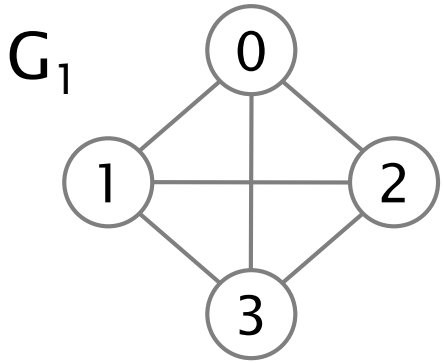
	0	1	2	3	4	5	6
0	0	1	1	0	0	0	0
1	1	0	0	1	1	0	0
2	1	0	0	0	0	1	1
3	0	1	0	0	0	0	0
4	0	1	0	0	0	0	0
5	0	0	1	0	0	0	0
6	0	0	1	0	0	0	0

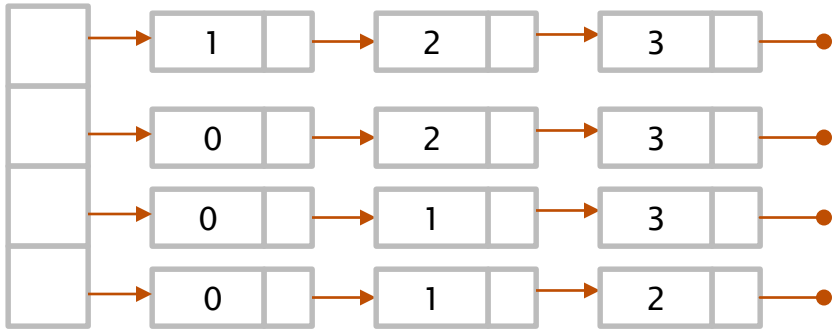
Matriz de adjacências



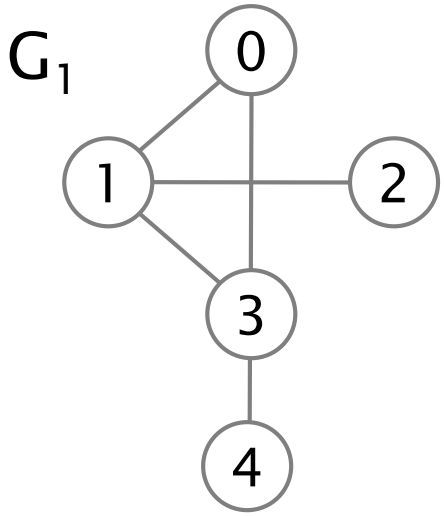
	0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0
2	1	0	0	1	0	0	0	0
3	0	1	1	0	0	0	0	0
4	0	0	0	0	0	1	0	0
5	0	0	0	0	1	0	1	0
6	0	0	0	0	0	1	0	1
7	0	0	0	0	0	0	1	0

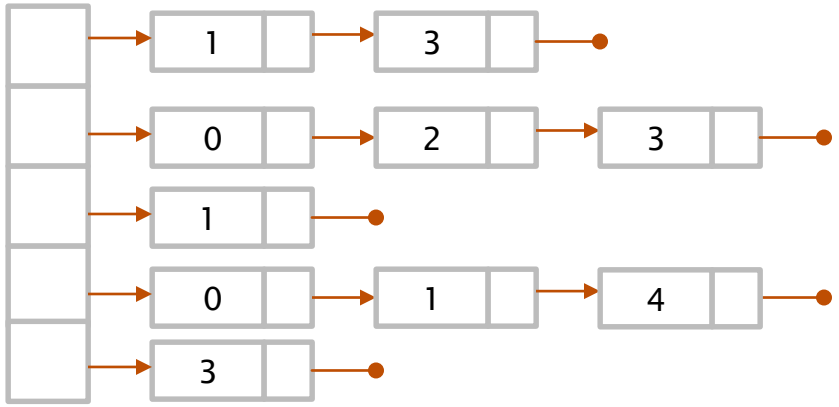
Listas de adjacências





Listas de adjacências





Listas de adjacências

o que vamos usar no lab:

```
typedef struct _viz Viz;
struct _viz {
  int noj;
  float peso;
  Viz*  prox;
};

struct _grafo {
  int nv;    /* numero de nos ou vertices */
  int na;    /* numero de arestas */
  Viz** viz; /* viz[i] aponta para a lista de arestas incidindo em i */
};
```

Percursos em grafos

passeios percorrendo todos os nós de um grafo

Percursos em grafos

em profundidade (*depth-first search - dfs*)

- arestas que partem do vértice visitado por último

em largura (*breadth-first search - bfs*)

- arestas que partem do vértice visitado primeiro

guloso (*greedy*)

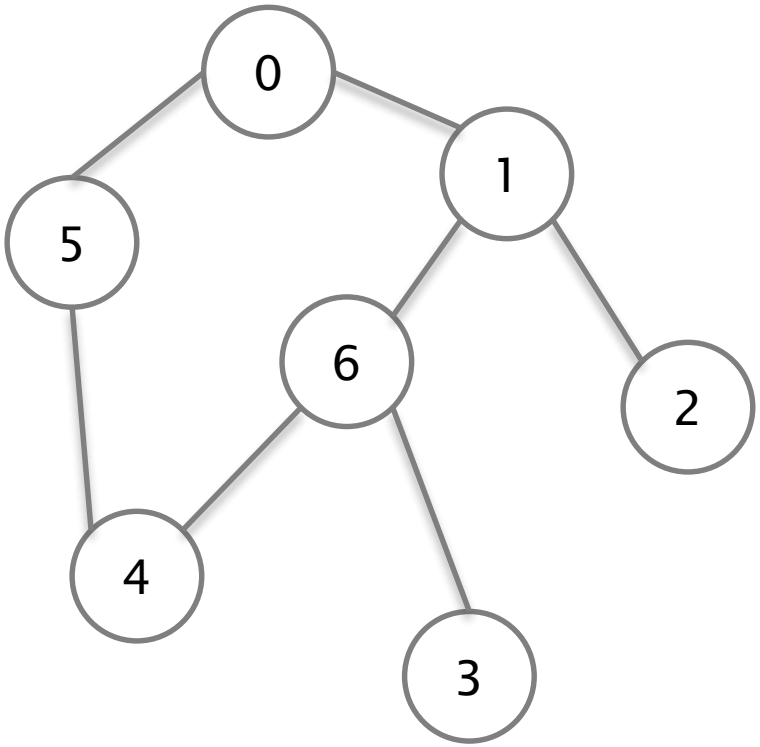
- arestas de menor custo (tipicamente procurando menor caminho)

Percursos em grafos

Cada vértice examinado deve ser marcado como visitado

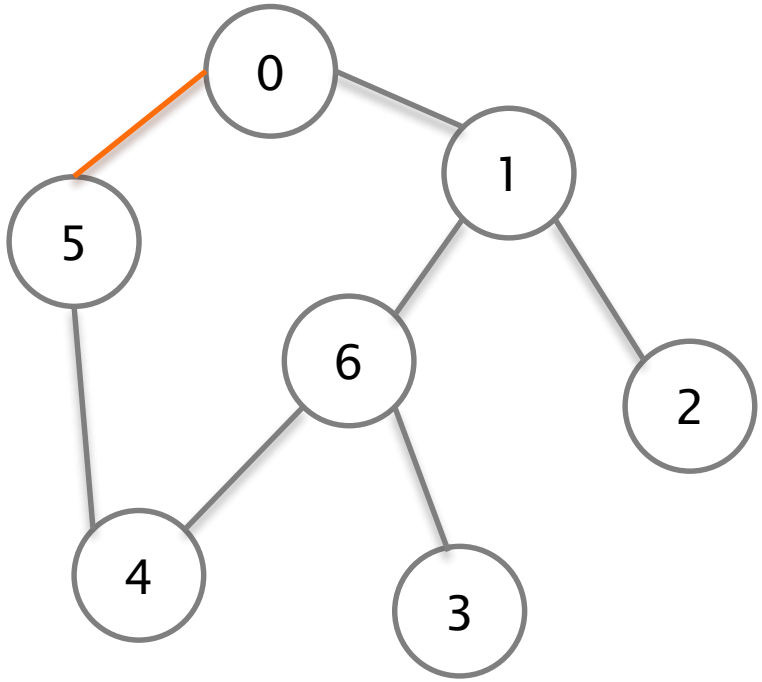
Por quê?

dfs iniciando em 0



dfs iniciando em 0

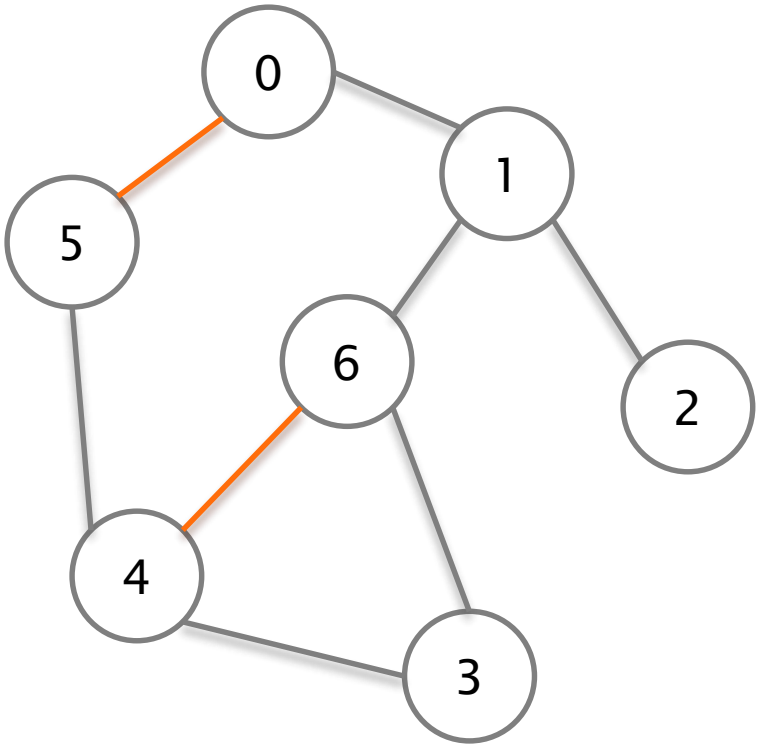
```
dfs(0)
    dfs(1)
        dfs(2)
        dfs(6)
            dfs(3)
            dfs(4)
                dfs(5)
```



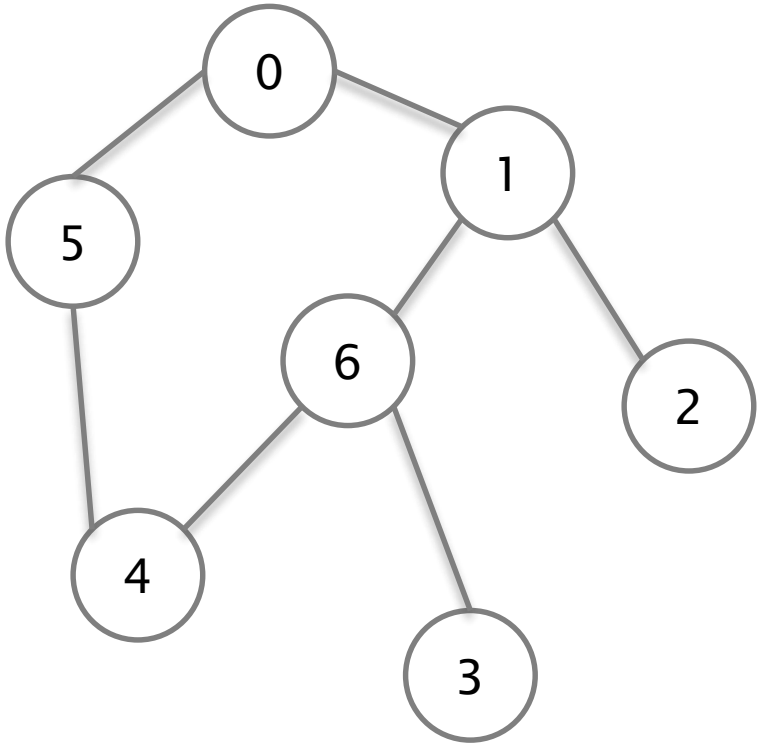
estado fica na pilha de chamadas recursivas

dfs iniciando em 0 – outro exemplo

dfs(0)
  dfs(1)
    dfs(2)
    dfs(6)
      dfs(3)
        dfs(4)
          dfs(6)
        dfs(5)
          dfs(0)

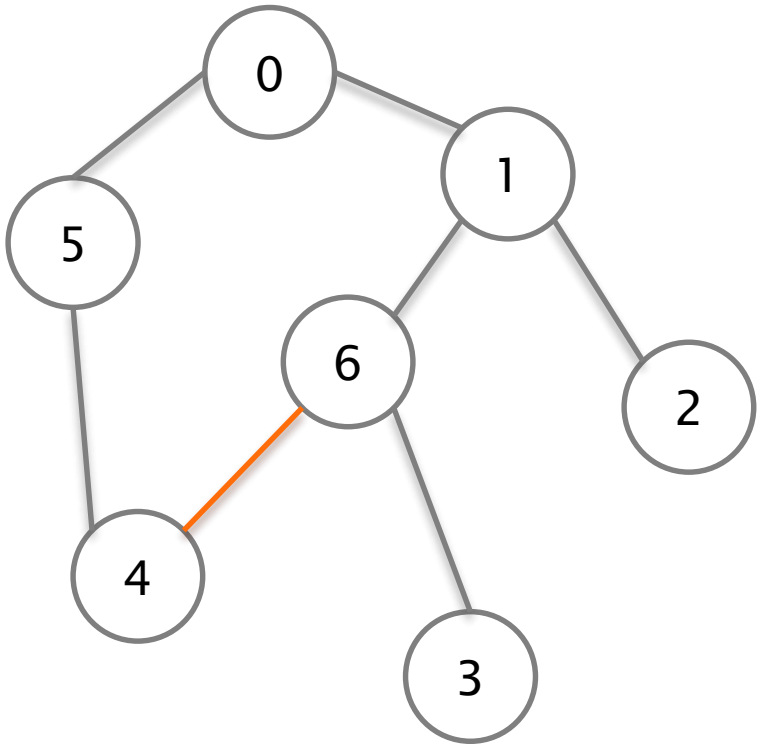


bfs iniciando em 0



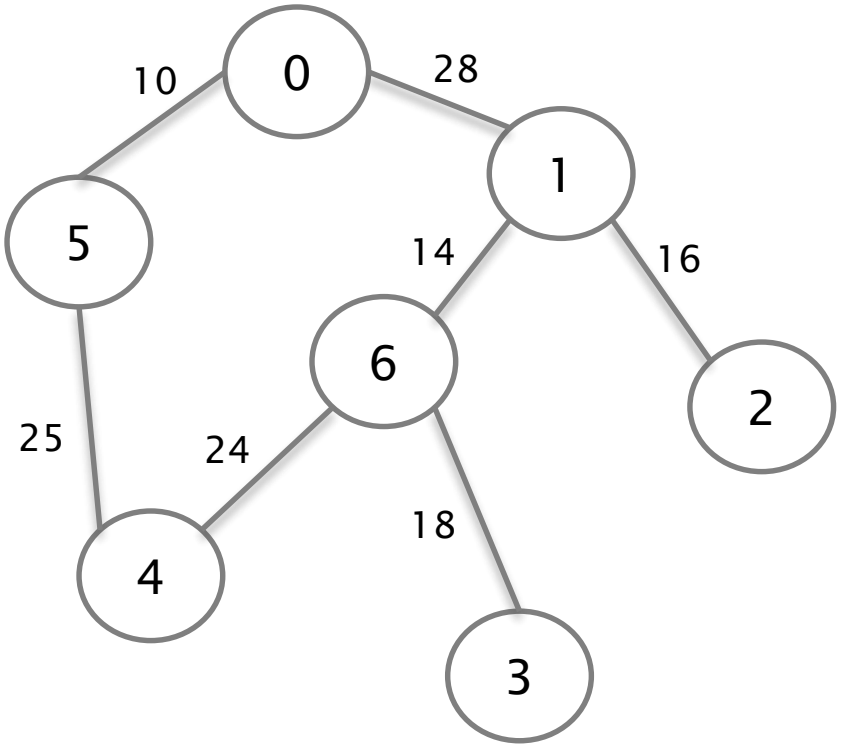
bfs iniciando em 0

bfs(0)	
-> enfileira 1, 5	[1,5]
bfs(1)	
-> enfileira 2, 6	[5,2,6]
bfs(5)	
-> enfileira 4	[2,6,4]
bfs(2)	
	[6,4]
bfs(6)	
-> enfileira 3	[4,3]
bfs(4)	
	[3]
bfs(3)	
	[]



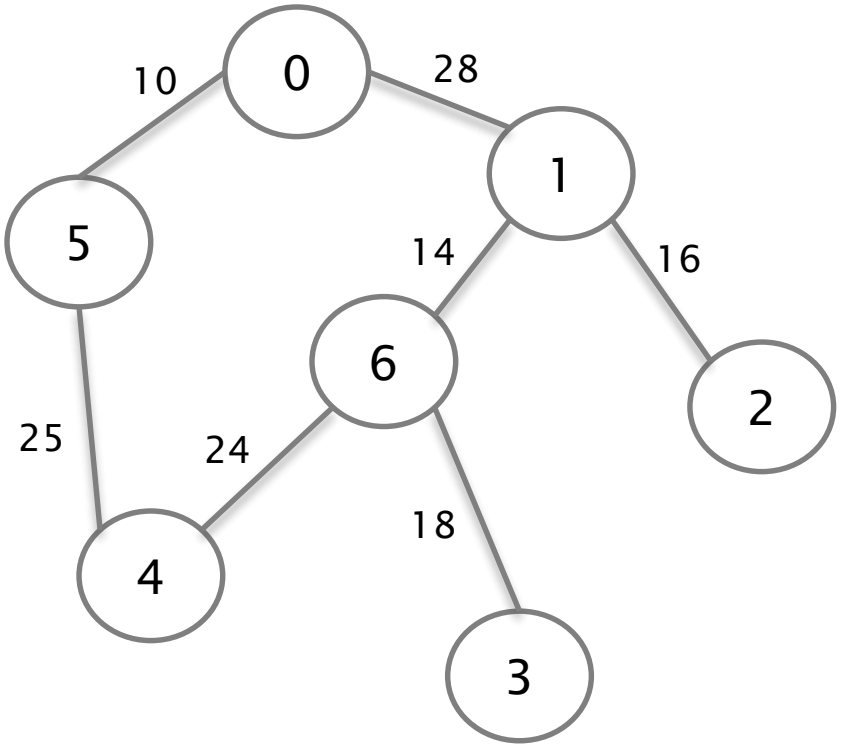
problemas comuns

- árvores geradoras
  - Diferentes formas de encontrar uma árvore geradora
  - 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

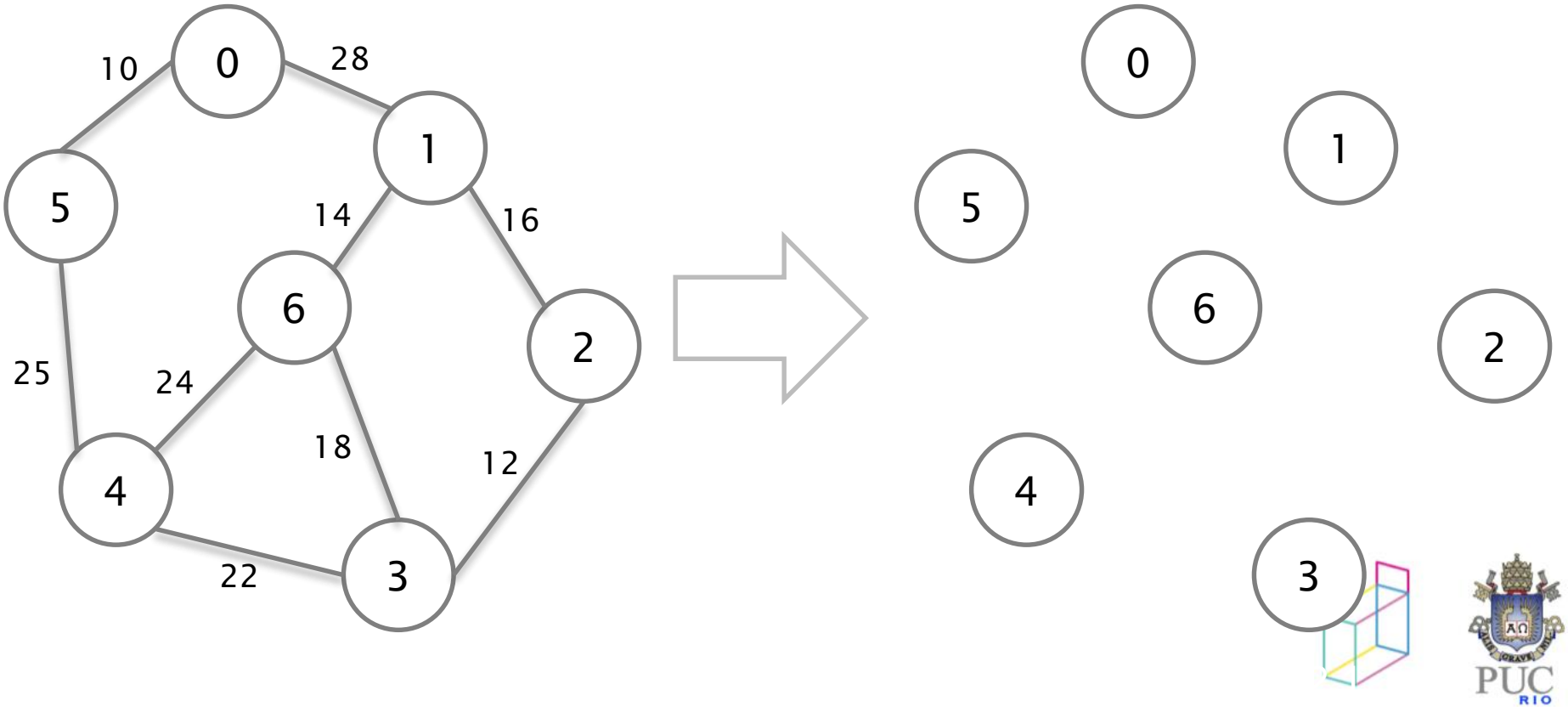
Entrada: Um grafo ponderado G = (V,E,p)

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)

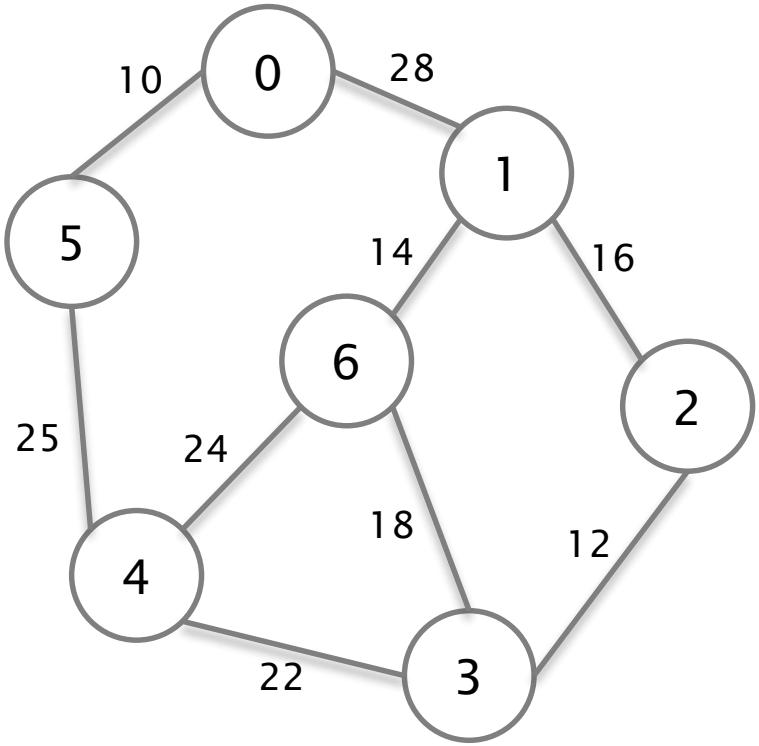


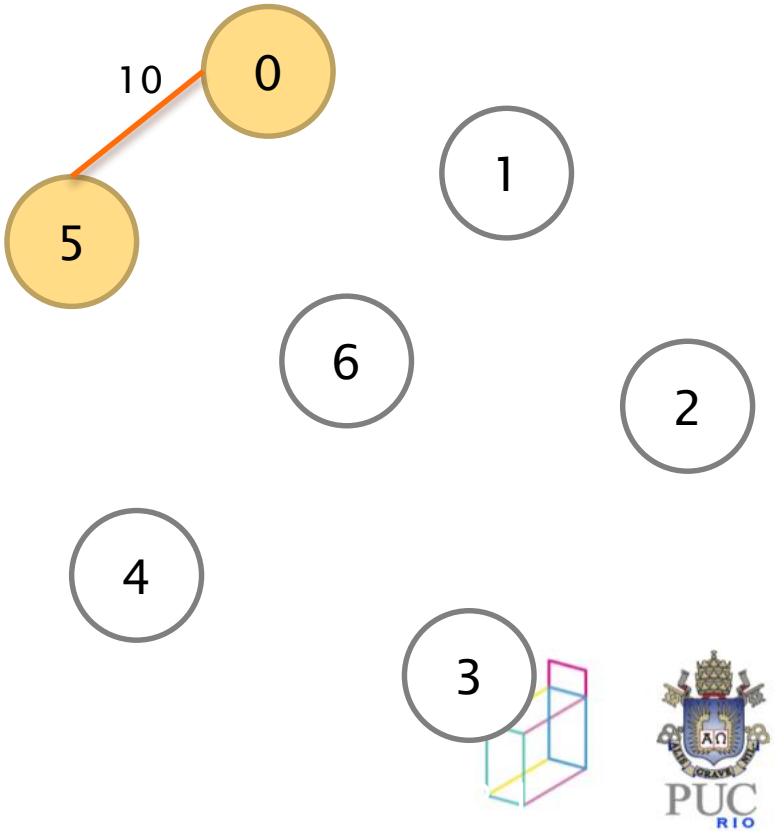
Algoritmo de Kruskal

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.



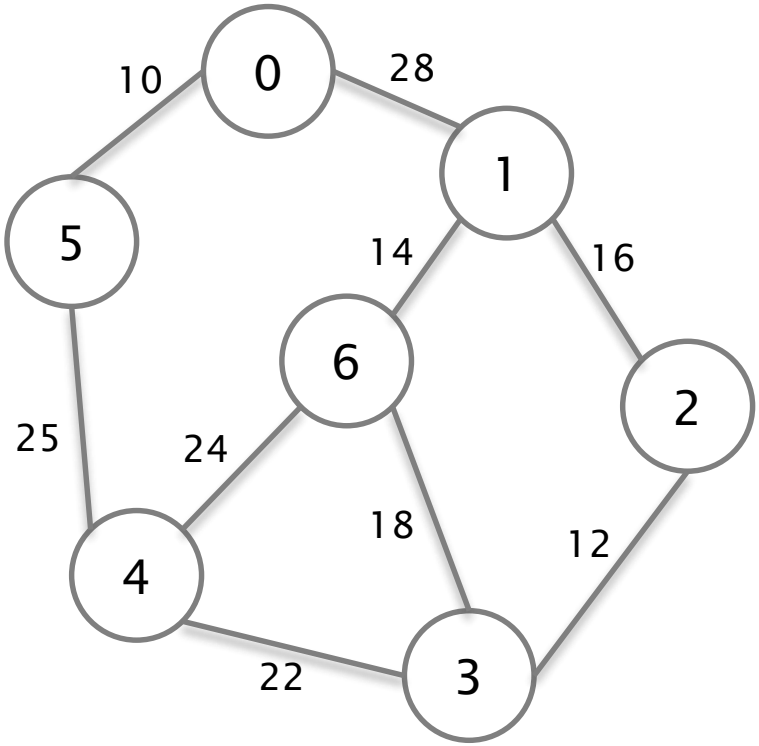


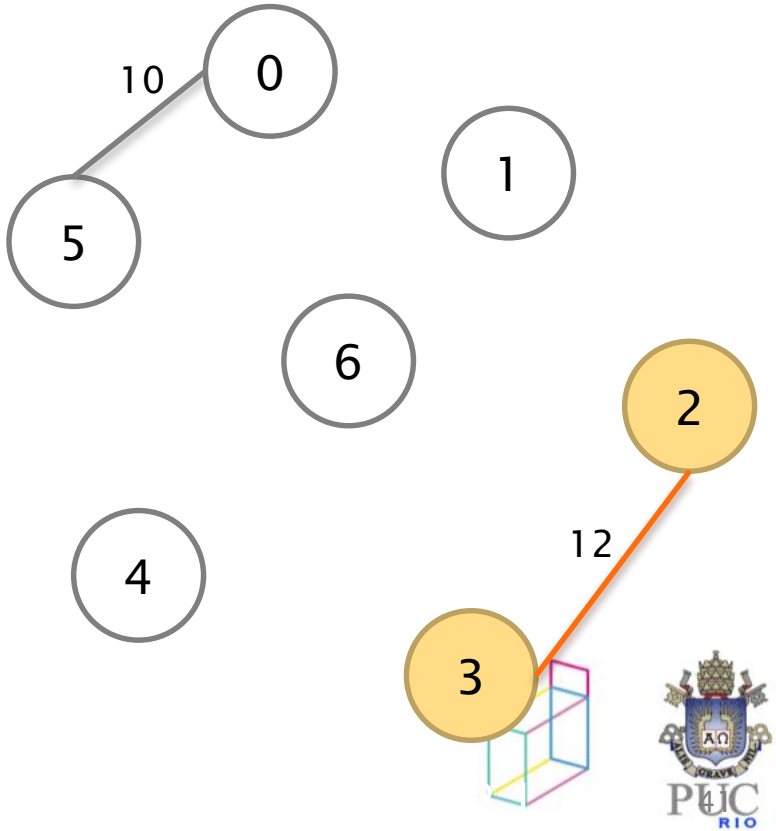
Algoritmo de Kruskal

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.



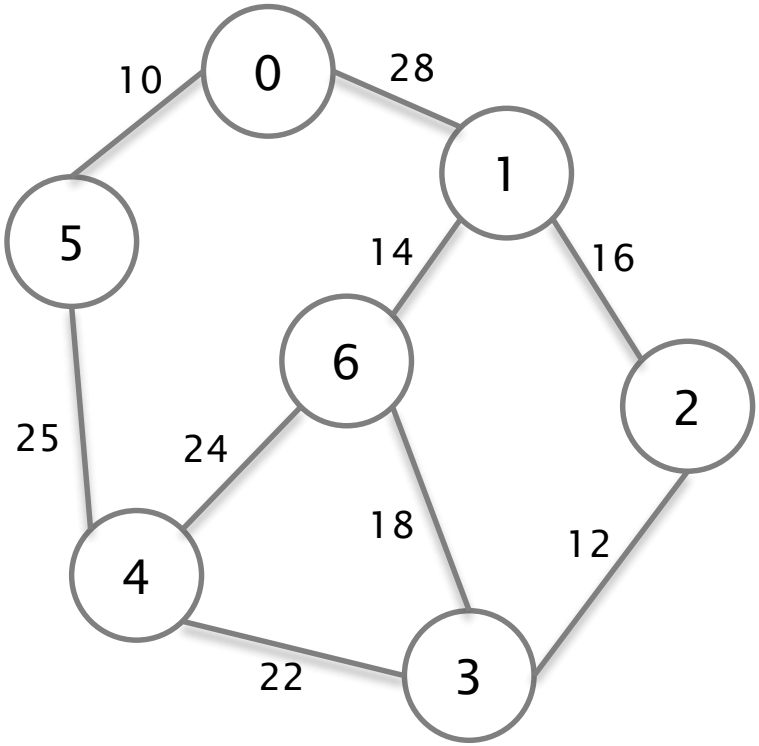


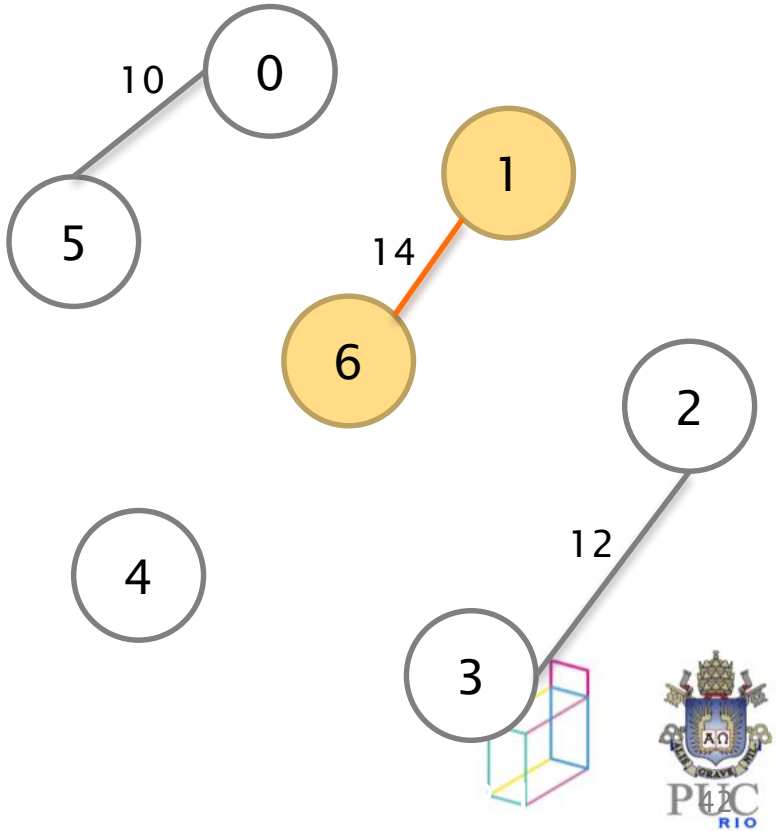
Algoritmo de Kruskal

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.



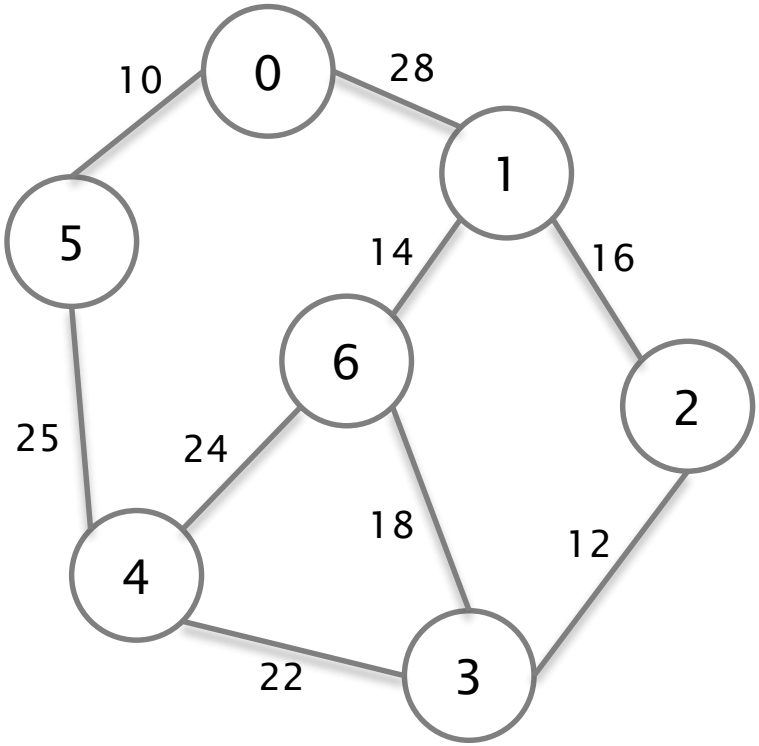


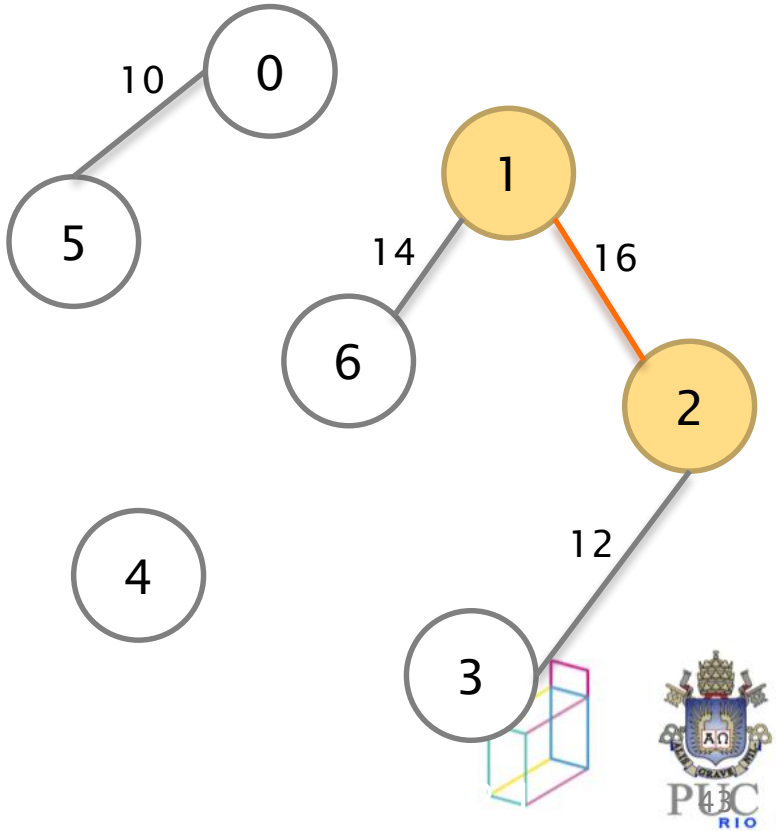
Algoritmo de Kruskal

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.



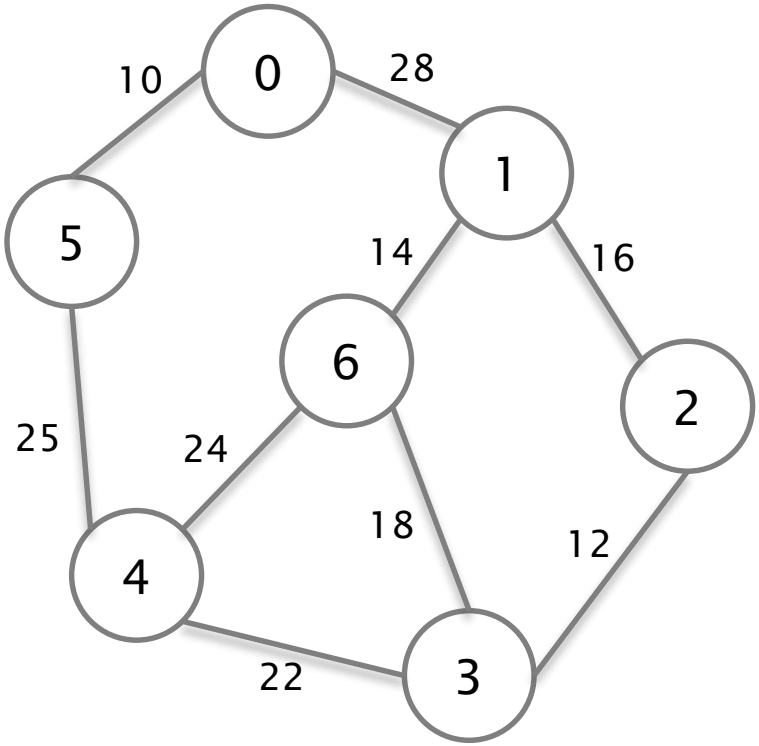


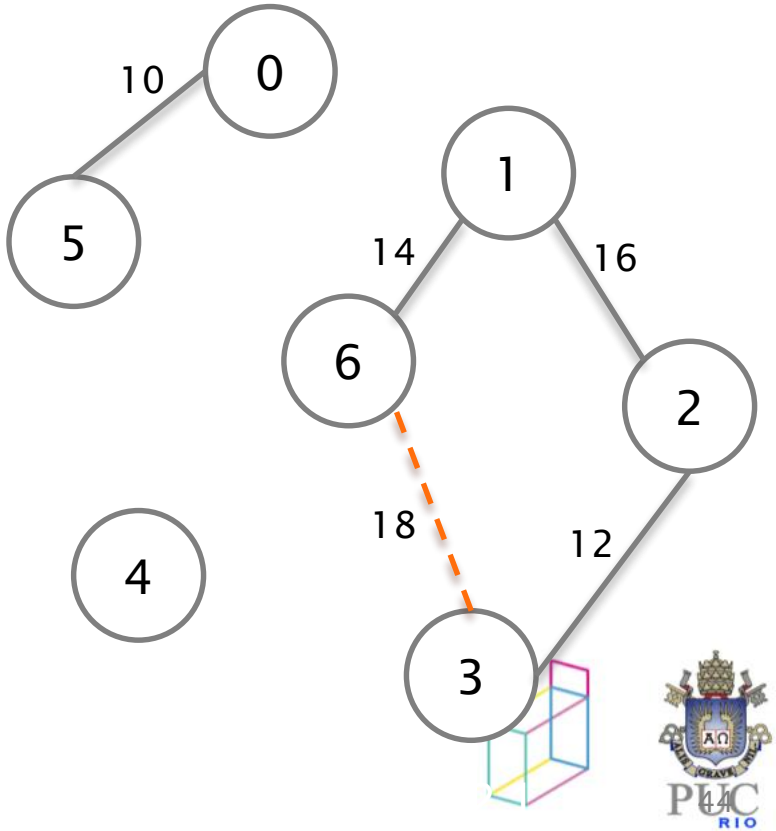
Algoritmo de Kruskal

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.



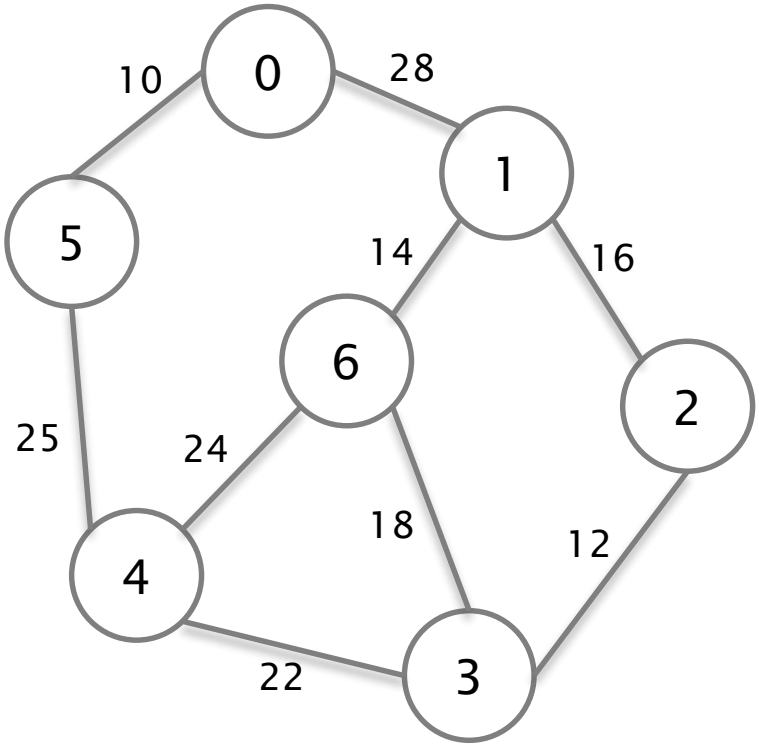


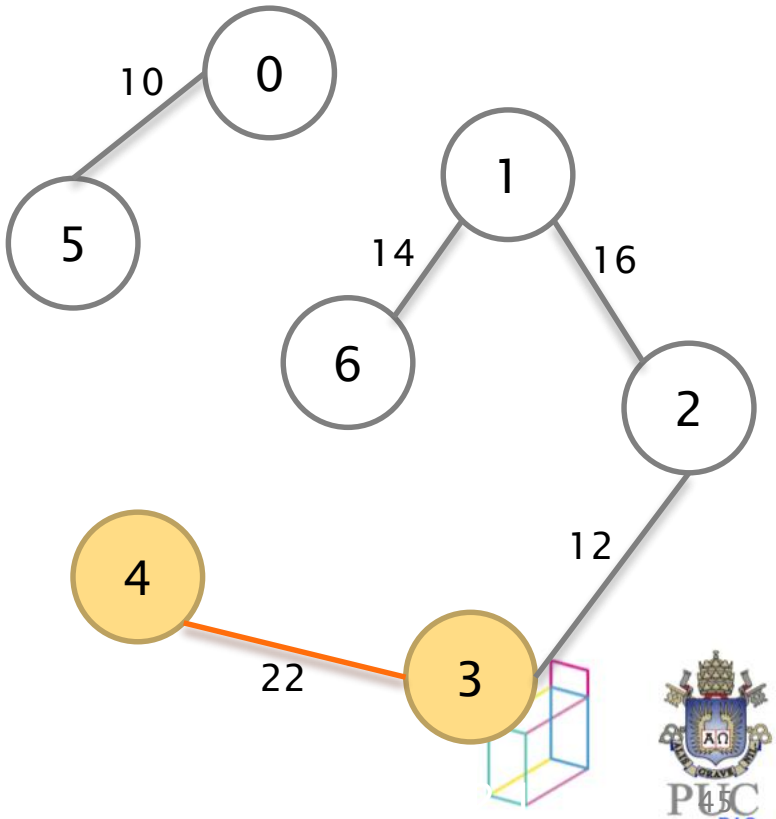
Algoritmo de Kruskal

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.



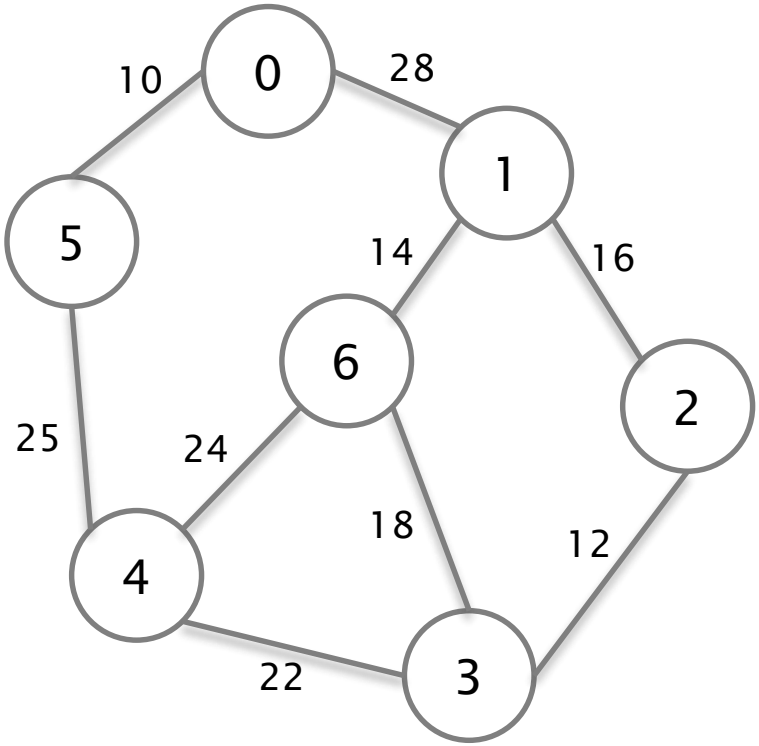


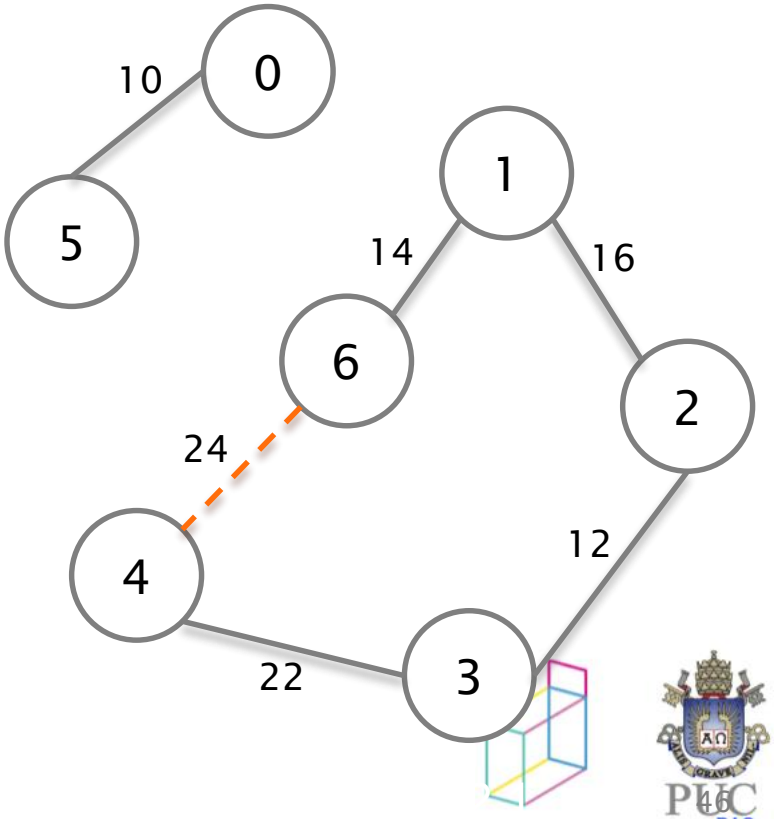
Algoritmo de Kruskal

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.