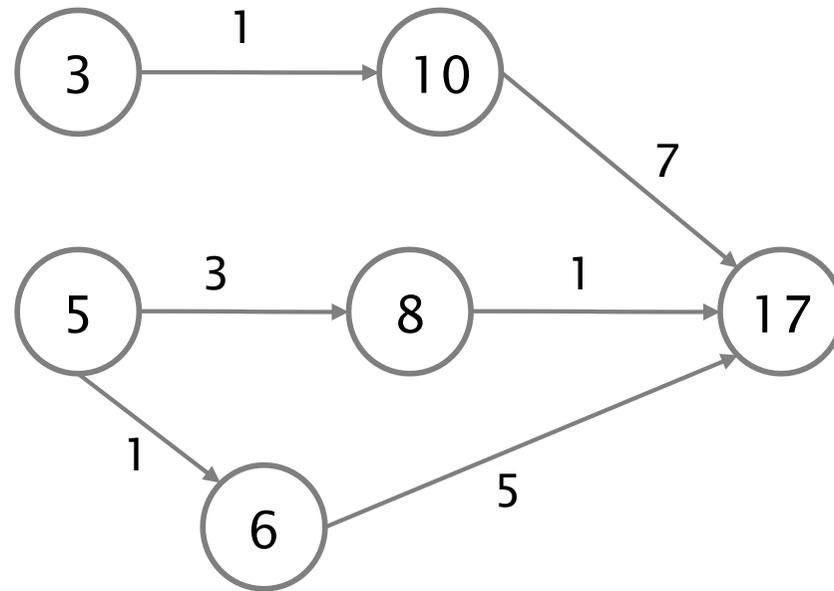
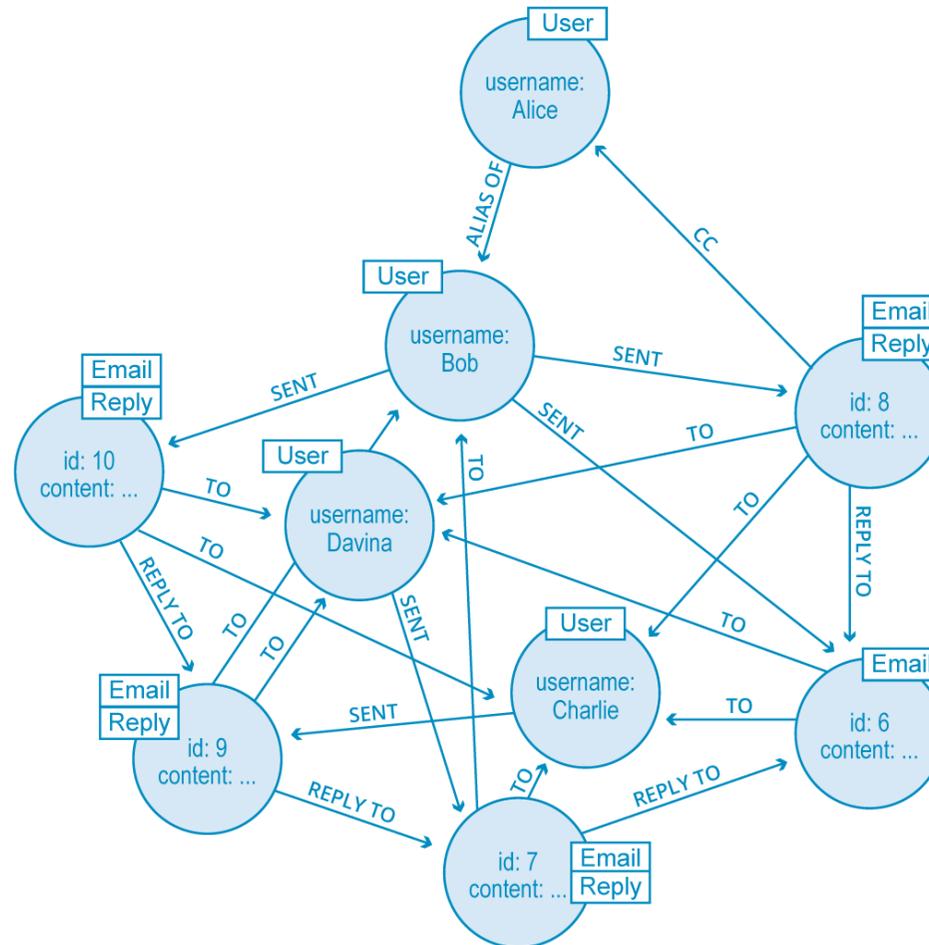


INF 1010
Estruturas de Dados Avançadas

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

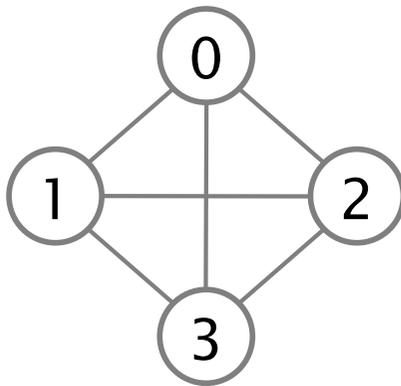
Um *grafo não dirigido* é um par $G = (V, E)$, onde

V é um conjunto de *nós* ou *vértices* e

E é um conjunto de *arestas*

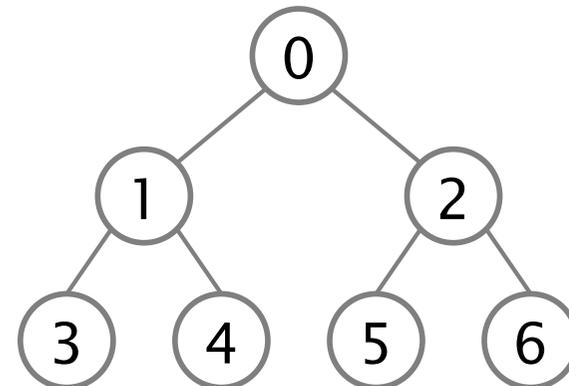
uma *aresta* é um conjunto de 2 vértices

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)

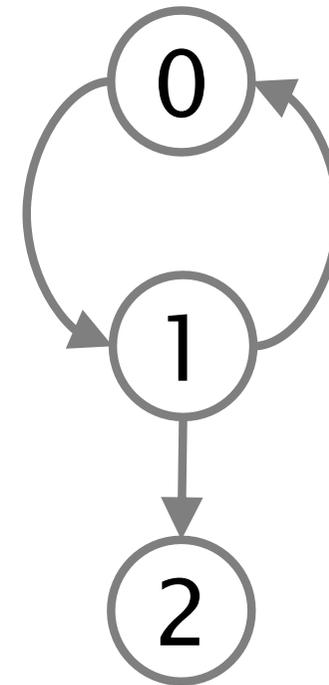
Um *grafo dirigido* é um par $G = (V, E)$, onde V é um conjunto de n nós ou *vértices* e E é um conjunto de m *arcos*

um *arco* é um par ordenado de vértices

Exemplo

vértices: $V = \{0, 1, 2\}$

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

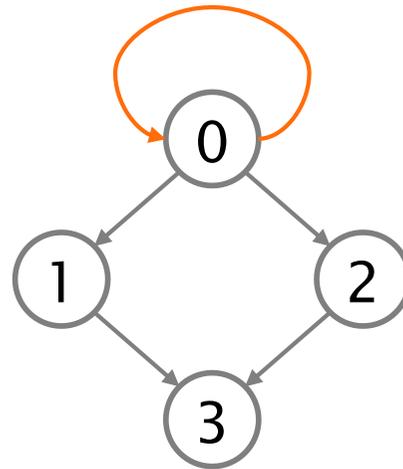


Grafo dirigido (orientado, ou digrafo)

Exemplo - Digrafo com auto-arco

vértices: $V = \{0,1,2,3\}$

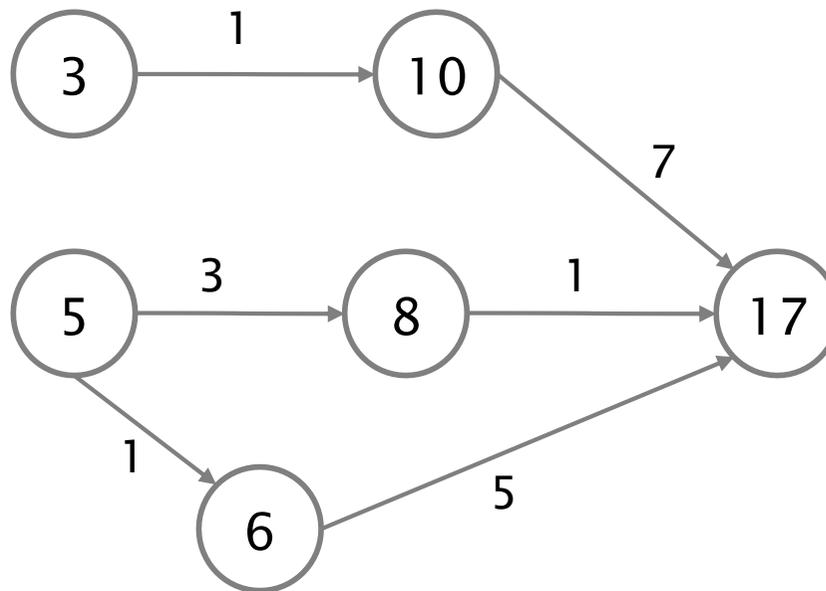
arcos: $E = \{(0,0), (0,1), (0,2), (1,3), (2,3)\}$



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*

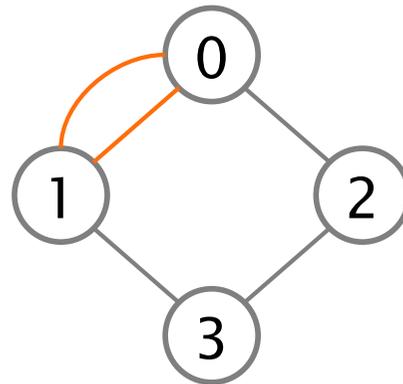
Exemplo



Multigrafo

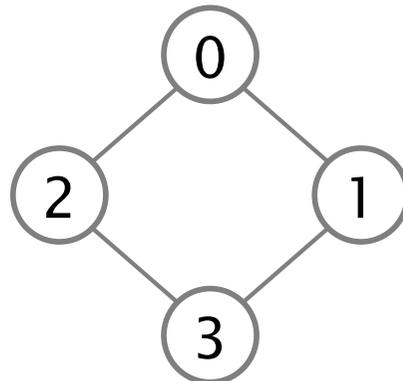
Um *multigrafo* é um grafo onde dois nós podem estar conectados por mais de uma aresta

Exemplo

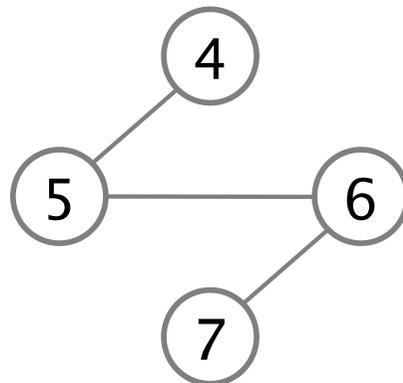


Vértices adjacentes

Vértices conectados por arestas



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

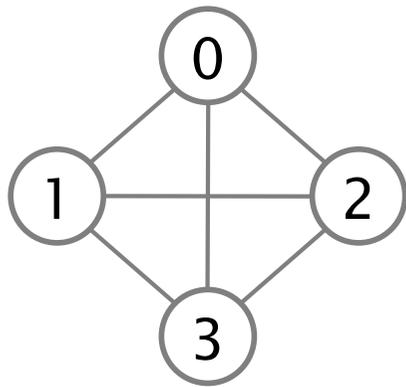


4 e 5
5 e 6
6 e 7

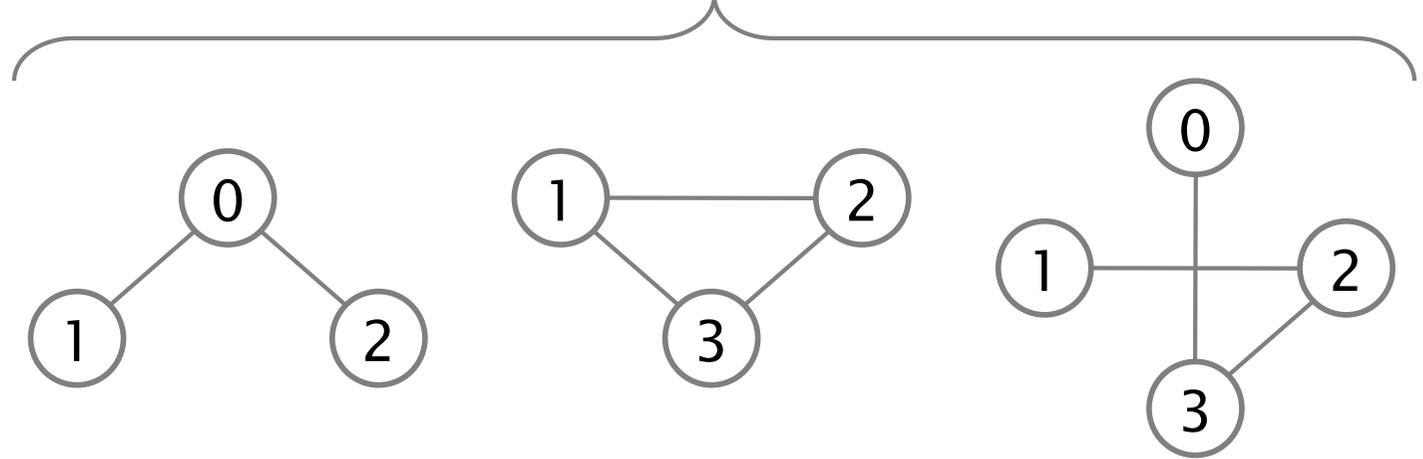


Subgrafo

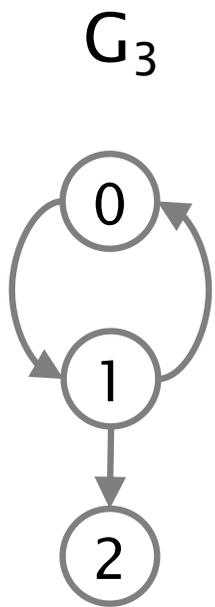
G_1



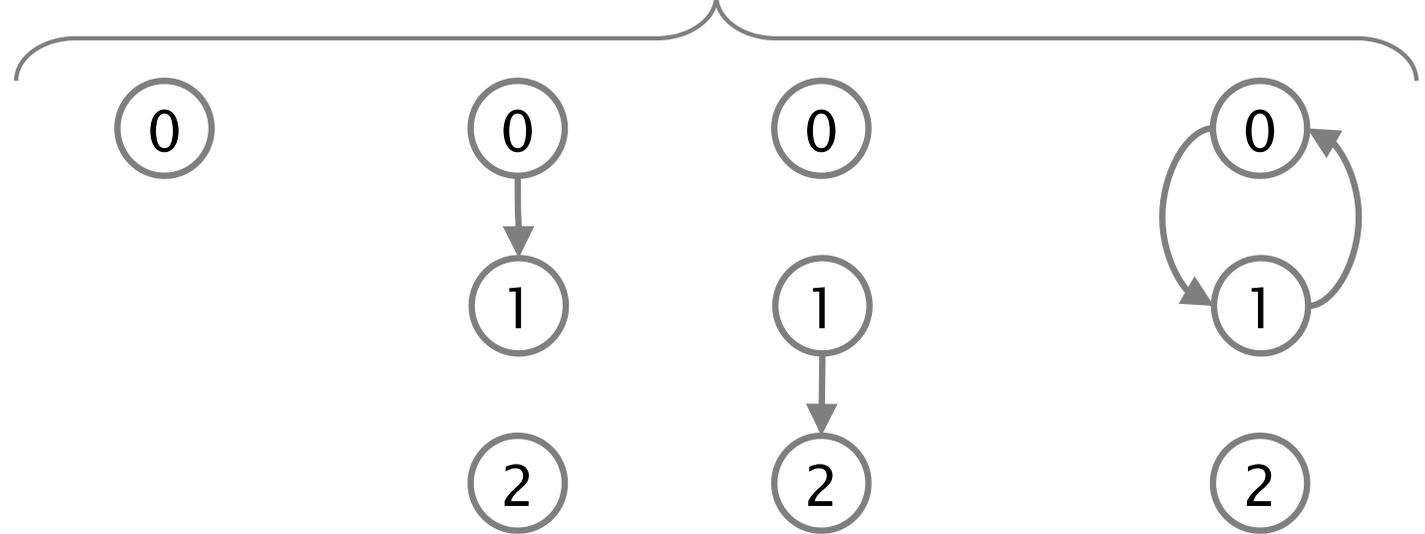
alguns subgrafos de G_1



Subgrafo

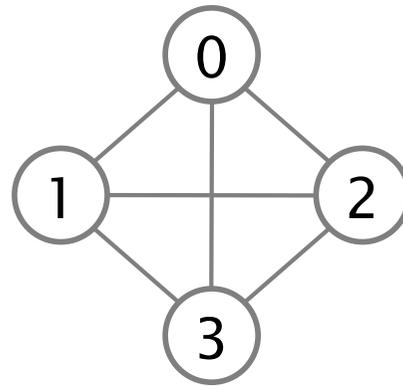


alguns subgrafos de G_3



Grafo completo

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

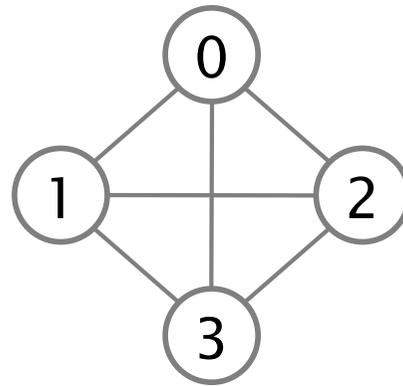


Quantas arestas há em um grafo completo de n vértices?



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?

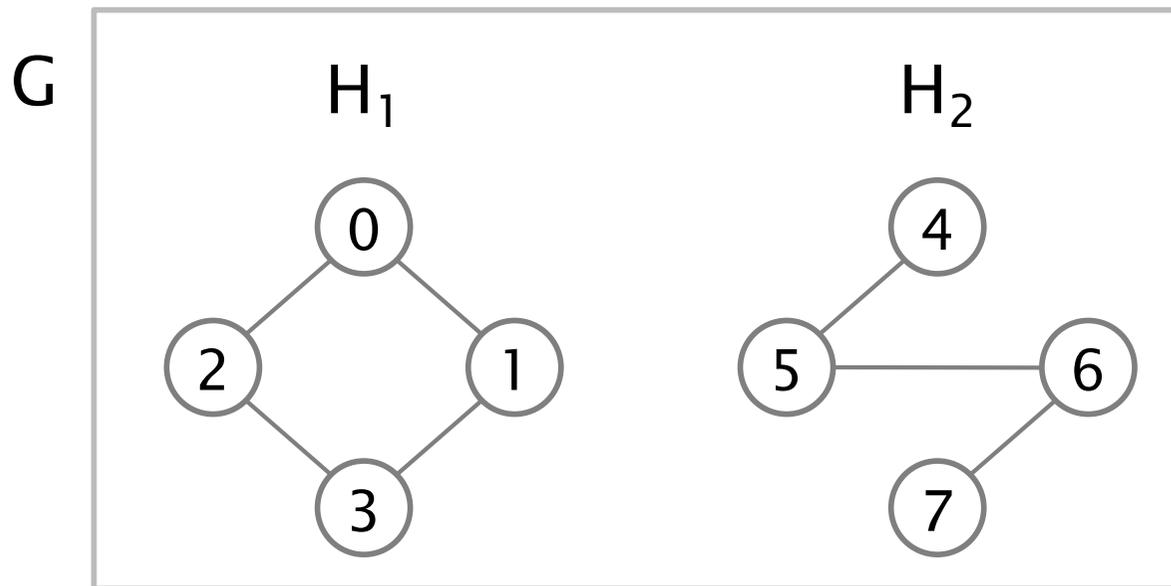
$$n(n-1)/2$$



Grafo conectado

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

Componente conexa de um grafo



Grau

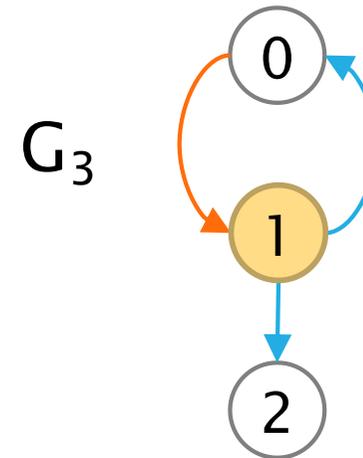
Um vértice possui *grau* n sse há exatamente n arestas incidentes ao vértice

Exemplo:

grau do vértice 1: 3

grau de **entrada** do vértice 1: 1

grau de **saída** do vértice 1: 2



Caminhos e ciclos

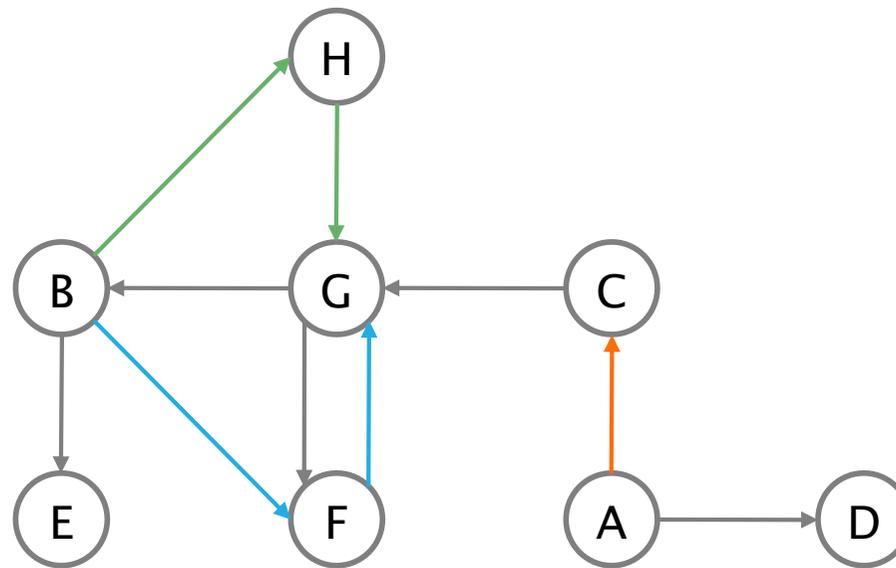
Caminho

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

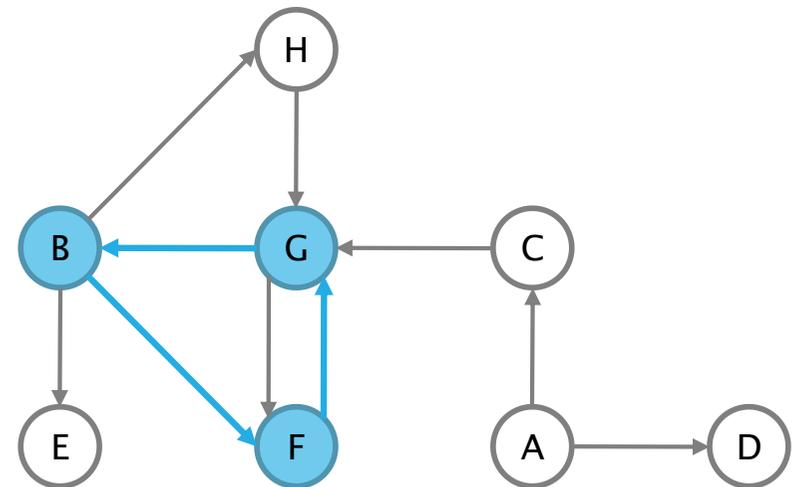
exemplo: B-F-G-B

Grafo cíclico

contém um ou mais ciclos

Grafo acíclico

não contém ciclos

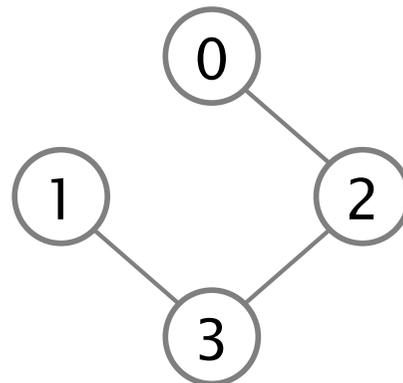
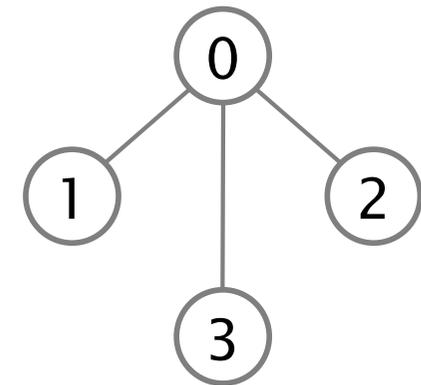
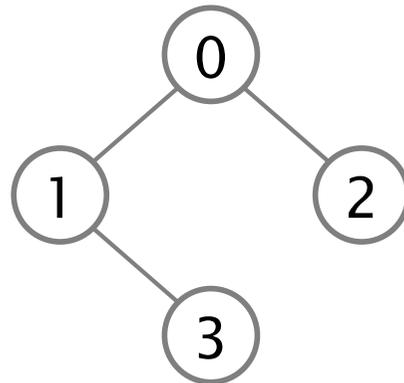
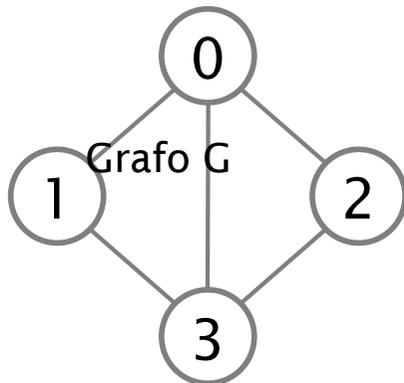


em grafos direcionados e não direcionados



árvore geradora

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



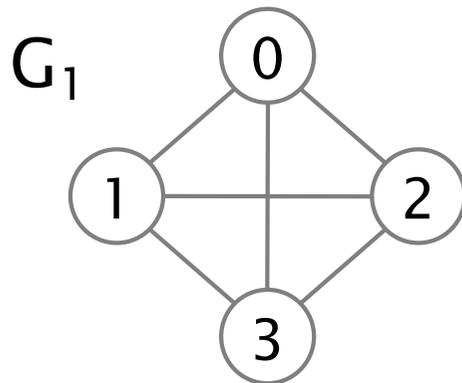
Representações de grafo

- Matriz de adjacências
- Listas de adjacências (incidências)



Matriz de adjacências

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

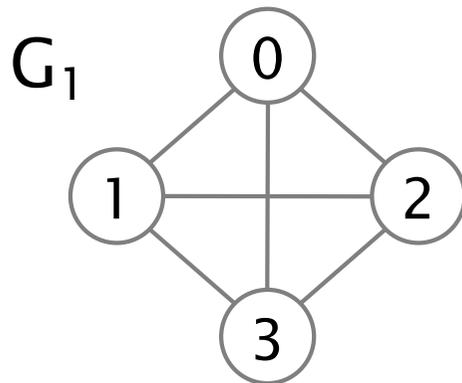


$$\begin{matrix} & 0 & 1 & 2 & 3 \\ 0 & \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix} \\ 1 & \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix} \\ 2 & \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix} \\ 3 & \begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix} \end{matrix}$$



Matriz de adjacências

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

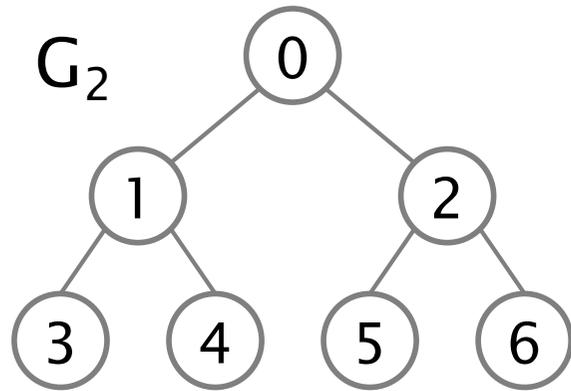


$$\begin{matrix} & 0 & 1 & 2 & 3 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{matrix}$$

matrizes simétricas para grafos não direcionados



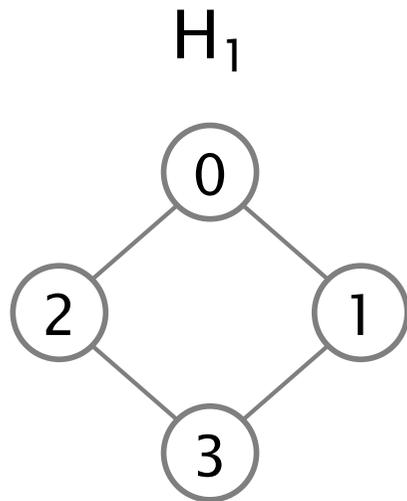
Matriz de adjacências



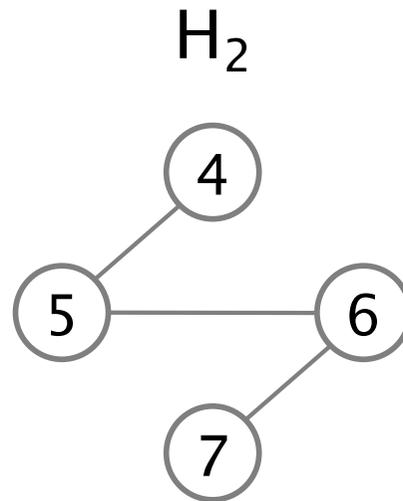
	0	1	2	3	4	5	6
0	0	1	1	0	0	0	0
1	1	0	0	1	1	0	0
2	1	0	0	0	0	1	1
3	0	1	0	0	0	0	0
4	0	1	0	0	0	0	0
5	0	0	1	0	0	0	0
6	0	0	1	0	0	0	0



Matriz de adjacências



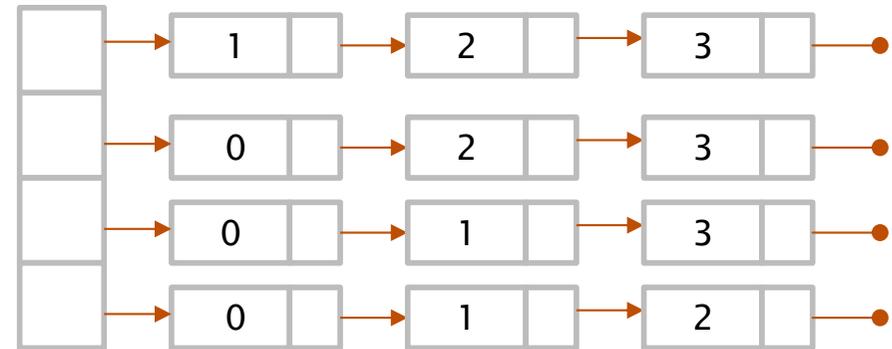
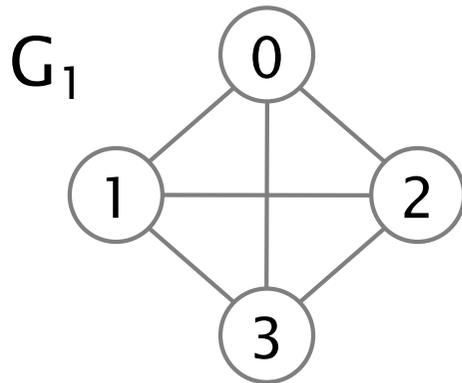
G



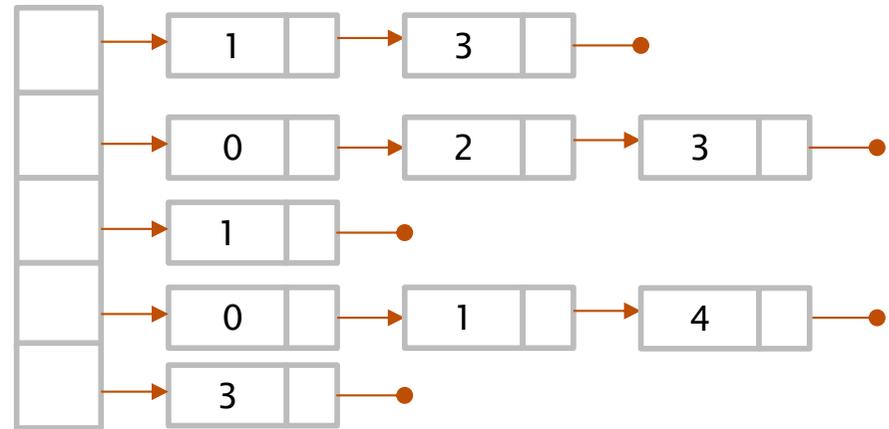
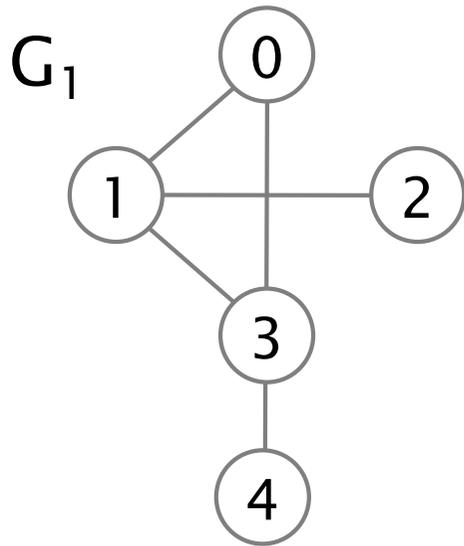
	0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0
2	1	0	0	1	0	0	0	0
3	0	1	1	0	0	0	0	0
4	0	0	0	0	0	1	0	0
5	0	0	0	0	1	0	1	0
6	0	0	0	0	0	1	0	1
7	0	0	0	0	0	0	1	0



Listas de adjacências



Listas de adjacências



Listas de adjacências

o que vamos usar no lab de 4a:

```
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

passaios 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)



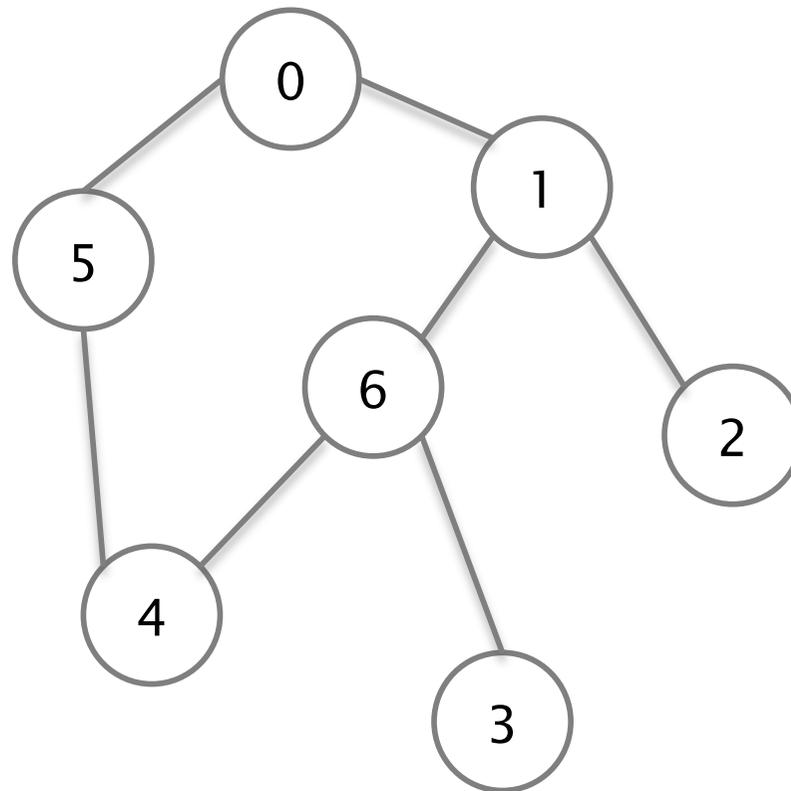
Percursos em grafos

Cada vértice examinado deve ser marcado como visitado

Por quê?



dfs iniciando em 0



dfs iniciando em 0

dfs(0)

dfs(1)

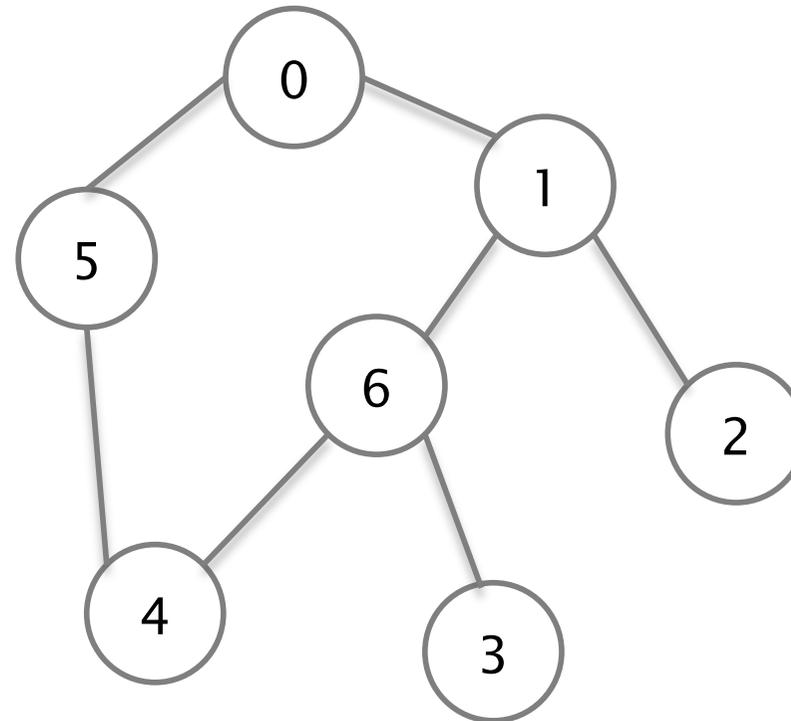
dfs(2)

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)

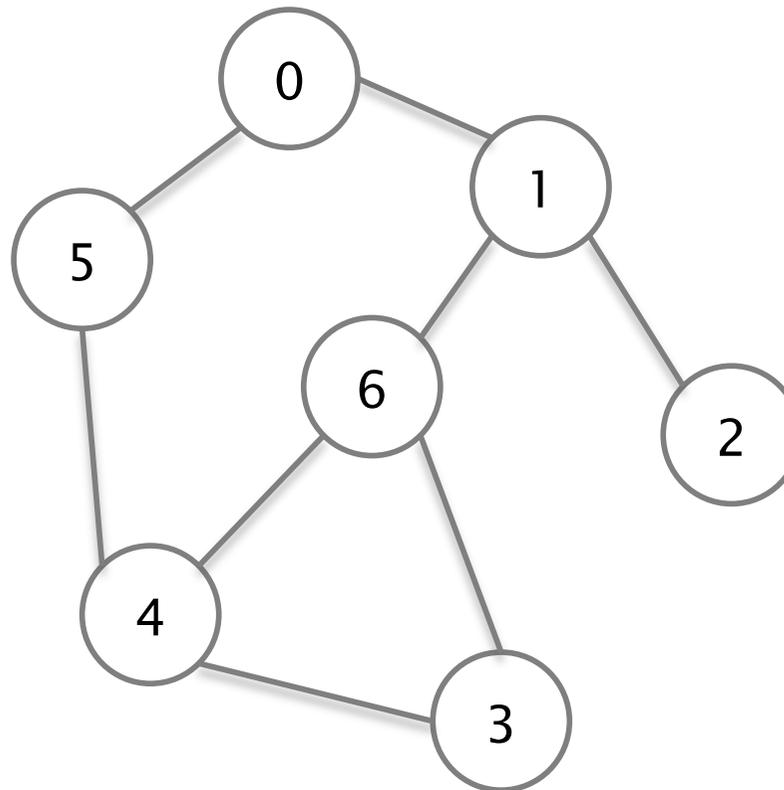
dfs(3)

dfs(4)

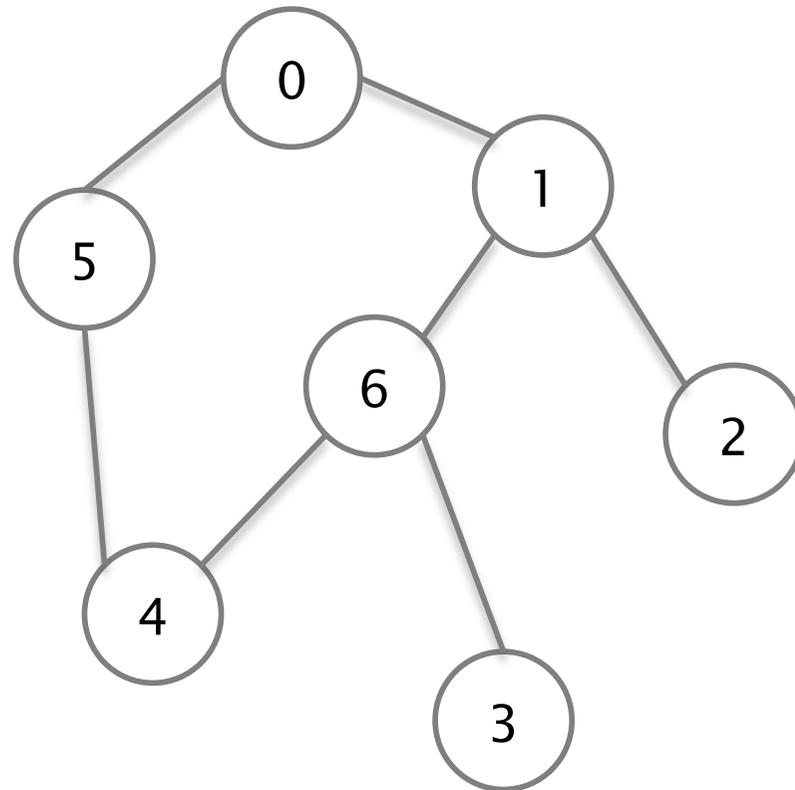
dfs(6)

dfs(5)

dfs(0)



bfs iniciando em 0



bfs iniciando em 0

bfs(0)

visita 0

visita 1

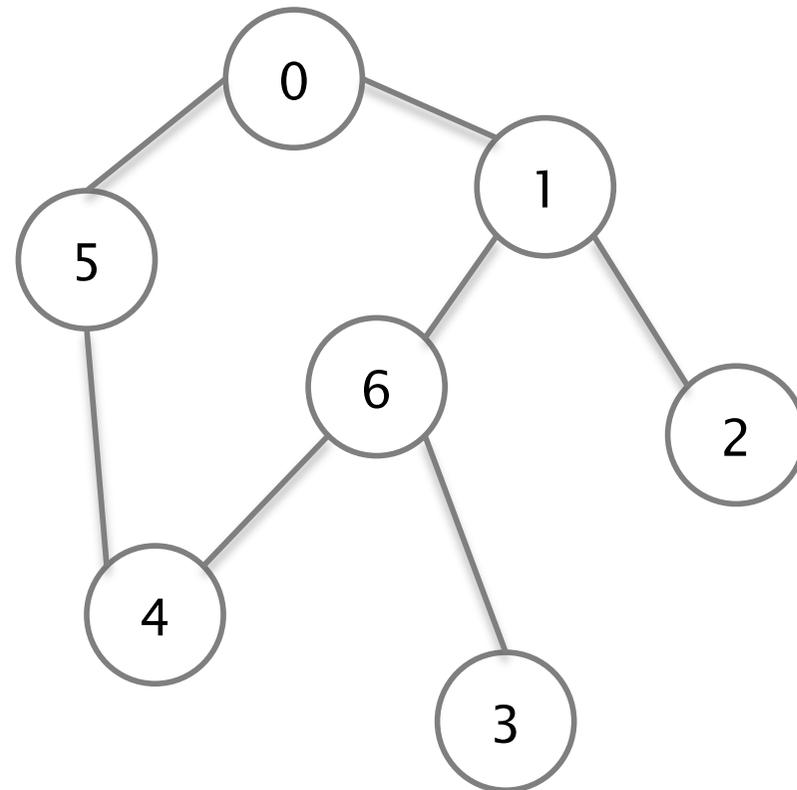
visita 5

visita 2

visita 6

visita 4

visita 3



bfs iniciando em 0

bfs(0)

visita 0

visita 1

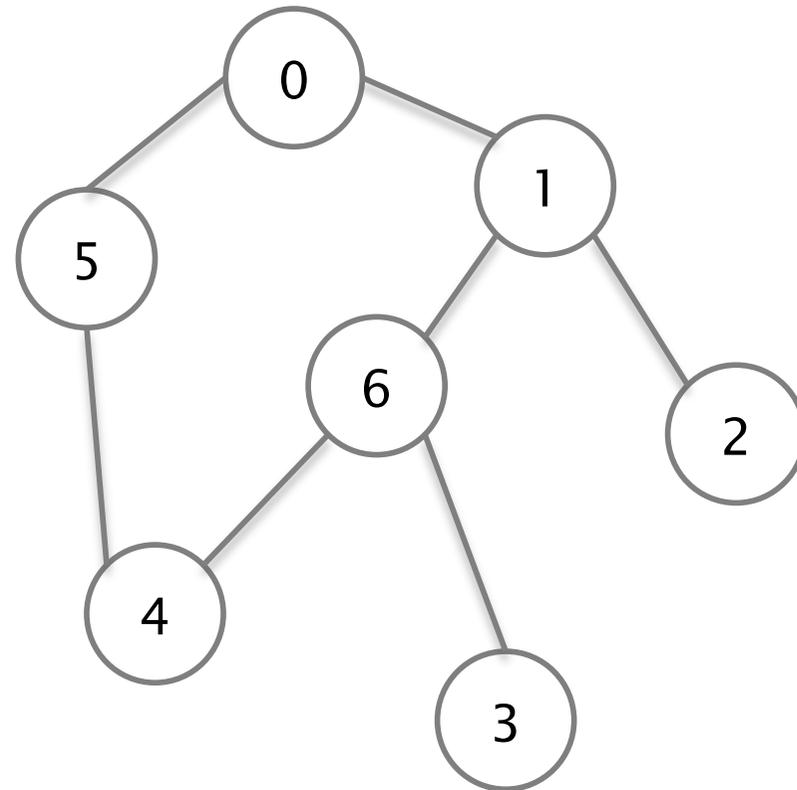
visita 5

visita 2

visita 6

visita 4

visita 3



bfs iniciando em 0

bfs(0)

visita 0

visita 1

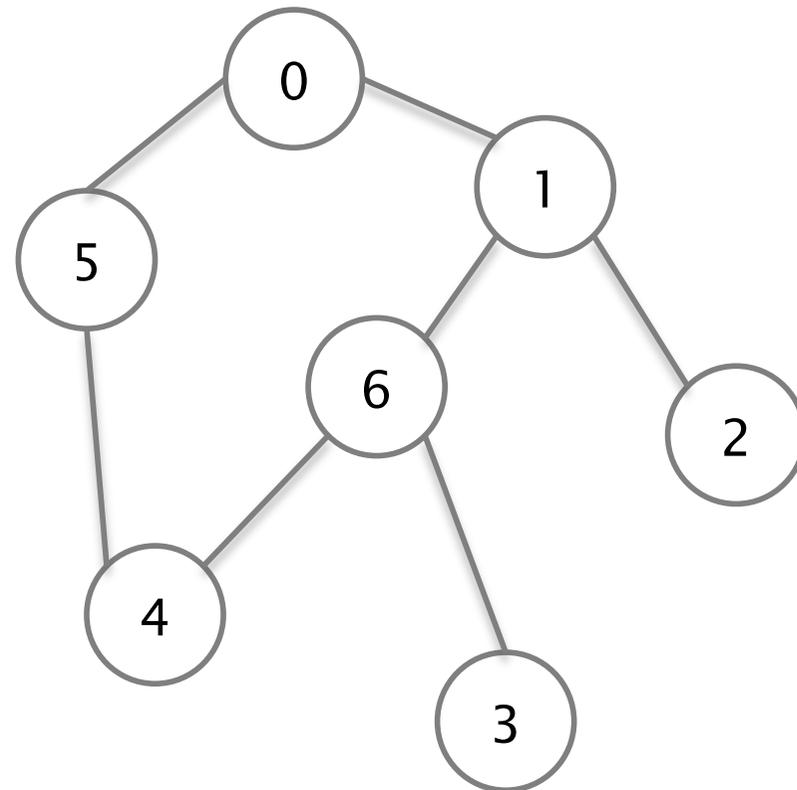
visita 5

visita 2

visita 6

visita 4

visita 3



Como registrar nós pendentes?



bfs iniciando em 0

bfs(0)

visita 0

visita 1

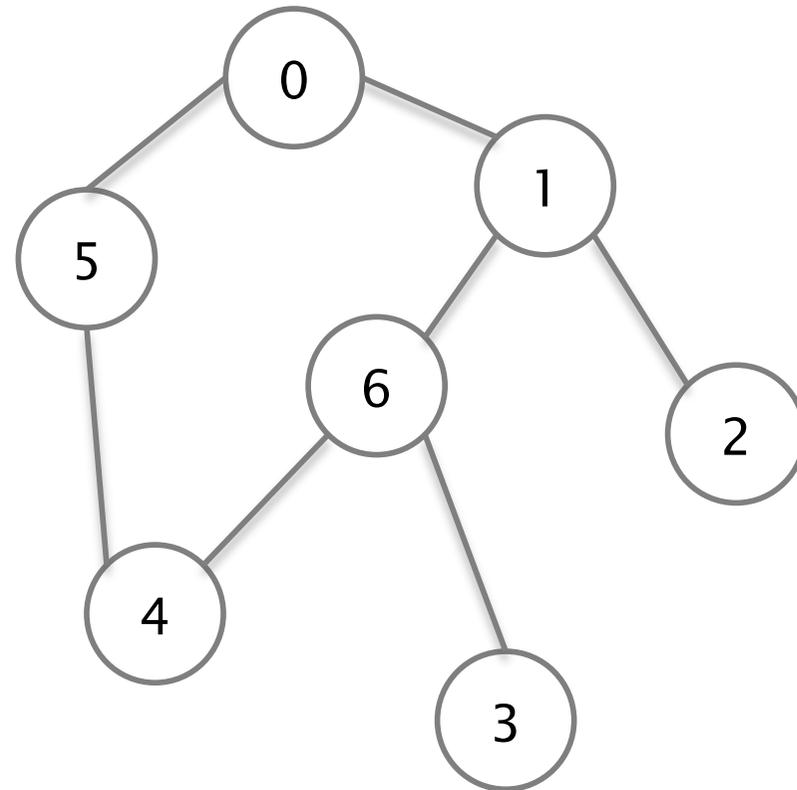
visita 5

visita 2

visita 6

visita 4

visita 3



Como registrar nós pendentes?

Uso de fila de nós!



bfs iniciando em 0

Uso de fila auxiliar!

bfs(0)

visita 0

-> enfileira 1, 5

visita 1

-> enfileira 2, 6

visita 5

-> enfileira 4

visita 2

visita 6

-> enfileira 3

visita 4

visita 3

[1,5]

[5,2,6]

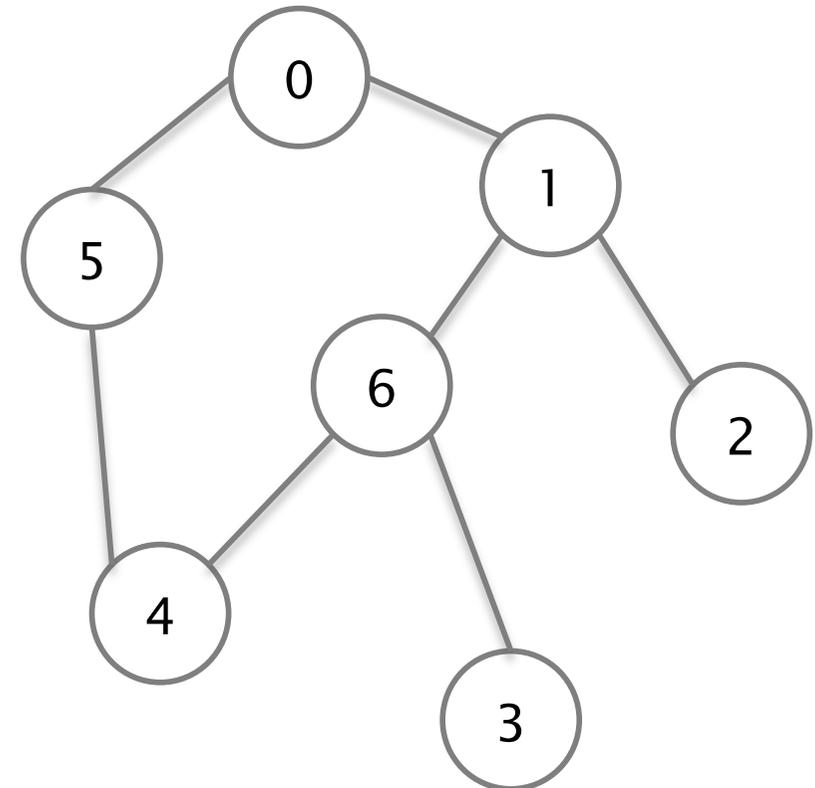
[2,6,4]

[6,4]

[4,3]

[3]

[]



bfs iniciando em 0

Uso de fila auxiliar!

bfs(0)

visita 0

-> enfileira 1, 5

visita 1

-> enfileira 2, 6

visita 5

-> enfileira 4

visita 2

visita 6

-> enfileira 3

visita 4

visita 3

[1,5]

[5,2,6]

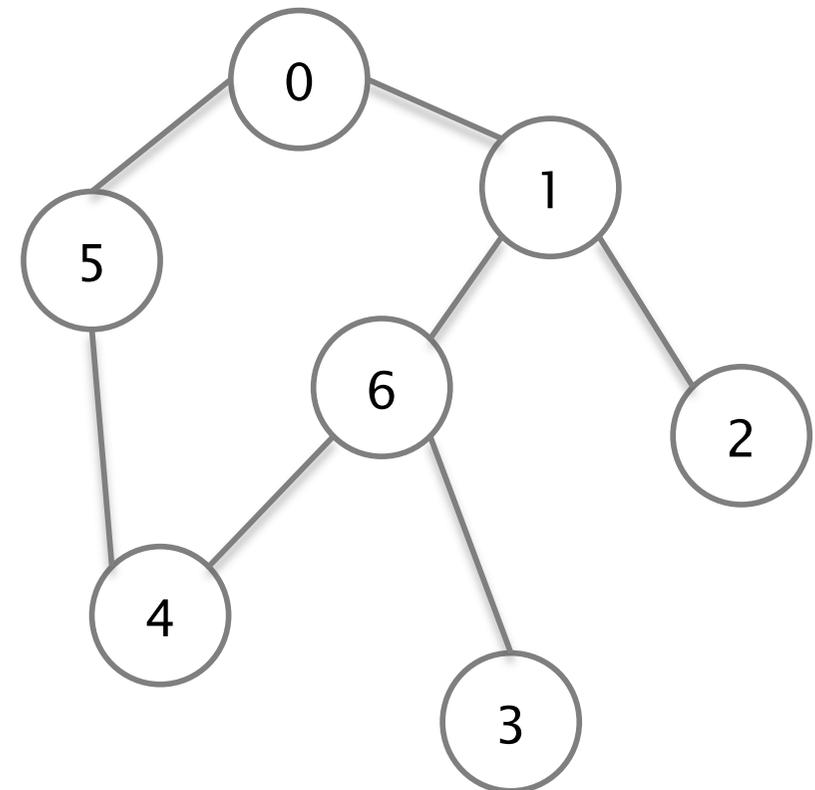
[2,6,4]

[6,4]

[4,3]

[3]

[]



bfs iniciando em 0

Uso de fila auxiliar!

bfs(0)

visita 0

-> enfileira 1, 5

visita 1

-> enfileira 2, 6

visita 5

-> enfileira 4

visita 2

visita 6

-> enfileira 3

visita 4

visita 3

[1,5]

[5,2,6]

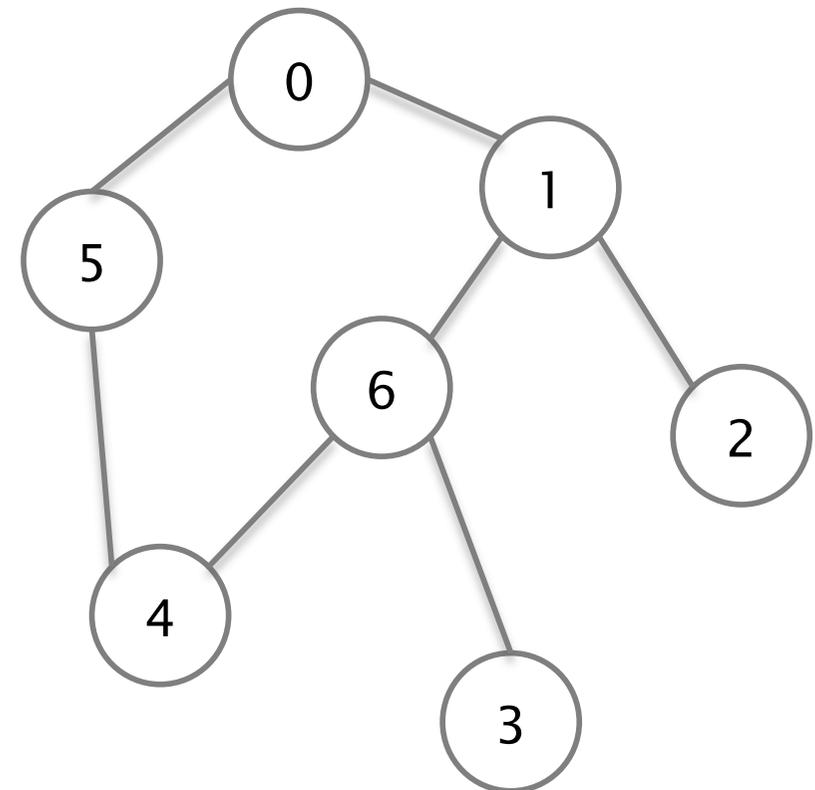
[2,6,4]

[6,4]

[4,3]

[3]

[]



bfs iniciando em 0

Uso de fila auxiliar!

bfs(0)

visita 0

-> enfileira 1, 5

visita 1

-> enfileira 2, 6

visita 5

-> enfileira 4

visita 2

visita 6

-> enfileira 3

visita 4

visita 3

[1,5]

[5,2,6]

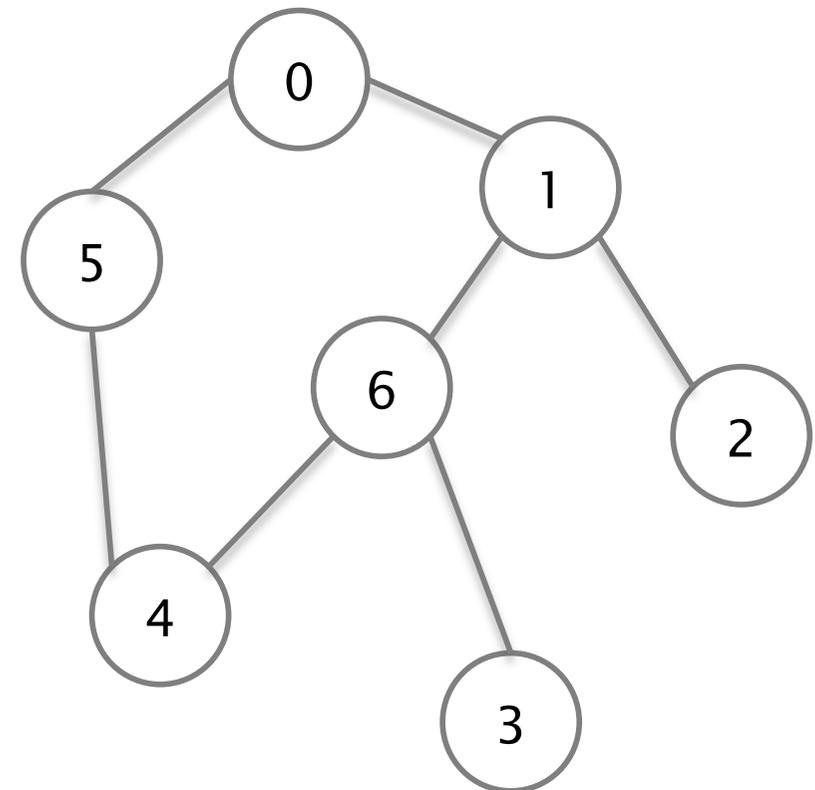
[2,6,4]

[6,4]

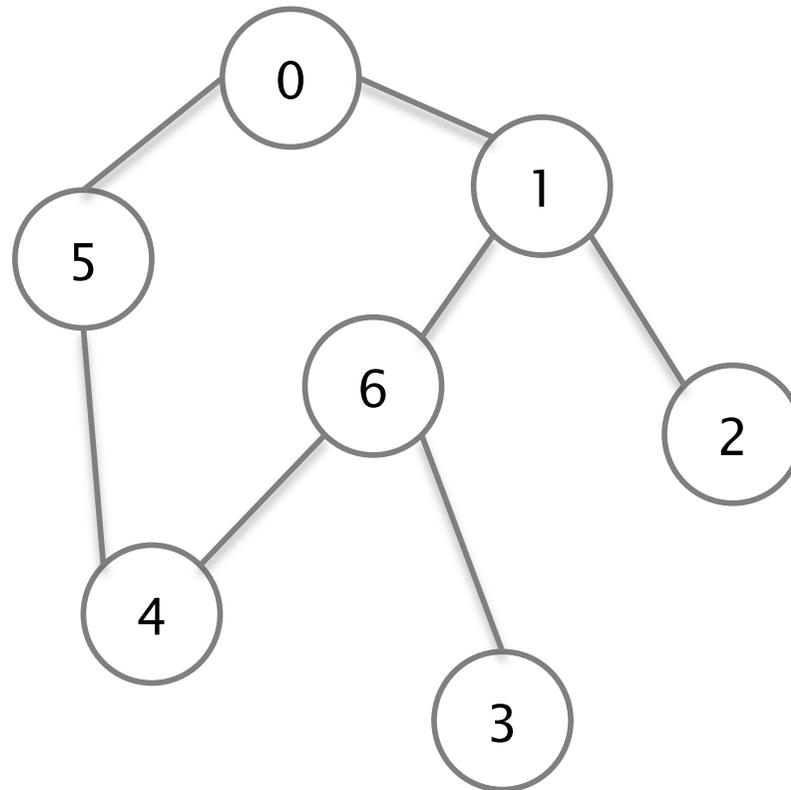
[4,3]

[3]

[]



dfs – podemos usar uma pilha



dfs iniciando em 0

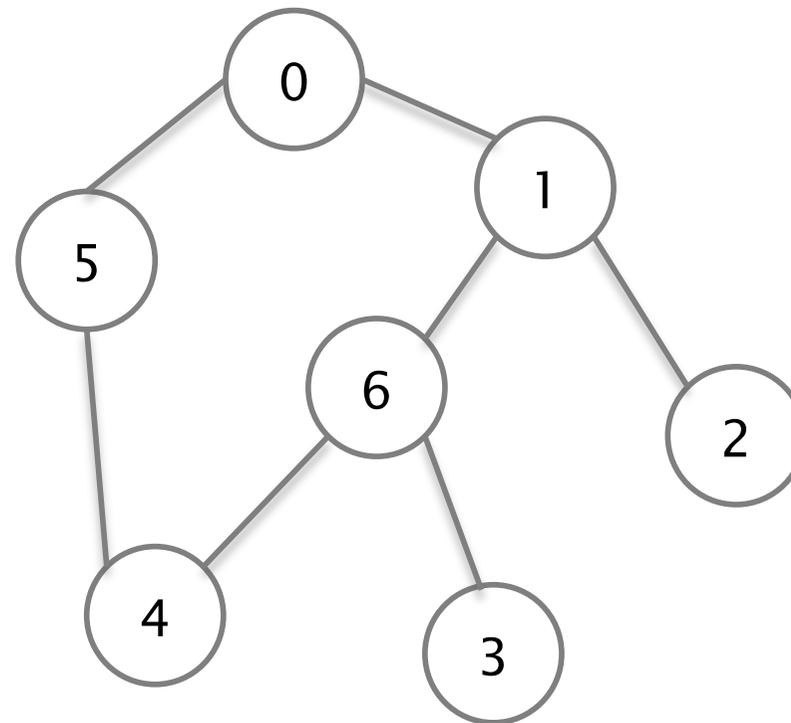
dfs(0)

empilha 0

desempilha 0 ()

visita 0

empilha: 5:1



estado agora na pilha



dfs iniciando em 0

dfs(0)

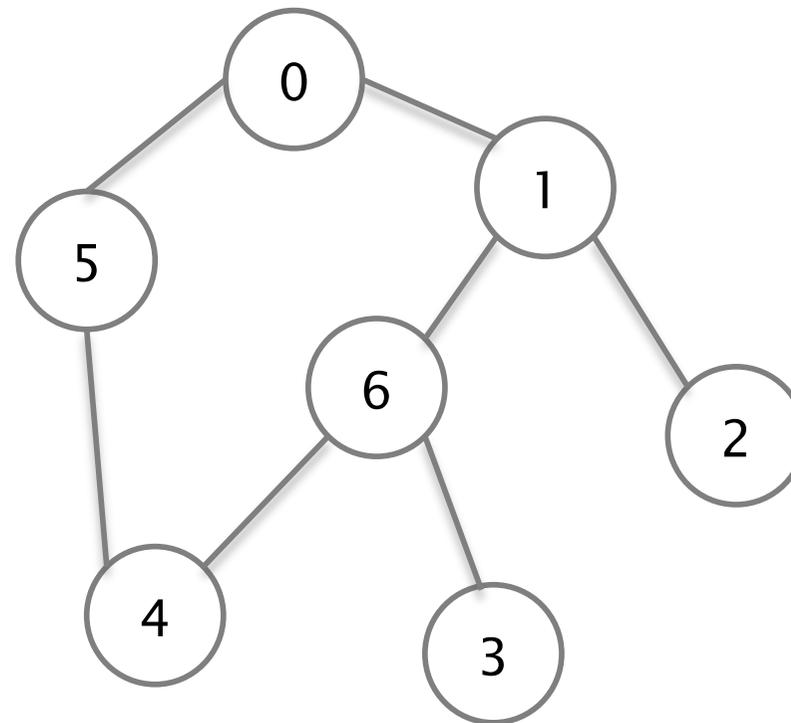
empilha 0

desempilha 0 (_ nada na pilha_)

visita 0

empilha: 5:1

desempilha 1 (5)



estado agora na pilha



dfs iniciando em 0

dfs(0)

empilha 0

desempilha 0 (_ nada na pilha_)

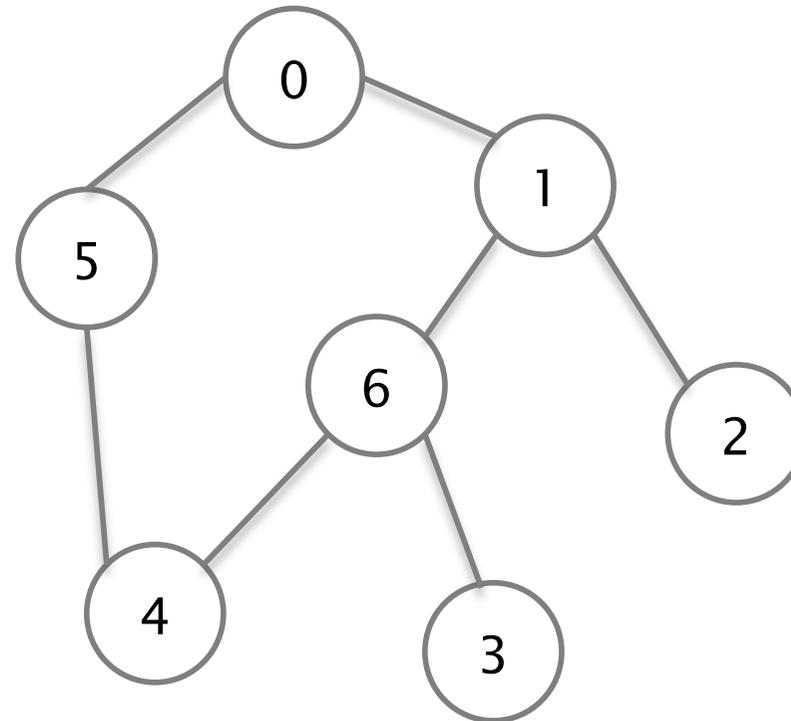
visita 0

empilha: 5:1

desempilha 1 (5)

visita 1

empilha 5:6:2



estado agora na pilha



dfs iniciando em 0

dfs(0)

empilha 0

desempilha 0 (_ nada na pilha_)

visita 0

empilha: 5:1

desempilha 1 (5)

visita 1

empilha 5:6:2

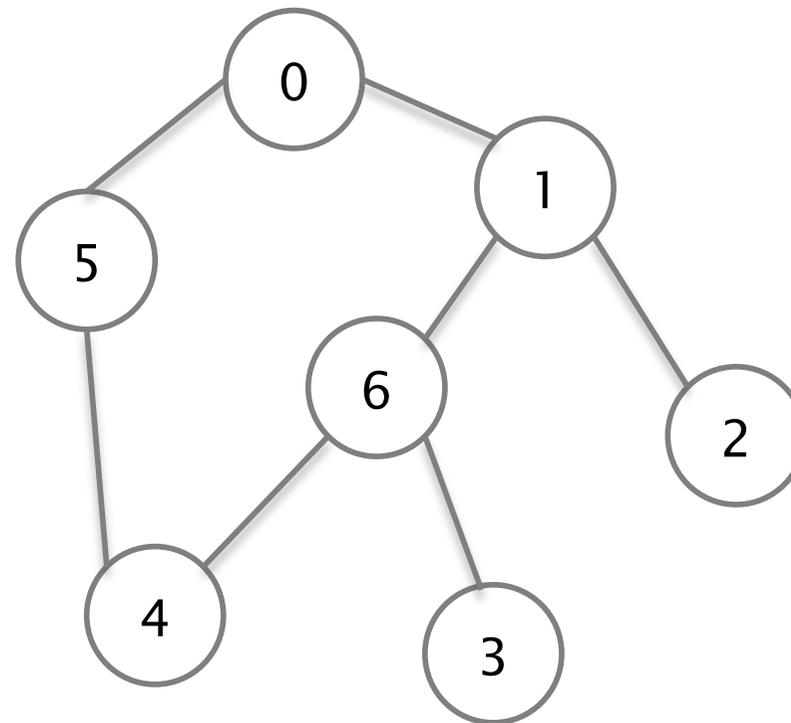
desempilha 2 (5:6)

visita 2

desempilha 6 (5)

visita 6

empilha 5:4:3



estado agora na pilha



dfs iniciando em 0

dfs(0)

empilha 0

desempilha 0 (_ nada na pilha_)

visita 0

empilha: 5:1

desempilha 1 (5)

visita 1

empilha 5:6:2

desempilha 2 (5:6)

visita 2

desempilha 6 (5)

visita 6

empilha 5:4:3

desempilha 3 (5:4)

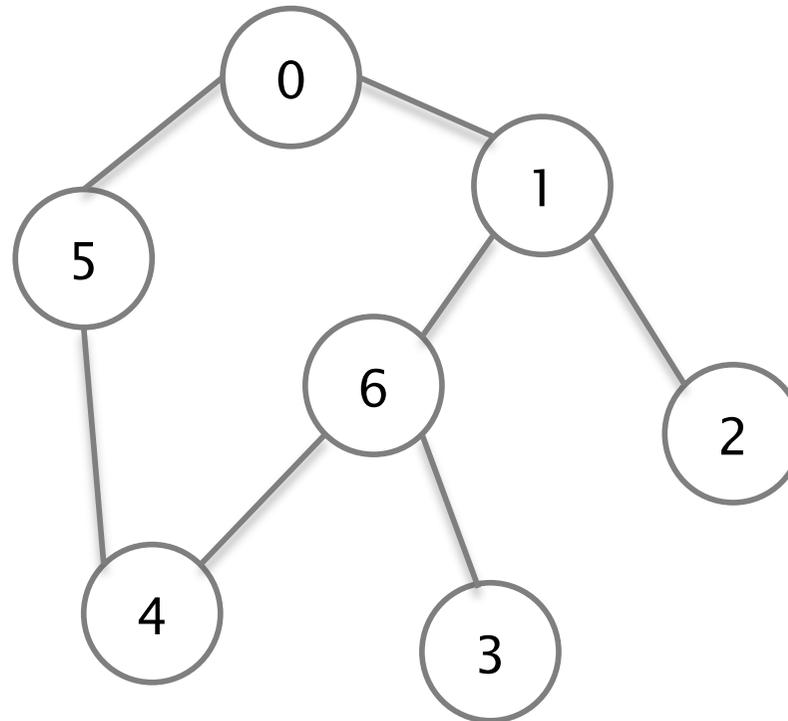
visita 3

desempilha 4 (5)

visita 4

desempilha 5

visita 5



estado agora na pilha

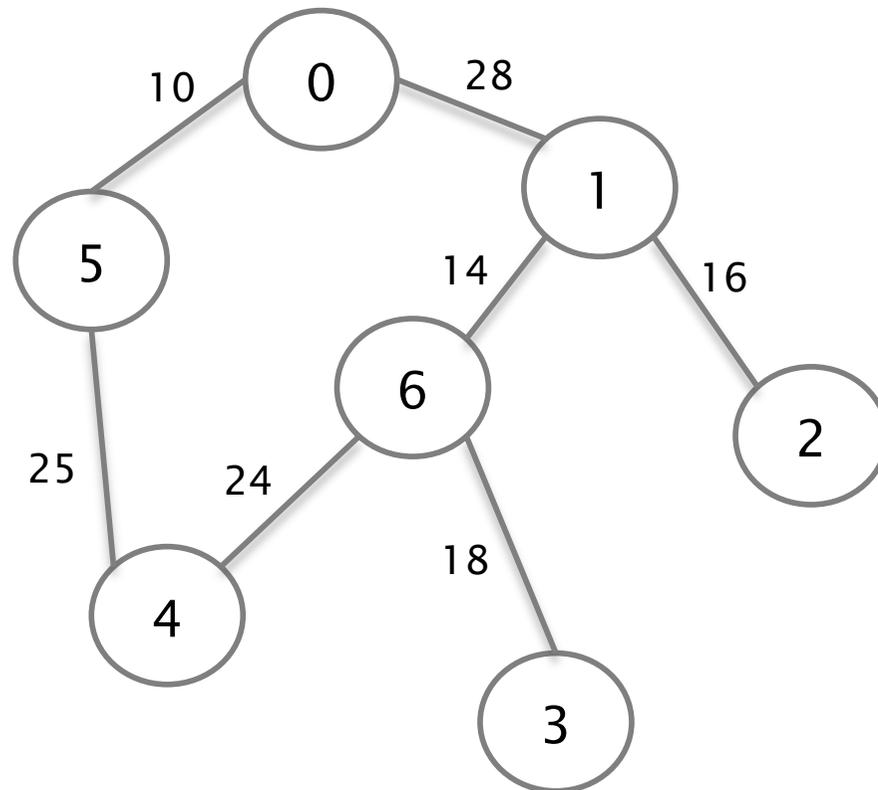


representações



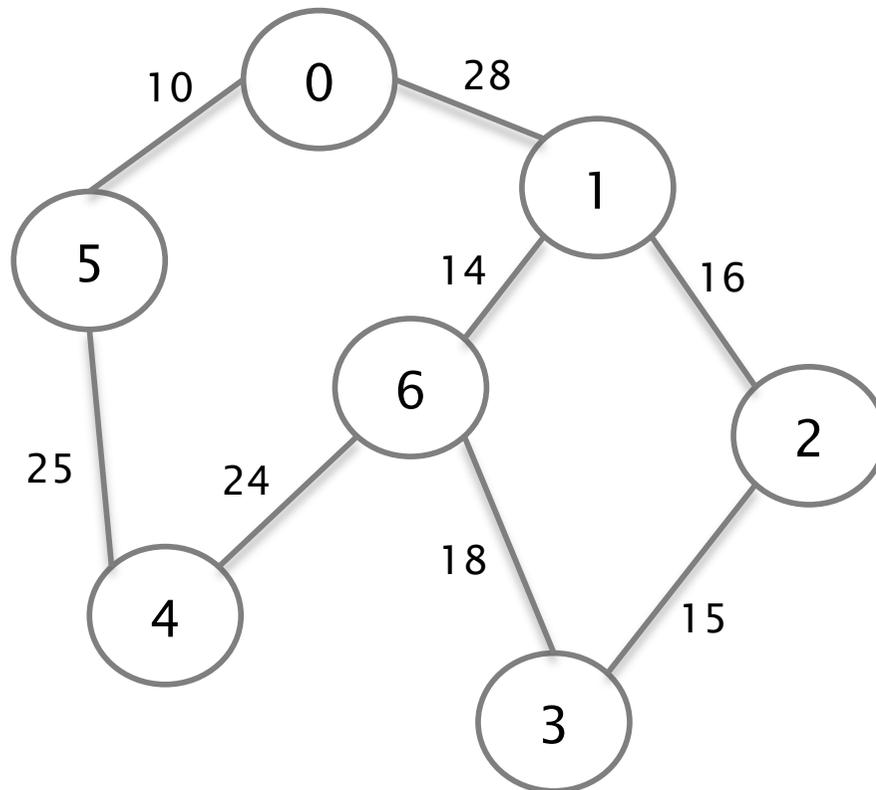
problemas comuns

- caminhos mais curtos
- árvores geradoras



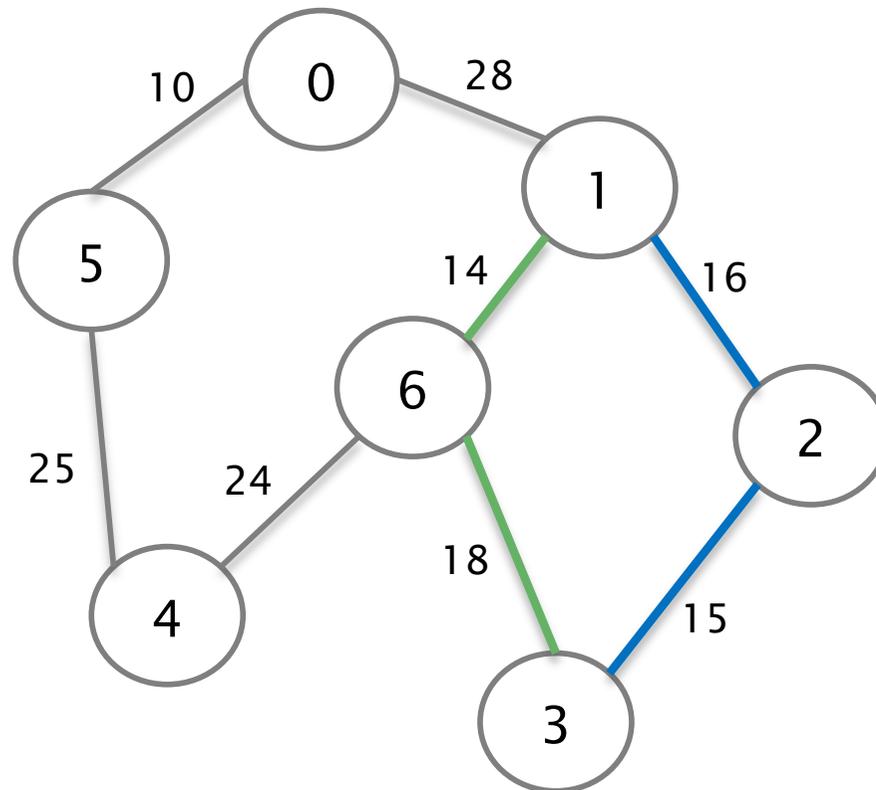
problemas comuns

- 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

Algoritmo de Dijkstra

Entradas:

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

Um vértice V do grafo

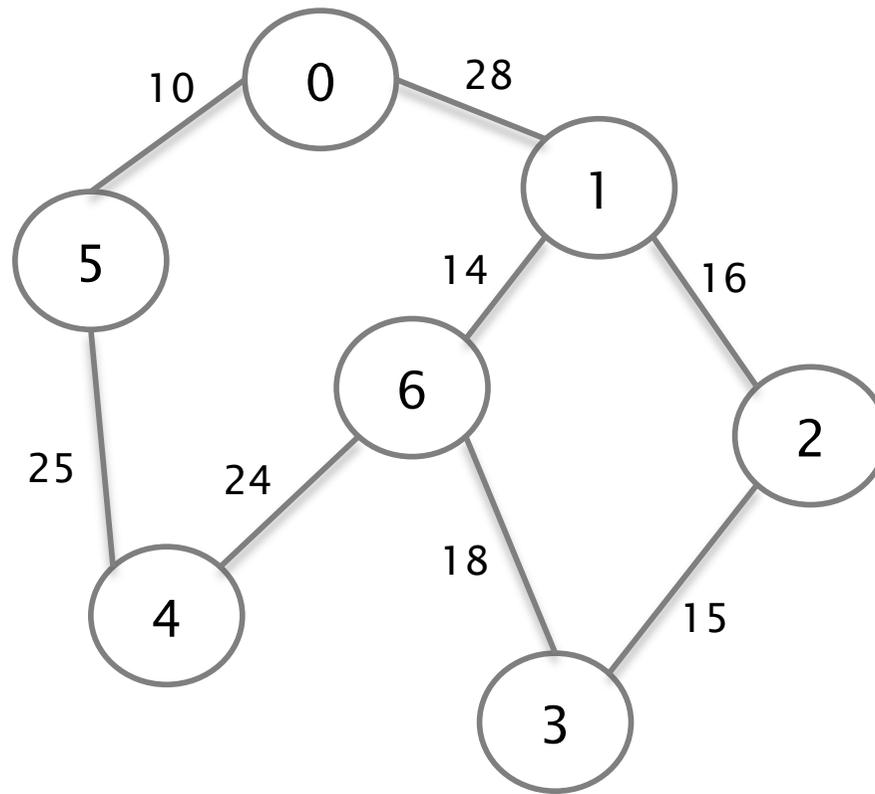
Saída:

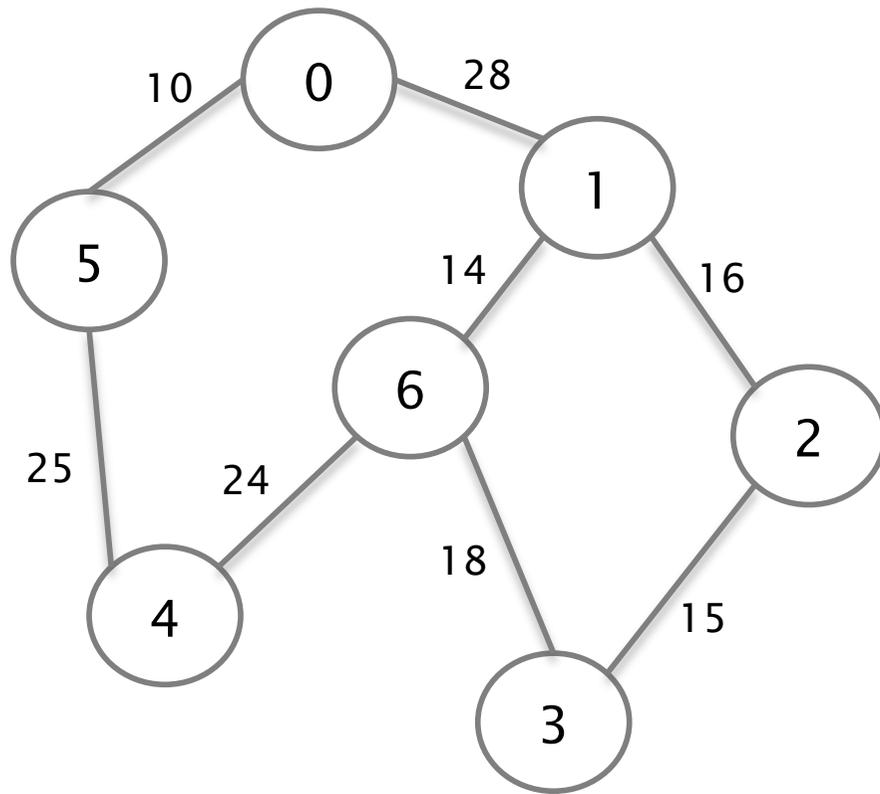
Menor caminho entre V e
cada um dos nós do grafo

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



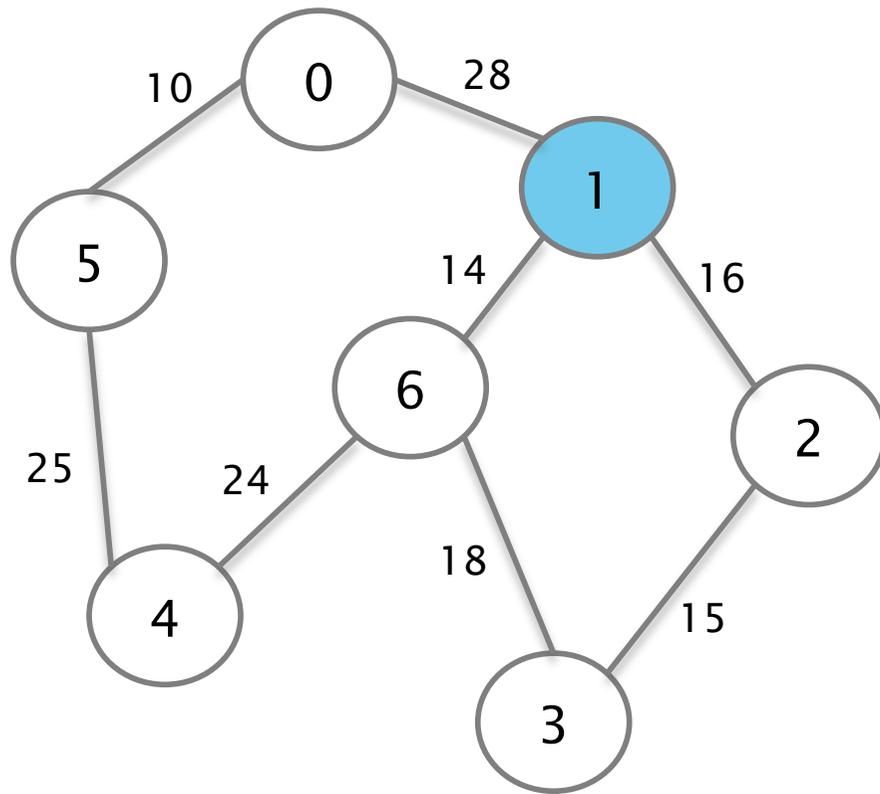
Dijkstra: caminhos mais curtos





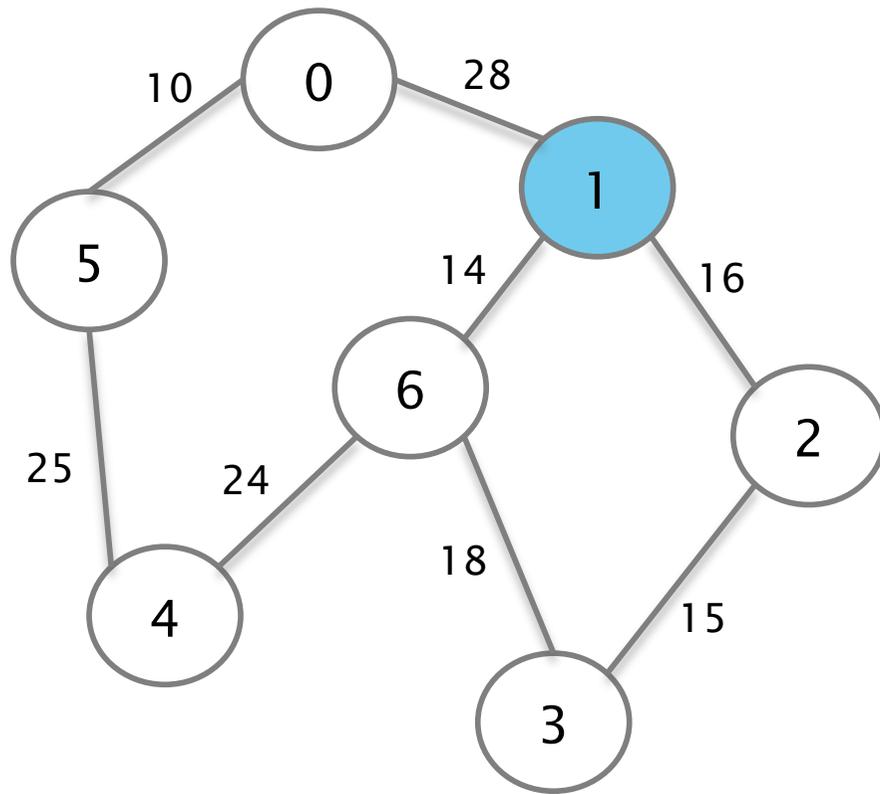
	dist
0	∞
1	0
2	∞
3	∞
4	∞
5	∞
6	∞





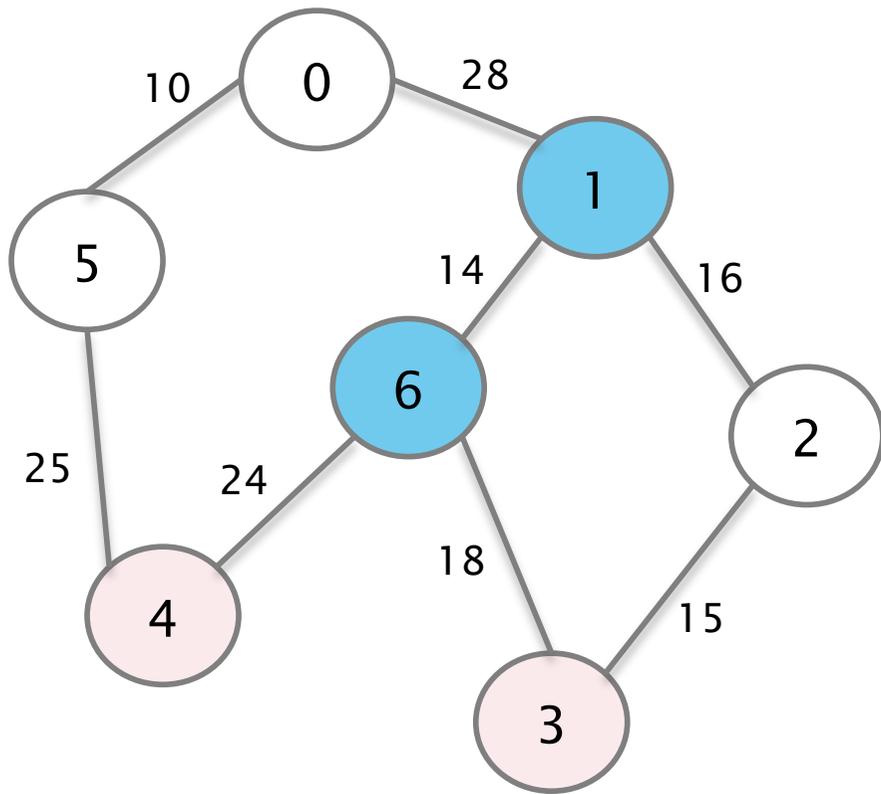
0	28
1	0
2	16
3	∞
4	∞
5	∞
6	14





0	28
1	0
2	16
3	∞
4	∞
5	∞
6	14

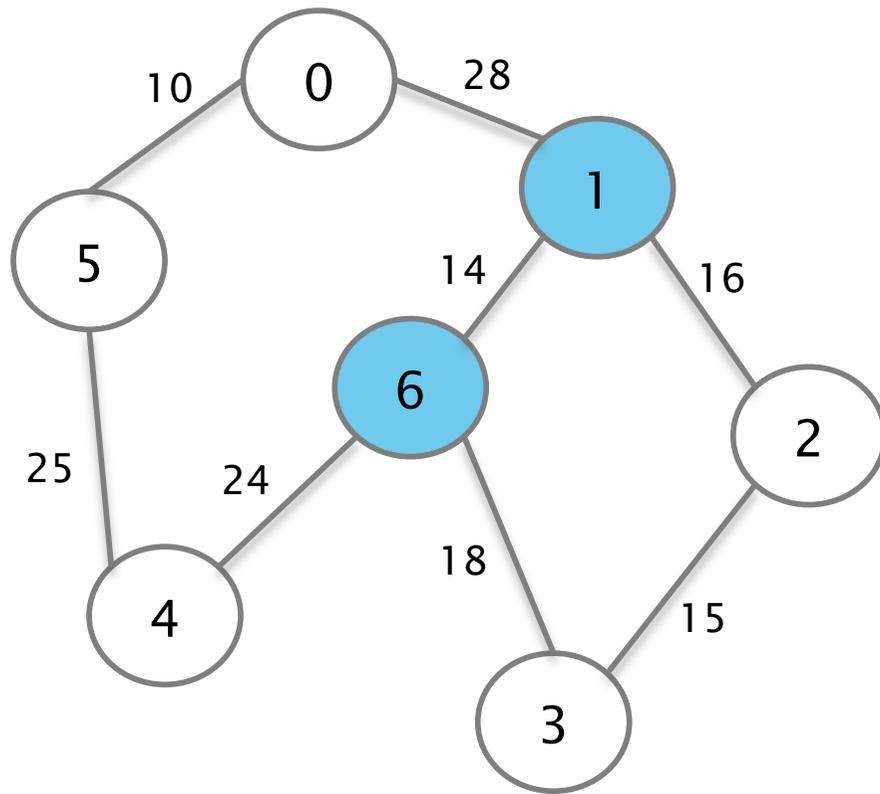




0	28
1	0
2	16
3	∞
4	∞
5	∞
6	14

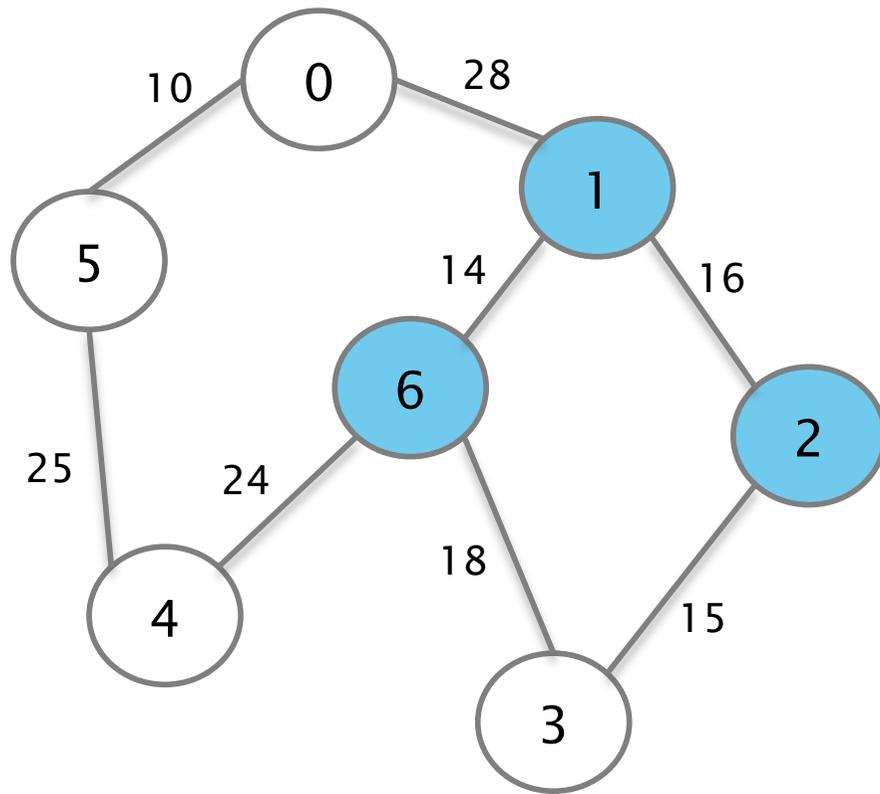
0	28
1	0
2	16
3	32
4	38
5	∞
6	14





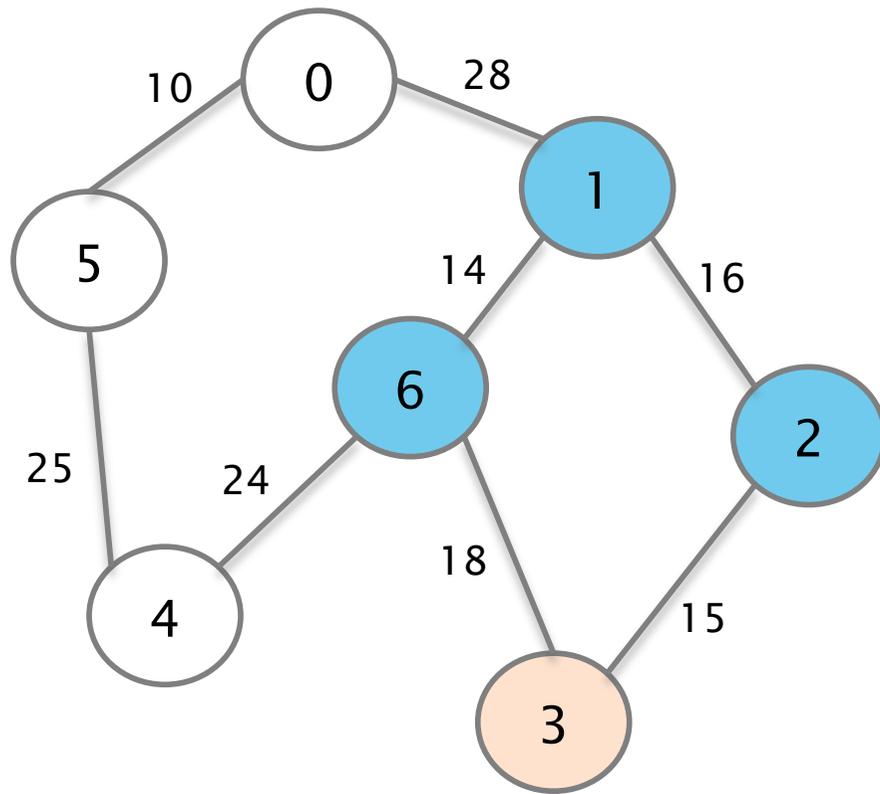
0	28
1	0
2	16
3	32
4	38
5	∞
6	14





0	28
1	0
2	16
3	32
4	38
5	∞
6	14

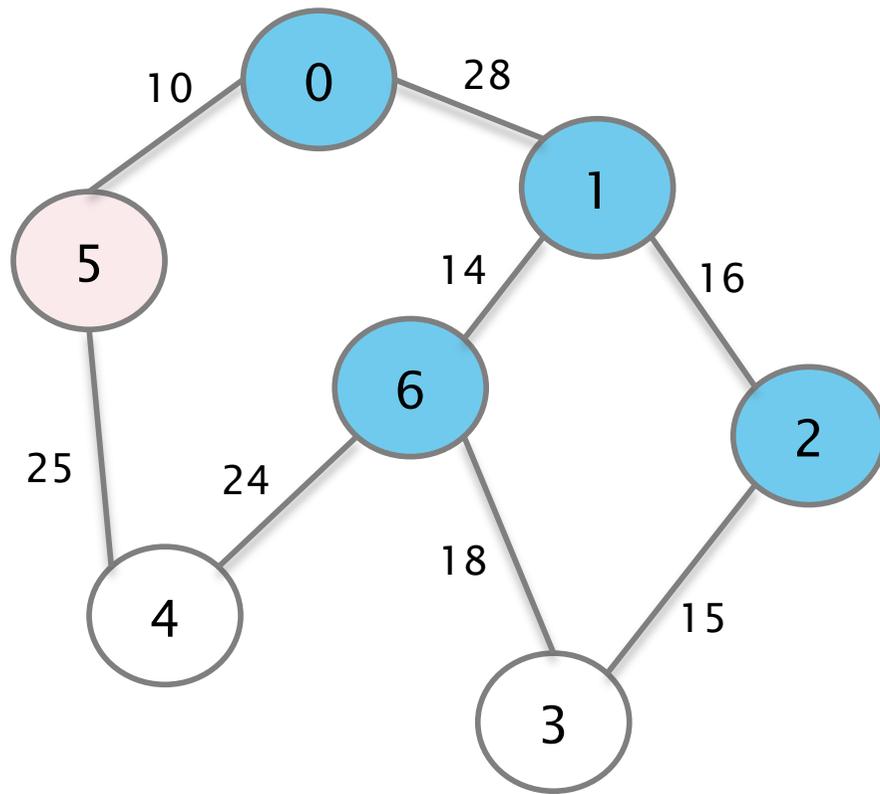




0	28
1	0
2	16
3	32
4	38
5	∞
6	14

0	28
1	0
2	16
3	31
4	38
5	∞
6	14

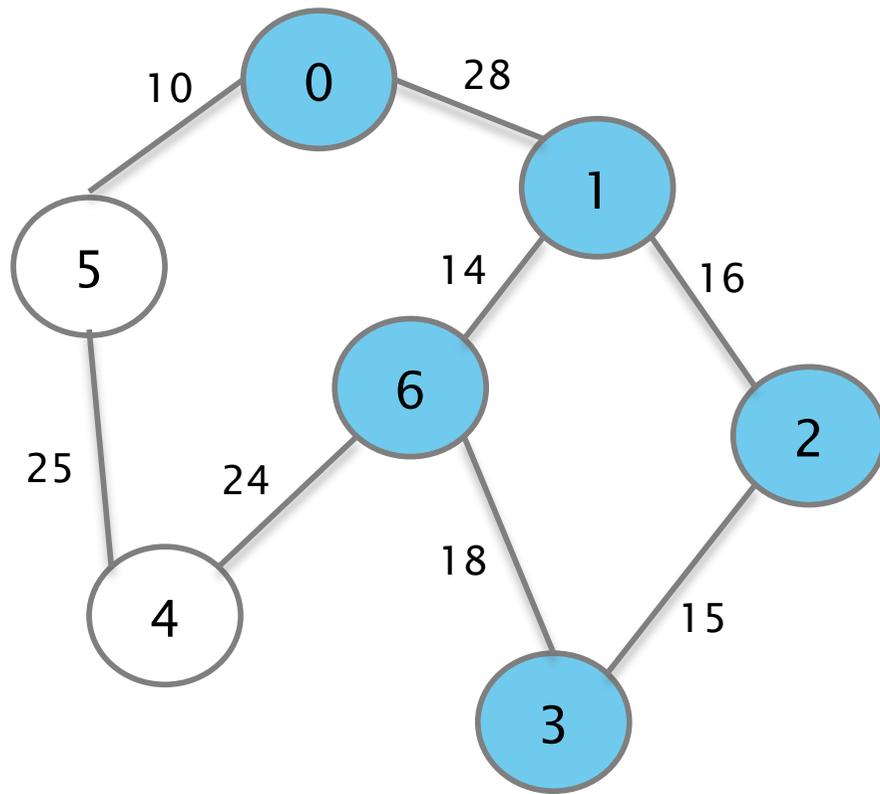




0	28
1	0
2	16
3	31
4	38
5	∞
6	14

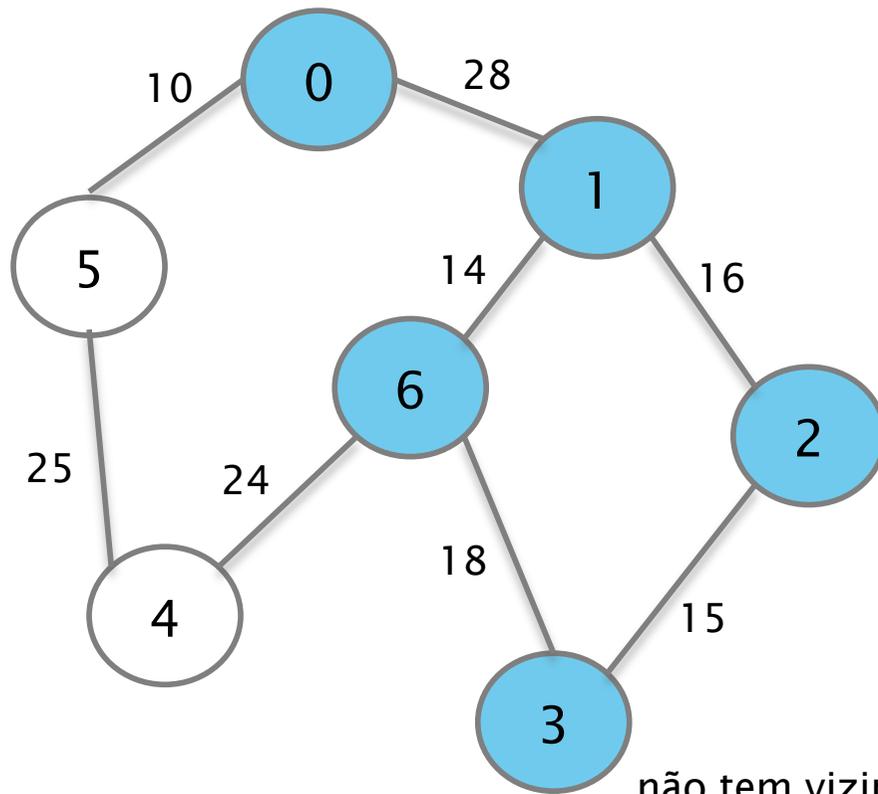
0	28
1	0
2	16
3	31
4	38
5	38
6	14





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

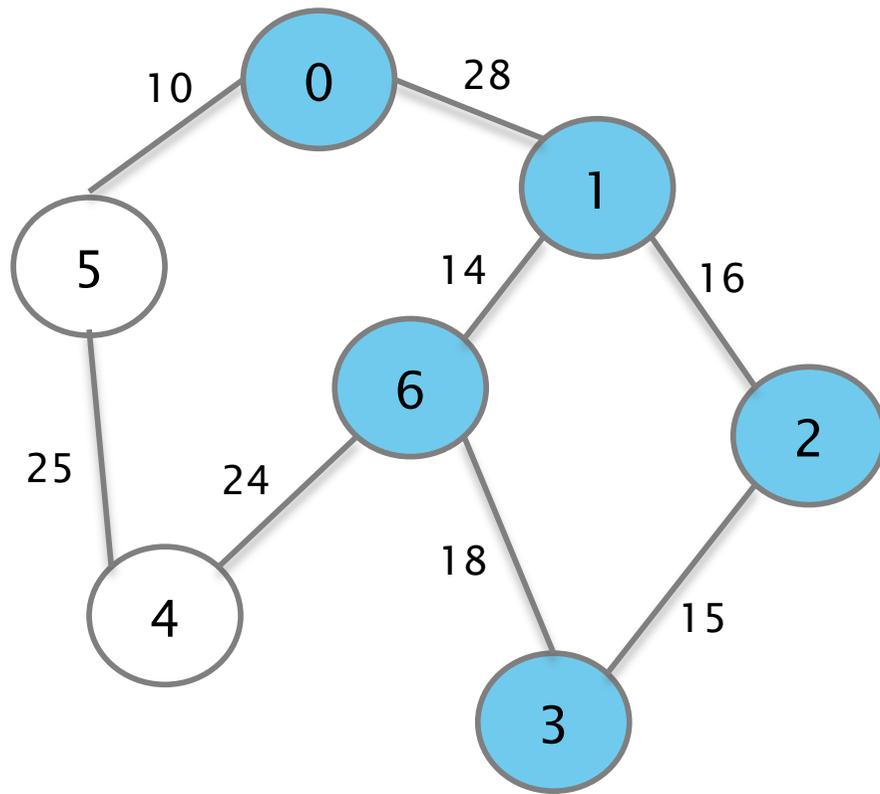




0	28
1	0
2	16
3	31
4	38
5	38
6	14

não tem vizinhos não visitados!





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



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.

1. 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.
2. Se essa distância for menor do que a distância registrada anteriormente (infinito, na primeira rodada; zero para o nó de origem), sobrescreva a distância de B.

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



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.
 1. 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.
 2. Se essa distância for menor do que a distância registrada anteriormente (infinito, na primeira rodada; zero para o nó de origem), 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.

