Grafos

Prof. Dr. Jonas Lopes de Vilas Boas

Definição

Grafos são estruturas de dados não lineares composta por **vértices** (ou nós) interconectados por **arestas** (ou arcos).

Um grafo G é um par (V, E).

Muito usados na matemática e computação para resolver uma vasta gama de problemas.

Aplicações

Algoritmos de busca;
Análise de redes complexas;
Planejamento de tarefas;
Análise de DNA;
Análise de redes de proteínas;
Simulação de sistema físicos;
Análise de mercados e fluxo de capiutal;
Recomendações;
Planejamento de rotas;
Etc.

Tipos de grafos

Direcionados ou não direcionados;

Conexo ou disconexo;

Cíclico ou acíclico;

Simples ou multigrafo;

Completo ou incompleto;

Bipartido, planar, euleriano ou hamiltoniano.

Representacão por Matriz de adjacência

Uma matriz quadrada com dimensão igual ao número de vértices do grafo.

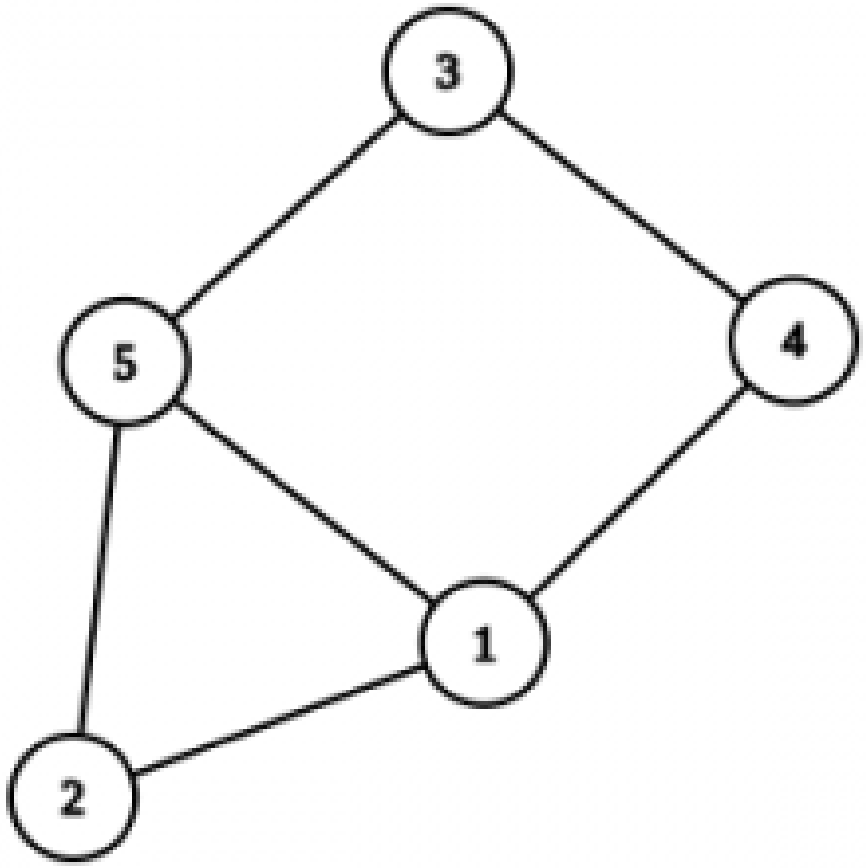
Cada entrada da matriz indica se existe uma aresta entre os vértices correspondentes (pode indicar o peso das arestas).

Vantagens:

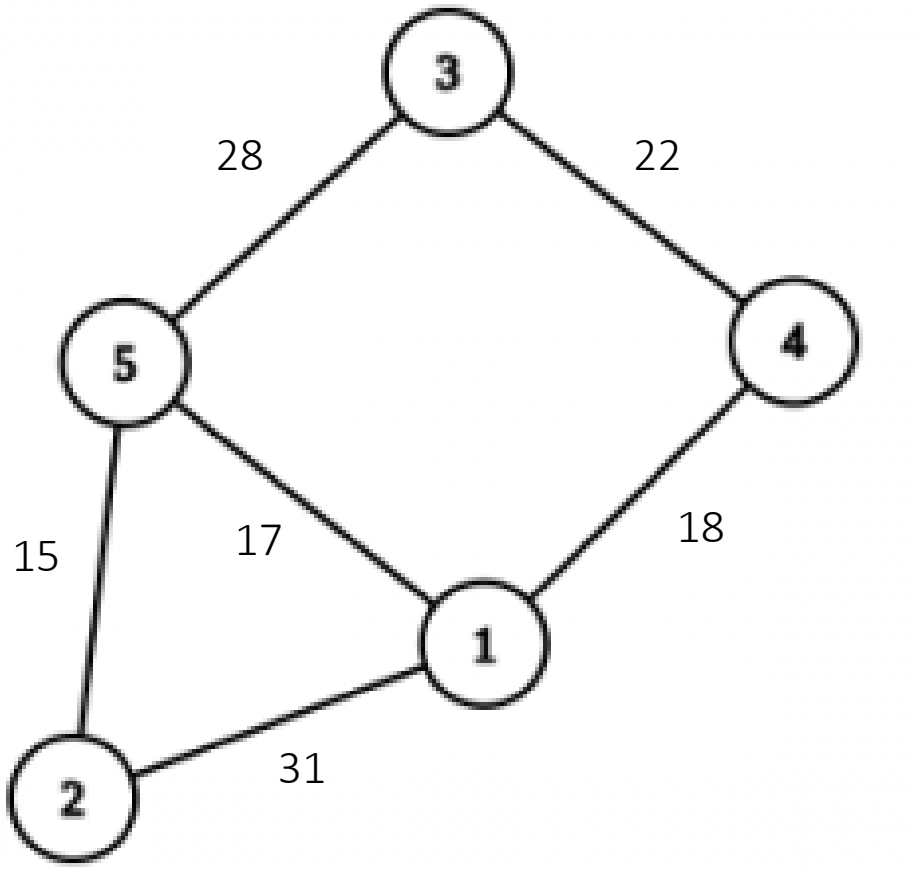
- Eficiente para verificar se existe uma aresta entre dois vértices.
- Simples de implementar.

Desvantagens:

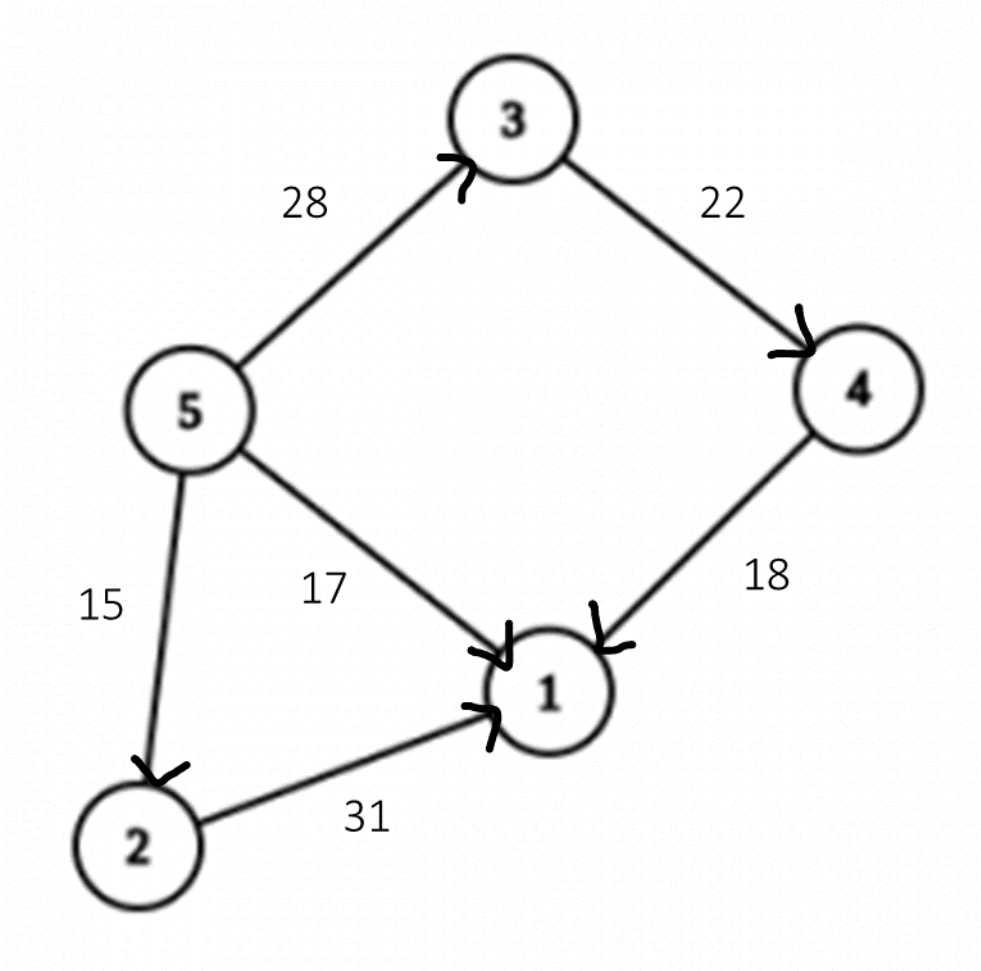
- Ocupa mais espaço em memória para grafos esparsos (com poucas arestas).
- Ineficiente para operações que envolvem todos os vértices adjacentes a um vértice específico.



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



	1	2	3	4	5
1	-	31	-	18	17
2	31	-	-	-	15
3	-	-	-	22	28
4	18	-	22	-	-
5	17	15	28	-	-



	1	2	3	4	5
1	-	-	-	-	-
2	31	-	-	-	-
3	-	-	-	22	-
4	18	-	-	-	-
5	17	15	28	-	-

Representação por Lista de ajacências

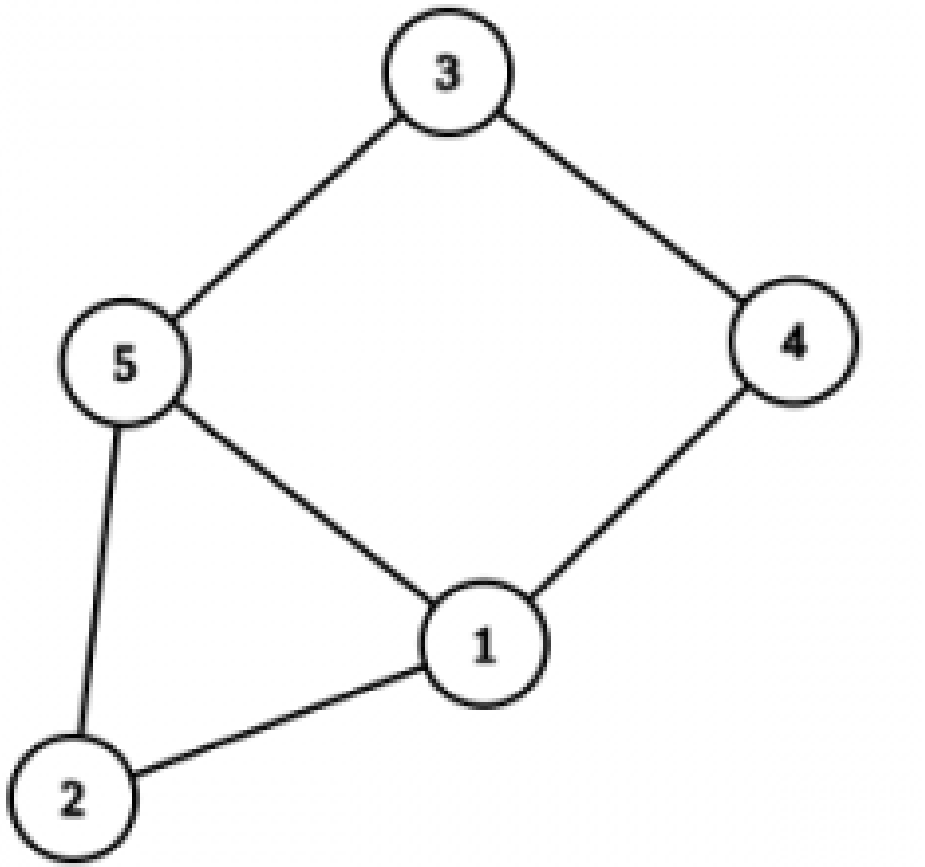
Uma lista para cada vértice do grafo, contendo os vértices adjacentes a ele.

Vantagens:

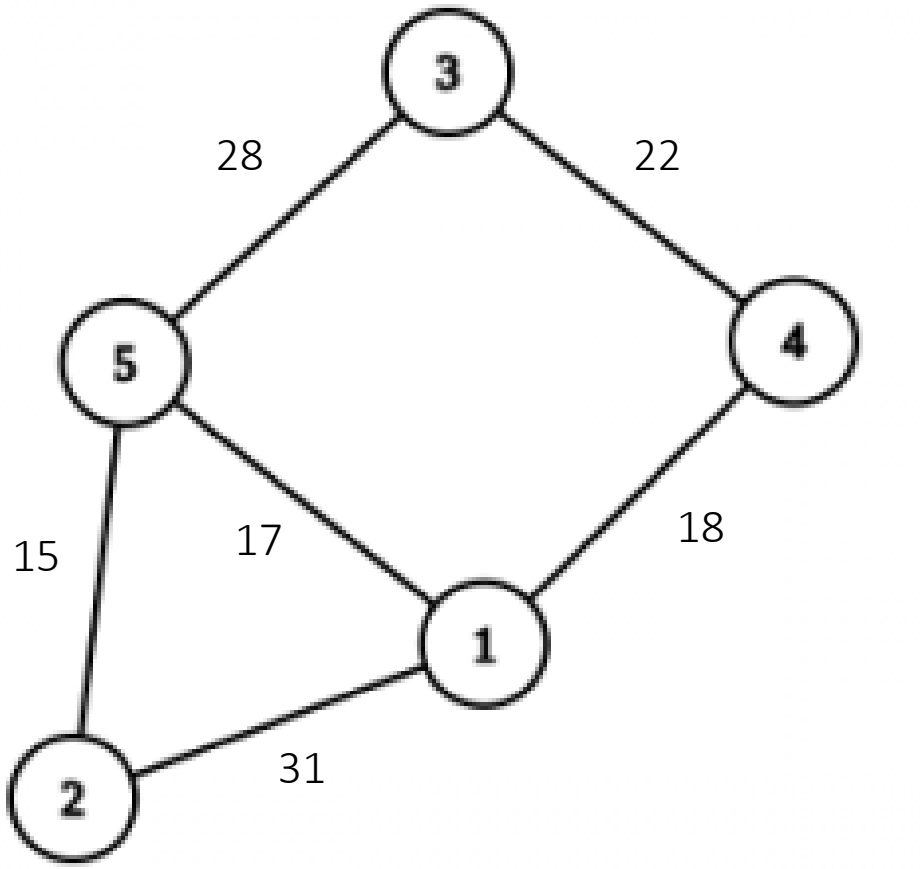
- Eficiente para operações que envolvem todos os vértices adjacentes a um vértice específico.
- Ocupa menos espaço em memória para grafos esparsos.

Desvantagens:

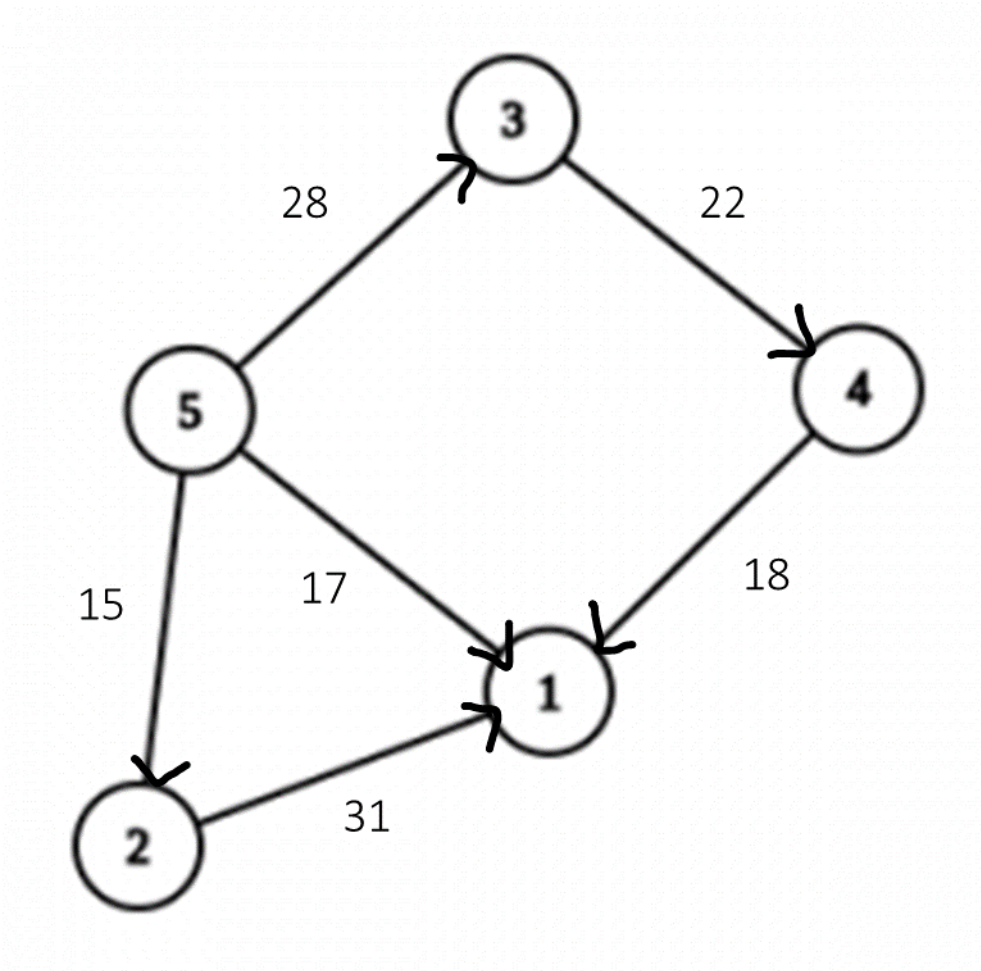
- Ineficiente para verificar se existe uma aresta entre dois vértices.
- Mais complexa de implementar.



1: [2,4,5]
2: [1,5]
3: [4,5]
4: [3,1]
5: [1,2,3]

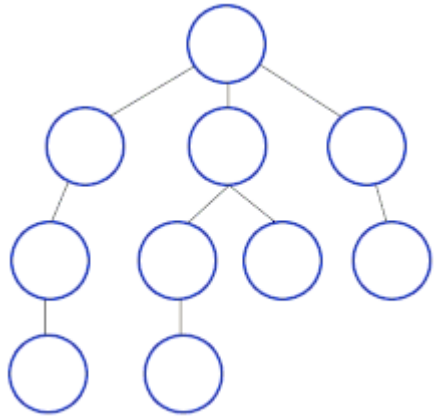


1: [(2,31),(4,18),(5,17)]
2: [(1,31),(5,15)]
3: [(4,22),(5,28)]
4: [(1,18),(3,22)]
5: [(1,17),(2,15),(3,28)]



1: []
2: [(1,31)]
3: [(4,22)]
4: [(1,18)]
5: [(1,17),(2,15),(3,28)]

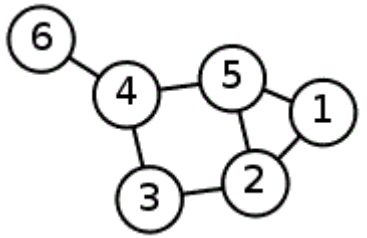
Varredura em largura



Do inglês *Breadth-First Search* (BFS), é uma estratégia de varredura de um grafo onde, partindo de um nó qualquer, os nós adjacentes são visitados primeiro.

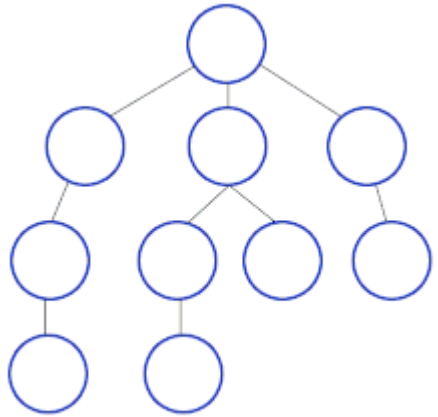
Cada nível de profundidade é verificada em cada passo dessa varredura.

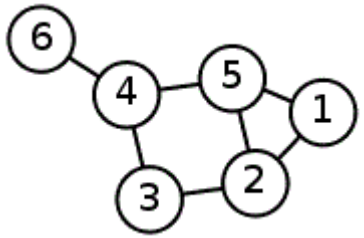
O nível de profundidade de um nó é equivalente a distância (em número de saltos) do nó de origem.



Algoritmo BFS

```
BuscaEmLargura
  escolha uma raiz s de G
  marque s
  insira s em F
  enquanto F não está vazia faça
    seja v o primeiro vértice de F
    para cada w ∈ listaDeAdjacência de v faça
      se w não está marcado então
        visite aresta entre v e w
        marque w
        insira w em F
      senao se w ∈ F entao
        visite aresta entre v e w
      fim se
    fim para
    retira v de F
  fim enquanto
```

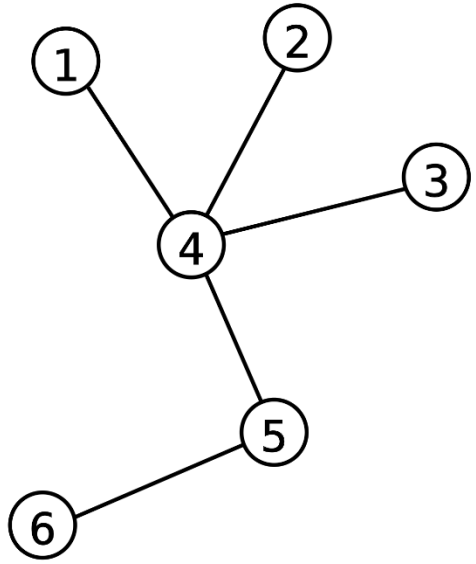




Árvores

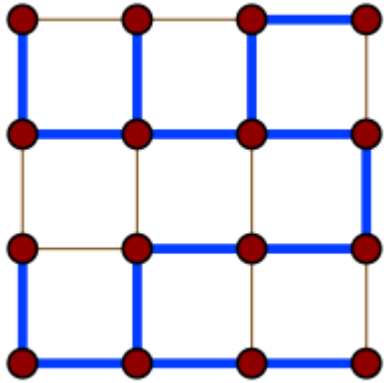
São grafos acíclicos e conexos (se desconexo, é chamado de floresta).

Vamos ver mais detalhes sobre árvores mais adiante na disciplina, quando formos desenvolver as árvores binárias de busca.



Árvore geradora

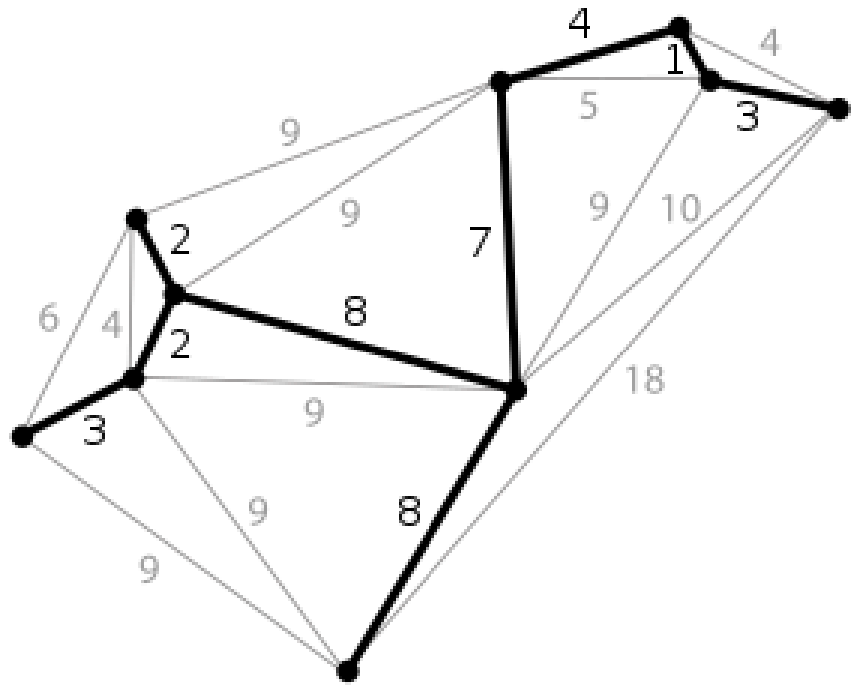
Do inglês *spanning tree*, é uma árvore formada pelo subconjunto de arestas de um grafo conexo, que conecta todos os vértices formando uma árvore.



Árvore geradora mínima

Do inglês *minimum spanning tree* (MST), é a árvore geradora formada pelo subconjunto de arestas com o custo mínimo (ou o peso total mínimo).

Podem existir mais de uma MST. Em um grafo não ponderado, todas as árvores geradoras são mínimas.



Algoritmo de Prim

Algoritmo guloso usado para encontrar a MST em um grafo ponderado, conexo e não direcionado.

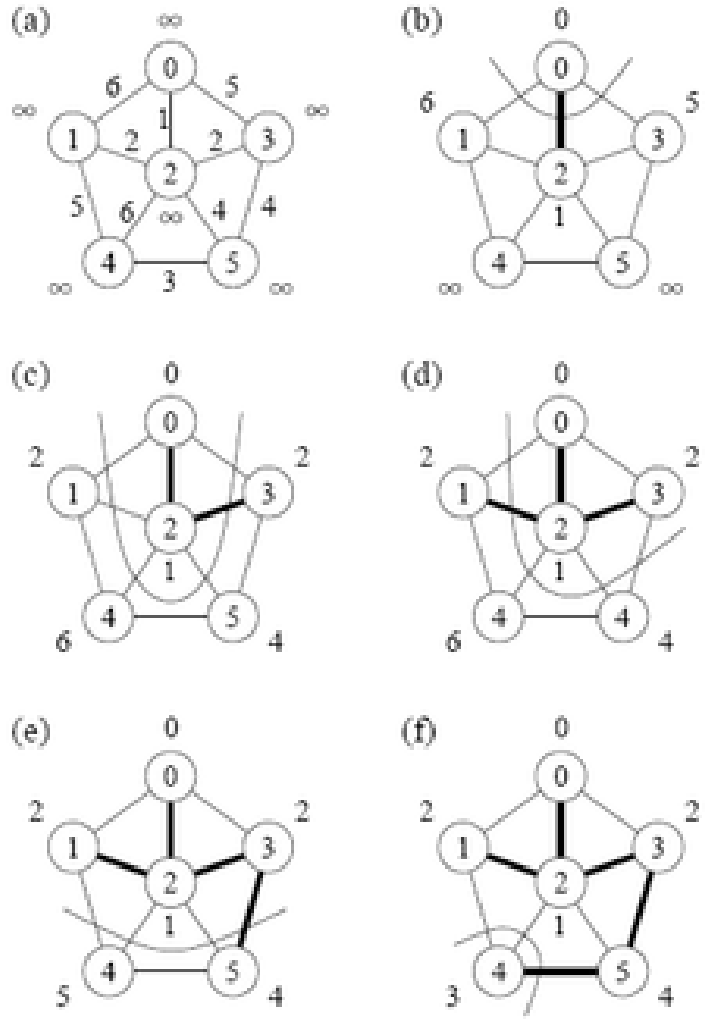
Algoritmo genérico:

*Escolha um vértice **S** para iniciar o subgrafo*

enquanto houver vértices que não estão no subgrafo

selecione uma aresta segura

insira a aresta segura e seu vértice no subgrafo



Algoritmo de Kruskal

Assim como o algoritmo de Prim, é um algoritmo guloso usado para encontrar a MST em um grafo ponderado, conexo e não direcionado.

Algoritmo genérico:

*Crie uma floresta **F**, onde cada vértice no grafo é uma árvore separada;*

*Crie um conjunto **S** contendo todas as arestas do grafo;*

*enquanto **S** for não-vazio, faça:*

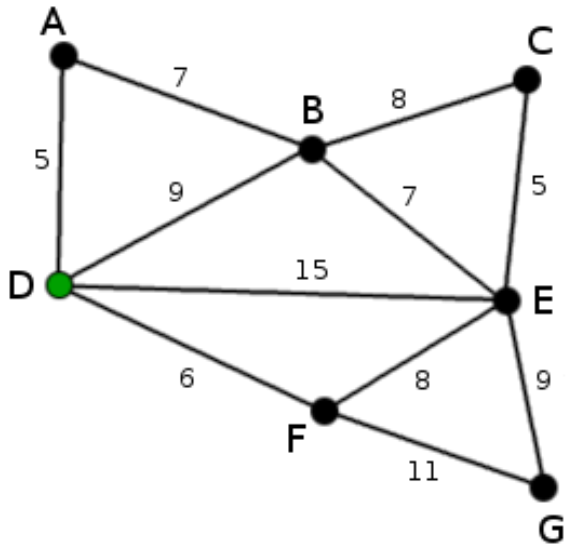
*remova uma aresta com peso mínimo de **S***

se essa aresta conecta duas árvores diferentes, adicione-a à floresta, combinando duas árvores numa única árvore parcial

do contrário, descarte a aresta

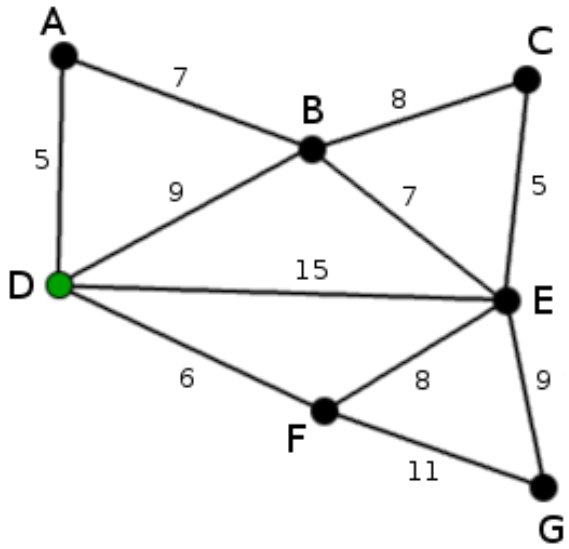
Caminho mínimo

Consiste na minimização do custo total da soma dos pesos de cada aresta considerando a travessia de um grafo partindo de um vértice de origem até encontrar um vértice de destino.



Caminho mínimo

Consiste na minimização do custo total da soma dos pesos de cada aresta considerando a travessia de um grafo partindo de um vértice de origem até encontrar um vértice de destino.

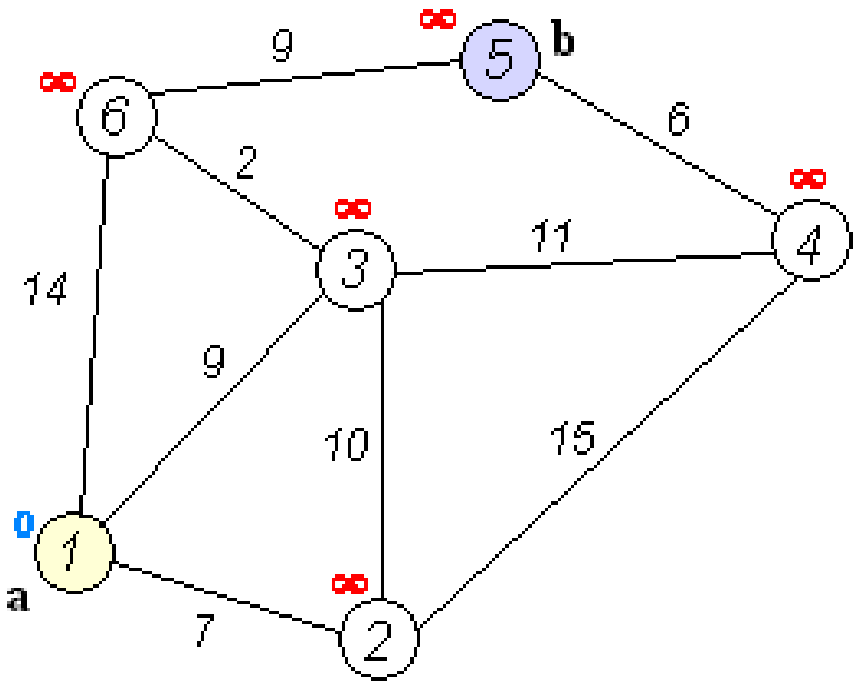


Algoritmo de Dijkstra

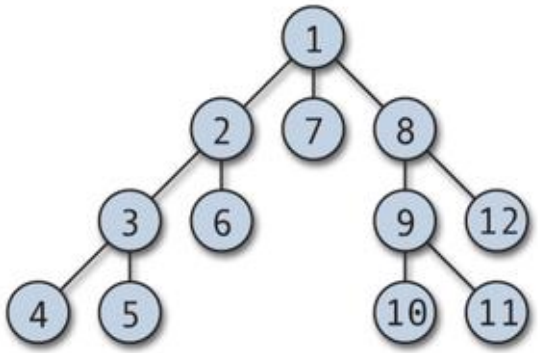
Considera um conjunto S de menores caminhos, iniciado com um vértice inicial I.

Enquanto existir um caminho de I para V e V não estiver em S

- *busca-se nas adjacências dos vértices pertencentes a S aquele vértice com menor distância relativa a I e adiciona-o a S.*
- *Arestas que ligam vértices já pertencentes a S são desconsideradas.*



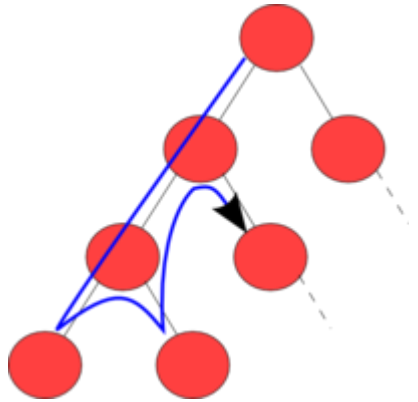
Varredura em profundidade



Do inglês *Depth-First Search* (DFS), é uma estratégia de varredura de um grafo onde, partindo de um nó qualquer, o primeiro nó vizinho é visitado, depois o primeiro vizinho desse novo nó e assim por diante, até que um nó sem vizinhos seja encontrado.

Em seguida, os outros vizinhos do nó anterior são visitados, usando a mesma lógica (*backtracking*).

O processo segue até que todos os nós tenham sido visitados.



Algoritmo DFS

```
BuscaEmProfundidade
  marque s
  para cada w ∈ listaDeAdjacência de s faça
     se w não está marcado então
        BuscaEmProfundidade(w)
     fim se
   fim para
```