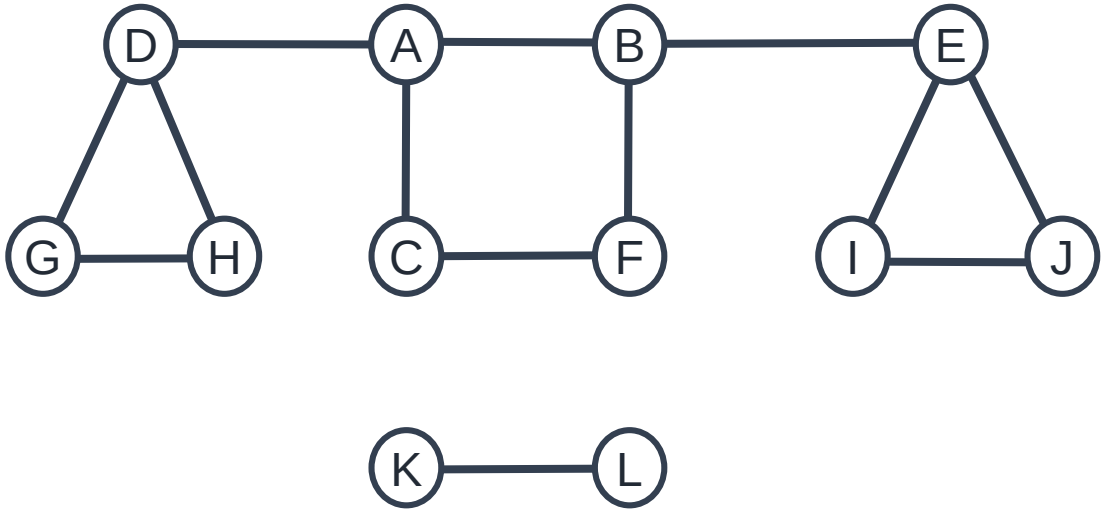
Grafos
Busca

ECM404

Busca em Grafos e Dígrafos

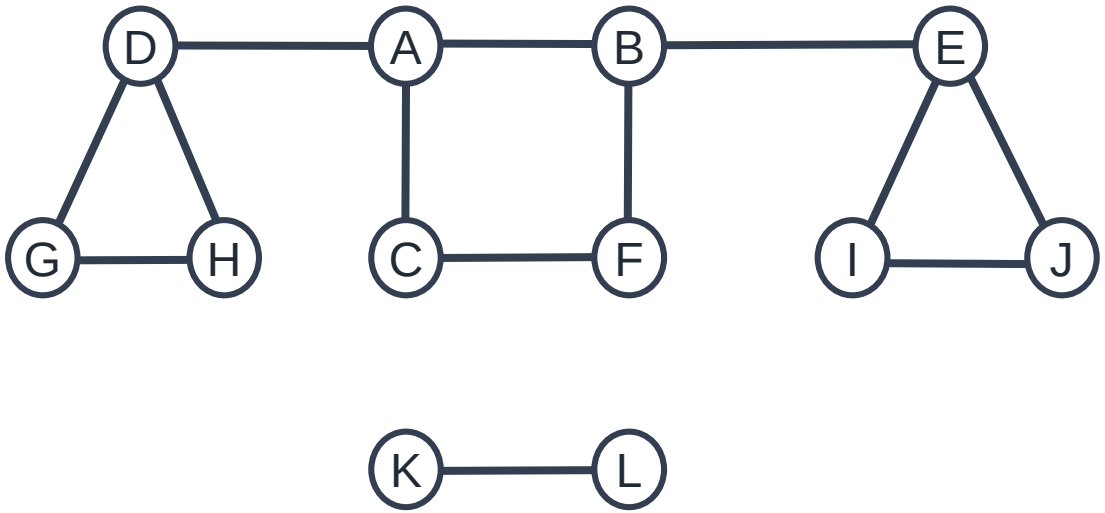
- Algoritmos utilizados para se explorar um grafo, ou seja, um processo sistemático de como caminhar por seus vértices e arestas.
- Estudaremos dois algoritmos:
  - Busca em Profundidade;
  - Busca em Largura.



Busca em Profundidade

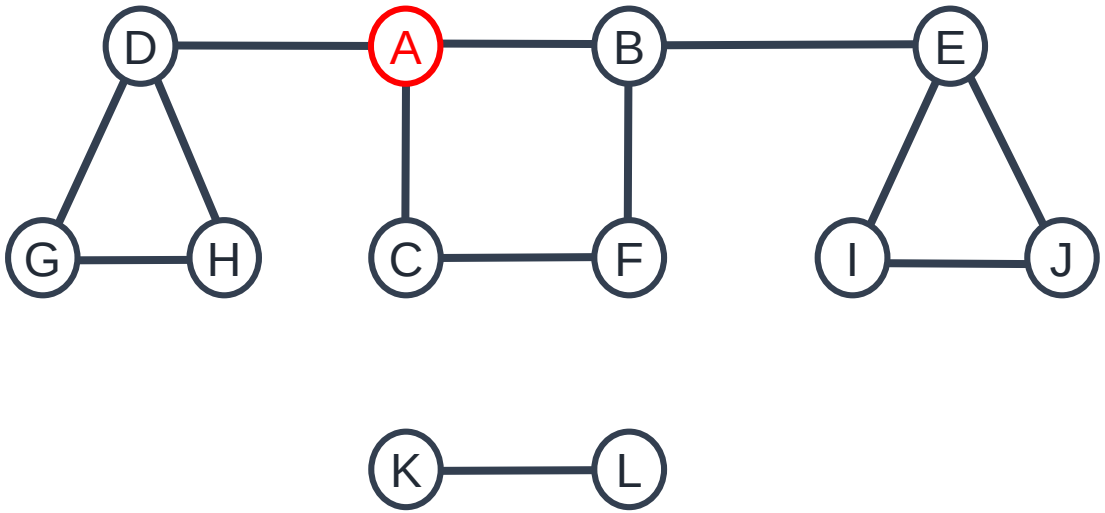
- **Depth First Search (DFS)**. Permite determinar quais partes de um grafo são alcançáveis a partir de um vértice.
- A partir do vértice de partida, explorar um dos vértices adjacentes não visitados. Essa repetição ocorre até que não hajam mais vértices não visitados, obrigando o algoritmo a retornar o caminho e explorar um outro vértice não visitado.
- Cada vértice visitado é inserido em uma pilha. Quando o retorno for necessário, a pilha é desfeita. Para automatizar a pilha, utilizaremos a abordagem recursiva.
- Mesmo que o grafo seja desconexo, todos os vértices serão visitados.

Busca em Profundidade



Nesse exemplo, a escolha dos vértices segue a ordem alfabética

Busca em Profundidade

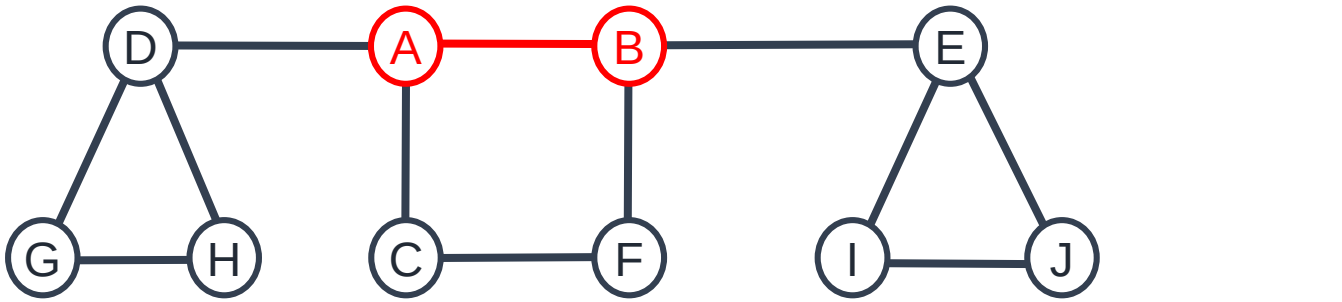


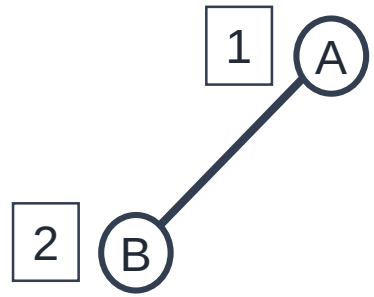
1 A

A

Pilha

Busca em Profundidade



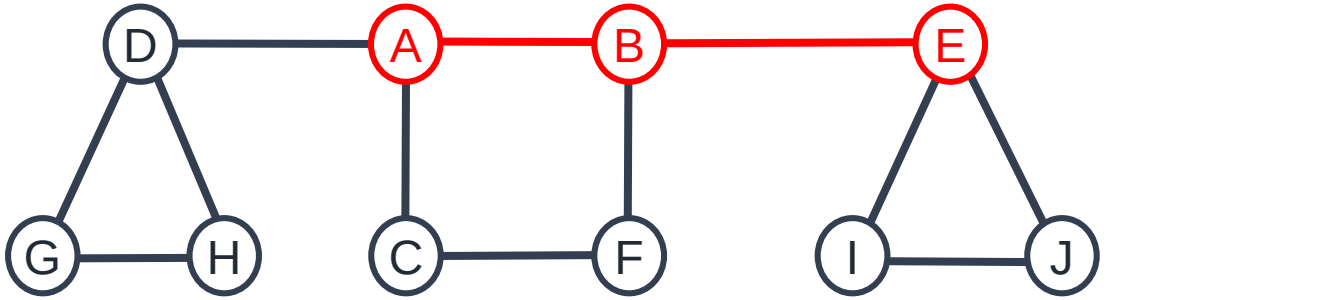


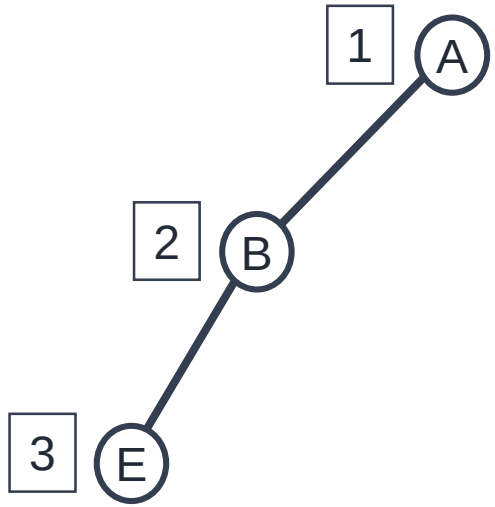
B

A

Pilha

Busca em Profundidade





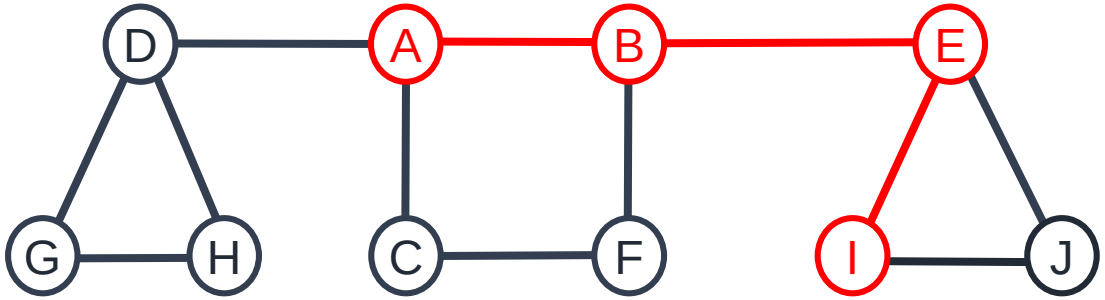
E

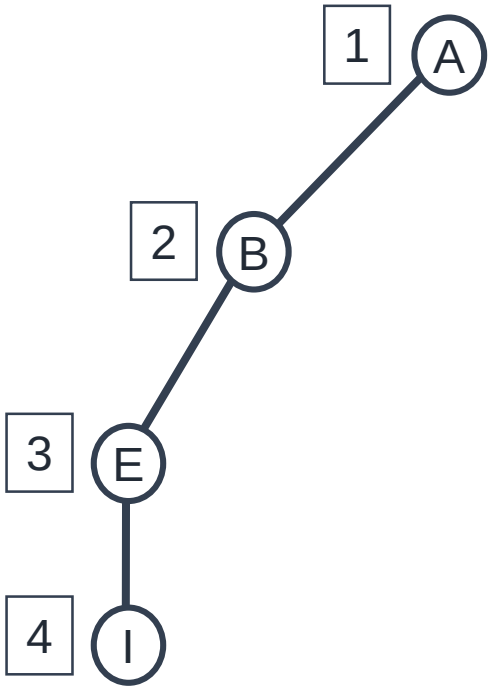
B

A

Pilha

Busca em Profundidade

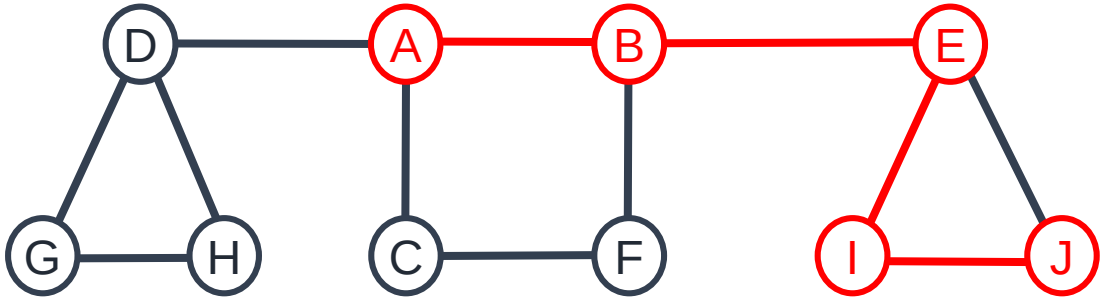


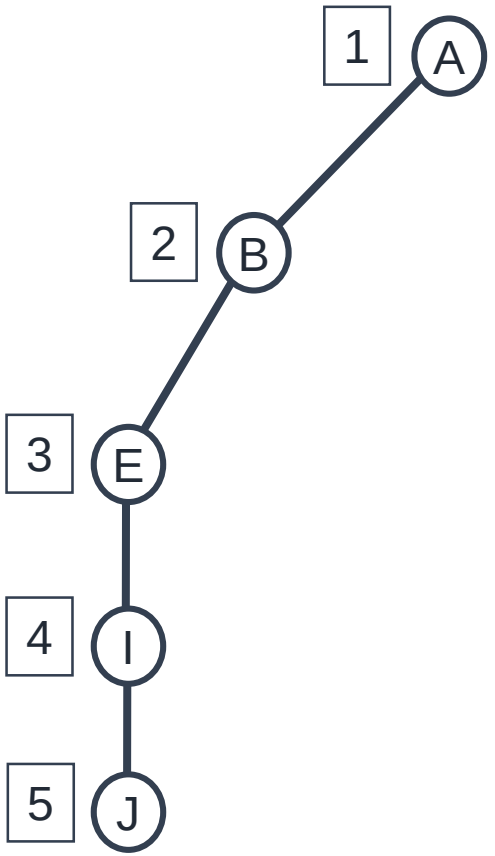


I
E
B
A

Pilha

Busca em Profundidade

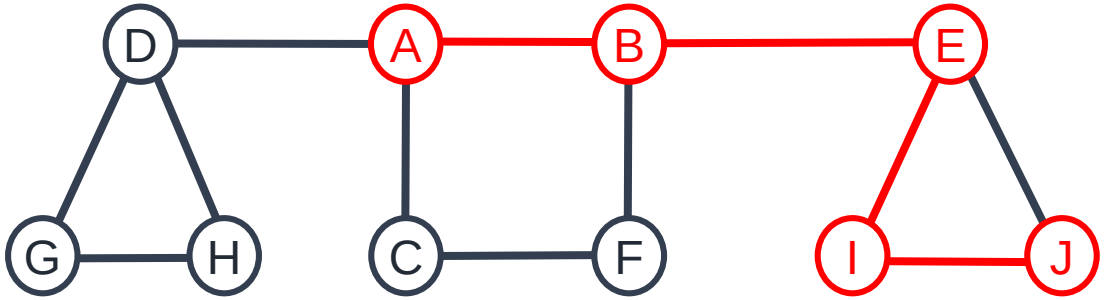


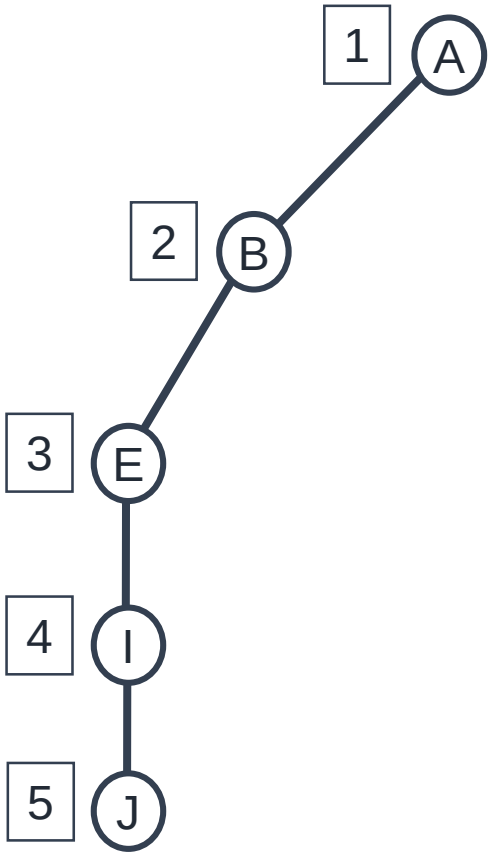


J
I
E
B
A

Pilha

Busca em Profundidade

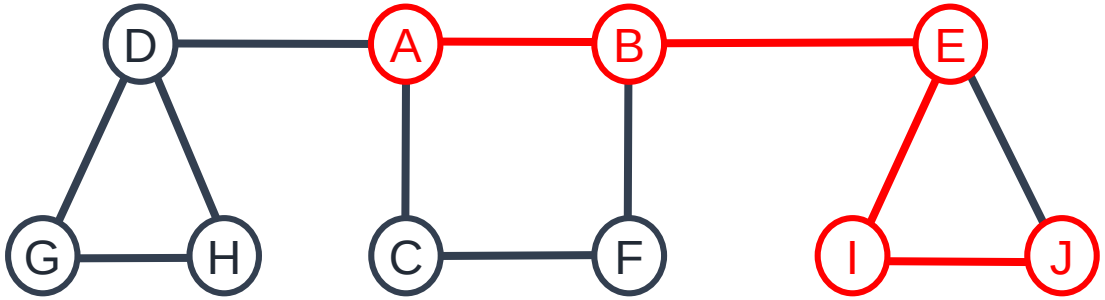


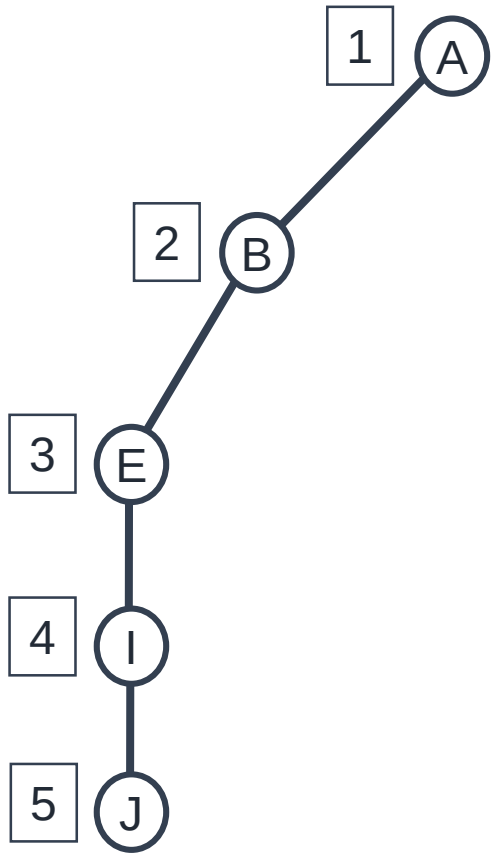


I
E
B
A

Pilha

Busca em Profundidade





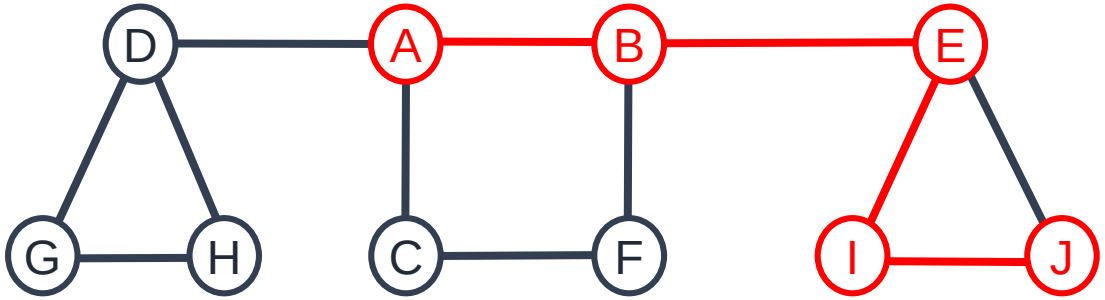
E

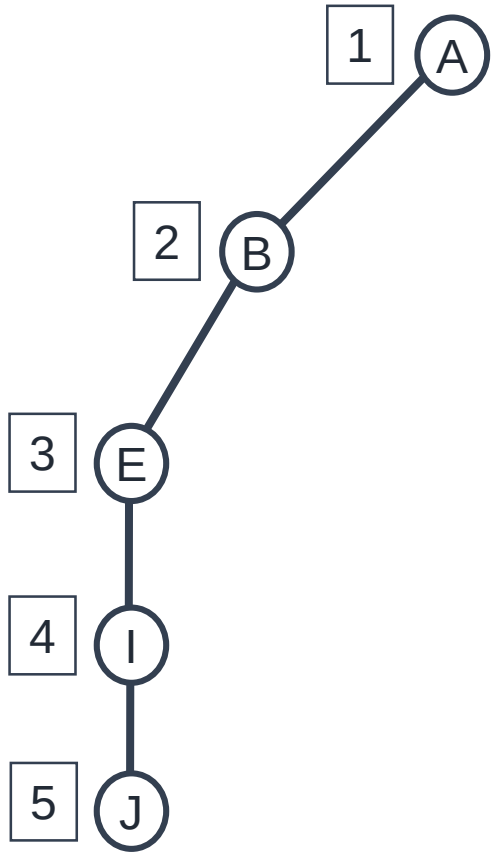
B

A

Pilha

Busca em Profundidade



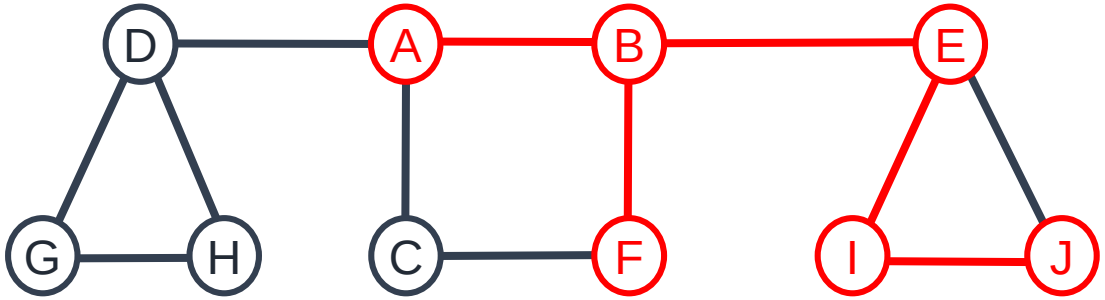


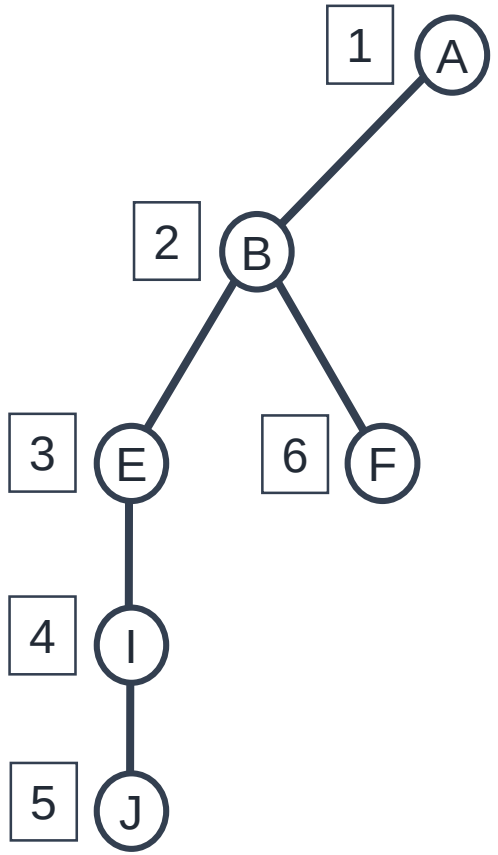
B

A

Pilha

Busca em Profundidade





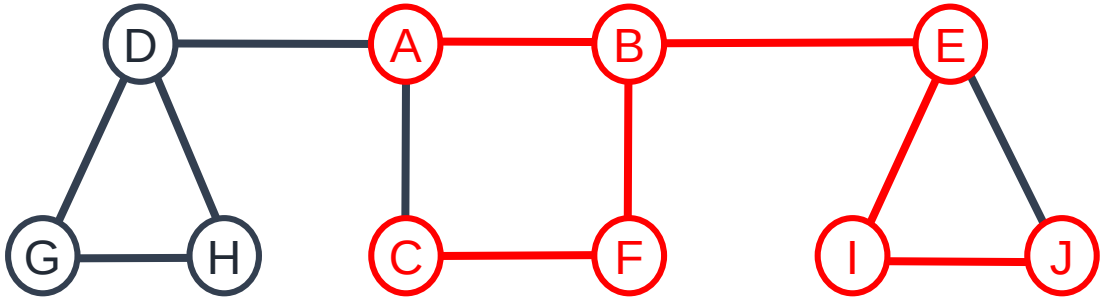
F

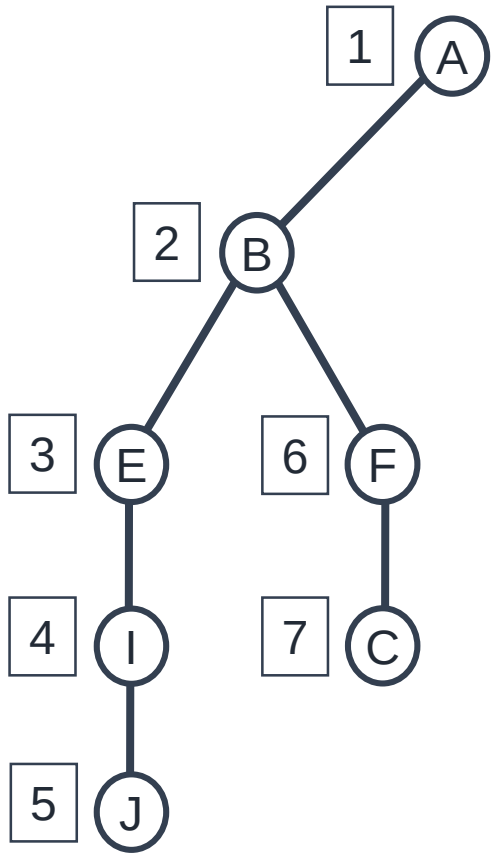
B

A

Pilha

Busca em Profundidade





C

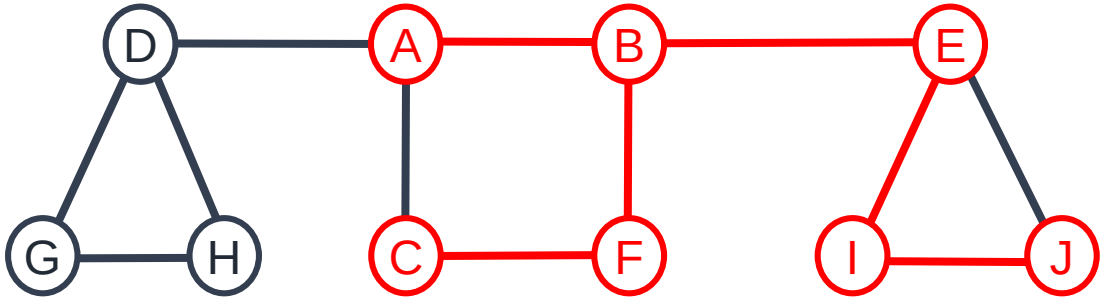
F

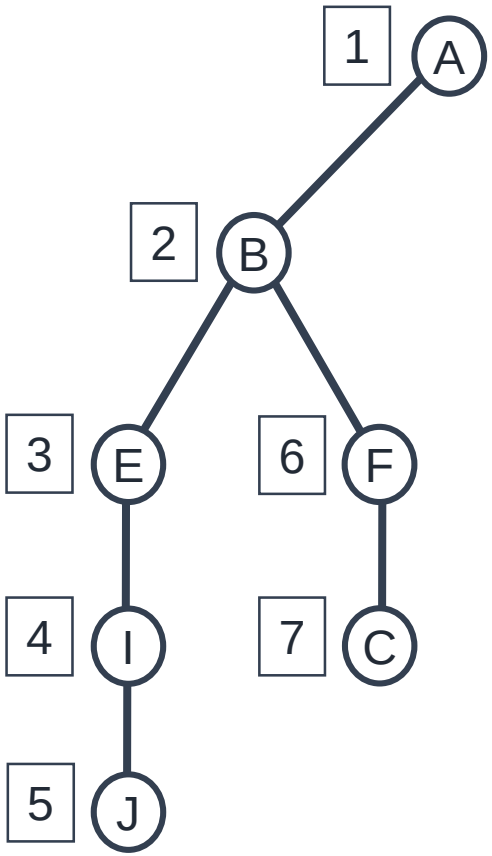
B

A

Pilha

Busca em Profundidade





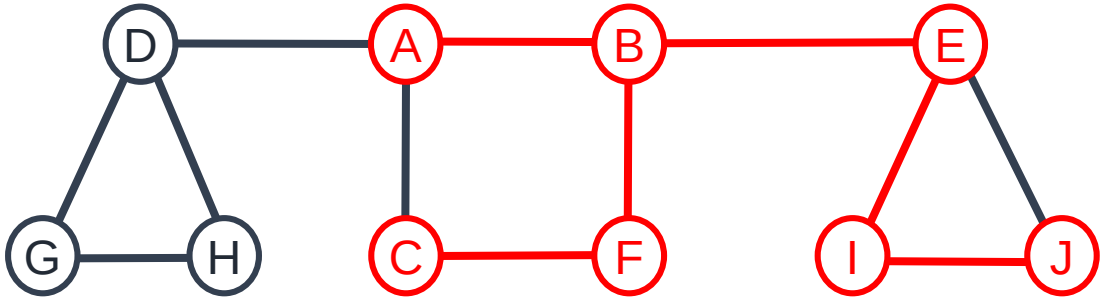
F

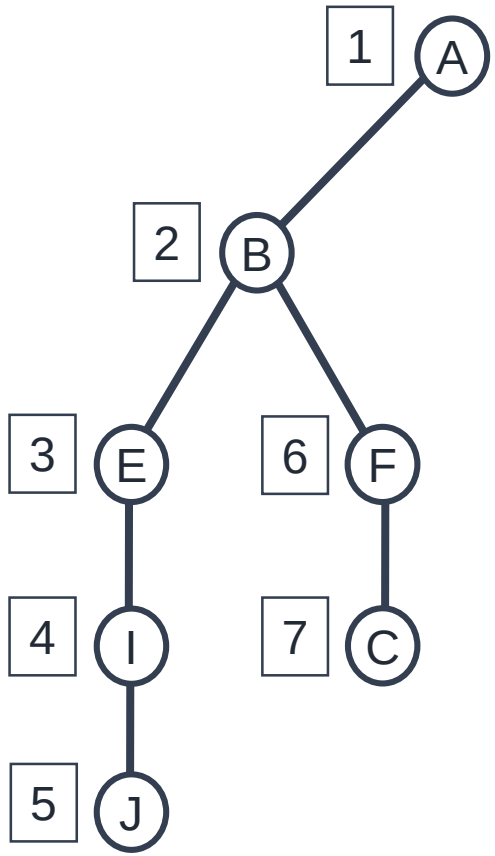
B

A

Pilha

Busca em Profundidade



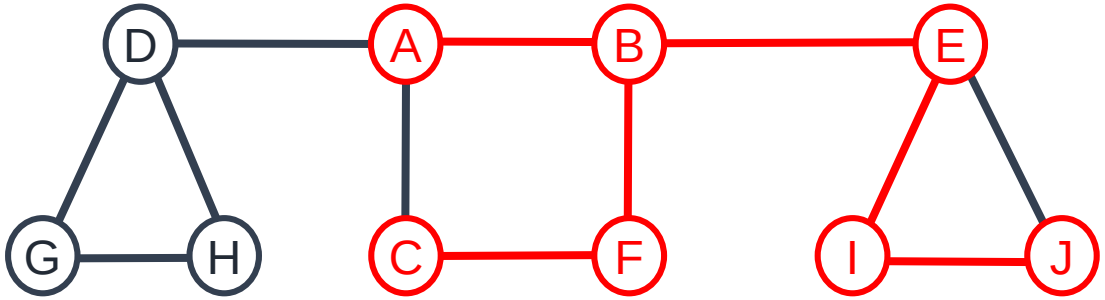


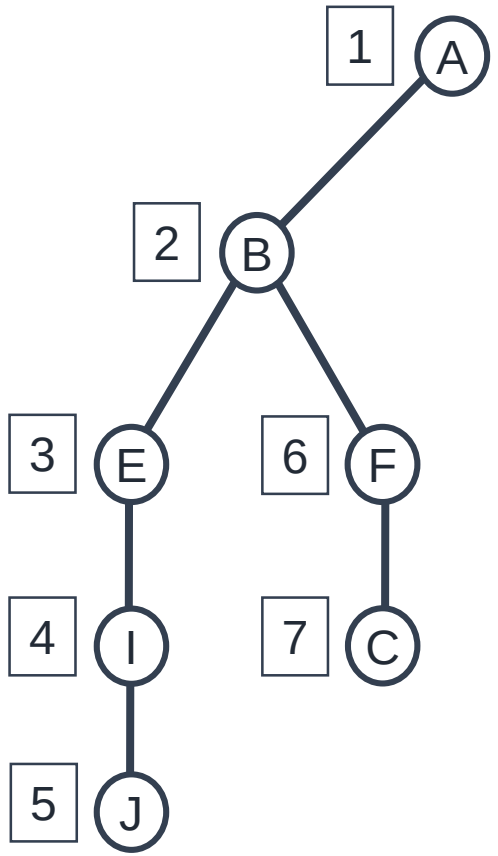
B

A

Pilha

Busca em Profundidade

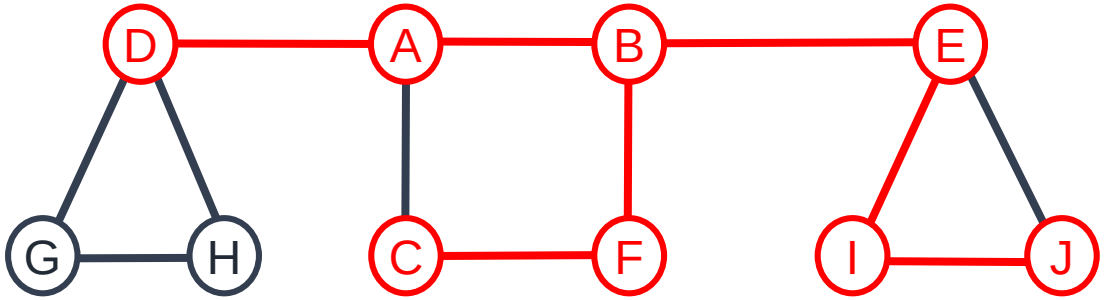


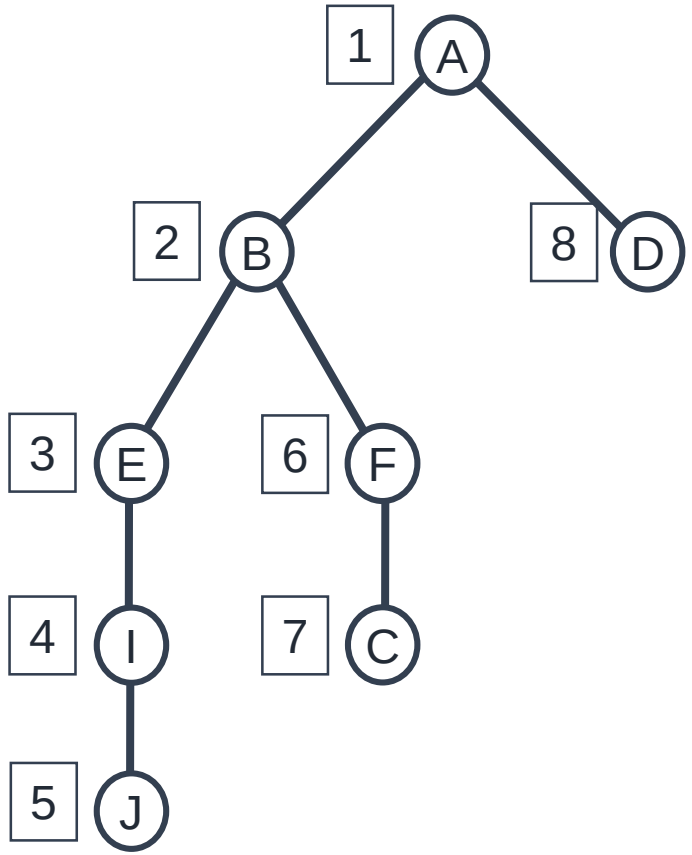


A

Pilha

Busca em Profundidade



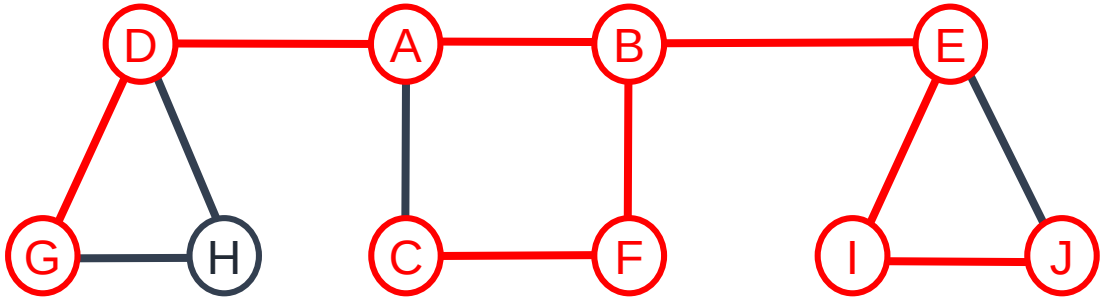


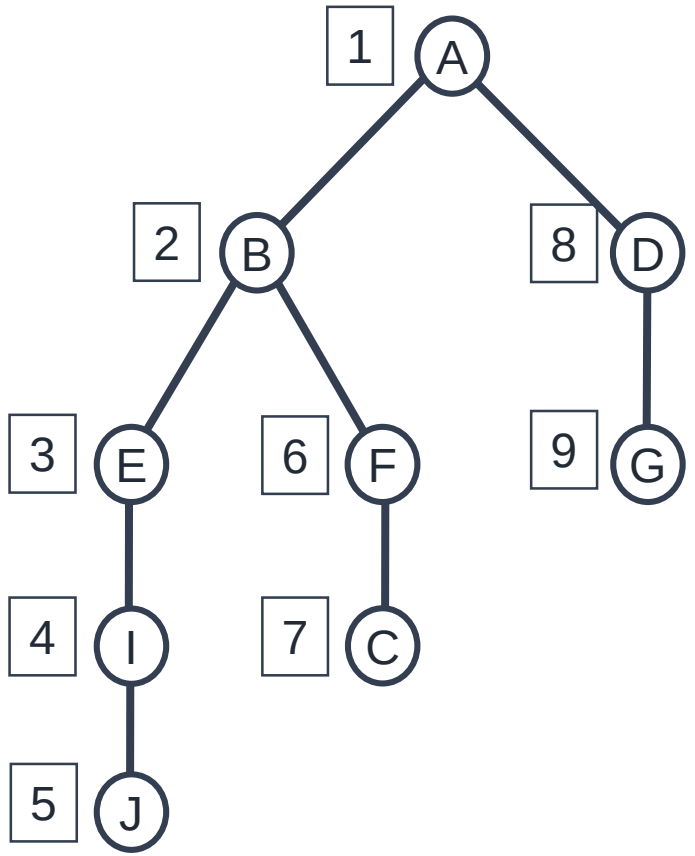
D

A

Pilha

Busca em Profundidade





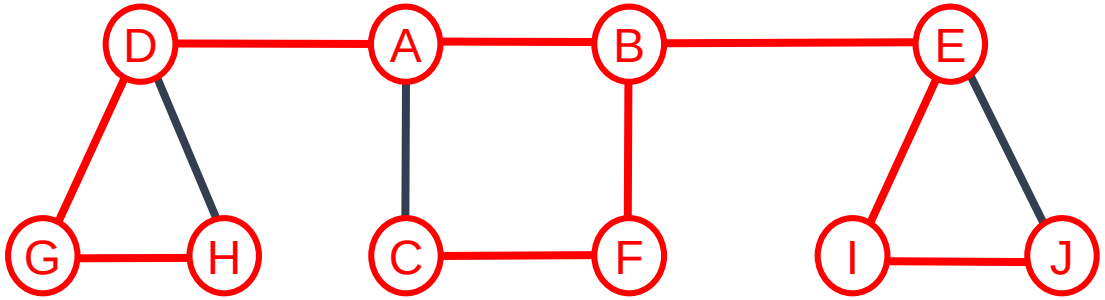
G

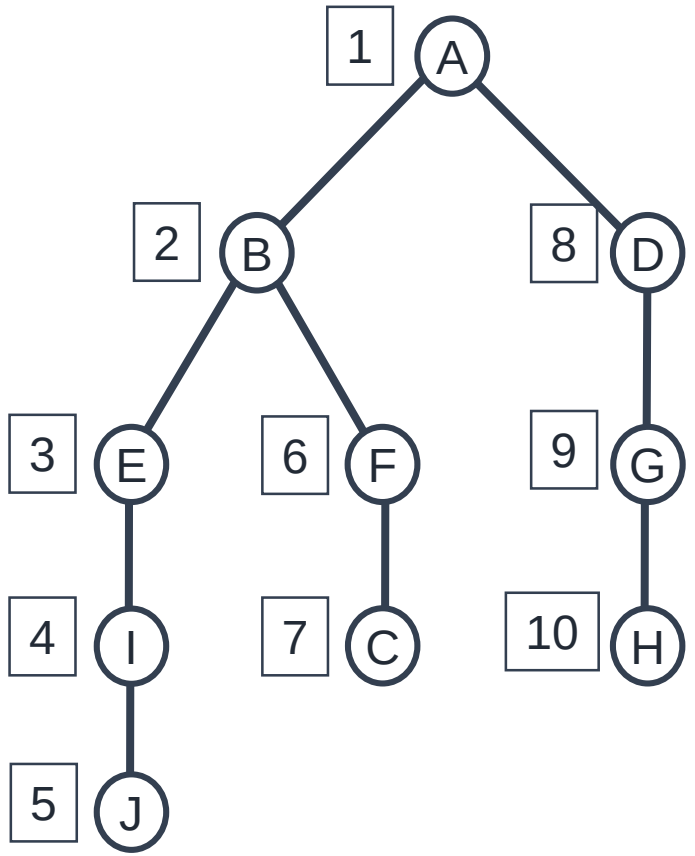
D

A

Pilha

Busca em Profundidade





H

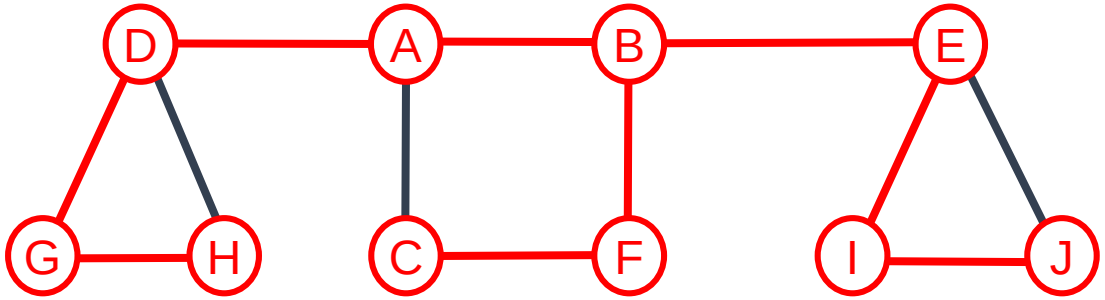
G

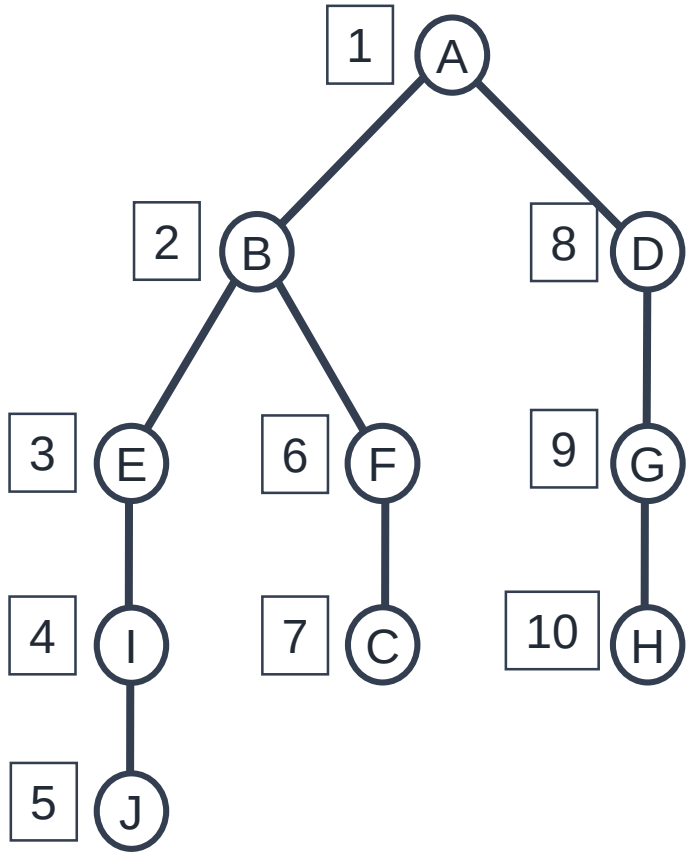
D

A

Pilha

Busca em Profundidade





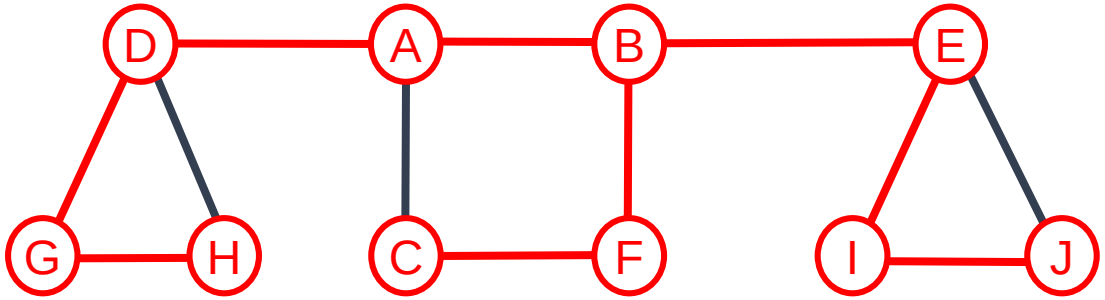
G

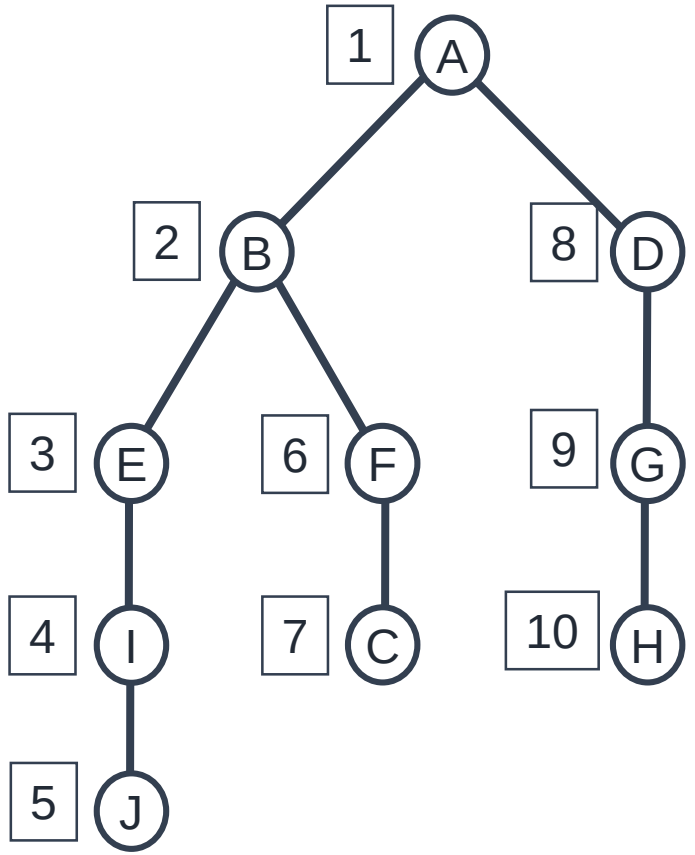
D

A

Pilha

Busca em Profundidade



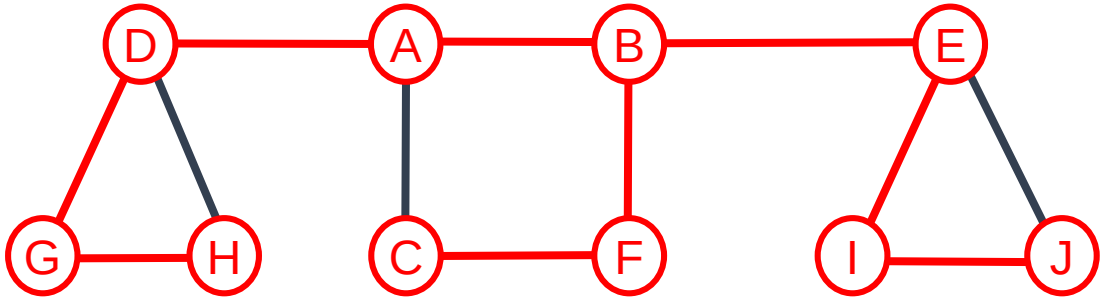


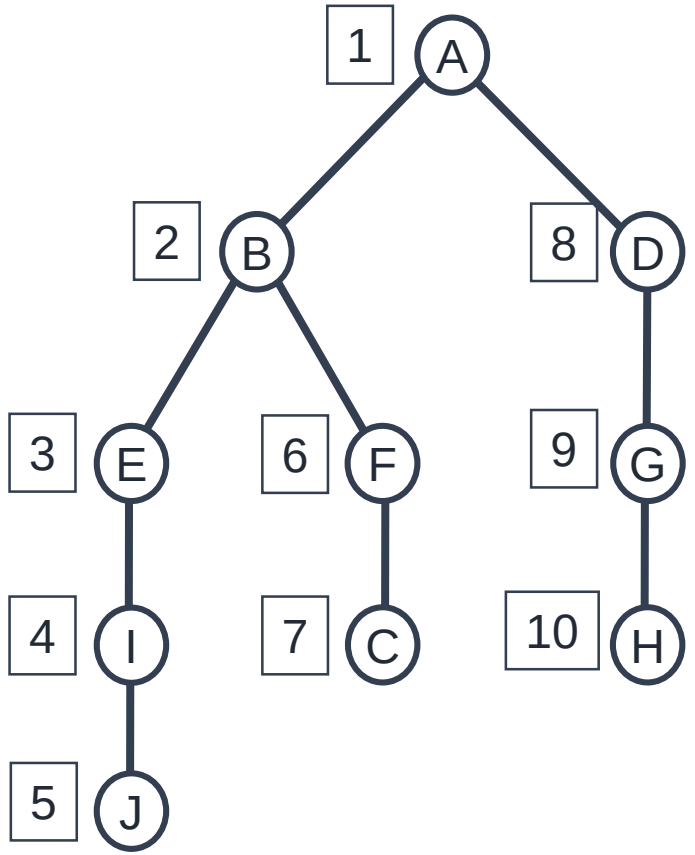
D

A

Pilha

Busca em Profundidade

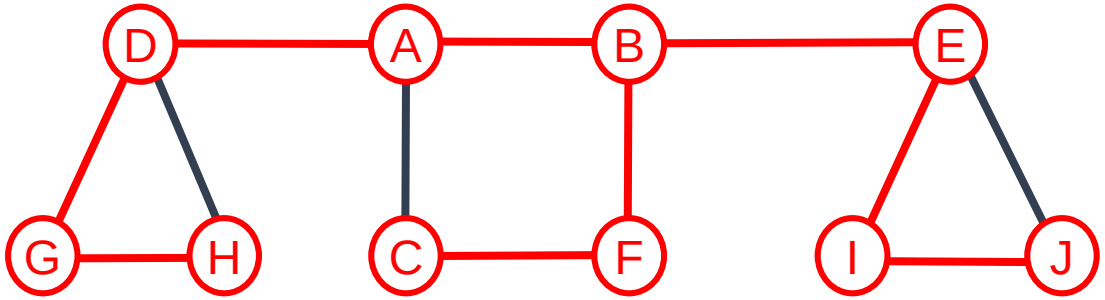


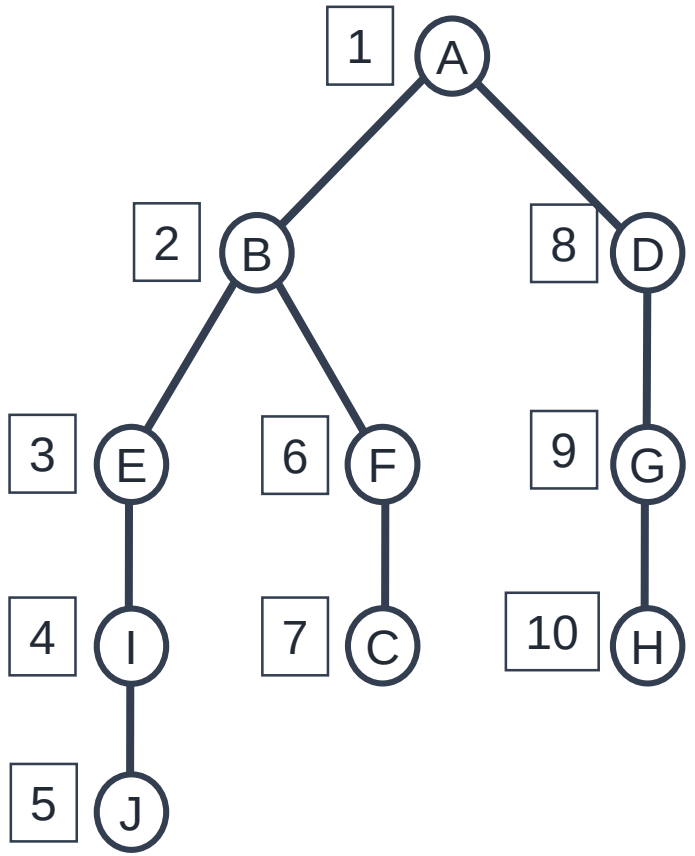


A

Pilha

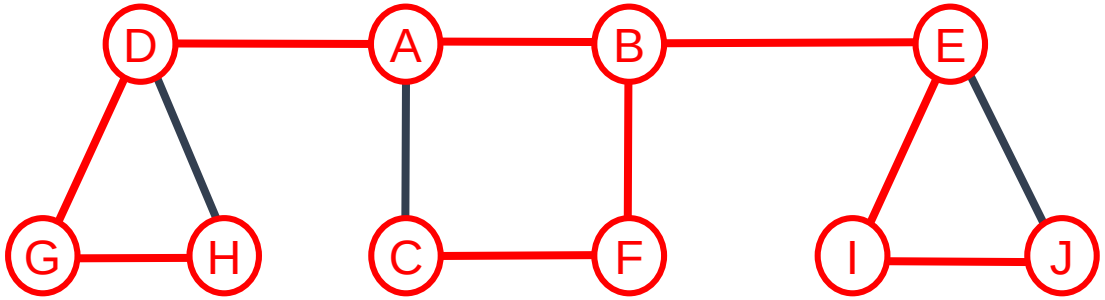
Busca em Profundidade

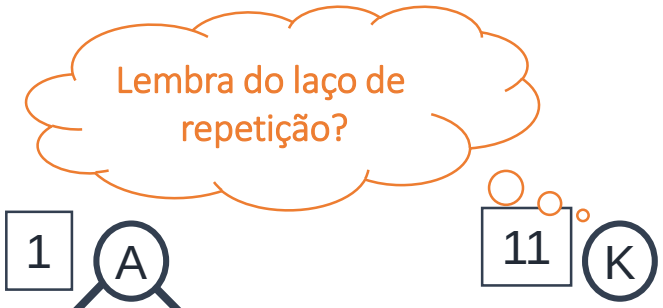


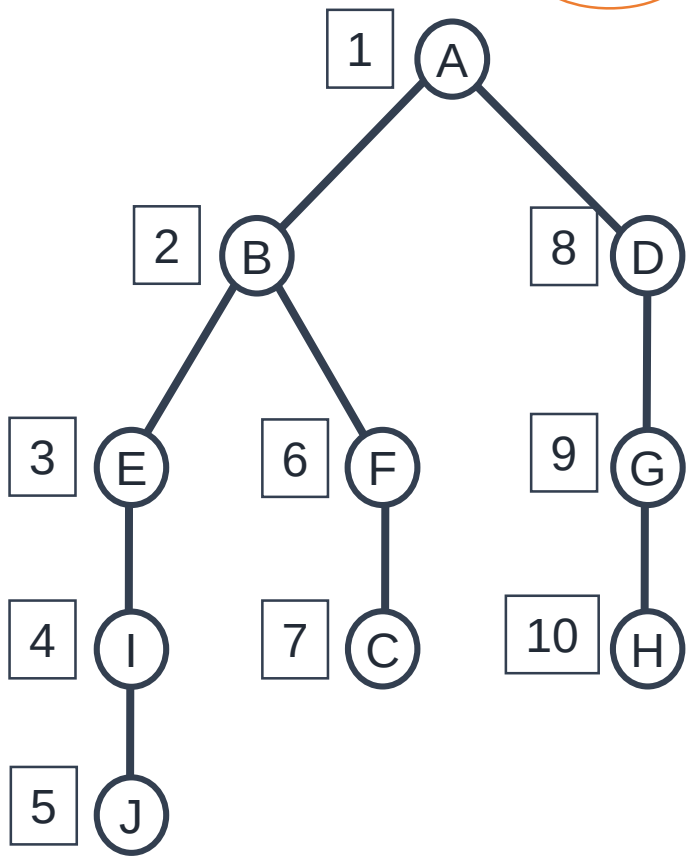


Pilha

Busca em Profundidade



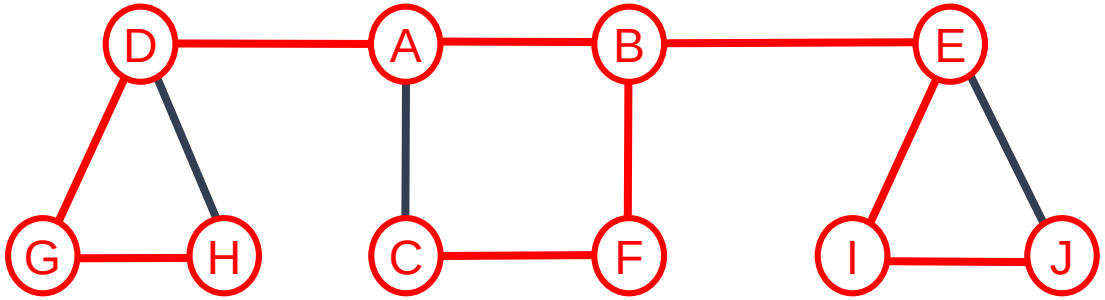


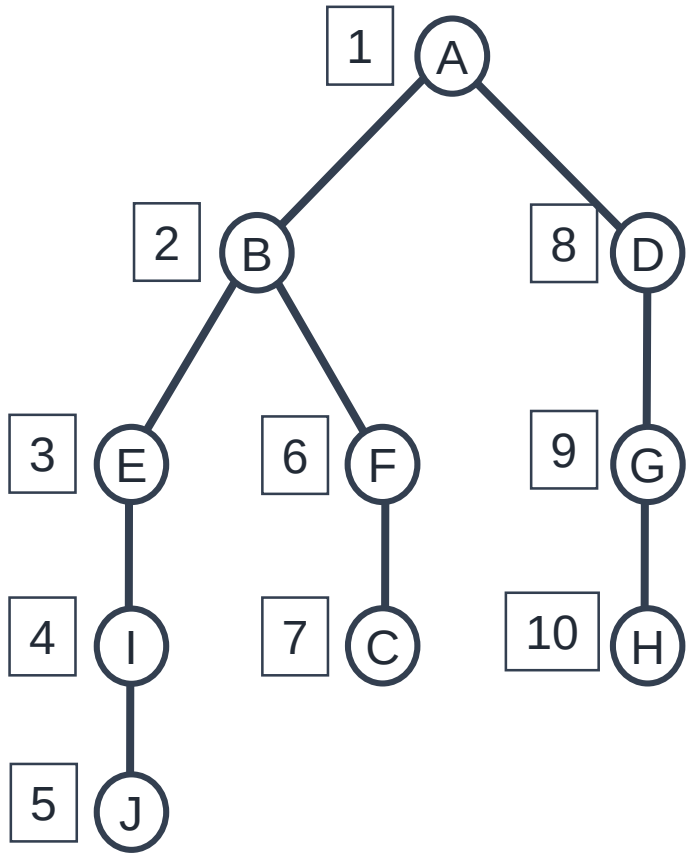


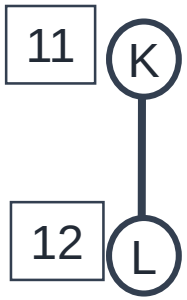
K

Pilha

Busca em Profundidade





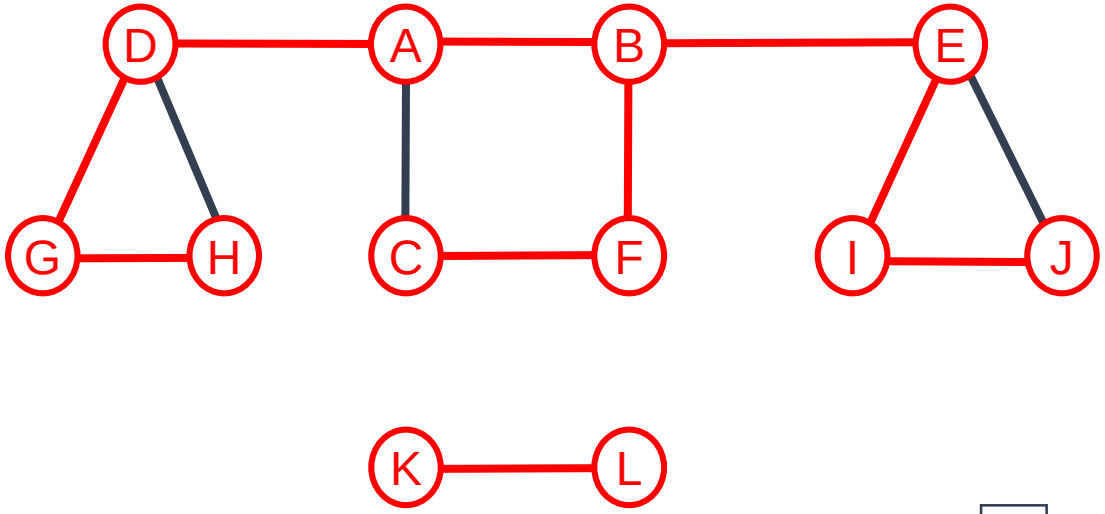


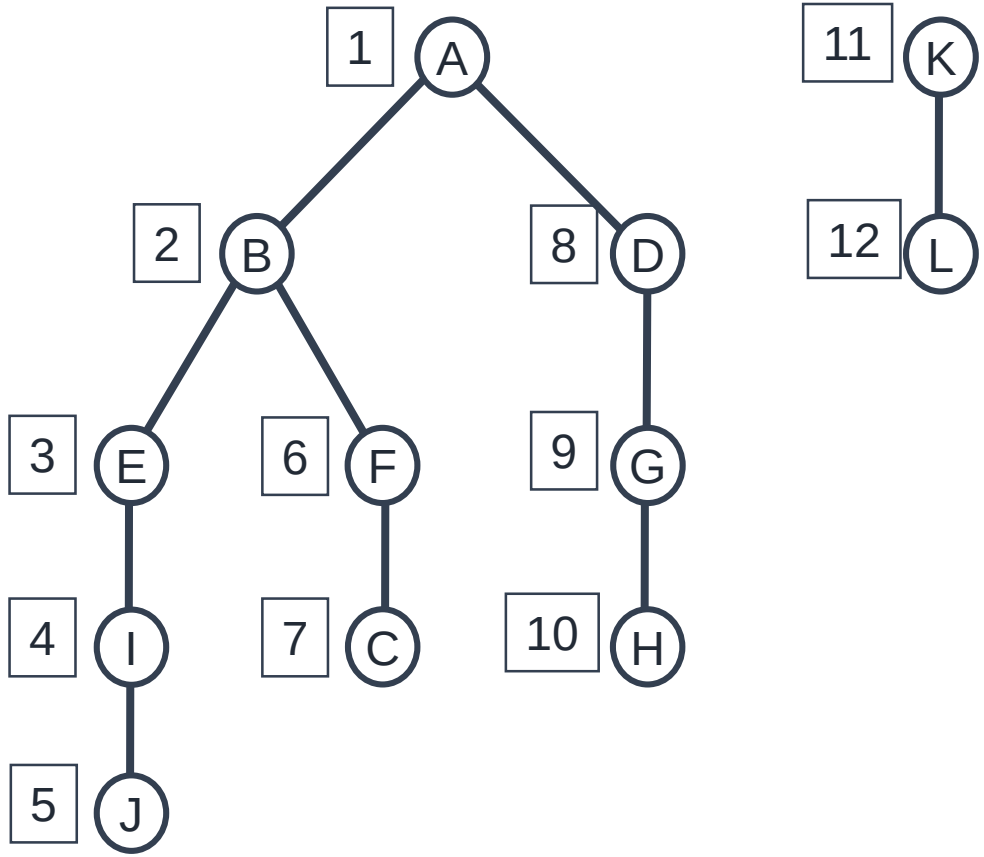
L

K

Pilha

Busca em Profundidade

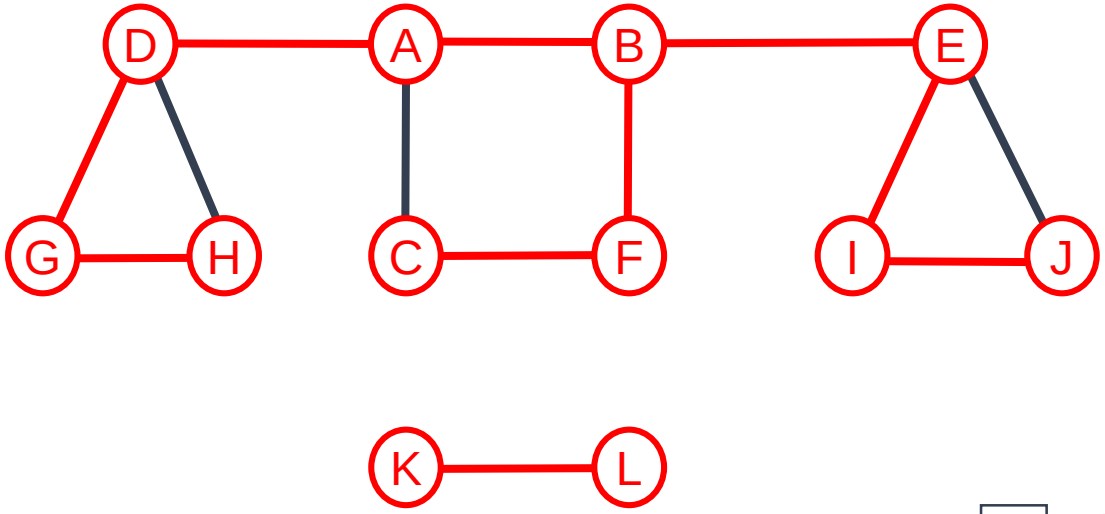


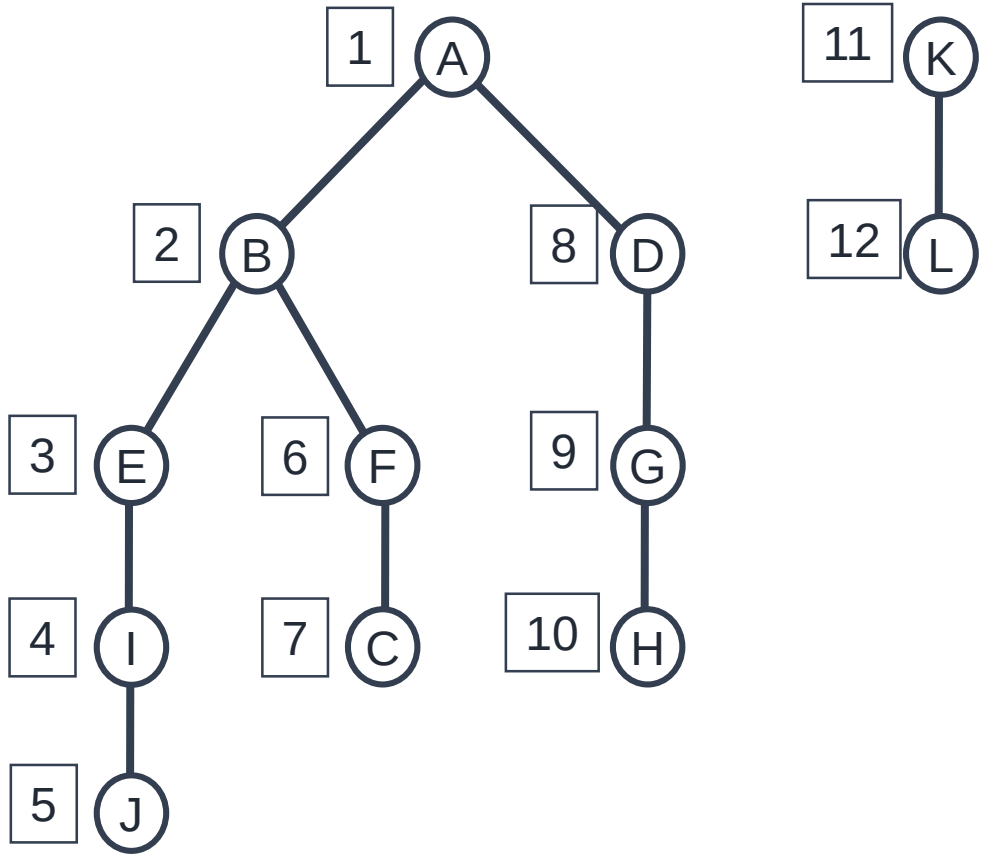


K

Pilha

Busca em Profundidade





Pilha

Busca em Profundidade

```
Procedimento DFS(Grafo g, int origem, int destino,
                                        ListaDeVertices *caminho){
  achei ← falso // se for procurar algo
  caminho->nVertices ← 0
  Para cada vértice v de g faça
    visitado[v] ← falso


  Explorar(g,origem,destino,visitado,caminho,&achei);

  Para cada vértice v de g faça
    Se (!visitado[v] && !achei) então
      Explorar(g,v,destino,visitado,caminho,&achei)
}
```

Busca em Profundidade

```
Procedimento Explorar (Grafo g, int v, int destino,
 int *visitado, ListaDeVertices *caminho, int *achei){
   visitado[v] ← verdadeiro



   Para cada vértice u de g faça
      Se (!visitado[u])
            && (u adjacente v)
              && (!(*achei)) então
         Explorar(g,u,destino,visitado,caminho,achei)


}
```

Busca em Profundidade

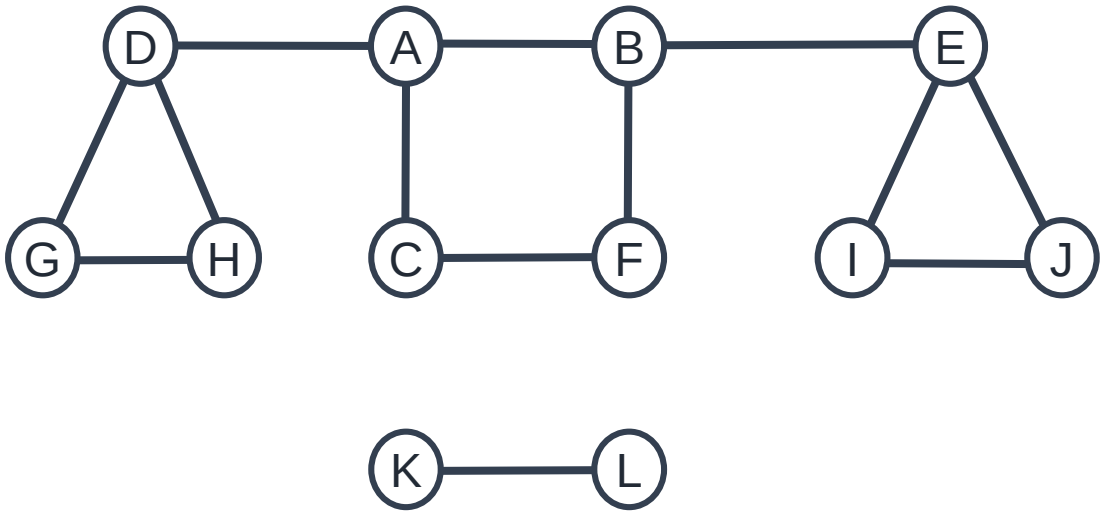
```
Procedimento Explorar (Grafo g, int v, int destino,
 int *visitado, ListaDeVertices *caminho, int *achei){
   visitado[v] ← verdadeiro

   Pre_visita_DFS(parâmetros) //opcional

   Para cada vértice u de g faça
      Se (!visitado[u])
            && (u adjacente v)
              && (!(*achei)) então
         Explorar(g,u,destino,visitado,caminho,achei)

   Pos_visita_DFS(parâmetros) // opcional
}
```

Os procedimentos Pre_visita e Pos_visita contém os algoritmos específicos para a resolução de algum problema

Busca em Largura

- **Breadth First Search (BFS)**. Normalmente utilizada para determinar caminhos mais curtos entre vértices.
- A partir do vértice de partida, expandir todos os seu vértices adjacentes em um ***novo nível***. Nesse novo nível, repetir a operação para todos os vértices que possuam adjacentes não visitados.
- Cada vértice visitado insere os vértices adjacentes a ele em uma fila. Quando todos os adjacentes a ele forem inseridos, o vértice analisado é retirado da ***fila***.
- Se o grafo for desconexo, alguns vértices não serão visitados.

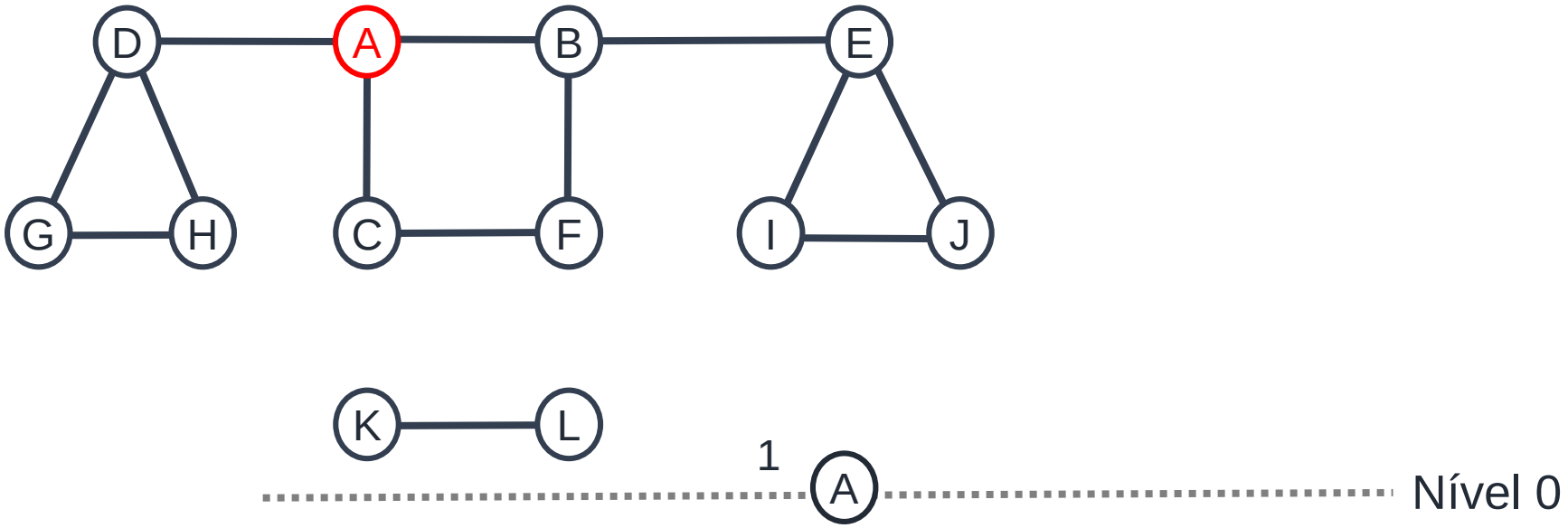
Busca em Largura



Nesse exemplo, a escolha dos vértices segue a ordem alfabética

Fila

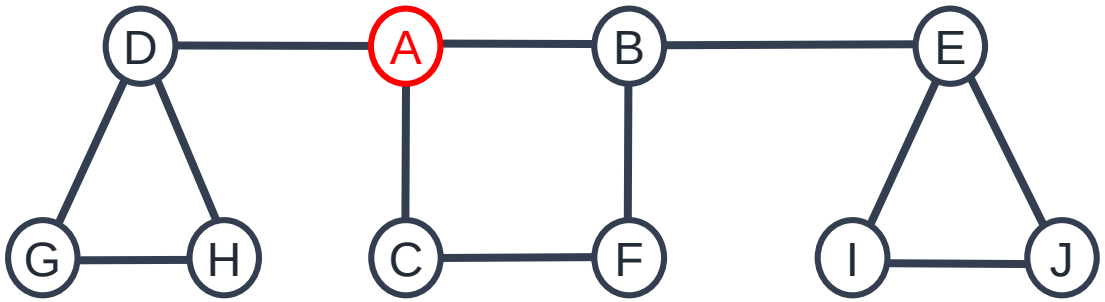
Busca em Largura



A

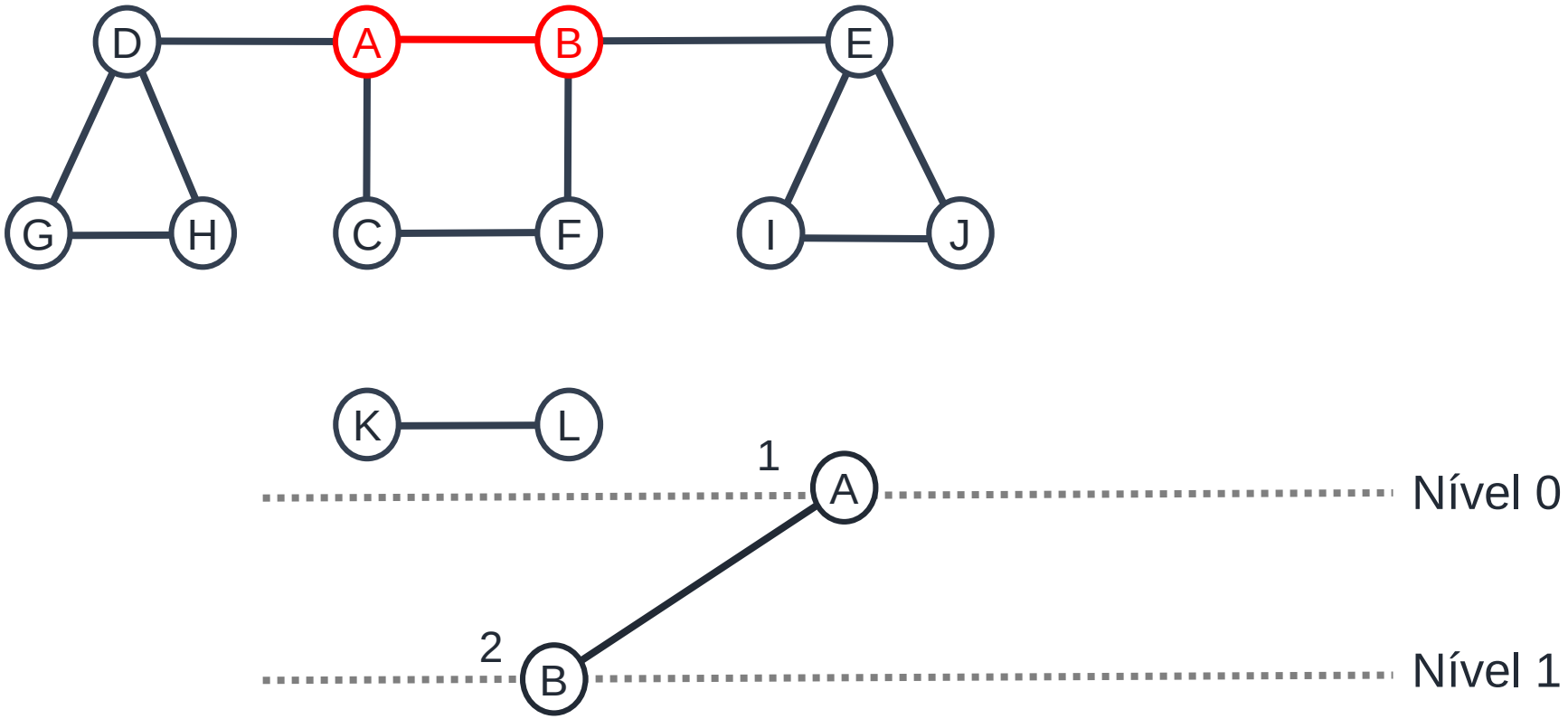
Fila

Busca em Largura



Fila

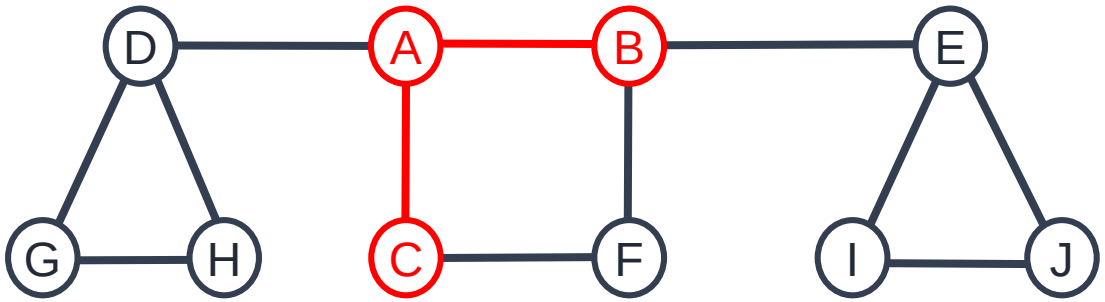
Busca em Largura

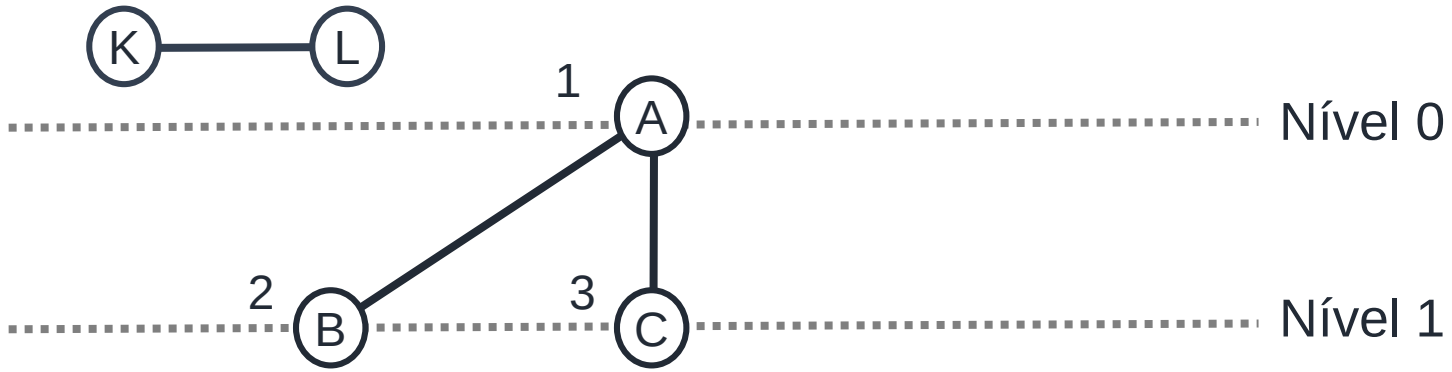


B

Fila

Busca em Largura



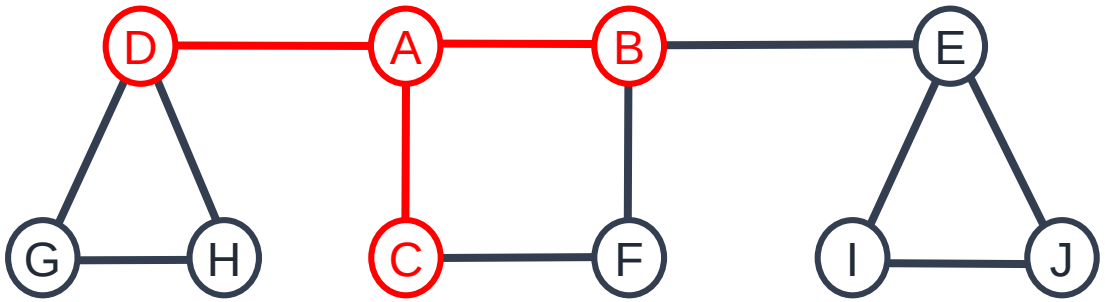


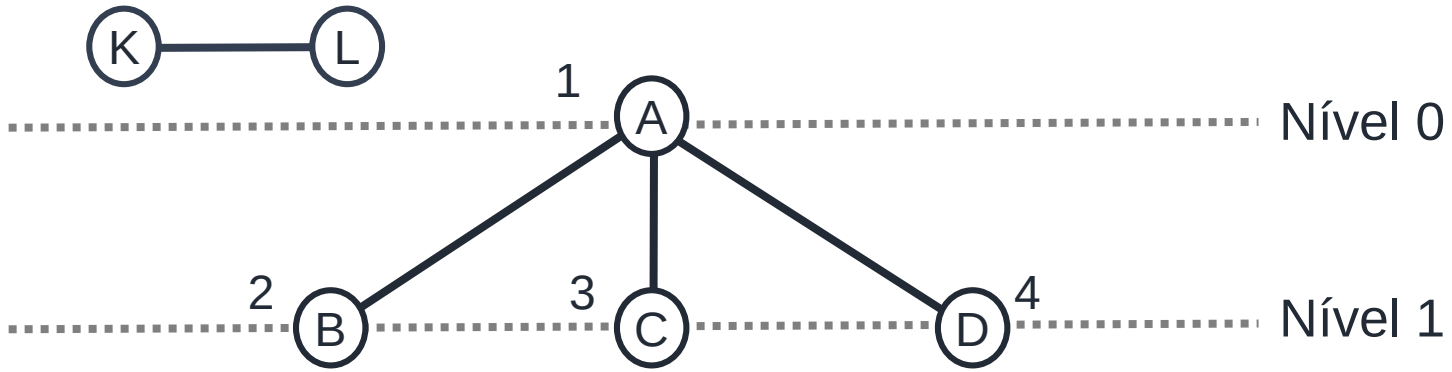
C

B

Fila

Busca em Largura





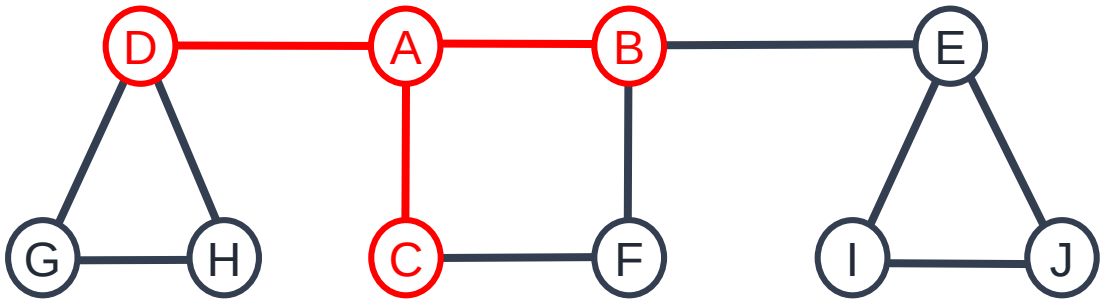
D

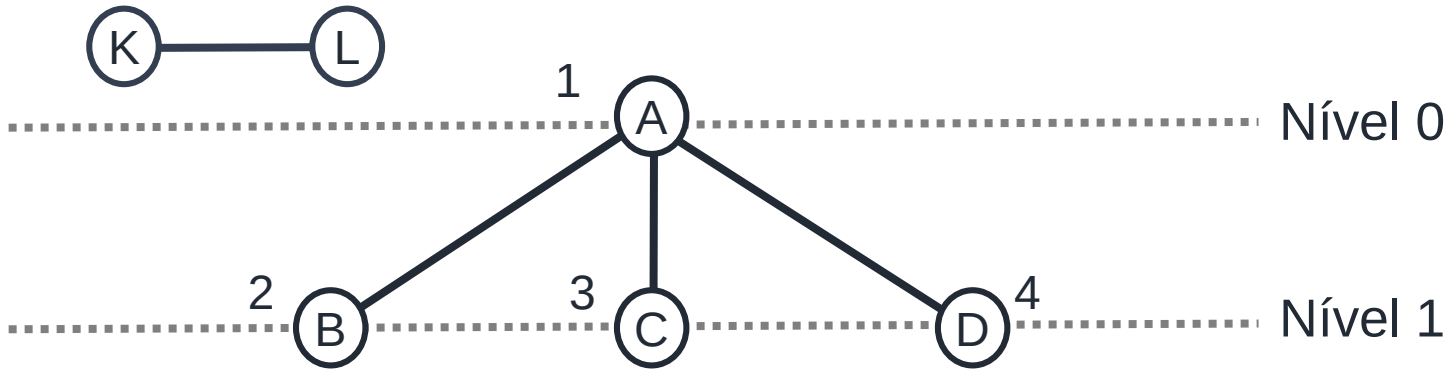
C

B

Fila

Busca em Largura



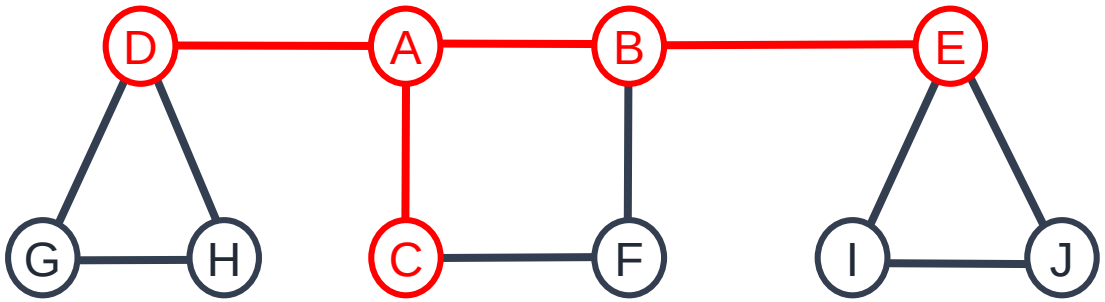


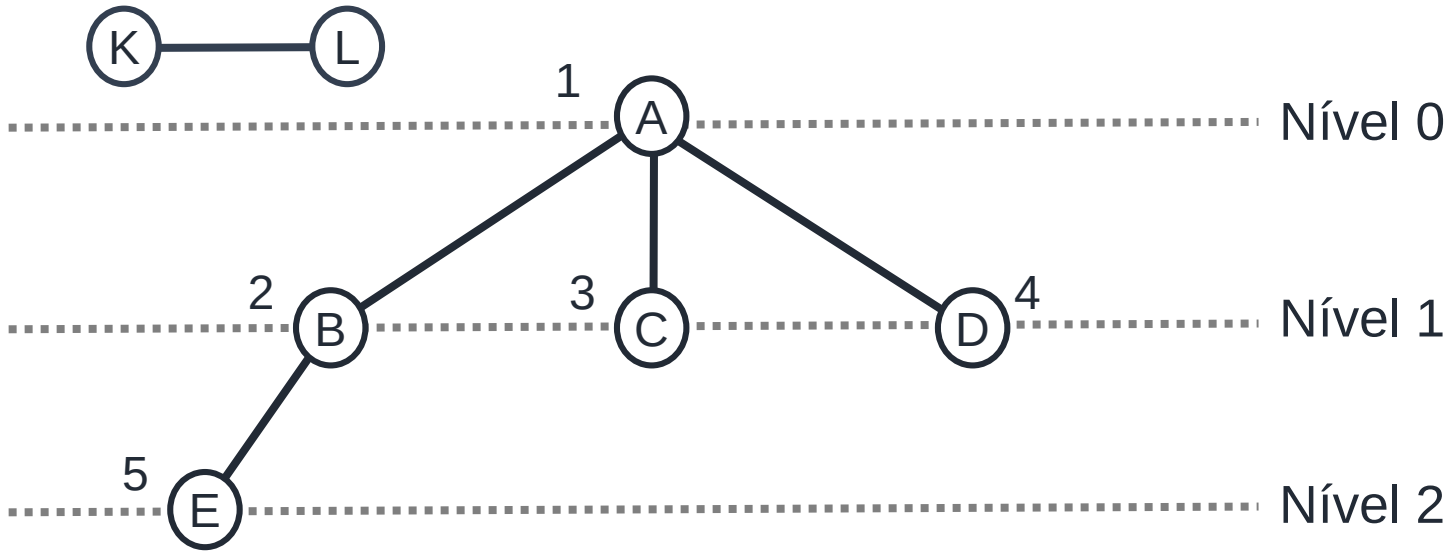
D

C

Fila

Busca em Largura





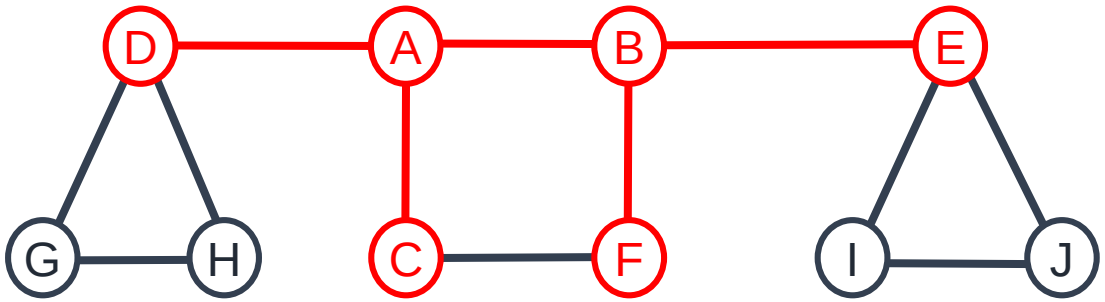
E

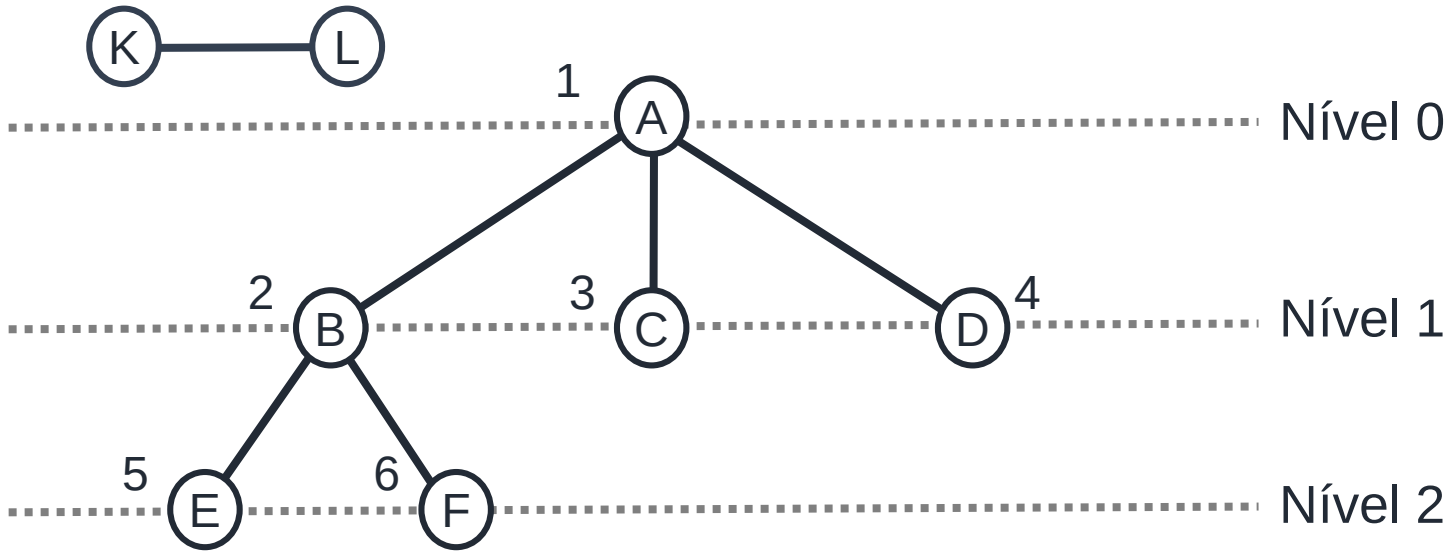
D

C

Fila

Busca em Largura

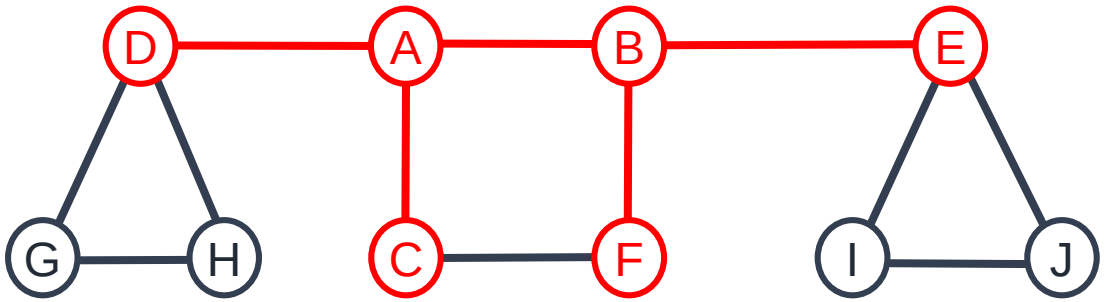


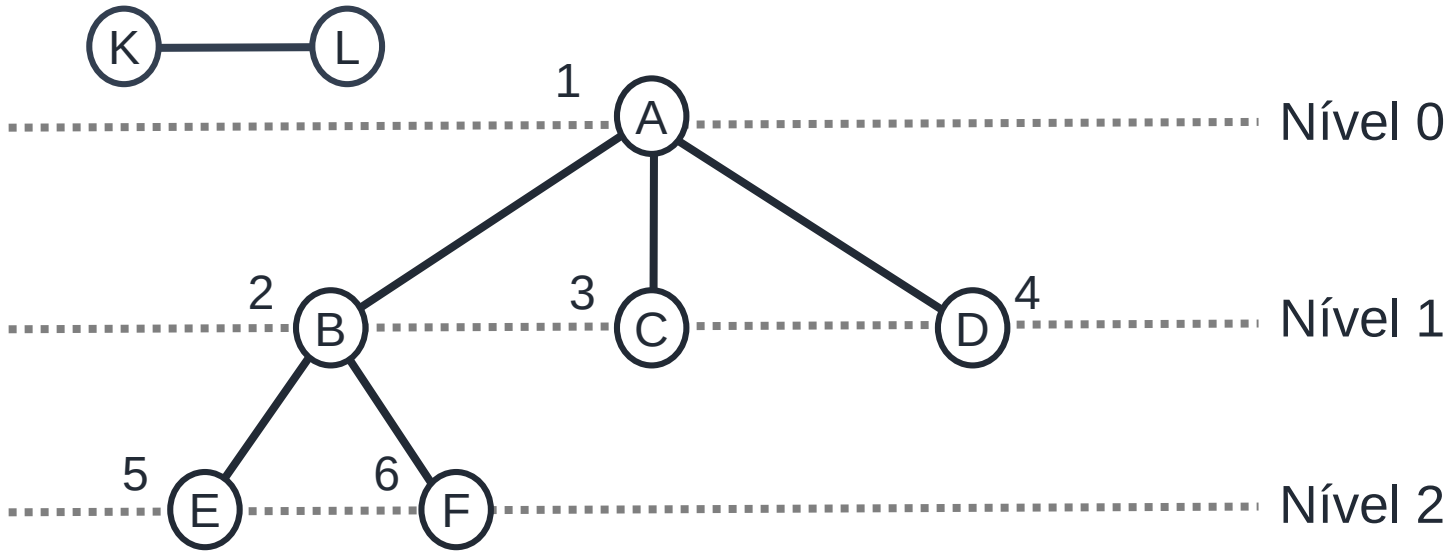


F
E
D
C

Fila

Busca em Largura





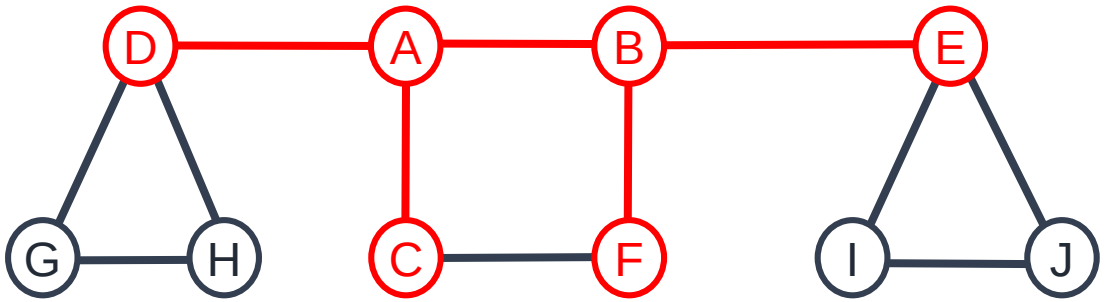
F

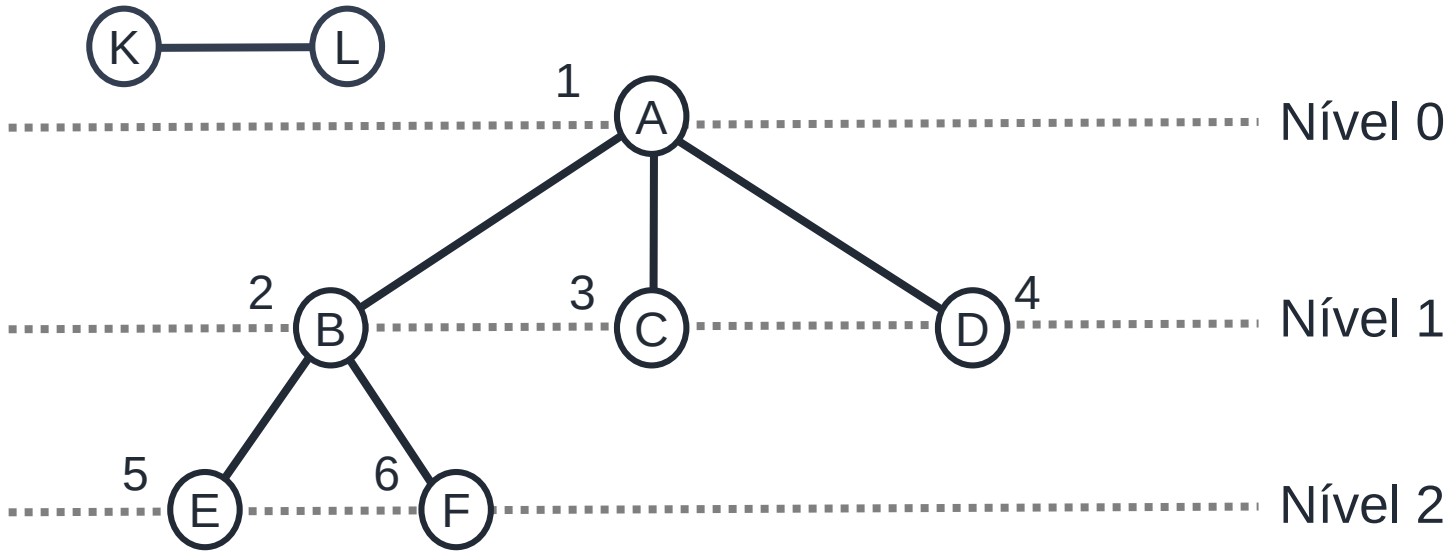
E

D

Fila

Busca em Largura



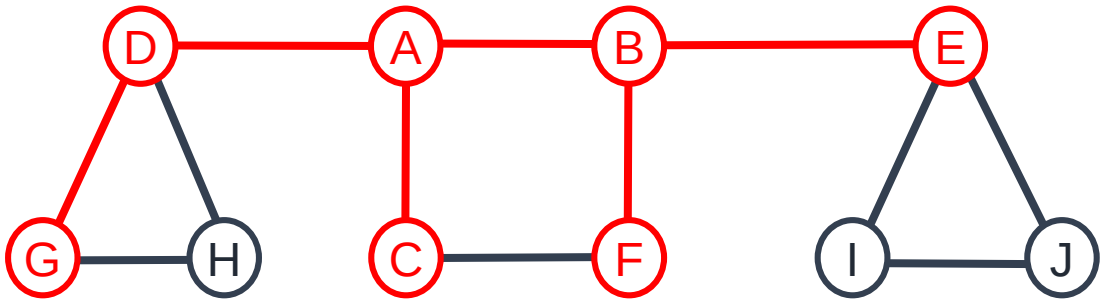


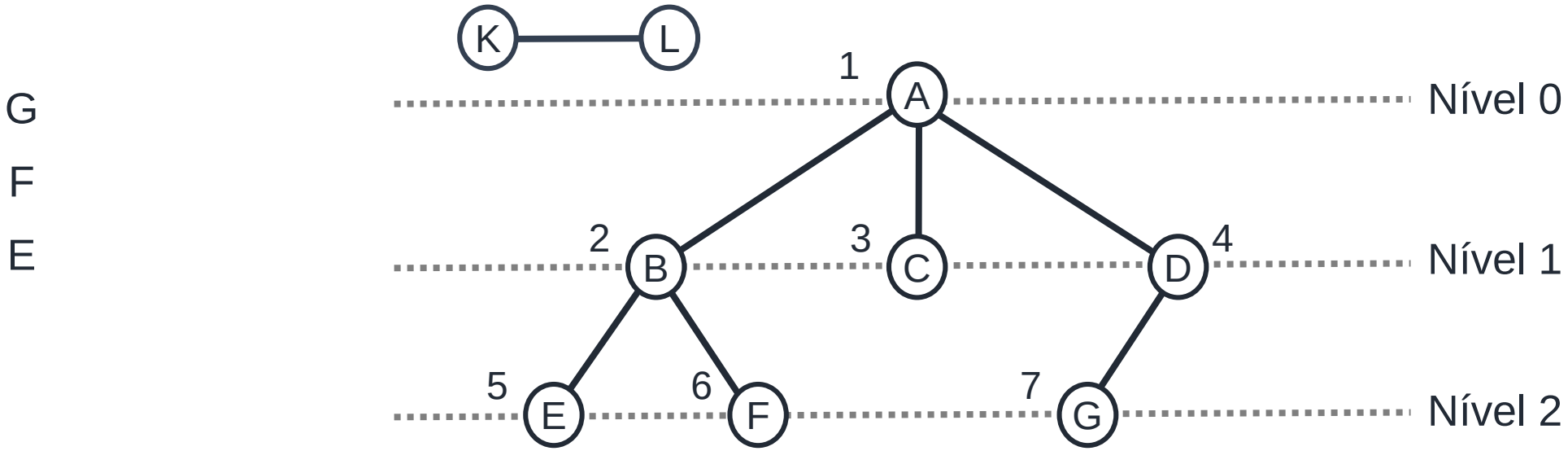
F

E

Fila

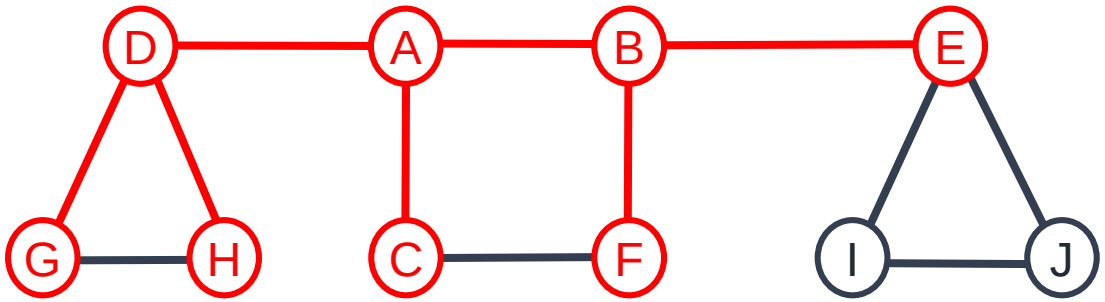
Busca em Largura



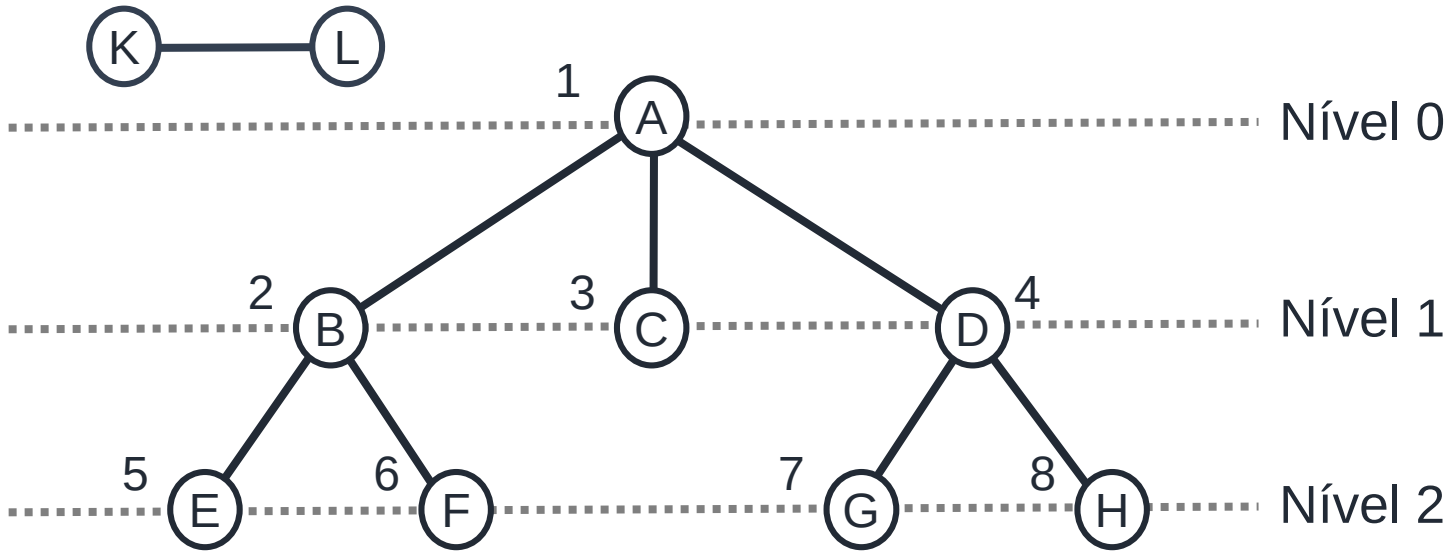


Fila

Busca em Largura

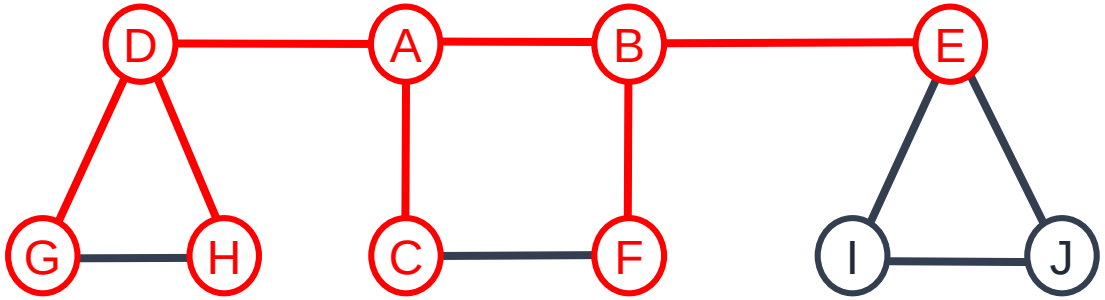


H
G
F
E

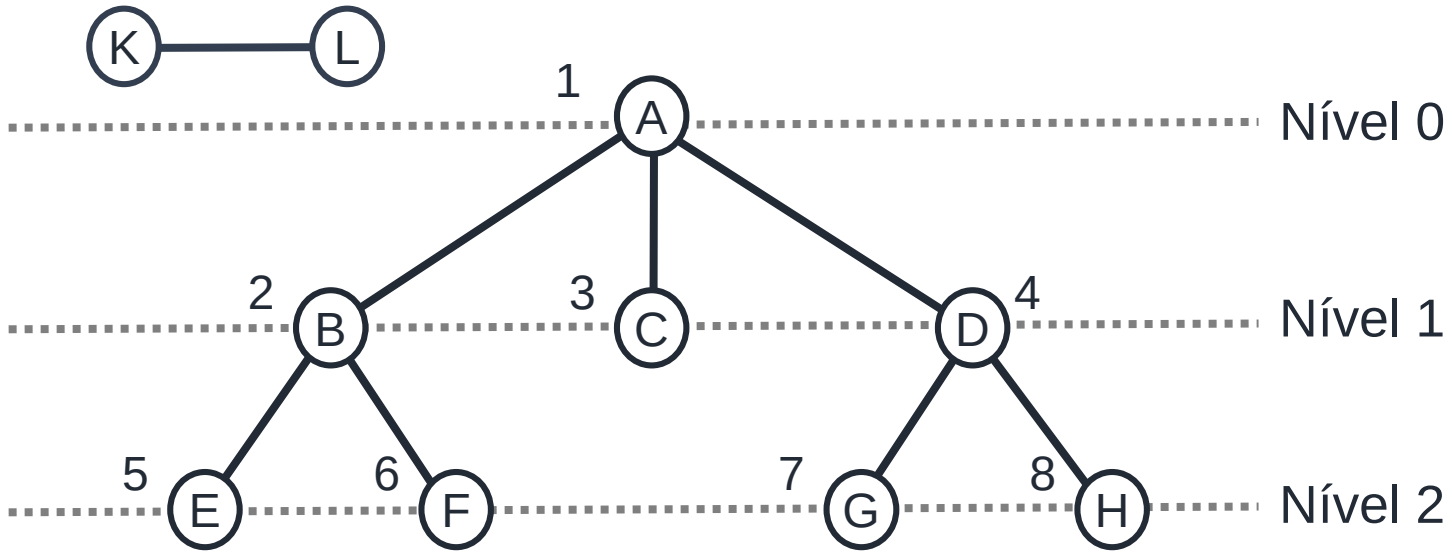


Fila

Busca em Largura

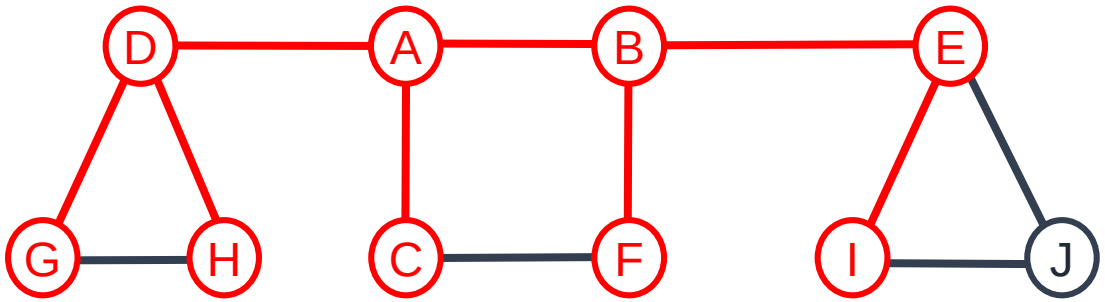


H
G
F

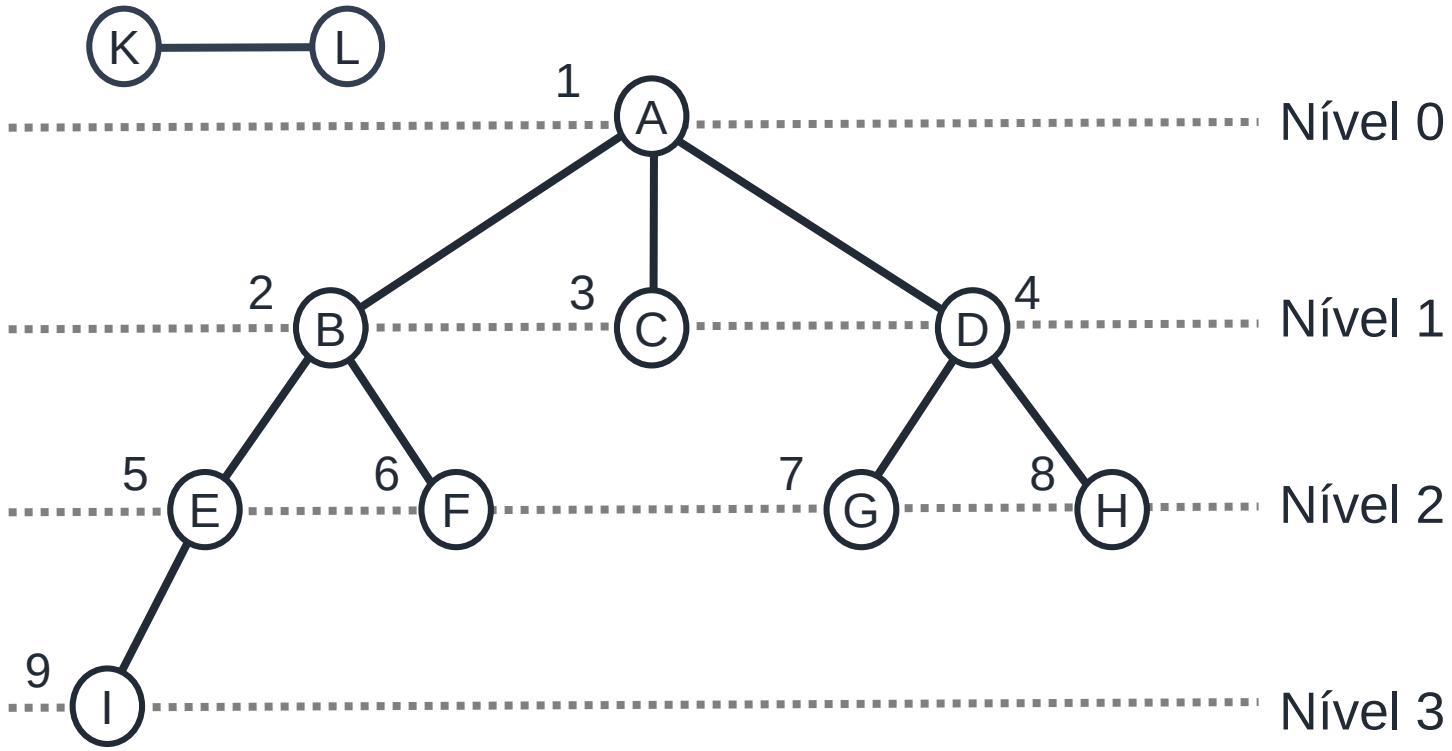


Fila

Busca em Largura

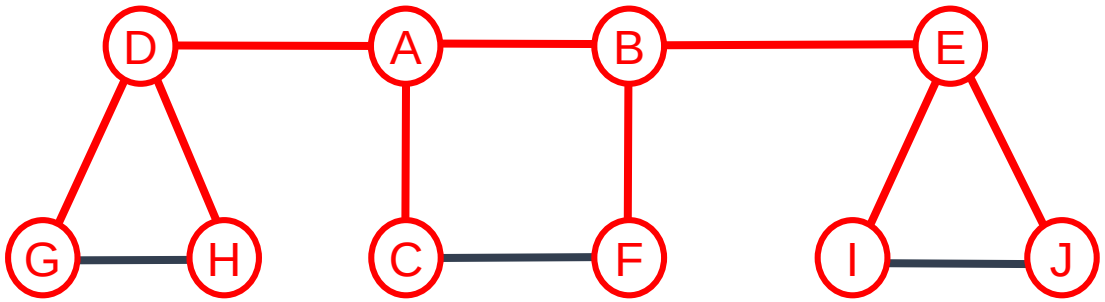


I
H
G
F



Fila

Busca em Largura



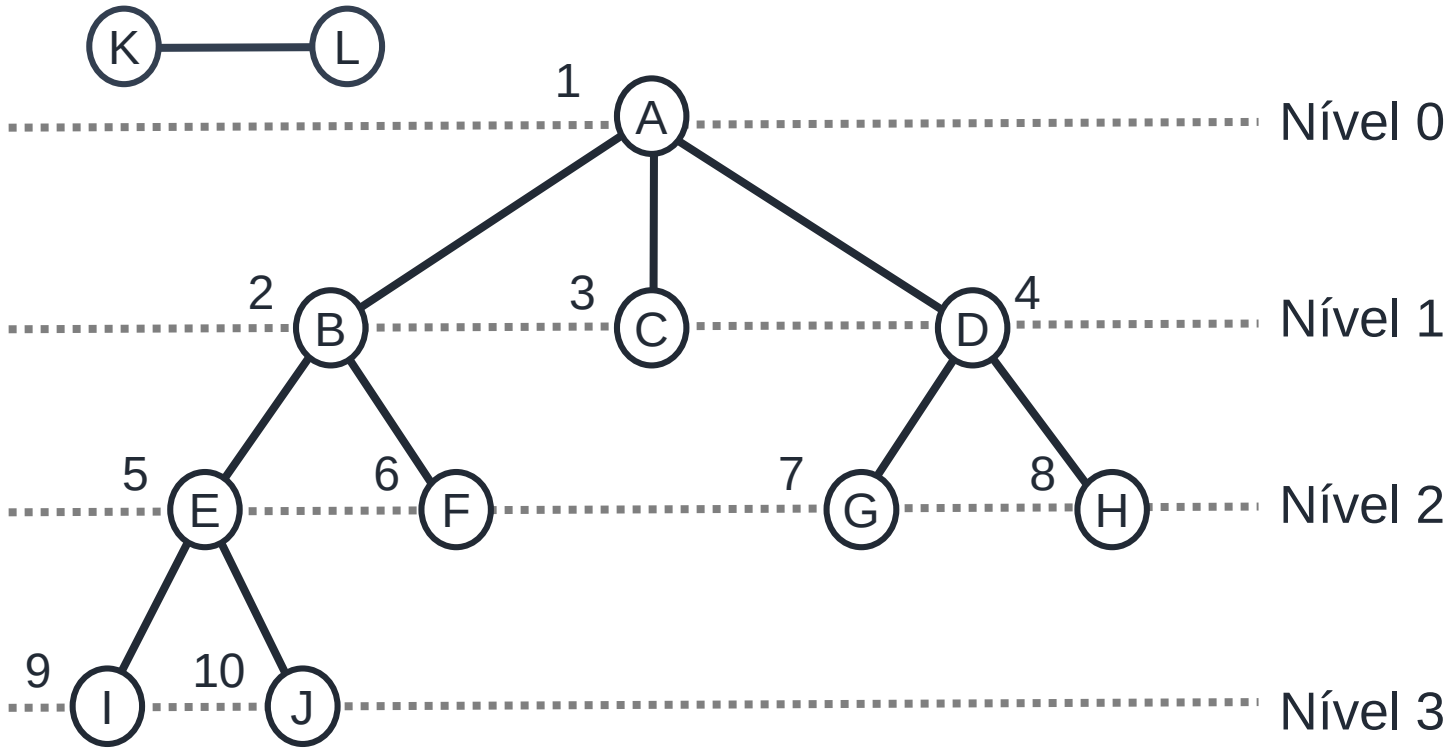
J

I

H

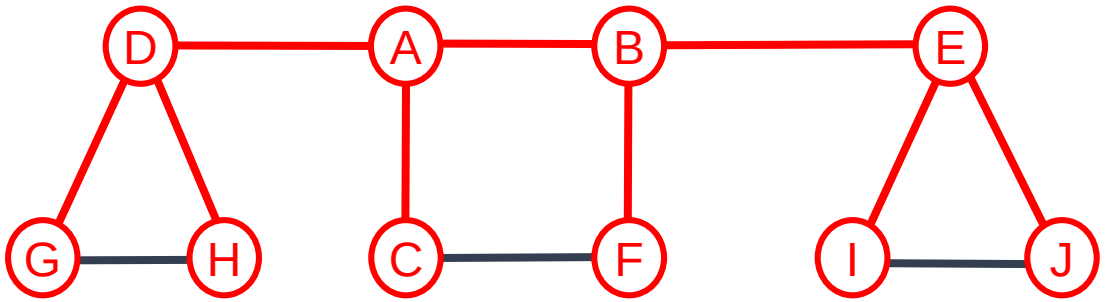
G

F



Fila

Busca em Largura

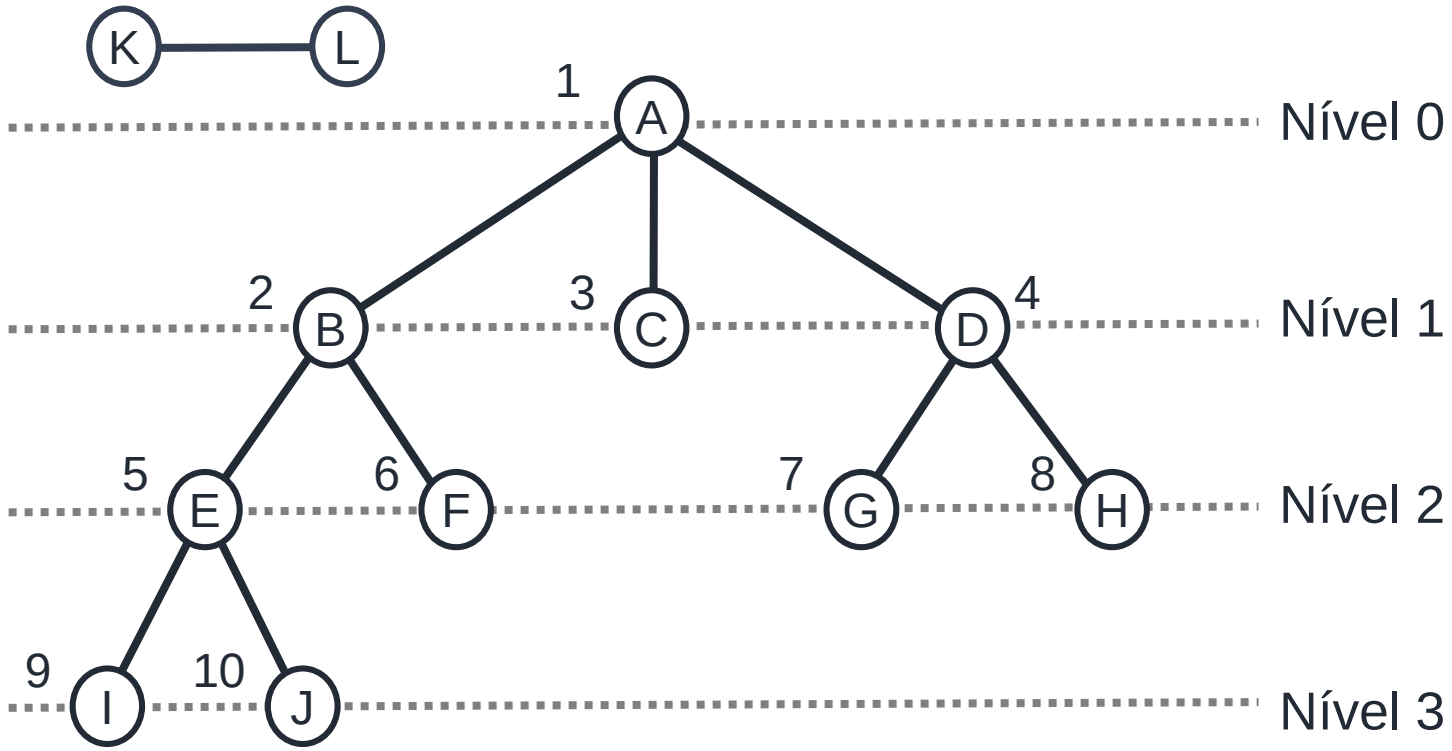


J

I

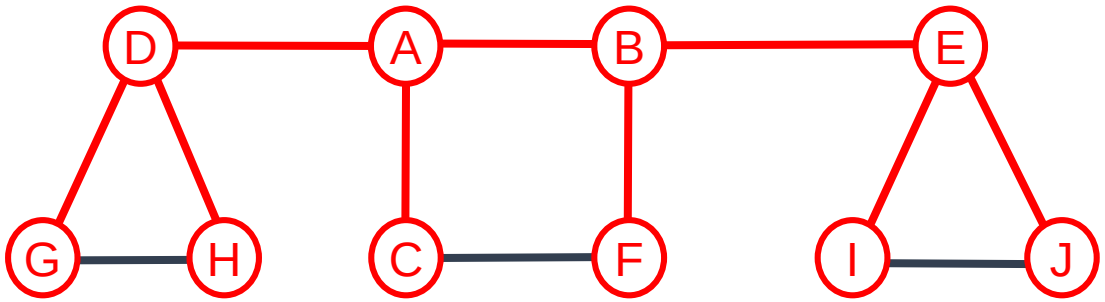
H

G



Fila

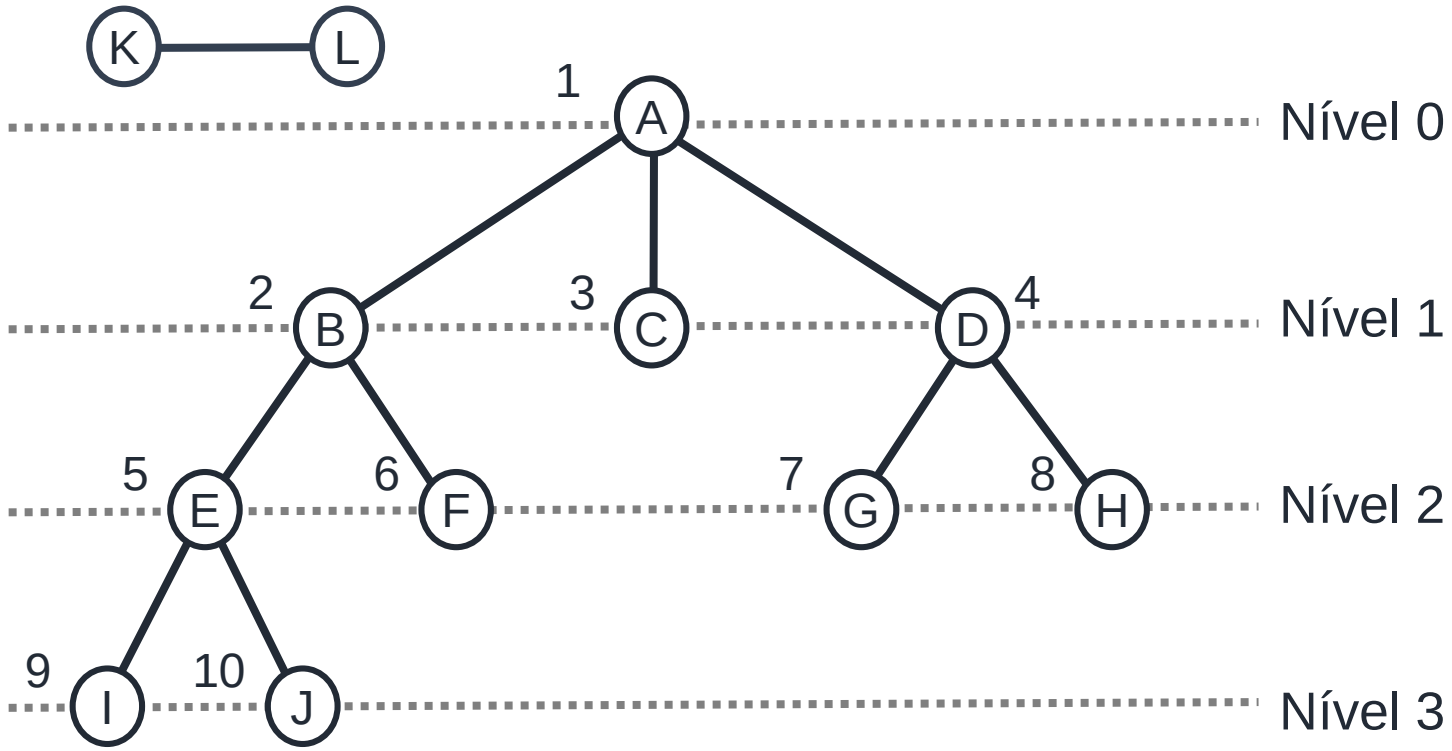
Busca em Largura



J

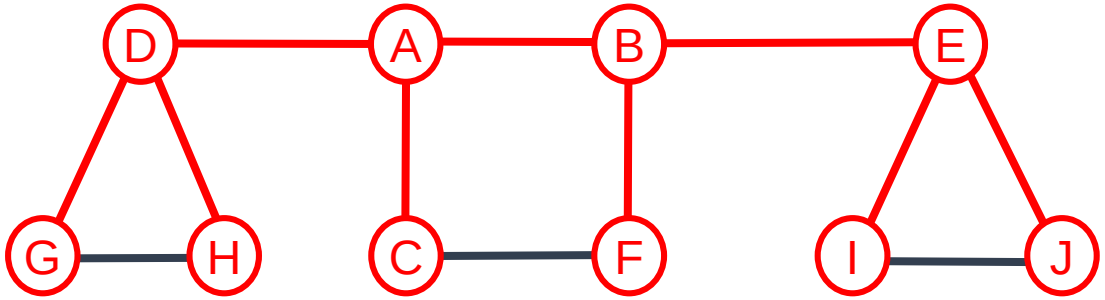
I

H



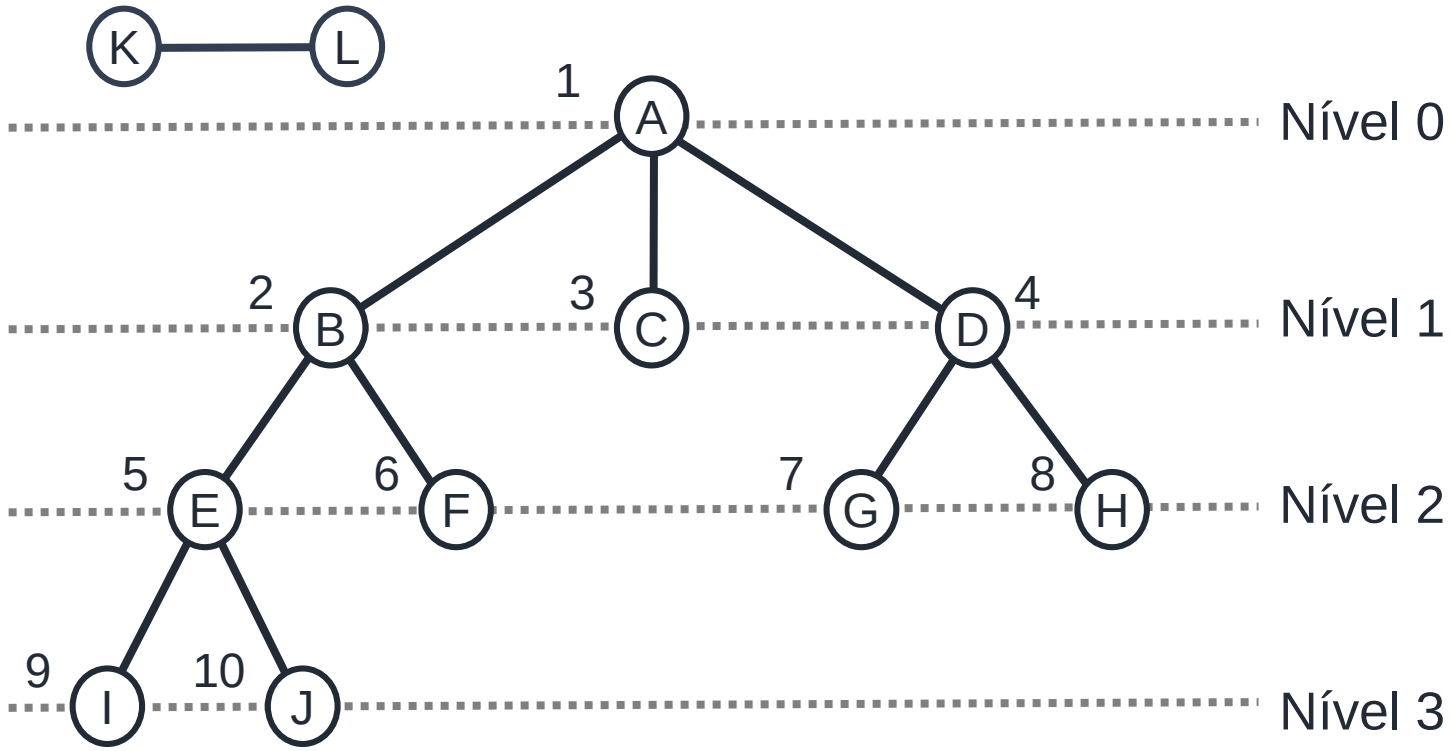
Fila

Busca em Largura



J

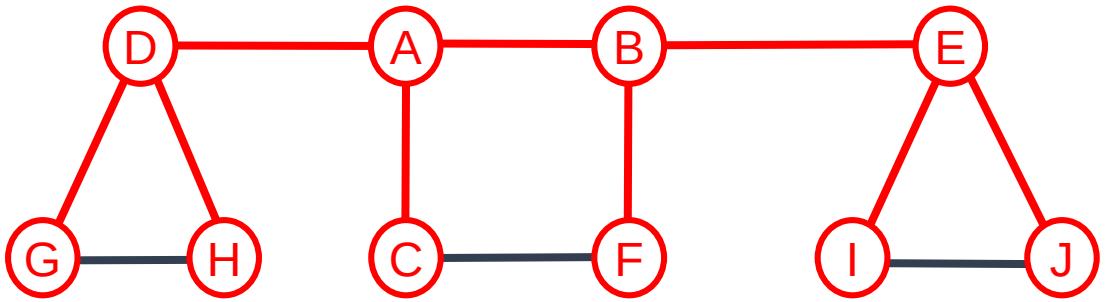
I

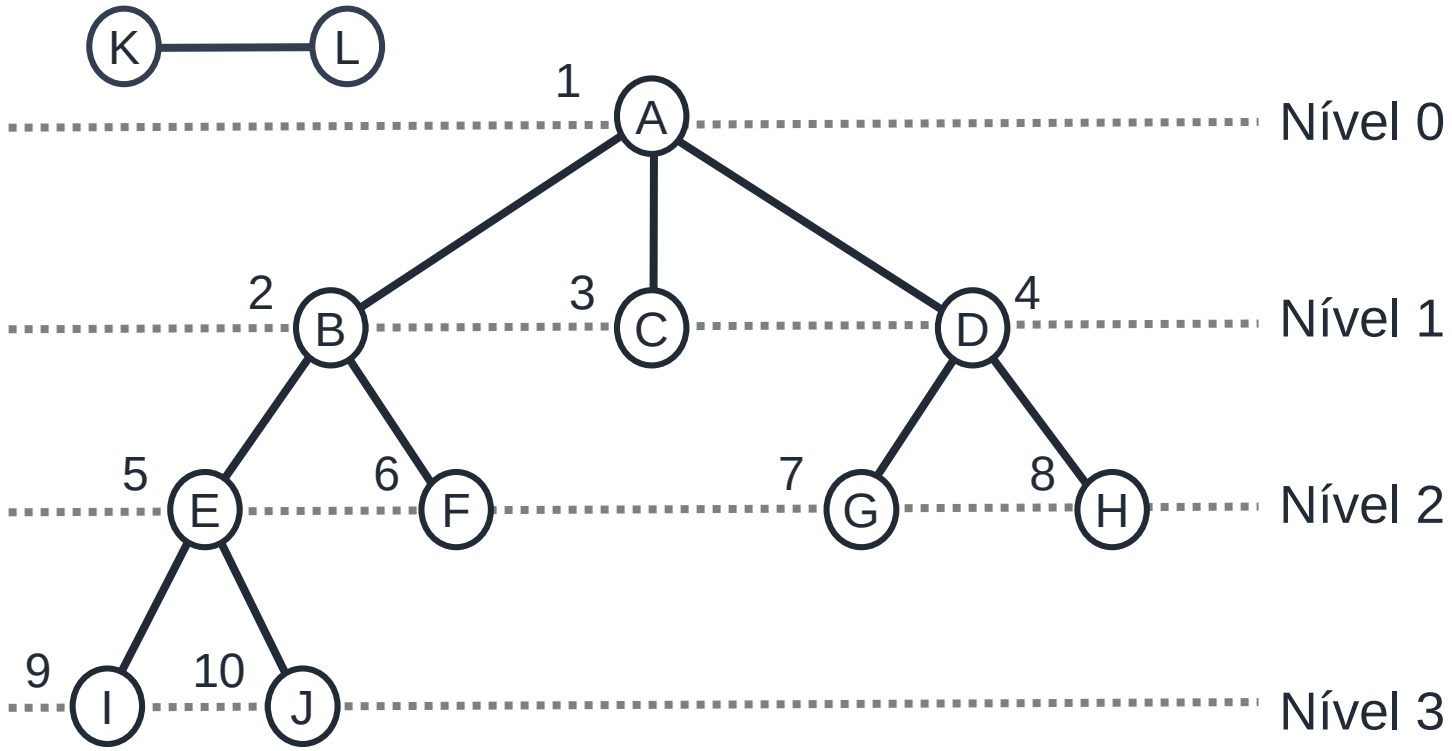


Fila

Busca em Largura

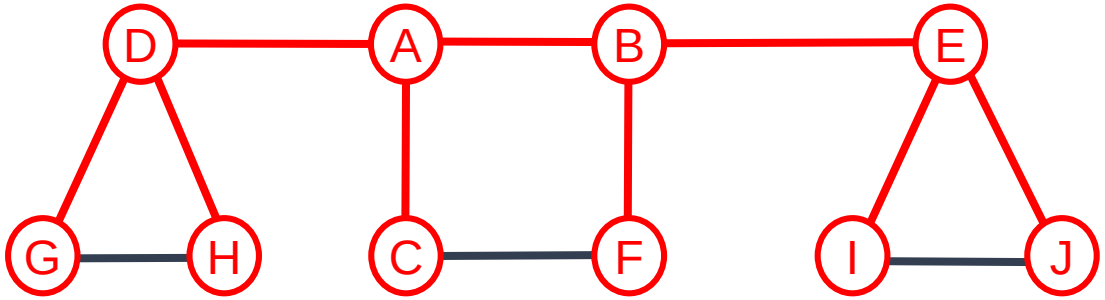
J

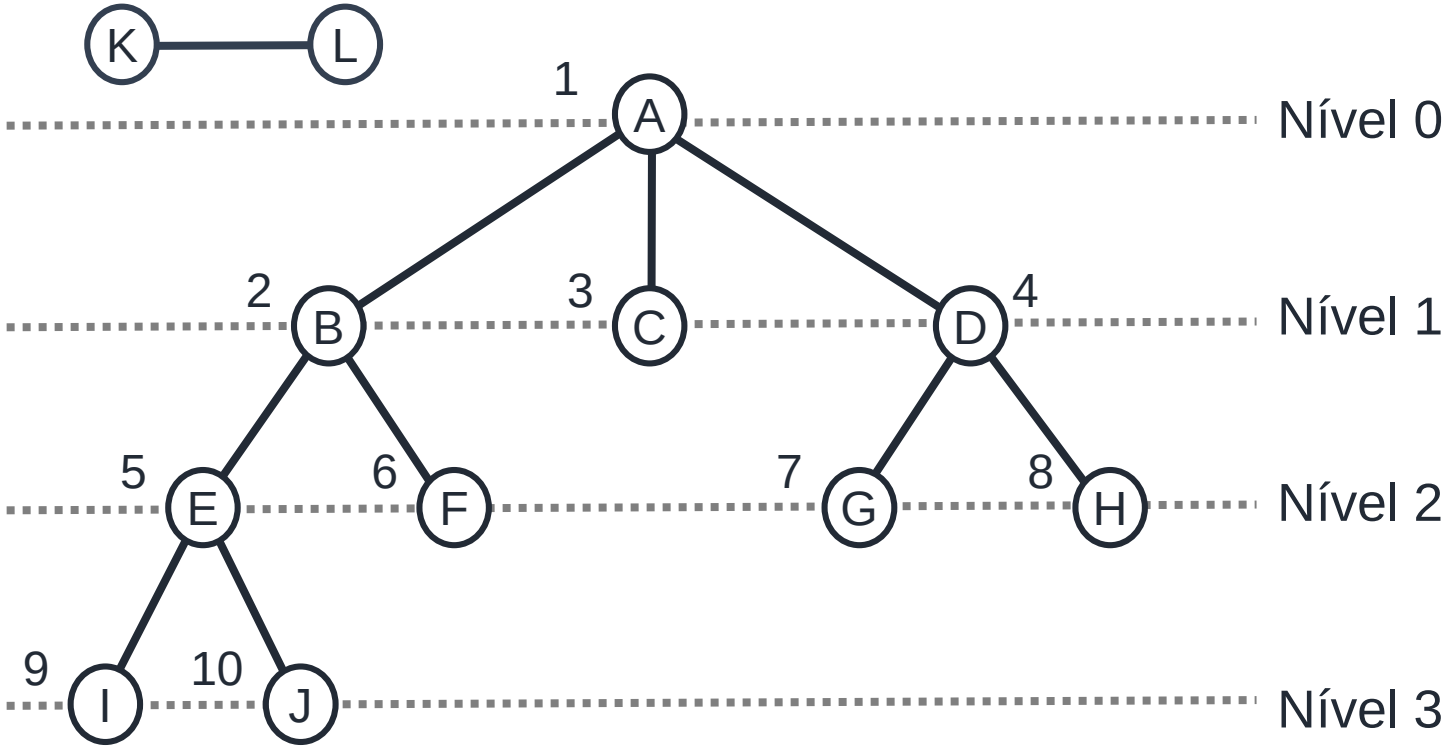




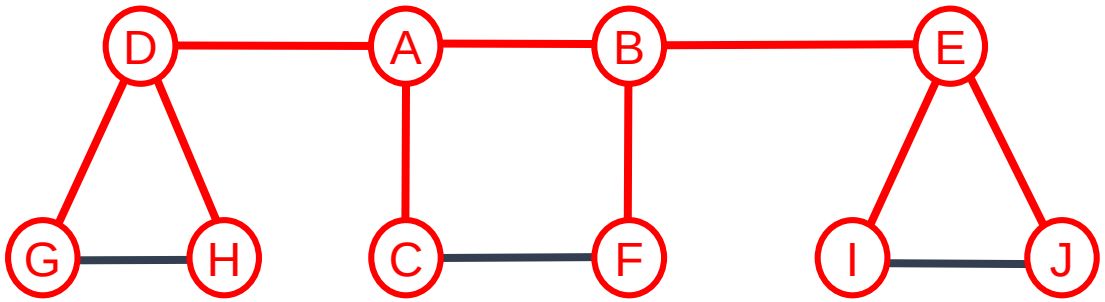
Fila

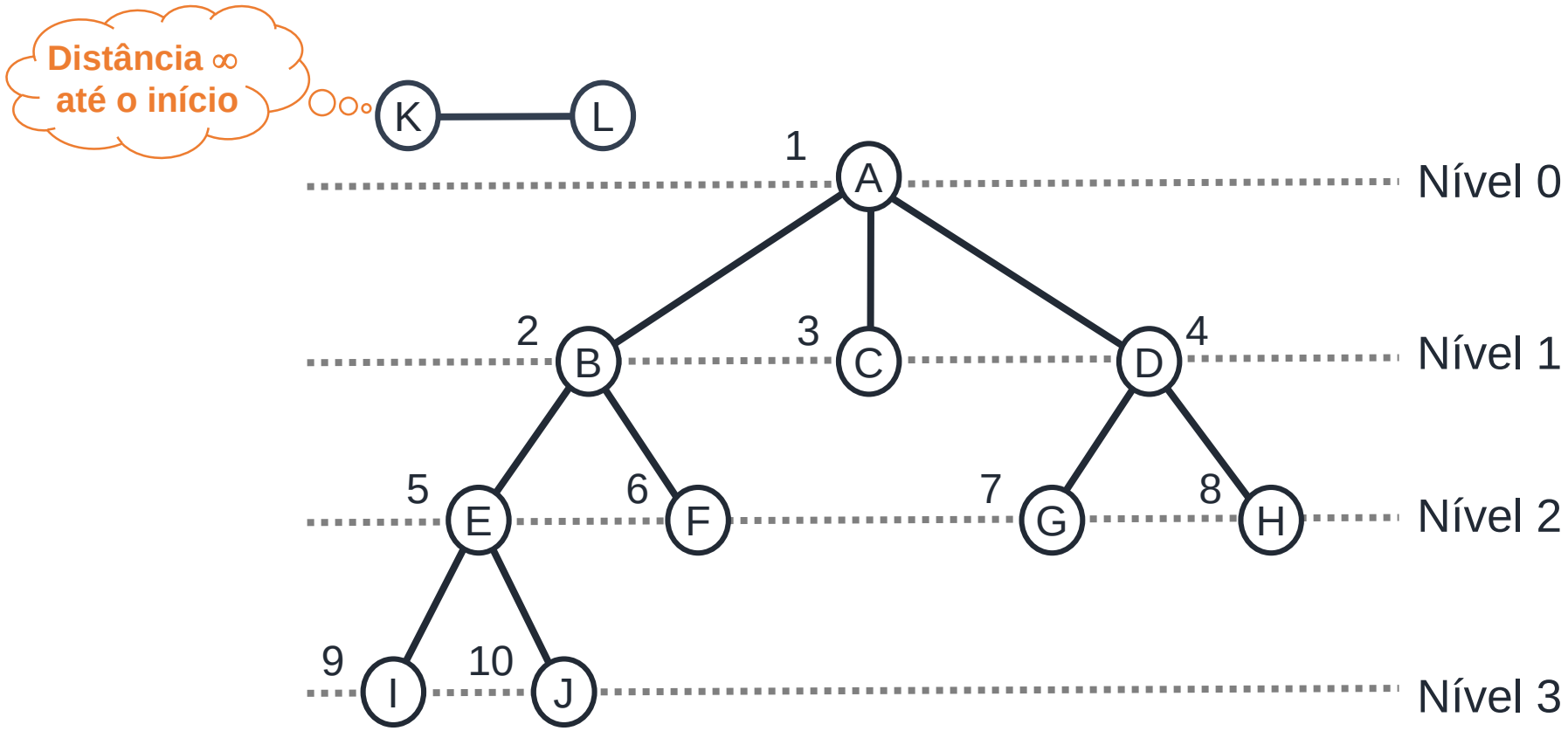
Busca em Largura





Busca em Largura





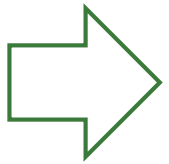
Fila

Busca em Largura

```
procedimento BFS(Grafo g, int origem, int destino,
                 float *dist, ListaDeVertices *caminho){
   CriarFila(&fila)

   achei ← falso
   caminho->nVertices ← 0
   Para cada vértice v de g faça
      dist[v] ← INF


   dist[origem] ← 0
   InserirNaFila(&fila,dados)
```

Busca em Largura

```
procedimento BFS(Grafo g, int origem, int destino,
                 float *dist, ListaDeVertices *caminho){
   CriarFila(&fila)

   achei ← falso
   caminho->nVertices ← 0
   Para cada vértice v de g faça
      dist[v] ← INF

   Pre_visita_BFS(parâmetros) //opcional
   dist[origem] ← 0
   InserirNaFila(&fila,dados)
   Pos_visita_BFS(parâmetros) //opcional
```



Pre_visita e Pos_visita para o vértice de origem

Busca em Largura

```
Enquanto (!FilaVazia(fila) && (!achei)) faça

  RemoverDaFila(&fila, &dados)

  Para cada vértice u de g faça
    Se (dist[u] == INF) && (u adjacente v) && (!achei) então
        InserirNaFila(&fila,dados)

        dist[u] ← dist[v] + 1
        // ou
        // dist[u] ← dist[v] + PesoDaAresta(g,v,u)
}
```

Busca em Largura

```
Enquanto (!FilaVazia(fila) && (!achei)) faça
  Pre_visita_BFS(parâmetros) //opcional
  RemoverDaFila(&fila, &dados)

  Para cada vértice u de g faça
    Se (dist[u] == INF) && (u adjacente v) && (!achei) então
        InserirNaFila(&fila,dados)
        Pos_visita_BFS(parâmetros) //opcional
        dist[u] ← dist[v] + 1
        // ou
        // dist[u] ← dist[v] + PesoDaAresta(g,v,u)
}
```

Exemplo

Armazenando os vértices descobertos

```
void Pre_visita_DFS(Grafo g, int v, ListaDeVertices *caminho){
    caminho->vertices[caminho->nVertices] = v;
    caminho->nVertices++;
}

void Pos_visita_BFS(Grafo g, int v, ListaDeVertices *caminho){
    caminho->vertices[caminho->nVertices] = v;
    caminho->nVertices++;
}
```