Algoritmos sobre Grafos

Eurismar, Geovane, Julliano, Luiz Henrique

Motivação

- Existe um caminho para ir de um objeto a outro seguindo as conexões?
- Qual é a menor distância entre um objeto e outro objeto?
- Quantos outros objetos podem ser alcançados a partir de um determinado objeto?

Conceitos Básicos

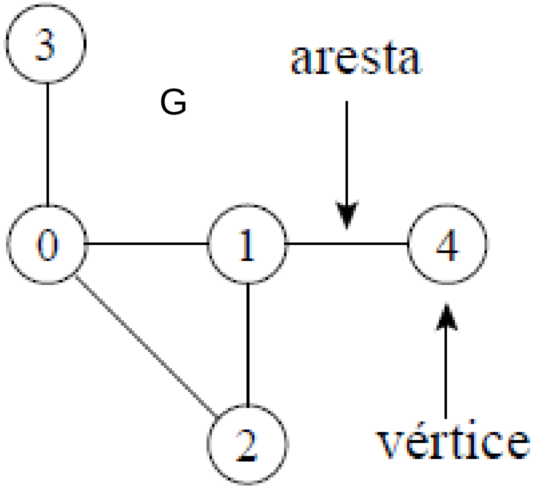
Grafo: conjunto de vértices e arestas.
Vértice: objeto simples que pode ter nome e outros atributos.
Aresta: conexão entre dois vértices.

Grafo Simples

$G = (V, E)$

$V(G) = \{0, 1, 2, 3, 4\}$

$E(G) = \{\{3,0\},\{0,1\},$
$\{1,4\},\{1,2\},\{0,2\}\}$

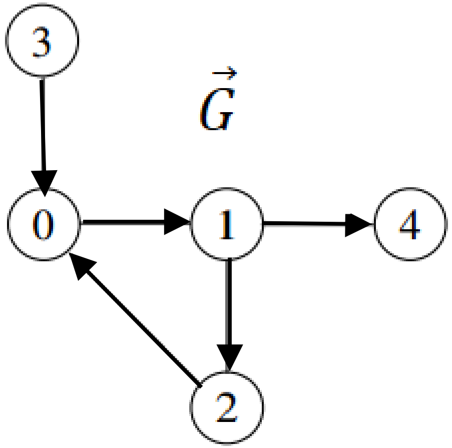


Grafo Orientado

$\overrightarrow{G} = (V, A)$

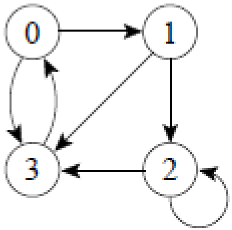
$V(\overrightarrow{G}) = \{0, 1, 2, 3, 4\}$

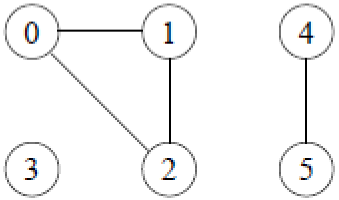
$A(\overrightarrow{G}) = \{(3,0),(0,1),$
$(1,4),(1,2),(2,0)\}$



Implementação

- Matriz de Adjacência
  - Grafos densos



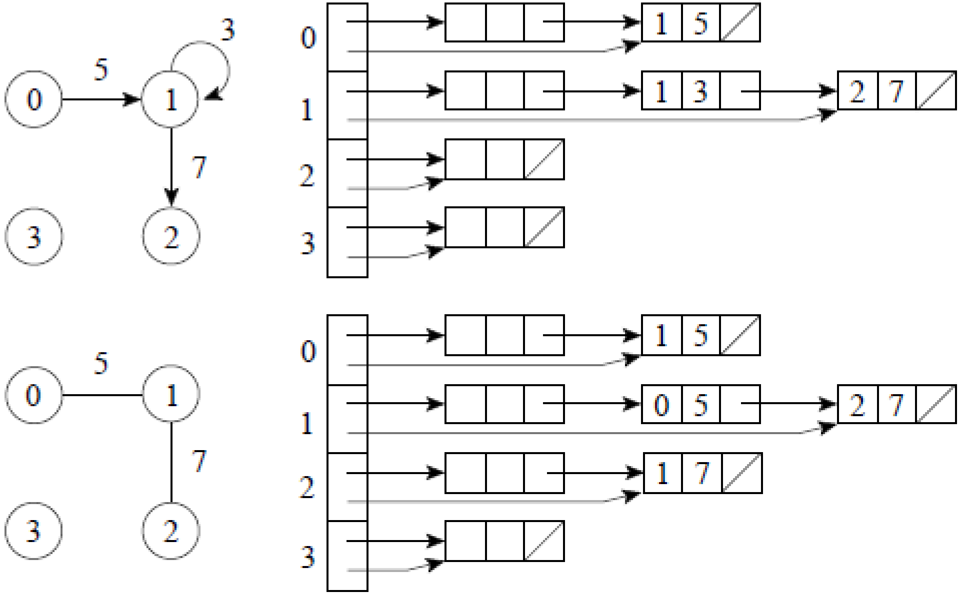


	0	1	2	3	4	5
0		1		1		
1			1	1		
2			1	1		
3	1					
4						
5						

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

Implementação

- Lista de Adjacência
  - Grafos esparsos



Algoritmos

- Algoritmos de Busca
  - Busca em Largura
  - Busca em Profundidade
- Caminho mais curto
  - Algoritmo de Dijkstra

Busca em Largura

- Também conhecido por Breadth-First Search (BFS)
- Busca ou travessia em um grafo ou estruturas do tipo árvore
- Se aplica em grafo direcionado ou não-direcionado
- Busca exaustiva
  - passa por todas as arestas e vértices do grafo

Motivação

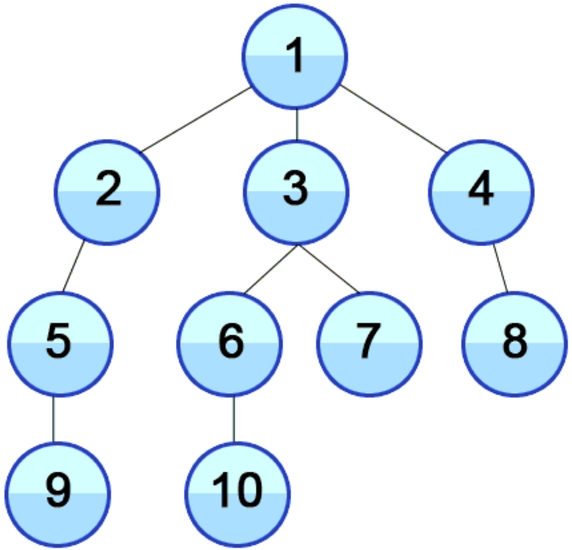
- Achar componentes conectados
- Achar todos os nódulos conectado a apenas um componente
- Pode ser estendido para resolver outros problemas em grafos:
    i. Encontrar um caminho entre um dado par de vértices, com a menor quantidade de arestas caso exista.
    ii. Encontrar um ciclo simples, caso exista

Como funciona

- Expande a fronteira entre vértices descobertos e não descobertos uniformemente através da largura da fronteira.
- O algoritmo descobre todos os vértices a uma distância k do vértice origem antes de descobrir qualquer vértice a uma distância k + 1.
- Utiliza uma fila para garantir a ordem de chegada dos vértices afim de que nenhum vértice ou aresta será visitado mais de uma vez.

Como funciona

- Ordem dos vértices explorados na busca

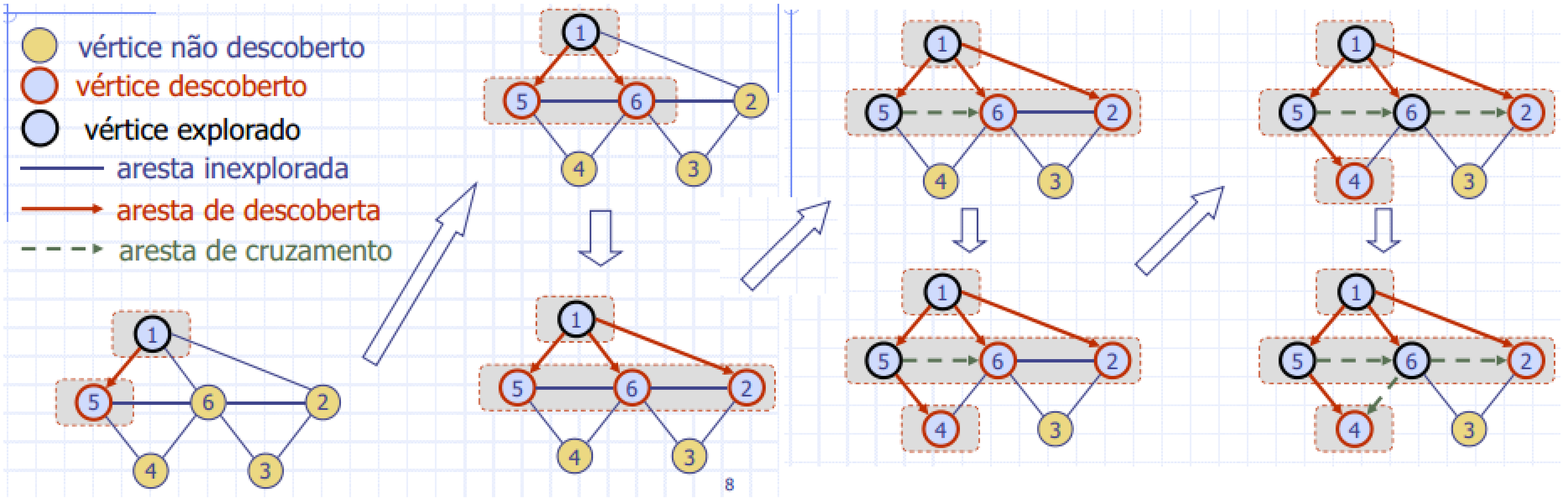


Pseudocódigo BFS

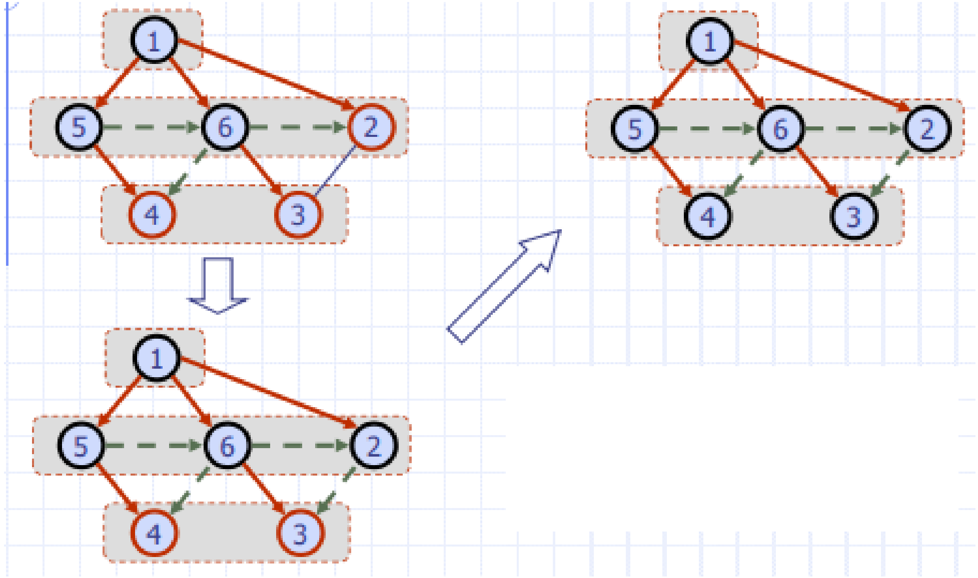
→ A letra F representa uma fila (FIFO) inicialmente vazia, G é o grafo em questão e s, v, w representam vértices do grafo onde listaDeAdjacência representa a lista de adjacência de um vértice.

```
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
```

Exemplo de Execução



Exemplo de Execução (cont.)



Complexidade

- O pior caso, aquele em que todos os vértices e arestas são explorados pelo algoritmo
- Expressão: O (|E| + |V|)
- |E| = o tempo total gasto nas operações sobre todas as arestas do grafo
- O(1) = tempo constante que cada operação requer sobre uma aresta
- |V| = o número de operações sobre todos os vértices que possui uma complexidade constante O(1), uma vez que todo vértice é enfileirado e desenfileirado pelo menos uma vez

Busca em Profundidade

- É um algoritmo também conhecido pela sigla em inglês DFS (*depth-first search*).

- O algoritmo de Busca em Profundidade resulta numa seqüência de vértices, onde cada um é adjacente ao próximo vértice.

Como Funciona

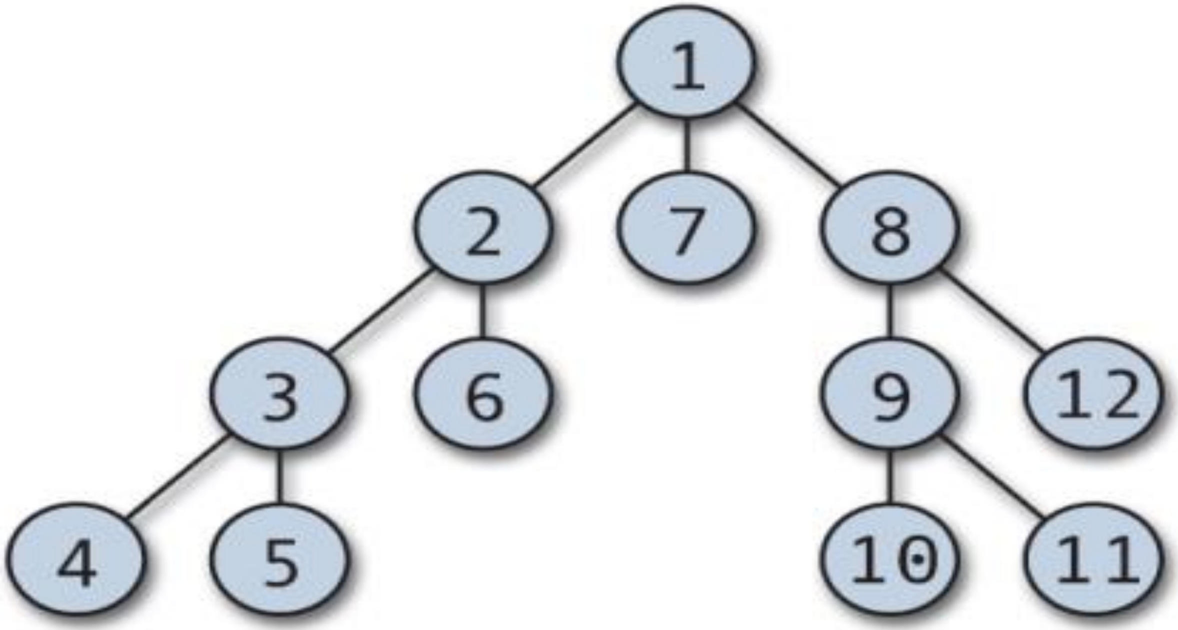
Expande o nó de maior profundidade que esteja na fronteira da árvore de busca. O algoritmo começa num nó raiz e explora tanto quanto possível cada um dos seus ramos antes de retroceder.

Como Funciona

```
procedimento Busca(G: Grafo)
    Para Cada vértice v de G:
        Marque v como não visitado
    Para Cada vértice v de G:
        Se v não foi visitado:
            Busca-Prof(v)

procedimento Busca-Prof(v: vértice)
    Marque v como visitado
    Para Cada vértice w adjacente a v:
        Se w não foi visitado:
            Busca-Prof(w)
```

Como Funciona



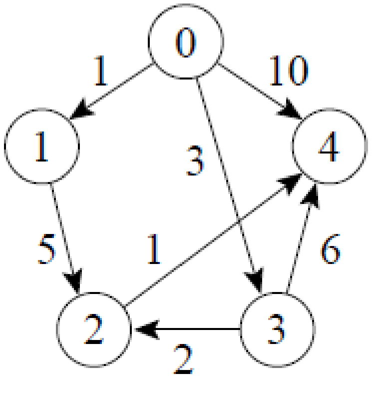
Problema

- Existem casos em que a árvore de busca possui profundidade infinita e, então, a busca em profundidade pode continuar a testar os nós mais profundos de um ramo que não contenham alguma solução e ser incapaz de retornar e testar outros ramos. Assim, ela não é uma busca completa.

- Por isso, muitas implementações deste método incorporam o conceito de limite de profundidade, sendo chamado de busca em profundidade limitada.

Caminho mais curto

- Dado um grafo ponderado G = (V, A), desejamos obter o caminho mais curto a partir de um dado vértice origem até cada vértice de V.



| V(G) | = 5

| A(G) | = 7

A(G) = {(0,1),(0,3),(0,4),(1,2),(2,4),(3,2),(3,4)}

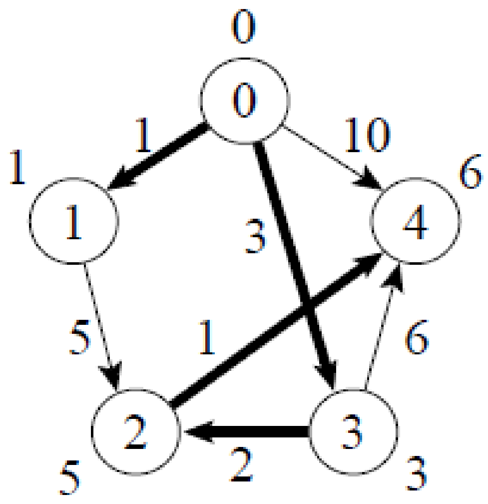
Caminho mais curto

- Algoritmo de Dijkstra
  - p[v] = peso de um caminho mais curto do vértice origem até v.
  - S = conjunto de vértices cujos caminhos mais curtos até um vértice origem já são conhecidos.

```
dijkstra (Grafo grafo, int raiz)
  for (int v = 0; v < grafo.numVertices (); v++)
    p[v] = Infinito;
    antecessor[v] = −1;
  p[raiz] = 0;
  Constroi heap sobre vértices do grafo;
  S = ∅;
  while (!heap.vazio ())
    u = heap.retiraMin ();
    S = S + u;
    for (v ∈ grafo.listaAdjacentes (u))
      if (p[v] > p[u] + peso da aresta (u,v))
        p[v] = p[u] + peso da aresta (u,v);
        antecessor[v] = u;
```

Caminho mais curto

- Resultado



Referências

Ziviani, Nivio. Projeto de algoritmos com implementações em Java e C++. Cengage Learning, 2006.