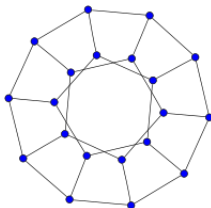


Estrutura de Dados

Grafos

prof. Frederico Santos de Oliveira

Universidade Federal de Mato Grosso
Instituto de Engenharia



Agenda

1 Introdução

2 Representação

- Matriz de Adjacência
- Lista de Adjacência

3 Percurso

- Busca em Largura
- Busca em Profundidade

4 Referências bibliográficas

5 Material Complementar

Representação

As principais formas para representar grafos são:

- **Matriz de Adjacência**
- **Lista de Adjacência**

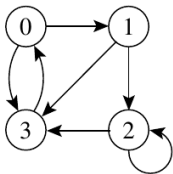
Representação

Matriz de Adjacência

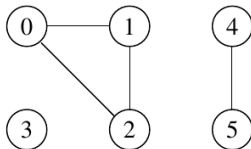
- A **Matriz de Adjacência** de um grafo $G = (V, A)$ contendo n vértices é uma matriz $n \times n$ de bits, onde $A[i, j]$ é 1 (ou verdadeiro) se e somente se existe um arco do vértice i para o vértice j .
- Para grafos ponderados $A[i, j]$ contém o rótulo ou peso associado com a aresta e, neste caso, a matriz não é de bits.
- Se não existir uma aresta de i para j então é necessário utilizar um valor que não possa ser usado como rótulo ou peso.

Representação

Matriz de Adjacência



	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						

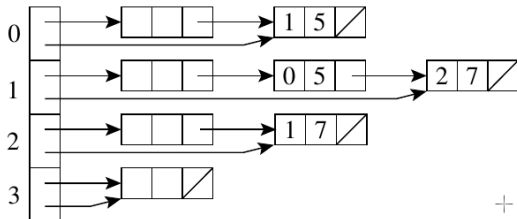
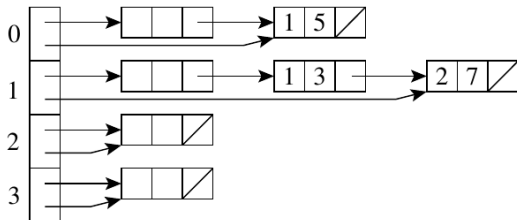
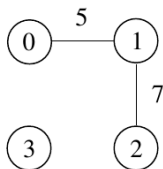
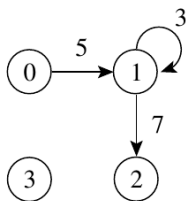
Representação

Lista de Adjacência

- A **Lista de Adjacência** consiste de um vetor, denominado *Adj*, contendo $|V|$ listas, uma para cada vértice de V .
- Para cada $u \in V$, $\text{Adj}[u]$ contém todos os vértices de G adjacentes a u .
- Os vértices são armazenados de forma arbitrária na lista.
- Também pode ser utilizada para representar grafos dirigidos.

Representação

Lista de Adjacência



+

- Fazer buscas em um grafo significa percorrer suas arestas sistematicamente, de modo a visitar seus vértices.
- As principais formas para percorrer grafos são:
 - ▶ **Busca em Largura** - em inglês *Breadth-First Search* ou BFS
 - ▶ **Busca em Profundidade** - em inglês *Depth-First Search* ou DFS

- A **Busca em Largura** é um dos algoritmos mais simples para exploração de um grafo.
 - ▶ Dados um grafo $G = (V, E)$ e um vértice s , chamado de fonte, a busca em largura sistematicamente explora as arestas de G de maneira a visitar todos os vértices alcançáveis a partir de s .
- 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 de origem s antes de descobrir qualquer vértice a uma distância $k + 1$.
- O grafo pode ser direcionado ou não-direcionado.
- Esse algoritmo é base para o algoritmo de Dijkstra, o qual encontra o menor caminho de um vértice aos demais.

Percurso

Busca em Largura

Algoritmo 1: BuscaEmLargura

Entrada: Grafo $G = (V, A)$, vértice inicial s .

Saída: Percurso armazenado no campo “predecessor” presente em cada vértice $v \in V$.

```
1 início
2   para cada vértice  $u \in V$  faça
3      $u.cor \leftarrow$  Branco
4      $u.d \leftarrow \infty$ 
5      $u.\pi \leftarrow$  NULL
6    $s.cor \leftarrow$  Cinza
7    $s.d \leftarrow 0$ 
8    $s.\pi \leftarrow$  NULL
9   CriaFilaVazia(Q)
10  Enfileira(Q, s)
11  enquanto ( $Q \neq \emptyset$ ) faça
12     $u \leftarrow$  Desenfileira(Q)
13    para cada vértice  $v \in u.ListaAdj$  faça
14      se  $v.cor =$  Branco então
15         $v.cor \leftarrow$  Cinza
16         $v.d \leftarrow u.d + 1$ 
17         $v.\pi \leftarrow u$ 
18      Enfileira(Q, v)
19   $u.cor \leftarrow$  Preto
```

Adaptado de Cormen et al. (2012).

Percurso

Busca em Profundidade

- Na **Busca em Profundidade**, a estratégia é buscar o vértice mais profundo no grafo sempre que possível:
 - ▶ As arestas são exploradas a partir do vértice v mais recentemente descoberto que ainda possui arestas não exploradas saindo dele.
- Quando todas as arestas adjacentes a v tiverem sido exploradas a busca anda para trás para explorar vértices que saem do vértice do qual v foi descoberto (*backtracking*).
- O algoritmo é a base para muitos outros algoritmos importantes, tais como:
 - ▶ Verificação de grafos acíclicos,
 - ▶ Ordenação topológica e
 - ▶ Componentes fortemente conectados.

Percurso

Busca em Profundidade

Algoritmo 2: BuscaEmProfundidade

Entrada: Grafo $G = (V, A)$.

Saída: Percurso armazenado no campo “predecessor” presente em cada vértice $v \in V$.

```
1 início
2   para cada vértice  $u \in V$  faça
3      $u.cor \leftarrow$  Branco
4      $u.\pi \leftarrow$  NULL
5   tempo  $\leftarrow$  0
6   para cada vértice  $u \in V$  faça
7     se  $u.cor =$  Branco então
8       DFS-Visita( $u$ )
```

Adaptado de Cormen et al. (2012).

Algoritmo 3: DFS-Visita


Entrada: Grafo $G = (V, A)$, vértice inicial s .

```
1 início
2   tempo  $\leftarrow$  tempo + 1
3    $u.d \leftarrow$  tempo
4    $u.cor \leftarrow$  Cinza
5   para cada vértice  $v \in u.ListaAdj$ 
6     faça
7       se  $v.cor =$  Branco então
8          $v.\pi \leftarrow u$ 
9         DFS-Visita( $G, v$ )
9    $u.cor \leftarrow$  Preto
10  tempo  $\leftarrow$  tempo + 1
11   $u.f \leftarrow$  tempo
```

Dúvidas?



Referências bibliográficas

 CORMEN, T. H. et al. *Algoritmos: Teoria e Prática*. 3. ed. São Paulo: Campus, 2012. ISBN 978-0-262-03384-8.

Material Complementar

- Material IME

- ▶ https://www.ime.usp.br/pf/algoritmos_para_grafos/aulas/bfs.html
- ▶ https://www.ime.usp.br/pf/algoritmos_para_grafos/aulas/dfs.html

- Animação Breadth-First Search (Busca em Largura)

- ▶ <https://www.cs.usfca.edu/galles/visualization/BFS.html>

- Animação Depth-First Search (Busca em Profundidade)

- ▶ <https://www.cs.usfca.edu/galles/visualization/DFS.html>

Material Complementar

- Youtube

- ▶ Estrutura de Dados Descomplicada - Busca em Largura
- ▶ Estrutura de Dados Descomplicada - Busca em Profundidade
- ▶ Univesp - Grafos
- ▶ Univesp - Busca em Largura
- ▶ Univesp - Busca em Profundidade