1

Algoritmos em Grafos: Introdução

R. Rossetti, A. P. Rocha, L. Ferreira, J. P. Fernandes, F. Ramos, G. Leão
CAL, MIEIC, FEUP

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

2

Índice

- Revisão de conceitos e definições
- Exemplos de aplicação
- Representação

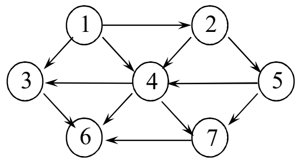
Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

Revisão de conceitos e definições

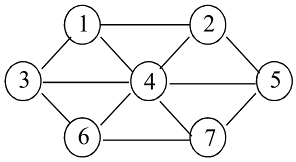
Conceito de grafo

- Grafo G = (V, E)
  - V — conjunto de vértices (ou nós)
  - E — conjunto de arestas (ou arcos)
  - cada aresta é um par de vértices (v, w), c/ v, w $\in$ V
  - se o par for ordenado, o grafo é dirigido, ou digrafo
  - um vértice w é adjacente a um vértice v se e só se (v, w) $\in$ E
  - num grafo não dirigido com aresta (v, w) e, logo, (w, v), w é adjacente a v e v adjacente a w

Grafos dirigidos e não dirigidos



G1= (Cruzamentos, Ruas)



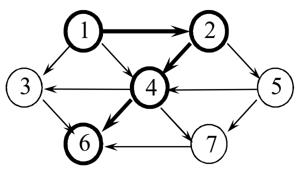
G2 = (Cidades, Estradas)

V1 = {1,2,3,4,5,6,7}

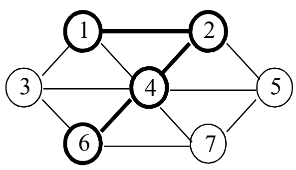
E1={(1,2), (1,3), (1,4), (2,4), (2,5),(3,6), (4,3), (4,7), (5,4), (5,7), (7,6)}

Caminhos

- Caminho - sequência de vértices $v_1, \ldots, v_n$ tais que $(v_i, v_{i+1}) \in E$, $1 \leq i < n$
- comprimento do caminho é o número de arestas, n-1
- se n = 1, caminho reduz-se a 1 vértice, comprimento 0
- caminho simples: todos os vértices distintos, excepto possivelmente o primeiro e o último



(1, 2, 4, 6)

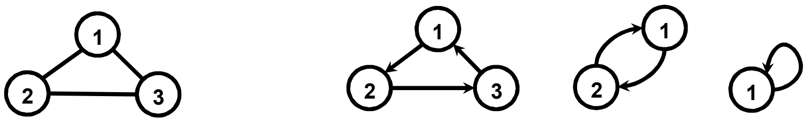


(1, 2, 4, 6)

7

Ciclos

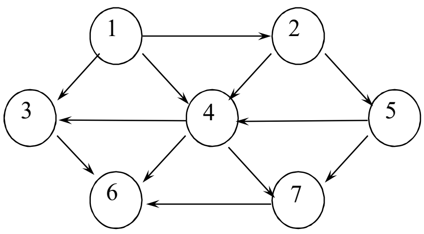
- Ciclo (ou circuito): caminho de comprimento ≥ 1 com $v_1 = v_n$
- Num grafo não dirigido, requer-se que as arestas sejam diferentes
- Anel: caminho v, v $\Rightarrow$ (v, v) $\in$ E , comprimento 1; raro



8

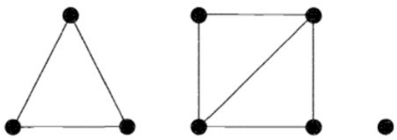
Grafo acíclico dirigido (*DAG - Directed acyclic graph*)

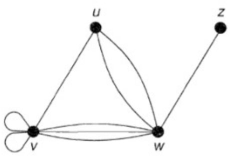
- Grafo dirigido sem ciclos. Para qualquer vértice v, não há nenhuma ligação dirigida começando e acabando em v.



Grafo simples

- Grafo sem arestas paralelas (várias adjacências, para o mesmo par de vértices), nem anéis:



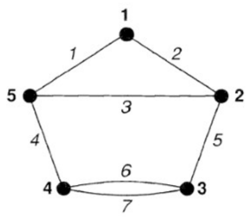


grafos simples

grafo complexo

Grafo pesado

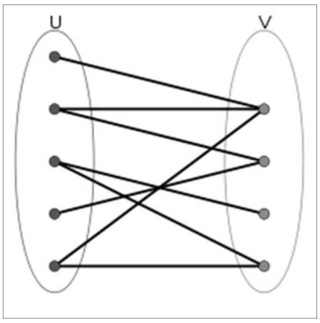
- As arestas são etiquetadas com um peso
  - Dependendo do tipo de grafo e problema, o peso pode representar uma distância, custo, etc.



11

Grafo bipartido

- Conjunto de vértices é partido em dois subconjuntos disjuntos V_1 e V_2
- Arestas ligam vértices de diferentes partições

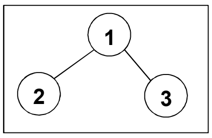


Algoritmos em Grafos: Introdução • CAL – MIEIC/FEUP

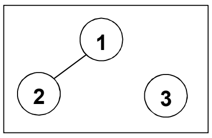
12

Conectividade (1/2)

- Grafo não dirigido é conexo sse houver um caminho a ligar qualquer par de vértices



Conexo

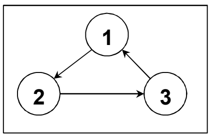


Não conexo

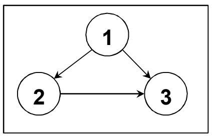
Algoritmos em Grafos: Introdução • CAL – MIEIC/FEUP

Conectividade (2/2)

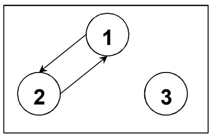
- Digrafo com a mesma propriedade: fortemente conexo, se p/ todo v, w $\in$ ***V*** existir em ***G*** um caminho de v para w, assim como de w para v
- Digrafo fracamente conexo: se o grafo não dirigido subjacente é conexo



Fortemente conexo



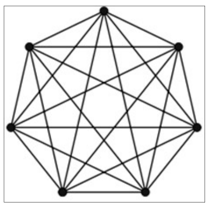
Fracamente conexo



Não conexo

Densidade

- Grafo denso — $|E| \sim \Theta(V^2)$
  - Grafo completo — existe uma aresta entre qualquer par de vértices



Grafo completo com 7 vértices (K_7)

- Grafo esparso — $|E| \sim \Theta(V)$

Exemplos de aplicação

Exemplos de aplicação: caminho mais curto

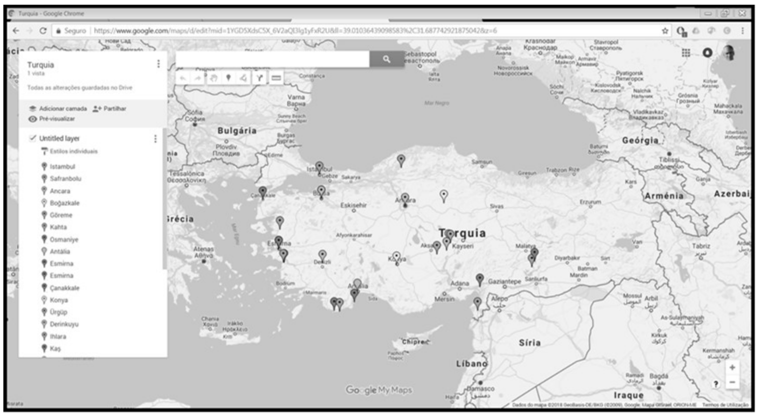
Qual o caminho mais curto / mais rápido / mais barato entre 2 pontos?

Abstraído como problema em grafos, resolúvel em tempo linear

Exemplos de aplicação: problema do caixeiro viajante

Qual o melhor circuito para passar nos pontos de interesse?

Abstraído como problema em grafos, em geral não resolúvel em tempo linear.



Exemplos de aplicação

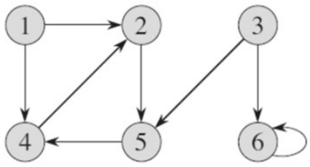
- Redes de transportes
  - Caminho mais curto (navegação GPS)
  - Controlo e gestão de tráfego
- Redes de abastecimento de água e saneamento
  - Gestão de carga
- Redes elétricas
  - Gestão da rede
- Redes de computadores
  - Encaminhamento (*routing*)

Exemplos de aplicação

- Redes de atividades ou tarefas
  - Problemas de escalonamento
  - Problemas de gestão de projectos
- Redes Bayesianas e probabilísticas (Processo de Manchester)
- Compiladores, sistemas de ficheiros
- Jogos
- Criptografia

Representação

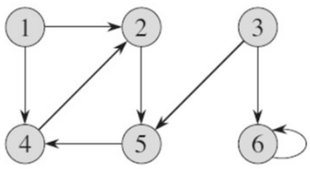
Matriz de adjacências

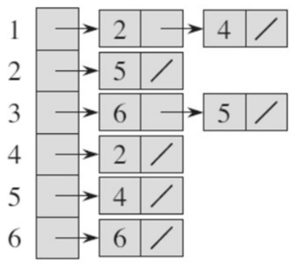


	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

- Matriz A de adjacências
- $a_{ij} = 1$, se $(i, j) \in E$ (0, no caso contrário)
- elementos da matriz podem ser os pesos
- apropriada para grafos densos
  - 3000 cruzamentos x 12 000 troços de ruas (4 por cruzamento) = 9 000 000 de elementos na matriz!

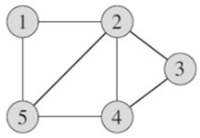
Lista de adjacências



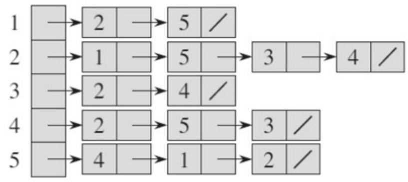


- para cada vértice, mantém-se a lista dos vértices adjacentes
- vetor de cabeças de lista, indexado pelos vértices
- espaço é O(|E| + |V|)
- pesquisa de adjacentes em tempo proporcional ao número destes
- estrutura típica para grafos esparsos

Representação de grafos não dirigidos



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



- Implicação para as matrizes de adjacência
  - Matriz simétrica
- Implicação para as listas de adjacência
  - Lista com dobro do espaço

Codificação

- Normalmente precisamos de guardar informação adicional em cada vértice e em cada aresta (nome, peso, etc.), pelo que se opta por uma representação mais complexa, como por exemplo (Java):

```
class Graph {
   ArrayList<Vertex> vertexSet;
}

class Vertex {
   String name;
   LinkedList<Edge> adj; //arestas a sair do vértice
}

class Edge {
   Vertex dest;
   double weight;
}
```

Referências e mais informação

- T.H. Cormen, C. E. Leiserson, R. L. Rivest , C. Stein. Introduction to Algorithms, 3rd Edition. MIT Press, 2009
  - Capítulo 22

- "Data Structures and Algorithm Analysis in Java", Second Edition, Mark Allen Weiss, Addison Wesley, 2006