1

Algoritmos em Grafos: Introdução

J. Pascoal Faria, R. Rossetti, L. Ferreira
CAL, MIEIC, FEUP

2

Índice

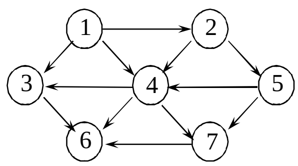
- Revisão de conceitos e definições
- Exemplos de aplicação
- Representação

Revisão de conceitos e definições

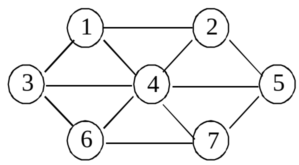
Conceito de grafo

- 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
  - as arestas têm por vezes um peso associado

Grafos dirigidos e não dirigidos



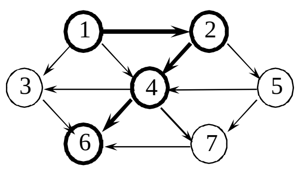
G1= (Cruzamentos, Ruas)



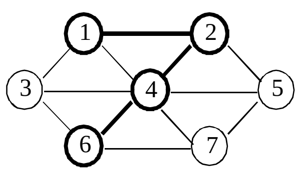
G2 = (Cidades, Estradas)

Caminhos

- Caminho - sequência de vértices v_1, …, 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



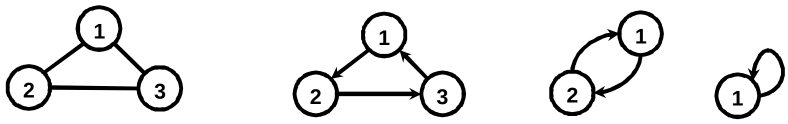
(1, 2, 4, 6)



(1, 2, 4, 6)

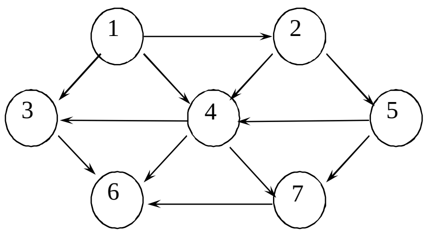
Ciclos

- Ciclo - 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



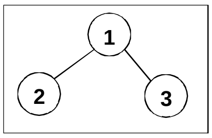
Grafo acíclico dirigido (*DAG – Directed acyclic graph*)

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

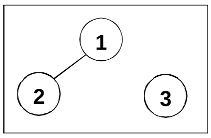


Conectividade (1/2)

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



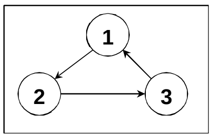
Conexo



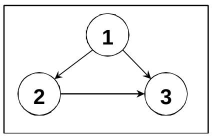
Não conexo

Conectividade (2/2)

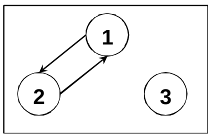
- Digrafo com a mesma propriedade: fortemente conexo, se p/ todo v, w $\in$ ***V*** existir em ***G*** um caminho de v para w.
- Digrafo fracamente conexo: se o grafo não dirigido subjacente é conexo



Fortemente conexo



Fracamente conexo

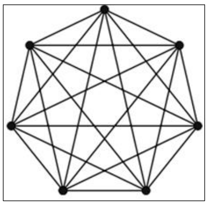


Não conexo

11

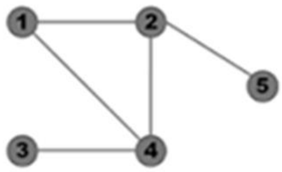
Densidade

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



Grafo completo com 7 vértices (K_7)

- Grafo esparso — $|E| = \Theta(V)$



12

Exemplos de aplicação

Exemplos de aplicação: caminho mais curto

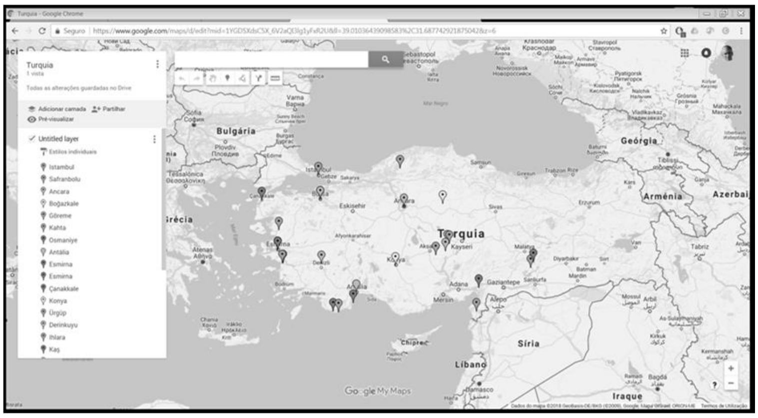
Qual o caminho mais curto / mais rápido / mais barato entre 2 pontos?

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
  - Navegação GPS
  - Controlo e gestão de tráfego
- Redes de abastecimento de água
  - Gestão de carga
- Redes de saneamento
  - Gestão de carga
- Redes de energia
  - Gestão da rede
- Redes de comunicações
  - Encaminhamento (*routing*)

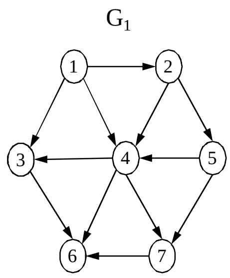
Exemplos de aplicação

- Workflows e cadeias de decisão
- Planeamento e gestão de projectos
- Compiladores, sistemas de ficheiros
- Jogos, criptografia
- Redes Bayesianas e probabilísticas (Processo de Manchester)

Representação

Matriz de adjacências

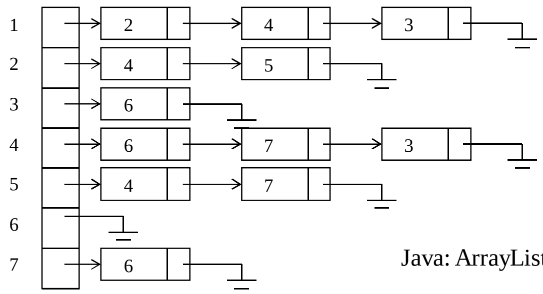
	1	2	3	4	5	6	7
1	0	1	1	1	0	0	0
2	0	0	0	1	1	0	0
3	0	0	0	0	0	1	0
4	0	0	1	0	0	1	1
5	0	0	0	1	0	0	1
6	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0



- a[u][v] = 1 sse (u, v) $\in$ E   (= 0 no caso contrário)
- elementos da matriz podem ser os pesos
- grafo não dirigido – matriz simétrica
- 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

- Estrutura típica para grafos esparsos
  - para cada vértice, mantém-se a lista dos vértices adjacentes
  - vector de cabeças de lista, indexado pelos vértices
  - espaço é O(|E| + |V|)
  - pesquisa de adjacentes em tempo proporcional ao número destes
- Grafo não dirigido: lista com dobro do espaço



Java: ArrayList<LinkedList<Integer>>

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