iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

iscte
TECNOLOGIAS
E ARQUITETURA

Desenho e Análise de Algoritmos

Notas - Aula TP 11 e 12

Grafos - Introdução

Ana Maria de Almeida

DCTI

2022/2023

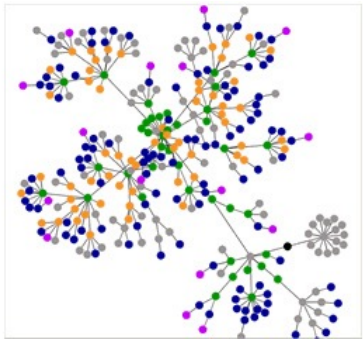
1

Plano para esta aula

Introdução ao modelo de Grafo ou Rede

- Grafos: conceitos elementares

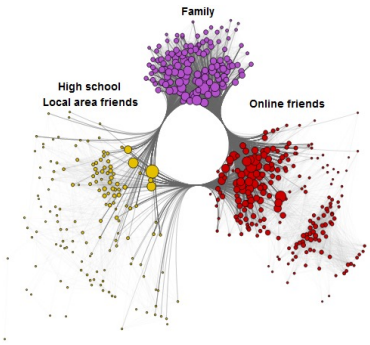
4



Grafos e Redes

Breve Introdução – Grafos e Redes

Na sua forma mais simples grafos são representações de objetos que se relacionam entre si. Formalmente a teoria dos grafos é uma área da Matemática discreta, com uma terminologia precisa.



É fácil de ver que uma rede pode ser representada por um grafo: a rede de amigos, da família, das ligações na rede social, das chamadas telefónicas...
Cada um destes exemplos é um grafo, em que os nós e as ligações ganham atributos semânticos, representando objetos reais ou abstratos.

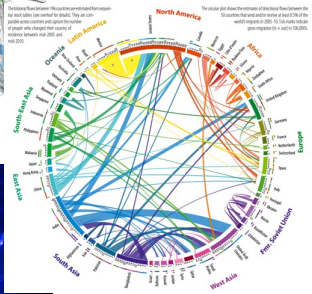
Qualquer que seja o sistema considerado, é possível encontrar uma rede (grafo) que define as interacções entre os seus componentes.

Source: Albert Bárabasi: Network Science: Introduction

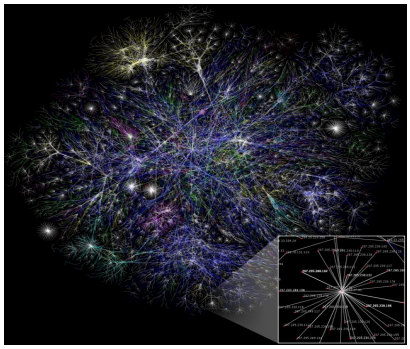
- O modelo de Grafo, ou Rede, representa conhecimento sobre iconetividade e interligações num sistema.
- Por exemplo, a rede elétrica é modelada através de um modelo de grafo, que é necessário para entender, por exemplo, como a **estrutura da rede real** afeta a robustez do sistema.
- O uso de *ferramentas* que avaliem o nível de interacção entre estrutura (e os processos dinâmicos que nela ocorrem) e o impacto de um corte na rede é uma área crítica para avaliação da robustez e escalabilidade.
- As "falhas de transporte" em redes (de transportes, de comunicação, infraestruturas) seguem leis que podem ser quantificadas, e até previstas, usando a Teoria de Grafos.

Algumas aplicações de Grafos

— *air transportation* network
— nodes are *world airports*
— links show *passenger flux*

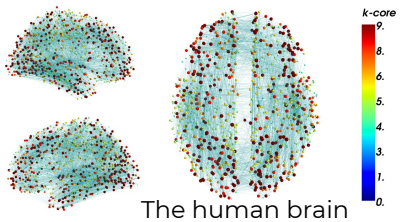


Immigration flows



https://upload.wikimedia.org/wikipedia/commons/d/d2/Internet_map_1024.jpg

The INTERNET network

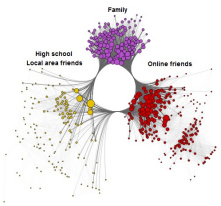


The human brain

2022/2023

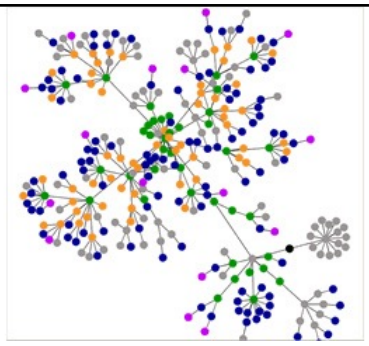
O facto de modelar sistemas reais como grafos é extremamente útil. Muitas aplicações necessitam representar conjuntos de ligações/relações entre pares de objetos, de modo a responder a questões tais como:

- Existe um caminho para ir de A para B? E qual o "melhor" caminho?
- Qual a menor distância entre 2 objetos?
- Quantos servidores são alcançáveis a partir deste?
- Qual é a minha rede de amigos?



Com o tipo abstrato de dados **Grafo**/**Rede**, conseguimos modelar os conjuntos de ligações, de modo a responder a estas e muitas outras situações.

11



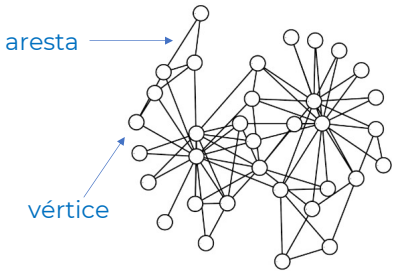
Grafos - Introdução

Conceitos elementares:

- Definição de grafo e tipos de grafos
- Algumas das principais propriedades dos grafos

13

Definição de Grafo

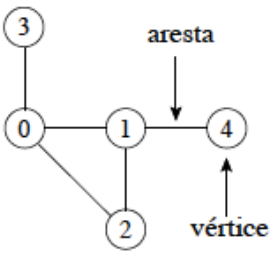


- Um grafo G = (V, E) é um par ordenado, onde:
  - V é o conjunto dos vértices (ou nós)
  - E é o conjunto das arestas (ou arcos)
- Os vértices representam os objetos ou entidades que podem estar em ligação.
- As arestas representam a ligação entre os vértices.
- Uma aresta pode ter uma direção, indicando que a ligação se estabelece do nó origem para o nó destino.

2022/2023

14

Definições básicas:



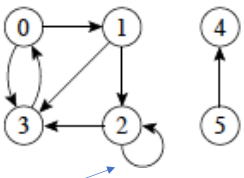
Grafo não orientado: todas as ligações -arestas- entre objetos – vértices- são bidirecionais

- **Dígrafo**: grafo dirigido ou orientado, ou seja, cada arco tem um nó origem e um nó destino

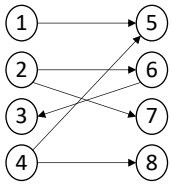
dito **sucessor**

dito **antecessor**

Grafo orientado ou dígrafo



laço ou lacete



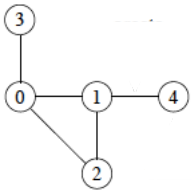
- Num digrafo, uma aresta costuma designar-se por arco e um vértice por nó.

Grafo bipartido: constituído dois subconjuntos de nós disjuntos e apenas existem arcos entre nós de subconjuntos diferentes.

17

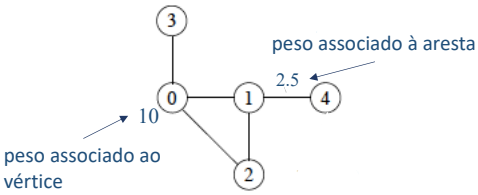
Definições básicas

- **Grafo não pesado** (*non-weighted*)



- Neste caso, as arestas (ou arcos) podem ser representadas por uma variável lógica (se não forem permitidas arestas paralelas)

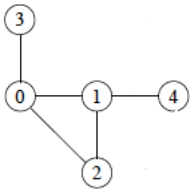
- **Grafo pesado** (*weighted*)



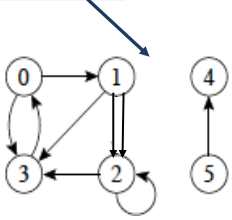
Já neste caso, as arestas/arcos (ou os vértices/nós) podem possuir informação associada, representando um "peso" da ligação (ou do nó).

Definições Básicas

- **Grau de um vértice:** número de arestas incidentes no vértice



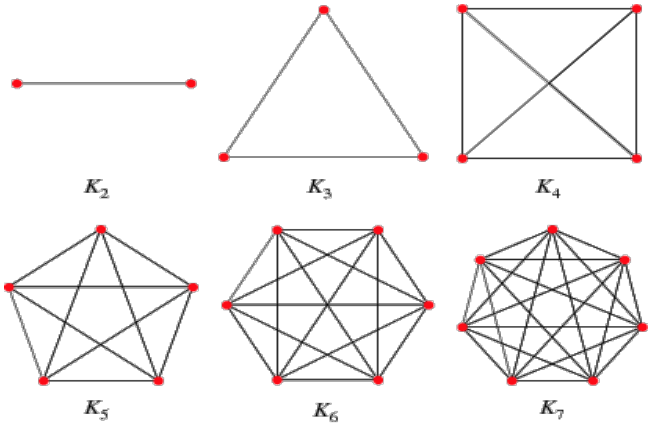
- **Adjacência**: o nó *a* diz-de **adjacente** ao nó b sse existe uma aresta (a,b)
- Denomina-se de **laço** ou **lacete** (tadpole ou selfloop) uma aresta com os seus extremos a iniciar e terminar no mesmo nó.
- **Multiarestas:** arestas **paralelas** que ligam **os mesmos vértices** (no mesmo sentido, no caso de um grafo dirigido).
- Um grafo que contém arestas paralelas chama-se um **multigrafo**.



Num grafo orientado, podemos definir **grau de entrada** e **grau de saída**

Definições Básicas

- Grafo **completo**: cada vértice está ligado **a todos** os outros



Source: http://mathworld.wolfram.com/CompleteGraph.html

20

Definições básicas

- Em grafos não dirigidos, sem lacetes nem arestas paralelas, a **densidade** é a razão entre o **número real de arestas** e o **número máximo possível de arestas** dado o número de vértices, logo, um valor em [0, 1].

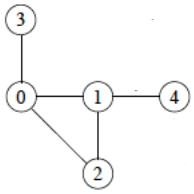
- Se ***n*** for o número de vértices e ***m*** for o número de arestas, a **densidade** de um **grafo não orientado** é dada por:

$$d = \frac{m}{\frac{n(n-1)}{2}}$$

quantidade de arestas real

quantidade máxima de arestas possível

Exemplo:



- n =
- m =
- d =

22

Conceitos importantes

Conetividade, caminhos, ciclos e travessias
Representações matemáticas

Conceitos importantes: caminho

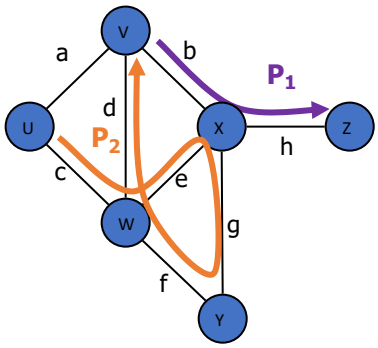
- **Caminho**: sequência de vértices
  - P_2 = {u, w, x, y, w, v}

  Note que:
  - P_2 = {u, w, x, y, w, v} ≡ {(u,w), (w,x), (x,y), (y,w), (w,v)}
    ≡ {c, e, g, f, d}

caminho descrito pelos seus vértices

caminho descrito pelas suas arestas

- **Caminho simples**: caminho onde todos os vértices são distintos
  - P_1 = {v, x, z} é um caminho simples

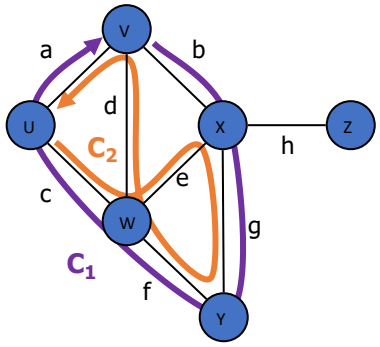


Grafo G=(V, E) com:
V={U, V, X, Y, W, Z} e E = {a, b, c, d, e, f, g, h, i, j}

27

Conceitos importantes: ciclo

- **Ciclo**: caminho **fechado** (começa e termina no mesmo vértice)
  - C_2 = {u, w, x, y, w, u}
- **Ciclo simples**: ciclo cujos vértices são todos diferentes (à excepção do inicial, que é também o final)
  - C_1 = {v, x, y, w, u, v}



28

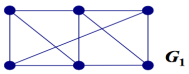
28

Conceitos Básicos: conetividade e árvore

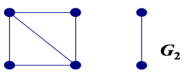
- **Grafo conexo**: grafo **não orientado** onde qualquer vértice é atingível a partir de outro outro (*existe caminho entre qualquer par de nós no grafo*)
- **Árvore**: grafo conexo e acíclico
- **Subgrafo** de G: Qualquer grafo H = (V', E') tal que,

$$V' \subseteq V \text{ e } E' \subseteq E .$$

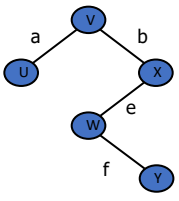
- **Árvore de cobertura** do grafo G: subgrafo conexo e acíclico, cujo conjunto de vértices contém (cobre) todos os vértices do grafo original G



Grafo conexo



Grafo desconexo

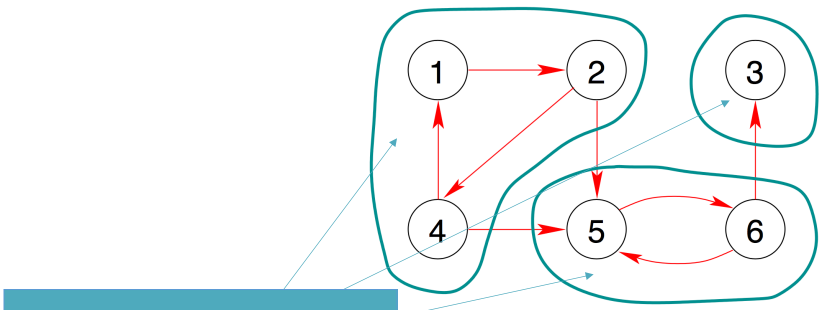


Árvore

29

Conceitos Básicos

- Grafo **fortemente conexo:** grafo **orientado** onde qualquer nó é atingível a partir de outro nó (existe caminho que liga qualquer par de nós no grafo, **nas duas direções**)



Cada uma destas três zonas mostra um (sub)grafo fortemente conexo

http://www4.di.uminho.pt/~jbj/AeC/page7/files/mp2-2.pdf

Representações de um grafo

Representações computacionais para um grafo

Matriz de adjacências

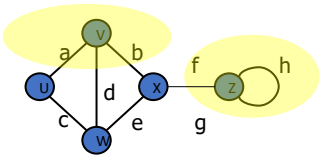
- Dado um grafo G = (V, A), com ***n*** vértices, v_1, ..., v_n, e ***m*** arestas, a_1, ..., a_m, a **matriz de adjacências** de G é uma matriz

  $M = [m_{ij}]_n$

de elementos em $\mathbb{N}_0$, definida como:

$m_{ij} = 1$ sse existe aresta a_{ij} **de** v_i **para** v_j

$m_{ij} = 0$ senão



	u	v	w	x	z
u	0	1	1	0	0
v	1	0	1	1	0
w	1	1	0	1	0
x	0	1	1	0	1
z	0	0	0	1	1

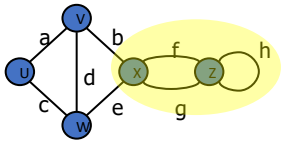
- O espaço ocupado por esta estrutura de dados é $S(n) = O(n^2)$.

35

- Se o grafo G = (V, A) tiver **multiarestas**, a matriz de adjacências de G é a matriz de elementos em $\mathbb{N}_0$ definida por:

$m_{ij} = \mathbf{k}$ sse existem k arestas a_{ij} **de** v_i **para** v_j

$m_{ij} = 0$ senão



Propriedades:

- Grafo não dirigido: matriz de adjacências é **simétrica**
- Grafo dirigido: matriz de adjacências não é necessariamente simétrica

	u	v	w	x	z
u	0	1	1	0	0
v	1	0	1	1	0
w	1	1	0	1	0
x	0	1	1	0	2
z	0	0	0	2	1

36

Matriz de incidências

- Dado um grafo G = (V, A) com *n* vértices, v_1, ..., v_n, e *m* arestas, a_1, ..., a_m, a **matriz de incidências** de G é uma matriz

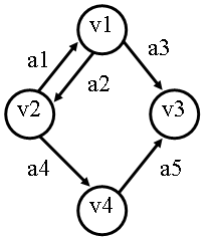
$$M = [m_{ij}]_{mxn}$$

1. Se o grafo for **não orientado**, os elementos são 0 ou 1, definidos como:

   $m_{ij} = 1$ sse x_i se encontrar num extremo da aresta j

   $m_{ij} = 0$ senão

   Espaço ocupado: S(n) = O(m x n)



2. Caso o grafo seja **dirigido,** os elementos definem-se em {0, 1, -1}:

   $m_{ij} = 1$ sse x_i é o antecessor do arco

   $m_{ij} = -1$ sse x_i é o sucessor do arco

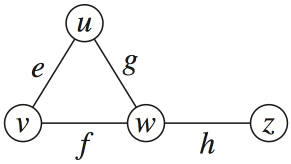
   $m_{ij} = 0$ sse não existe arco

	a1	a2	a3	a4	a5
v1	-1	1	1	0	0
v2	1	-1	0	1	0
v3	0	0	-1	0	-1
v4	0	0	0	-1	1

Representações computacionais

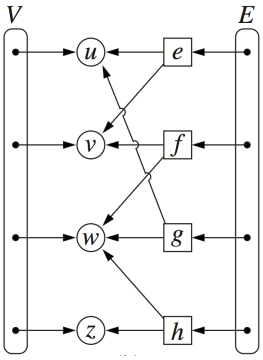
- As representações matemáticas – matrizes - em termos de espaço apenas são computacionalmente eficientes **se** o grafo for completo ou muito **denso**.
- A maioria dos grafos utilizados são **esparsos**, ou seja, $m << n^2$, e, neste caso, esta ineficiência pode ser significativa.
- Para representar computacionalmente um grafo existem diferentes hipóteses que podem reduzir o espaço ocupado em comparação com a utilização de uma representação com uma matriz de adjacência ou incidência. As mais comuns são:
  - uma estrutura baseada em listas de arestas
  - uma estrutura baseada em listas de adjacências
  - uma estrutura baseada em mapas de adjacências

Estrutura: lista de arestas



- Sequência de vértices (**V**) que aponta cada um dos vértices
- Sequência de arestas (**E**), que aponta cada uma das arestas

- Objeto **vértice**:
  - Identificação
  - A posição (referência) do vértice na sequência de vértices V
  - Informação adicional (se existir)
- Objeto **aresta**:
  - Identificação
  - Referência para o vértice *origem*
  - Referência para o vértice *destino*
  - A posição (referência) da aresta na sequência de arestas E
  - Informação adicional (peso), caso exista



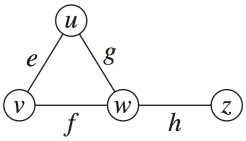
41

41

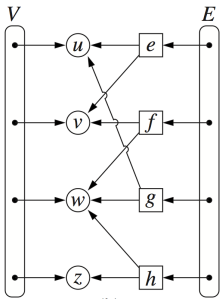
lista de arestas

- A ocupação de espaço nesta representação é:
  - n objectos do tipo vértice
  - m objectos do tipo aresta

S(n): O(n + m)



- A complexidade temporal será:



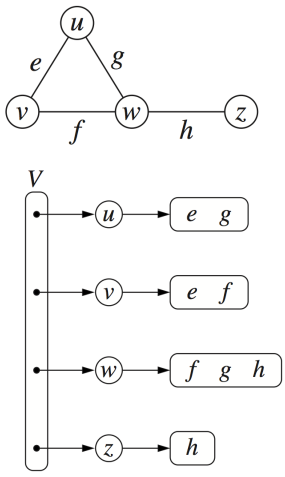
Operation	Running Time
vertex_count( ), edge_count( )	$O(1)$
vertices( )	$O(n)$
edges( )	$O(m)$
get_edge(u,v), degree(v), incident_edges(v)	$O(m)$
insert_vertex(x), insert_edge(u,v,x), remove_edge(e)	$O(1)$
remove_vertex(v)	$O(m)$

43

43

Estrutura: lista de adjacências

- Esta estrutura guarda, para cada vértice n, uma "lista" com todas as arestas incidentes em n.
- No caso de um grafo orientado, existem 2 listas: a lista das arestas que entram e a lista das que saiem de n.
- Sequência de vértices (**V**)
- Cada vértice aponta a sequência das arestas incidentes
- Nesta implementação, tipicamente, o objecto aresta indica (a posição d)o vértice terminal (na sequência de vértices).



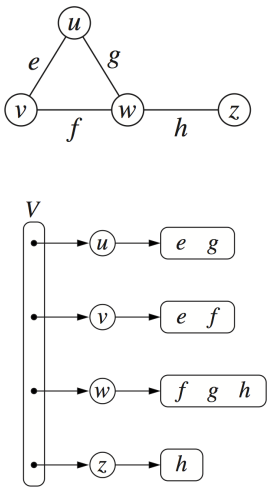
44

Lista de adjacências

- A ocupação de espaço nesta representação é:
  - **n** objectos do tipo vértice
  - **2m** objectos do tipo aresta

  S(n): O(n + m)
- A complexidade temporal será:

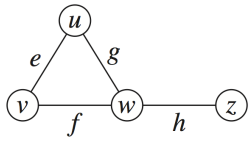
Operation	Running Time
vertex_count( ), edge_count( )	$O(1)$
vertices( )	$O(n)$
edges( )	$O(m)$
get_edge(u,v)	$O(\min(\deg(u), \deg(v)))$
degree(v)	$O(1)$
incident_edges(v)	$O(\deg(v))$
insert_vertex(x), insert_edge(u,v,x)	$O(1)$
remove_edge(e)	$O(1)$
remove_vertex(v)	$O(\deg(v))$

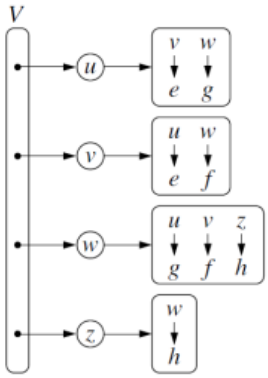


46

Estrutura: mapas de adjacência

Em comparação com a estrutura baseada em listas de adjacências, esta estrutura altera os "contentores" associados a cada vértice para um "dicionário", tendo por *chave* o vértice adjacente e por *valor* a aresta.





- Sequência de vértices (V) S(n): O(n + m)
- Cada vértice aponta o dicionário de vértices adjacentes → arestas

47

47

Quadro comparativo de ordens de complexidade temporal

Operation	Edge List	Adj. List	Adj. Map	Adj. Matrix
vertex_count( )	$O(1)$	$O(1)$	$O(1)$	$O(1)$
edge_count( )	$O(1)$	$O(1)$	$O(1)$	$O(1)$
vertices( )	$O(n)$	$O(n)$	$O(n)$	$O(n)$
edges( )	$O(m)$	$O(m)$	$O(m)$	$O(m)$
get_edge(u,v)	$O(m)$	$O(\min(d_u, d_v))$	$O(1)$ exp.	$O(1)$
degree(v)	$O(m)$	$O(1)$	$O(1)$	$O(n)$
incident_edges(v)	$O(m)$	$O(d_v)$	$O(d_v)$	$O(n)$
insert_vertex(x)	$O(1)$	$O(1)$	$O(1)$	$O(n^2)$
remove_vertex(v)	$O(m)$	$O(d_v)$	$O(d_v)$	$O(n^2)$
insert_edge(u,v,x)	$O(1)$	$O(1)$	$O(1)$ exp.	$O(1)$
remove_edge(e)	$O(1)$	$O(1)$	$O(1)$ exp.	$O(1)$

source: Cormen, Leiserson et al., MIT Press

d_v : grau do nó v exp. : tempo esperado

48

TAD Grafo: definição do conjunto mínimo de operadores

Algumas questões típicas em algoritmos que utilizam grafos:

- O vértice v_i é adjacente ao v_j?
- Qual o grau de v_i?
- Quais são os vizinhos de v_i?

Dado um grafo, G = (V, E), para definiro conjunto mínimo de operações de um grafo é necessário definir, pelo menos, as operações seguintes:

- indicar a quantidade de vértices em G
- indicar a quantidade de arestas em G
- indicar a coleção de todos os vértices em G
- indicar o grau de um dado vértice
- uma listagem com todos os arcos incidentes num dado vértice
- uma listagem de todos os vértices adjacentes a um dado vértice
- indicação se dois vértices dados são adjacentes

Para responder a muitas destas questões, necessitamos de um algoritmo para fazer a **travessia do Grafo**

49

Para estudar

- **Cap. 2 e 3 – *Introduction to Algorithms*.** Cormen, Leiserson et al., 4rd ed. MIT Press, 2020
- **Cap. 2 – Algorithm *Design***, John Kleinberg, Eva Tàrdos, Addison-Wesley, 2005.