IPS Instituto Politécnico de Setúbal

Escola Superior de Tecnologia de Setúbal

Programação Avançada 2021-22

[3a] Grafos | Conceitos

Bruno Silva, Patrícia Macedo

Sumário 📝

- Contexto Histórico
- Conceitos Gerais sobre Grafos
- Aplicações
- Percorrer Grafos

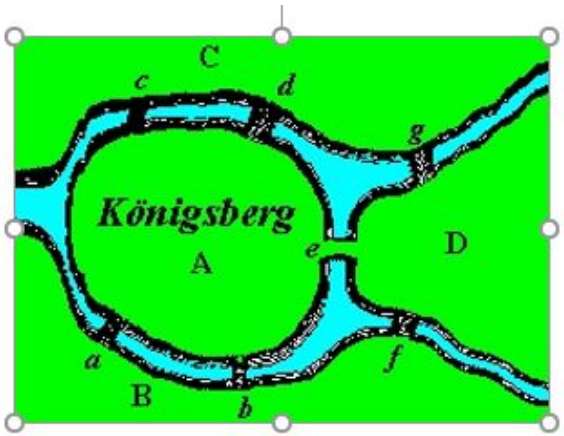
Contexto Histórico

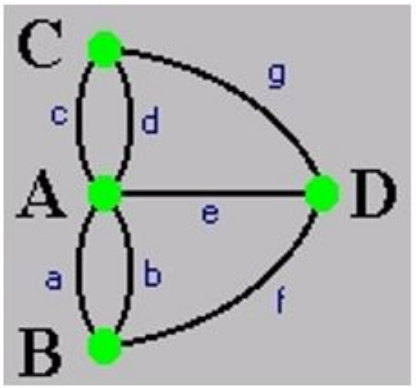
Problema das Pontes de Königsberg

No século 18, na cidade de Königsberg, um conjunto de sete pontes cruzavam o rio Pregel, ligando 4 ilhas entre si.

- Os habitantes perguntavam: É possível cruzar as sete pontes numa caminhada contínua sem passar duas vezes por qualquer uma delas?

Em 1736, Euler apresenta a solução deste problema na Academia de São Petersburgo, sendo este o primeiro trabalho sobre Grafos.

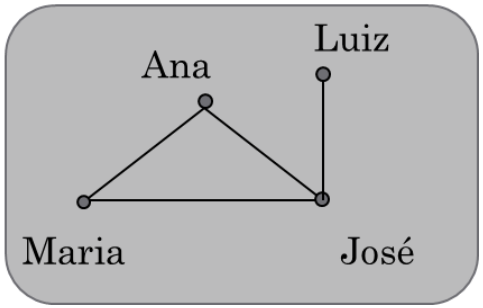




Grafos | Conceitos

Um **grafo G(V, A)** é definido pelos conjuntos `V` e `A`, onde:

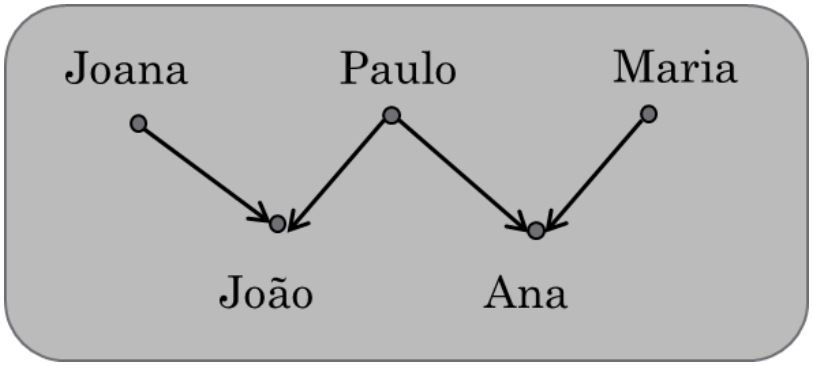
- `V` é um conjunto não vazio: Vértices, Nodos ou Nós do grafo (vertex);
- `A` é um conjunto de pares ordenados `a=(v,w)` com `v` e `w` pertencente a `V`: Arestas, Linhas ou Ramos do grafo (edge).
  Exemplo
  - V = {p | p é uma pessoa}
  - A = {(v,w) | v é amiga de w}
  - V = {Maria, José, Ana, Luiz}
  - A = {(Maria, José), (Maria, Ana), (José, Luiz), (José, Ana)}



Digrafo | Grafo orientado

Exemplo

- `V = {p | p é uma pessoa}`
- `A = {(v,w) | v é pai ou mãe de w}`



Ordem | Adjacência

Ordem - É o número de vértices do grafo.

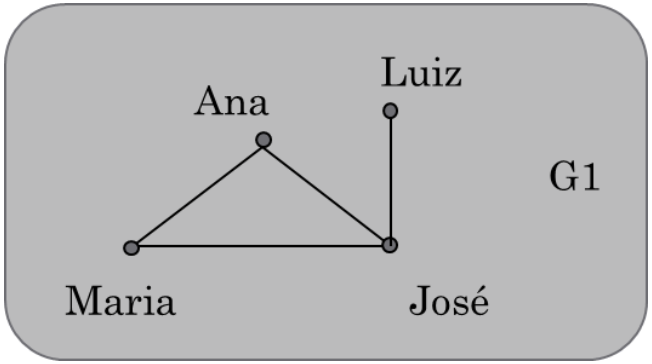
- Ordem(G1) = 4

Adjacência:
Dois vértices (`v` e `w`) são adjacentes se há uma aresta `a=(v,w)` em G.

- Ex: José e Luiz em G1

Duas arestas são adjacentes se incidem sobre o mesmo vértice.

- Ex: (Ana, Maria) e (Ana, José) em G1



Graus

Grau de um vértice : É o número de arestas incidentes no vértice.

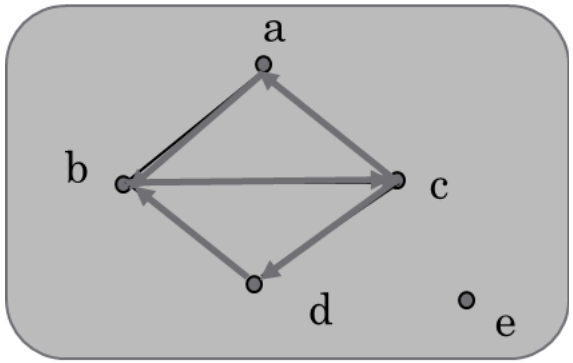
- Grau(b)=3

Grau de saída (outdegree): Número de arestas que têm ponta inicial no vértice.

- GrauSaída(b) = 1

Grau de entrada (indegree): Número de arestas têm ponta final no vértice.

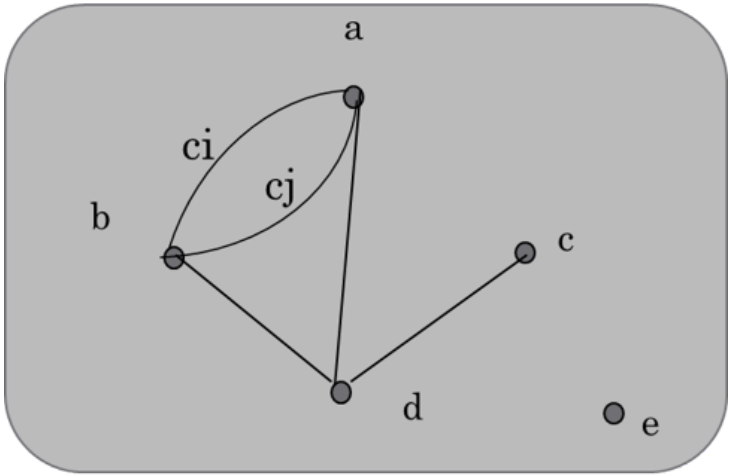
- GrauEntrada(b) = 2



Vértices Isolados | Arestas Paralelas

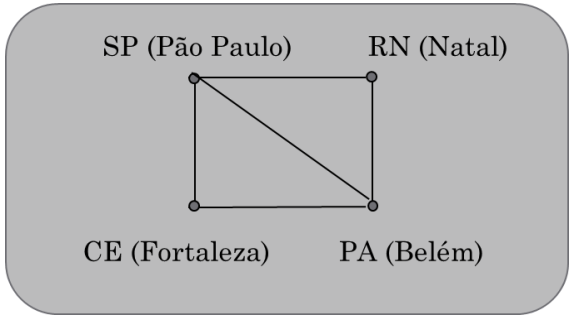
Vértice isolado : É aquele que possui grau igual a zero.

- Ex: Vértice `e`
  Arestas paralelas : Possuem os mesmos vértices terminais.
- Exemplo `ci=(a,b)` e `cj=(a,b)`

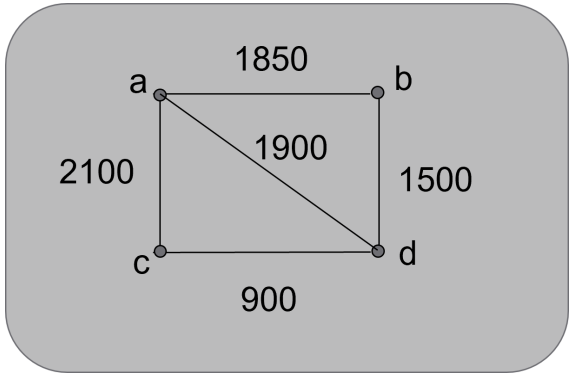


Grafo Rotulado | Grafo Valorado

Rotulado - Grafo em que cada vértice está associado a um rótulo.



Valorado - Grafo em que cada aresta está associado a um valor.



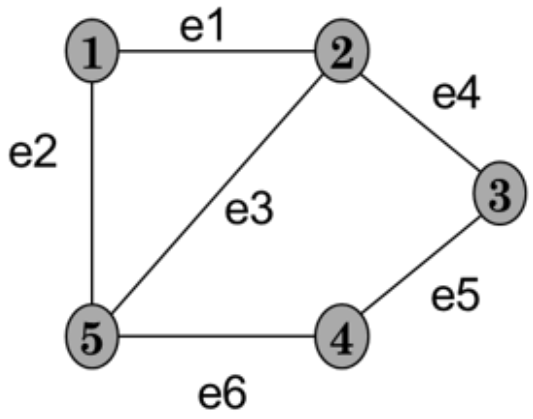
Algumas áreas de aplicação

GRAPH	VERTICES	EDGES
circulatory	organ	blood vessel
skeletal	joint	bone
nervous	neuron	synapse
social	person	relationship
epidemiological	person	infection
chemical	molecule	bond
n-body	particle	force
genetic	gene	mutation
biochemical	protein	interaction
transportation	intersection	road
Internet	computer	cable
Web	web page	link
distribution	power station	power line
mechanical	joint	beam
software	module	call
financial	account	transaction

Exercício 1

Para o grafo da imagem seguinte indique quais as frases verdadeiras:

- A ordem do grafo é 3;
- Os vértices com maior grau são os vértices 5 e 2;
- As arestas incidentes ao vértice 2 são as arestas e1 e e4;
- O vértice oposto ao vértice 2 na aresta e3 é o vértice 5;
- O vértice 1 e 5 são adjacentes.



Exercício 2

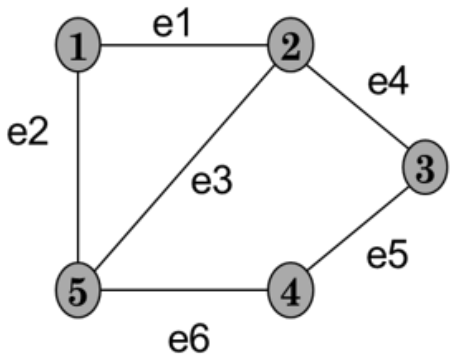
Desenhe um grafo que obedeça aos seguintes critérios

- É um digrafo;
- Tem ordem 4;
- Tem um vértice isolado;
- Tem duas arestas paralelas;
- O vértice com maior grau de saída tem grau 4;
- O vértice com maior grau de entrada tem grau 2.

Percorrer Grafos| Finalidade

Qual a finalidade de Percorrer um grafo?

- Procurar se um vértice, ou uma aresta existem;
- Cópia de um grafo ou conversão entre representações diferentes;
- Contagem de números de vértices e/ou arestas;
- Determinação de caminho entre dois vértices ou ciclos, caso existam.



Percorrer Grafos

As **estruturas lineares** são percorridas sequencialmente, normalmente de uma extremidade para a outra.

As **estruturas não lineares** podem ser percorridas de diversas formas:

- As árvores pode ser percorridas em:
  - **Largura**;
  - **Profundidade** (pos-order e pre-order).
- Os grafos também podem ser percorridos em
  - **Largura**;
  - **Profundidade**.

Atividade : Percorrer Grafos

Aceda aos seguintes links para, de forma autónoma, percorrer um grafo em Largura e em Profundidade.
- https://www.cs.usfca.edu/~galles/visualization/BFS.html
- https://www.cs.usfca.edu/~galles/visualization/DFS.html

- **Qual a principal diferença entre os dois algoritmos que explorou ?**

Percorrer Grafos

Os algoritmos que percorrem **grafos** devem levar em conta a:

- **Eficiência**: um mesmo local não deve ser visitado mais do que uma vez.
  - Estratégia: Marcam-se os vértices já visitados.
- **Corretude**: o percurso deve ser feito de modo a que não se perca informação.
  - Estratégia: Mantém-se uma coleção (pilha ou fila) com todos os vértices ainda por visitar.
    - Fila [FIFO] – Percorrer em Largura (BFS);
    - Pilha [LIFO] – Percorrer em Profundidade (DFS).

Breath First Search (BFS) & Depth First Search

BFS Largura - Exploração por níveis

- Inicia-se num vértice (denominado *vértice raiz*) e exploram-se todos os vértices adjacentes a este;
- Para cada um dos vértices adjacentes, exploram-se os **seus vértices vizinhos** ainda não visitados, repetindo os passos até já não existirem mais vértices por visitar. Para tal usa-se uma estrutura com politica de acesso **FIFO** para guardar os nós não visitados.

DFS Profundidade - Explora-se um ramo completo.

- Inicia-se num vértice (denominado *vértice raiz*) e explora-se, tanto quanto possível, cada um dos seus ramos antes de retroceder. Para tal usa-se uma estrutura com politica de acesso **LIFO** para guardar os nós não visitados.

Breath First Search (BFS) Algoritmo

```
Algorithm: BFS(Graph,vertice_root)
BEGIN
    setAsVisited(vertice_root)
    enqueue(q,vertice_root)
    WHILE q is not EMPTY
        BEGIN
            v <- dequeue(q)
            process(v)
            FOR EACH w adjacents(v)
                IF w is not visited THEN
                    BEGIN
                        setAsVisited(w)
                        enqueue(q,w)
                    END
        END
END
```

Depth First Search (DFS) Algoritmo

```
Algorithm: DFS(Graph,vertice_root)
BEGIN
    setAsVisited(vertice_root)
    push(s,vertice_root)
    WHILE s is not EMPTY
        BEGIN
            v <- pop(s)
            process(v)
            FOR EACH w adjacents(v)
                IF w is not visited THEN
                    BEGIN
                        setAsVisited(w)
                        push(s,w)
                    END
        END
END
```

Exercício 3

Percorra o grafo da imagem seguinte usando as travessias:

- Em profundidade (DFS);
- Em largura (BFS).

(Nota: inicie a travessia no vértice 3).