Algoritmos - Aula 7

Fernando Raposo

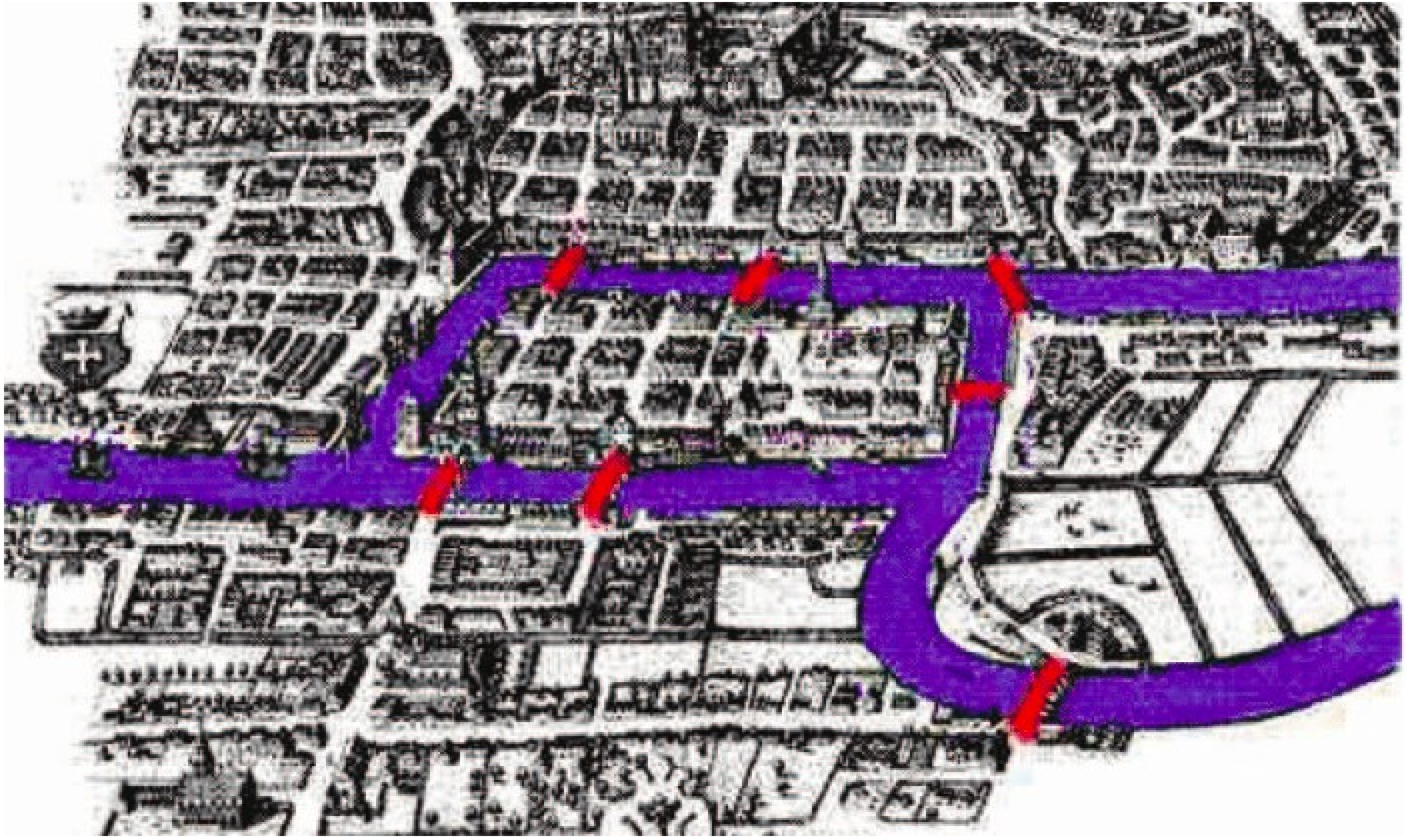
Vamos ver

- Grafos
  - Conceitos Introdutórios
  - Definição
  - Vizinhança
  - Grau
  - Representação
  - Parte Prática
  - Caminho
  - Problemas apoiados em Grafos
- Busca em Profundidade
  - Parte Prática
  - O que respondemos utilizando busca em profundidade?

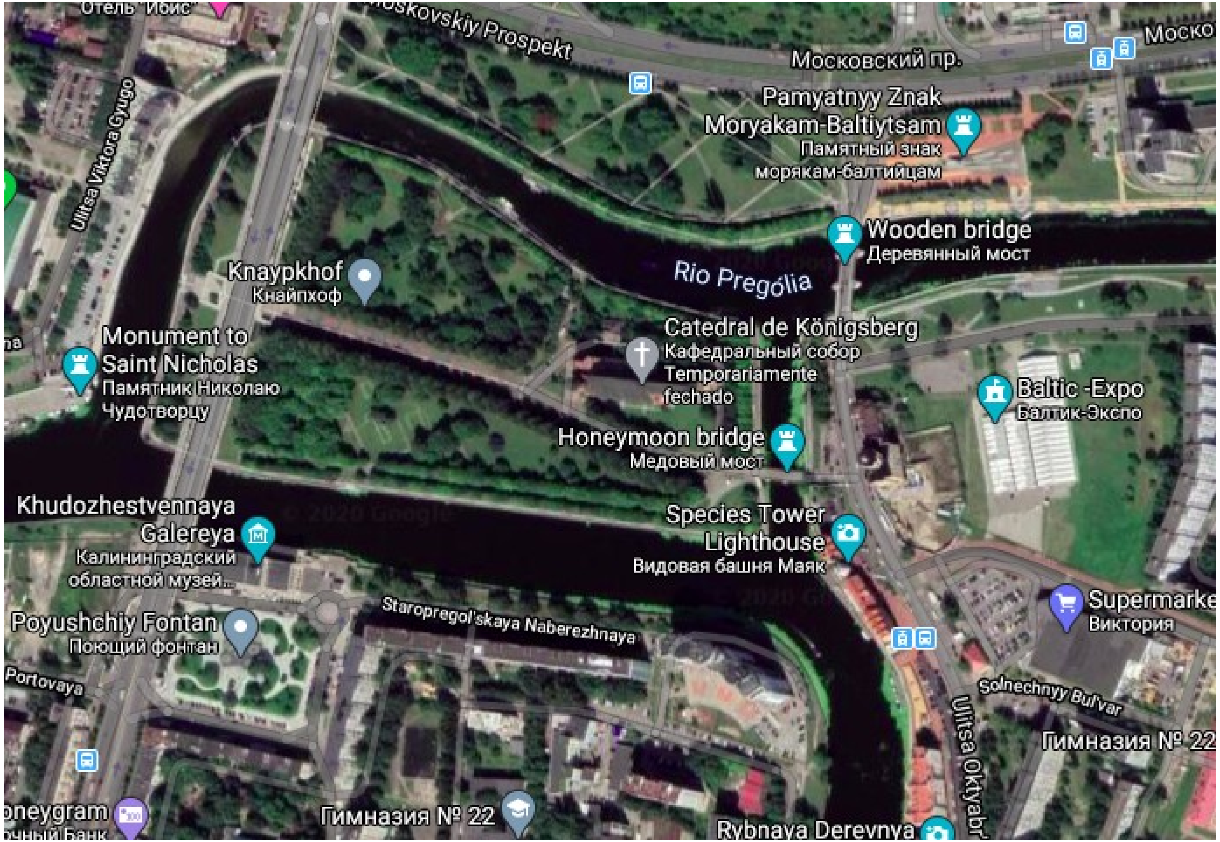
Teoria dos Grafos: Início

- Início com Euler em **Königsberg** (atual Kaliningrado)
    - A cidade é cortada pelo rio Pregel, que possui duas ilhas, e essas ilhas eram conectadas uma a outra e ao continente por **7 pontes**.
    - A questão: Encontrar um percurso que partisse de uma margem, e atravessando cada ponte uma única vez, atravessar cada uma das sete pontes e retornar à margem de saída.

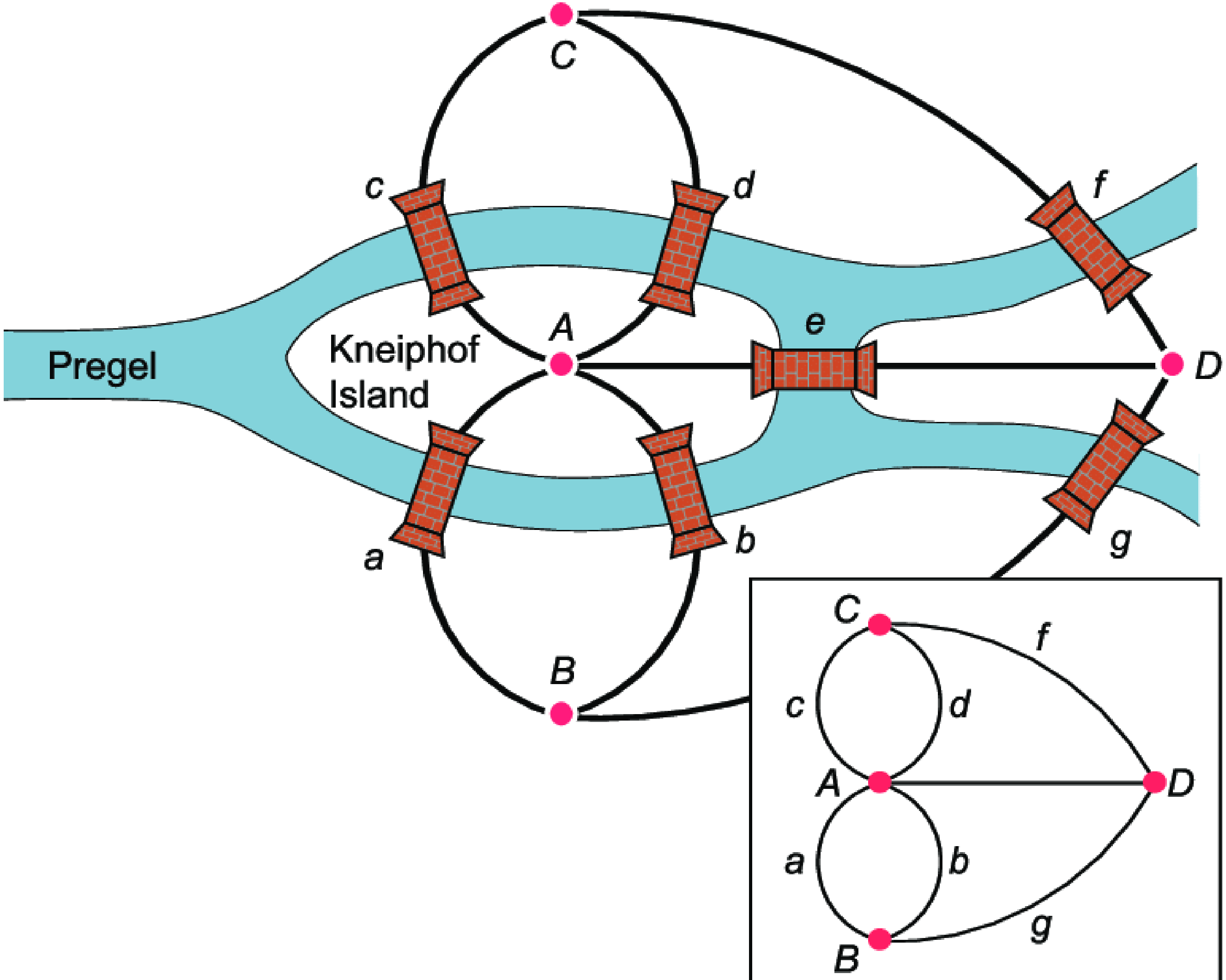
Königsberg, Prússia, 1736



Kaliningrado, Rússia, 2020

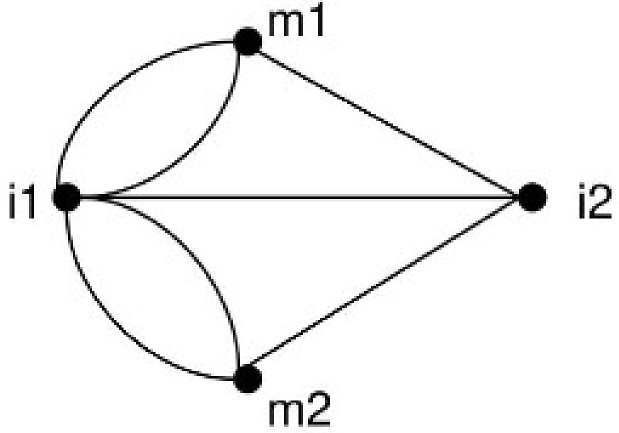


Resumo



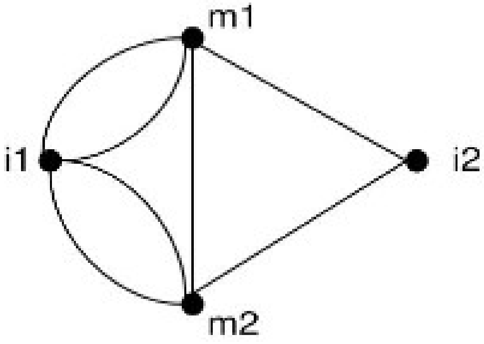
Análise do problema

- Tentem olhar para o grafo e ver se há uma combinação de caminhos possível



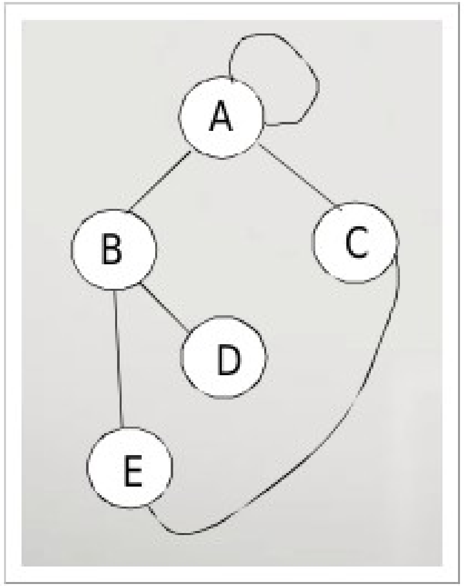
Análise do problema

- Chegamos a uma conclusão de que **NÃO É POSSÍVEL**.
- Cada vértice do Grafo tem grau ímpar;
- Utilizamos grafos para resumir ou simplificar nosso entendimento de algo;
- Um grafo é uma **simplificação** da realidade;
- Poderíamos encontrar uma solução se mudássemos as pontes de lugar:



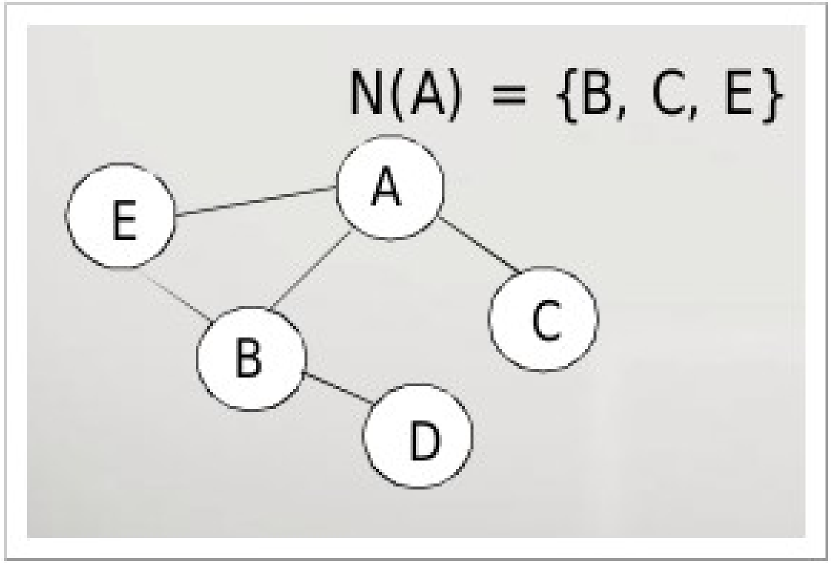
Definição: Grafo

- É um par de conjuntos: um conjunto de vértices e um conjunto de arcos (arestas). Cada arco é um par ordenado de vértices. O primeiro vértice do par é a ponta inicial do arco e o segundo é a ponta final. G = (V, A)
- No grafo a seguir temos o seguintes conjuntos:
  - V = {A, B, C, D, E} e
  - A = {(A,A), (A,B), (A,C), (B,D), (B,E), (C,E)}



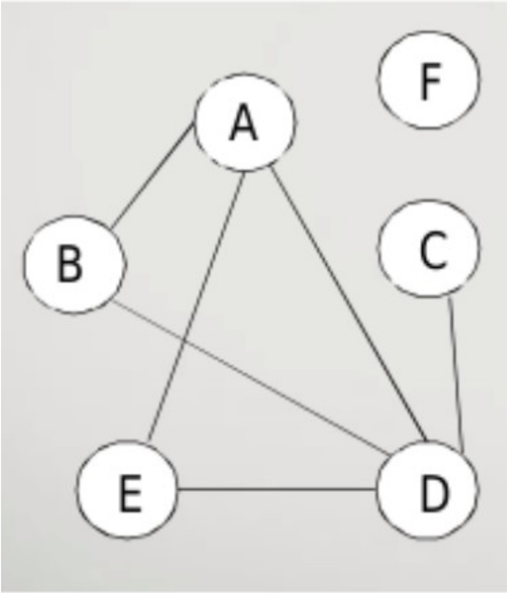
Grafo: Vizinhança

- A vizinhança de um nó é dada por:
- *N(v) = {w pertence a V | v-w pertence a A}*



Grafo: Grau

- O grau de um vértice é a quantidade de arestas que incide nele.
- Um vértice de grau ZERO é aquele que está isolado (sem arestas incidentes)



G(A) = 3
G(B) = 2
G(C) = 1
G(D) = 4
G(E) = 2
G(F) = 0

Grafo: Representação

- Podemos exprimir um grafo utilizando uma Matriz de adjacências;
- A matriz de adjacências de um grafo é uma matriz de 0s e 1s com colunas e linhas indexadas pelos vértices. Se adj[][] é uma tal matriz então, para cada vértice v e cada vértice w.

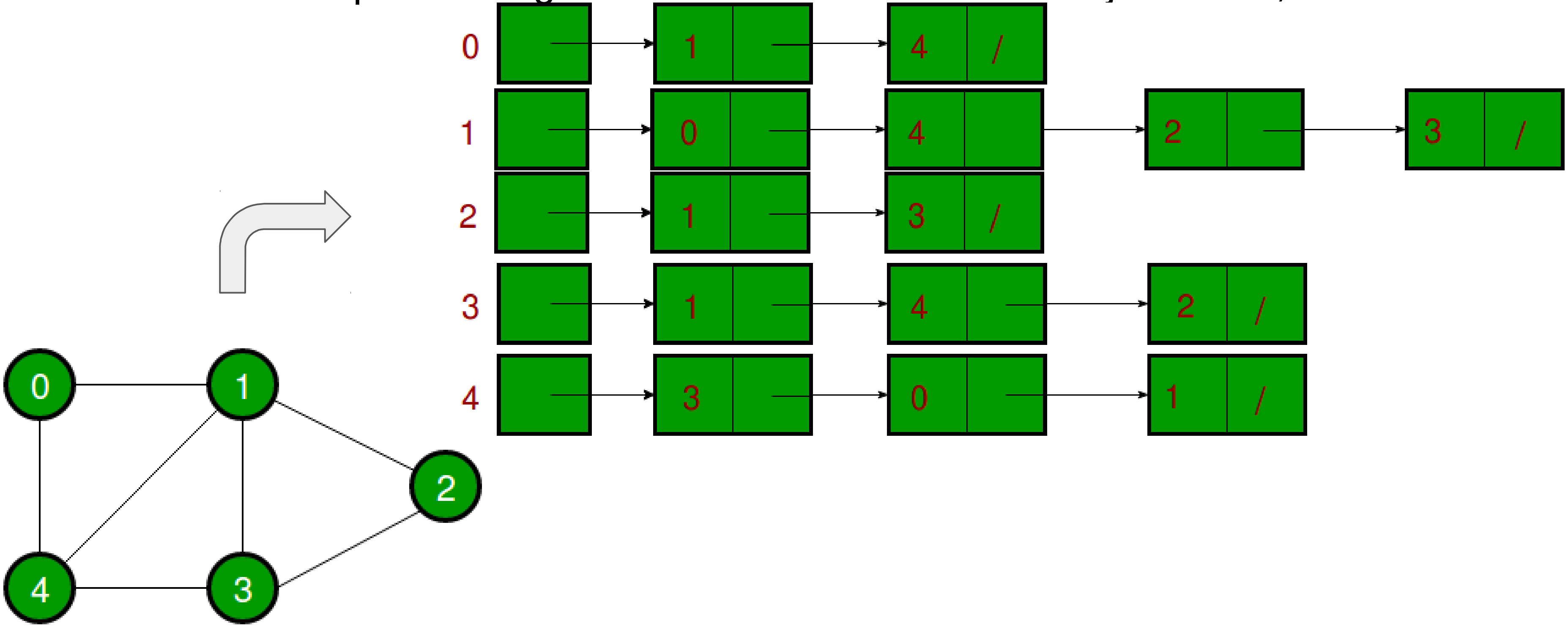
	1	2	3	4	5	6	7
1	0	1	0	1	0	0	0
2	1	0	1	1	0	0	0
3	0	1	0	1	0	0	0
4	1	1	1	0	1	0	0
5	0	0	0	1	0	1	0
6	0	0	0	0	1	0	1
7	0	0	0	0	0	1	0

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

Qual a particularidade do vértice 6?

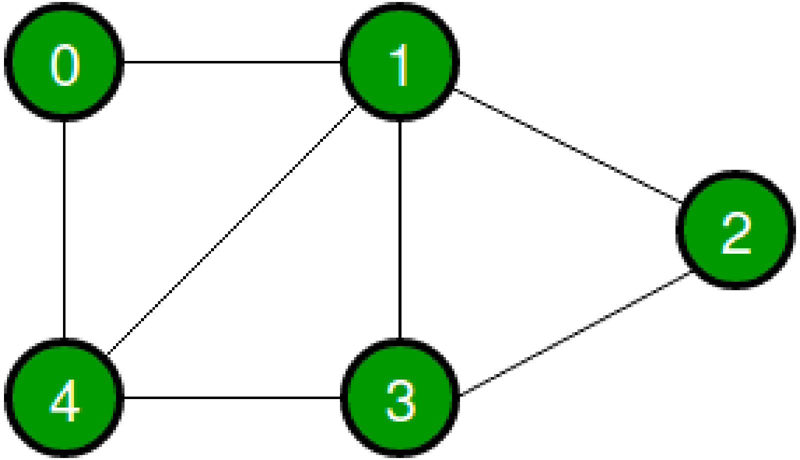
Grafo: Representação

- Podemos exprimir um grafo utilizando uma Lista de adjacências;



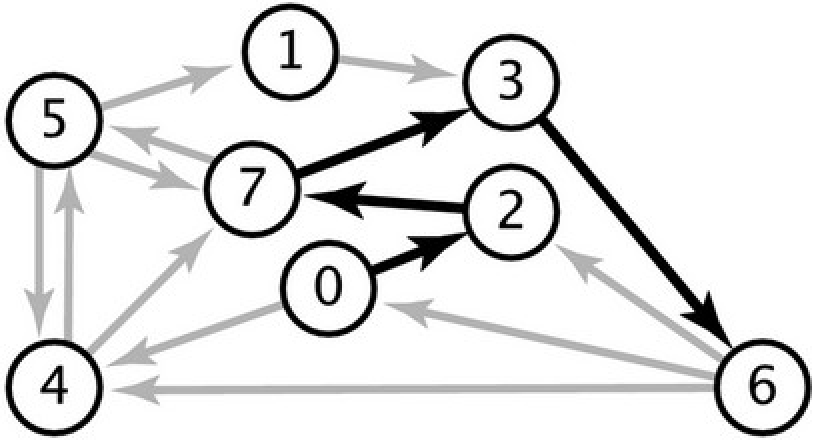
Parte Prática...

- Vamos tentar construir o Grafo anterior como uma lista de adjacências?



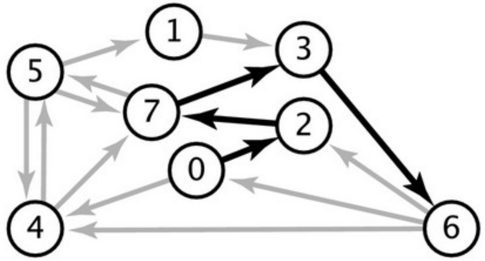
Grafo: Caminho

- Achar o caminho entre dois vértices é uma resposta **falsamente simples**;
- Pois:
  - Deslocamentos implicam custos (**dinheiro**/ tempo);
  - Podemos ter que obrigatoriamente passar em um ou mais vértices (em uma ordem específica);
- Caminho
  - É um passeio <u>sem arcos repetidos</u>, ou seja, um passeio em que os arcos são todos diferentes entre si. Um caminho é simples se não tem vértices repetidos.
    Por exemplo, **0-2-7-3-6** é um caminho simples no grafo da figura.



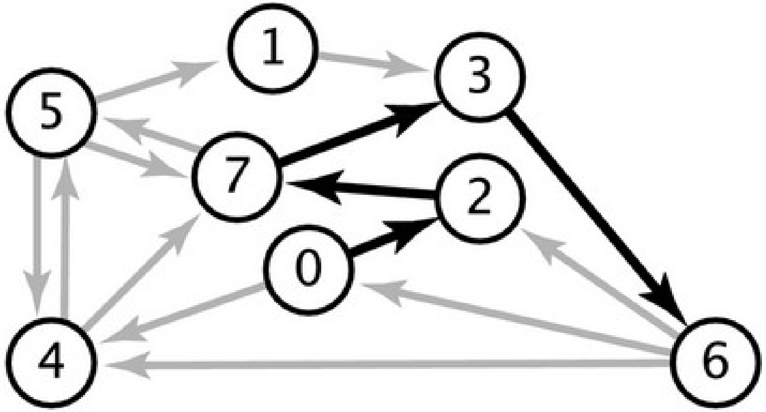
Grafo: Caminho

- A **origem** de um caminho é o seu primeiro vértice. O **término** é o seu último vértice. Se um caminho tem origem em **0** e término em **6**, dizemos que vai de **0** a **6**.
- O comprimento de um caminho é o número de arcos do caminho. Se um caminho tem n vértices, seu comprimento é pelo menos n−1; se o caminho é simples, seu comprimento é exatamente n−1.
- Atenção: A sequência 5-1-3-6-4-5 não é um caminho simples, pois há repetição de vertices;



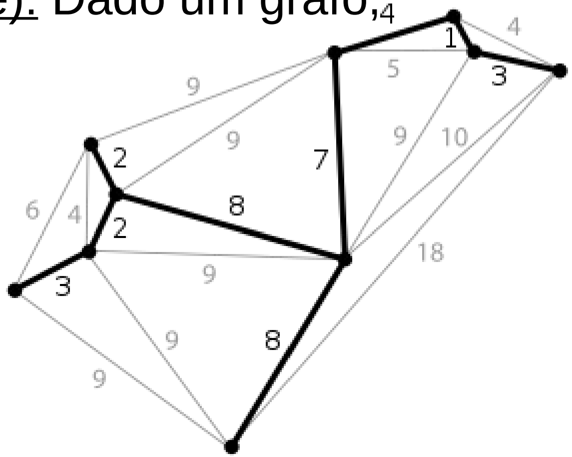
Grafo: Caminho

- Ciclos são caminhos **fechados**.
- Dizemos que um arco v-w pertence a um dado ciclo (ou que o ciclo passa pelo arco) se o vértice w é o sucessor de v no ciclo.   Um ciclo é simples se não tem vértices repetidos exceto pelo último (que coincide com o primeiro).
- Exemplos de ciclos:
  - **5-7-5**
  - **4-7-5-4**
  - **6-2-7-3-6**



Problemas apoiados em Grafos

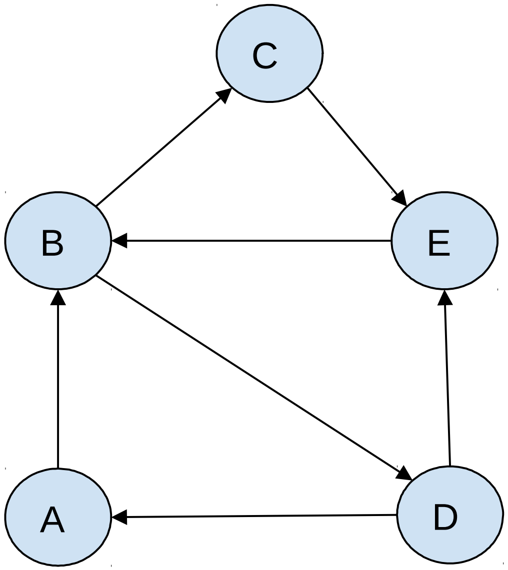
- <u>Simplificando Königsberg:</u> Dado um grafo com vértices origem ***x*** e destino ***y***. Precisamos responder se é possível ter um caminho que vá de ***x*** a ***y***.
  - Este problema é típico de labirintos (Mazes)
- <u>Detecção de ciclos:</u> Como descobrir que um caminho entre os vértices x e y tem um ciclo?
- <u>Árvore de Extensão Mínima (Minimum Spanning Tree):</u> Dado um grafo, detectar subgrafo que tem menor peso;



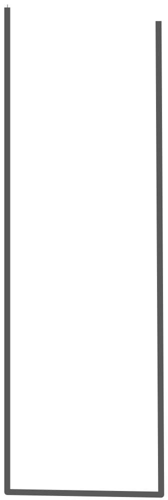
Busca em Profundidade

- Para responder o problema de Königsberg (e muitos outros...) poderíamos utilizar uma estratégia chamada **Busca em Profundidade** (*Depth First Search*), ou DFS.
- Estratégia:
    a. Marcar todos os vértices como não-visitados
    b. Escolhemos um vértice inicial;
    c. Exploramos ao máximo cada “ramo” do vértice antes de retroceder;

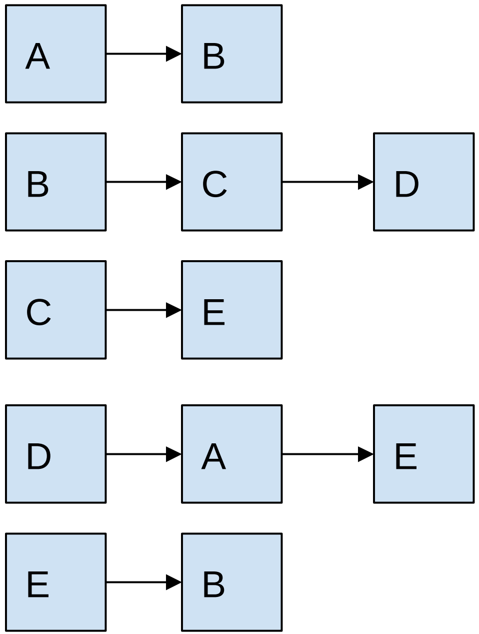
Busca em Profundidade



Grafo



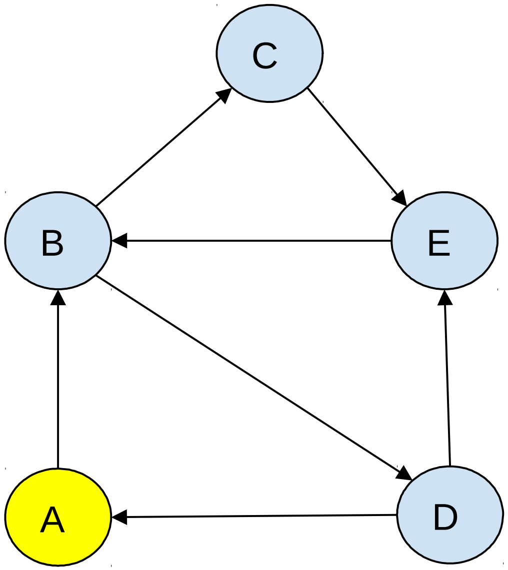
Pilha



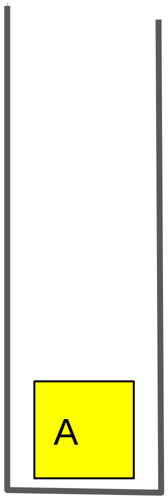
Lista de Adjacências

Busca em Profundidade

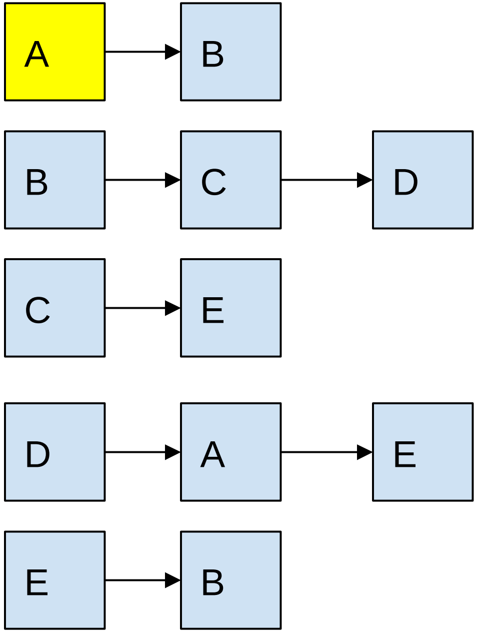
- **Início: Marcamos o primeiro vértice que escolhemos como visitado e o colocamos no topo da pilha.**



Grafo



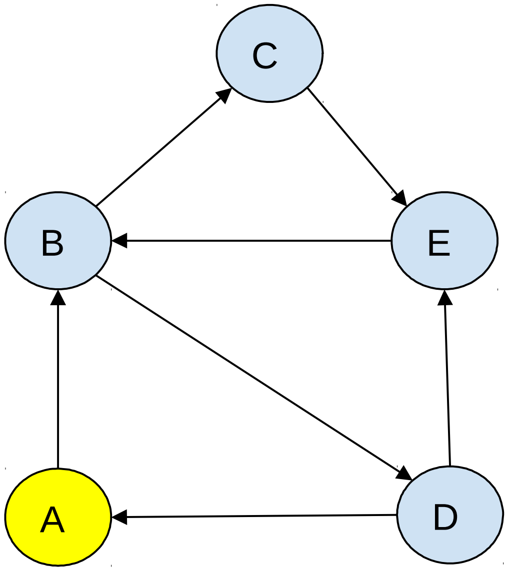
Pilha



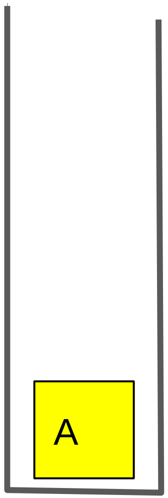
Lista de Adjacências

Busca em Profundidade

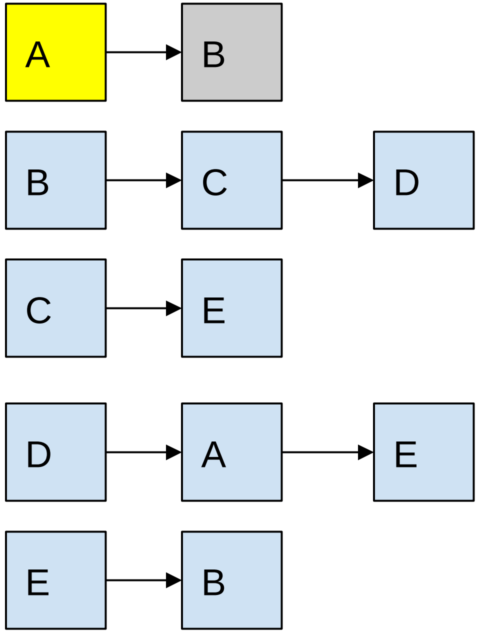
- **Olhamos o item no topo da pilha e pegamos a letra mais baixa próxima a ele que ainda NÃO foi visitada...**



Grafo



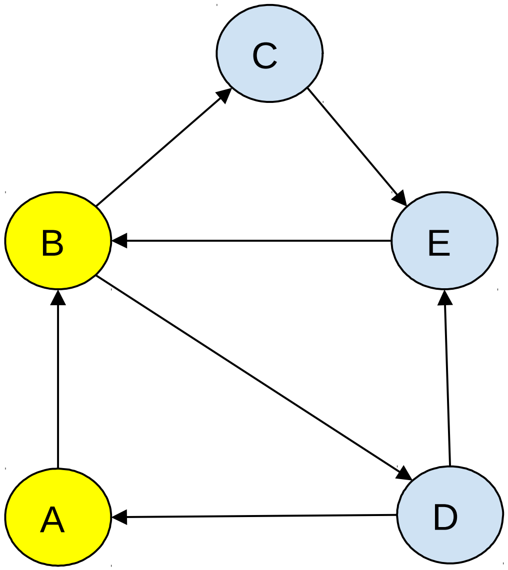
Pilha



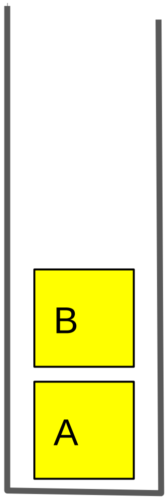
Lista de Adjacências

Busca em Profundidade

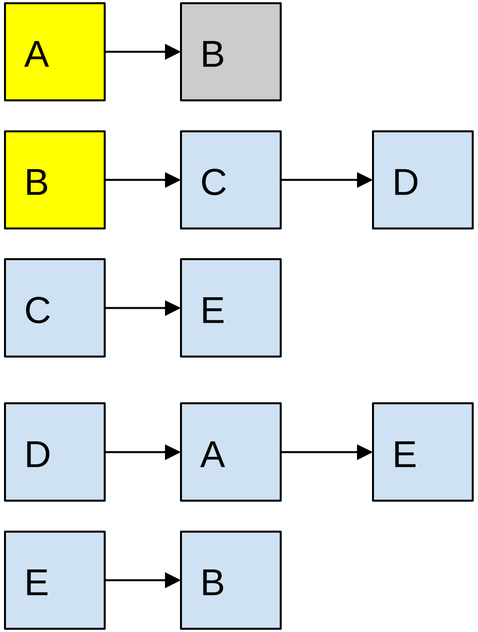
- **...E o colocamos na pilha. Marcando-o como visitado.**



Grafo



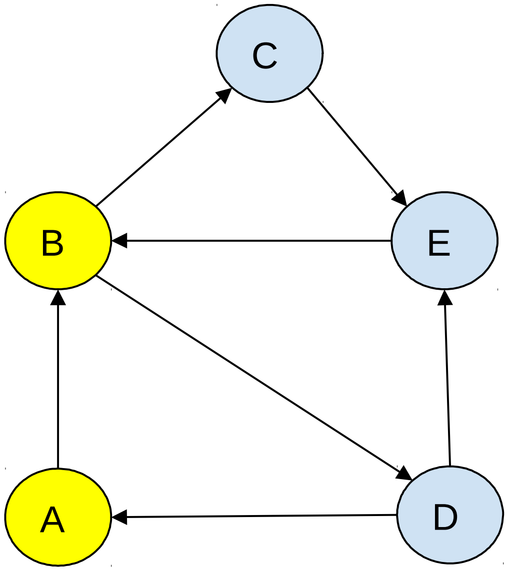
Pilha



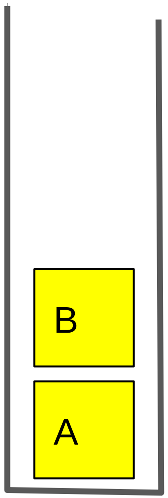
Lista de Adjacências

Busca em Profundidade

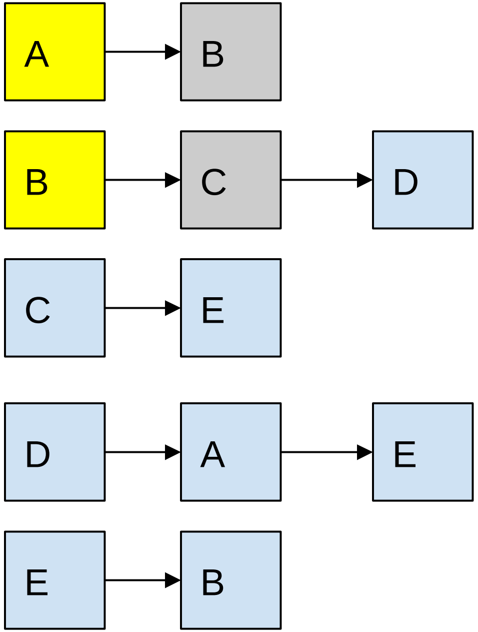
- **Olhamos o item no topo da pilha e pegamos a letra mais baixa próxima a ele que ainda NÃO foi visitada...**



Grafo



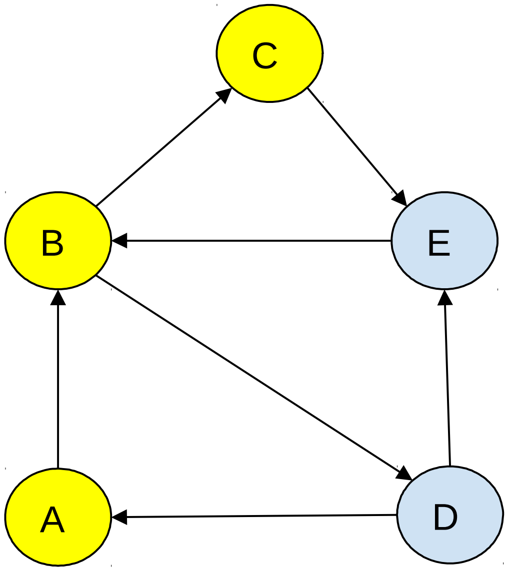
Pilha



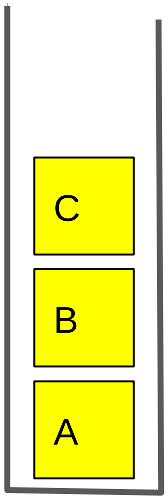
Lista de Adjacências

Busca em Profundidade

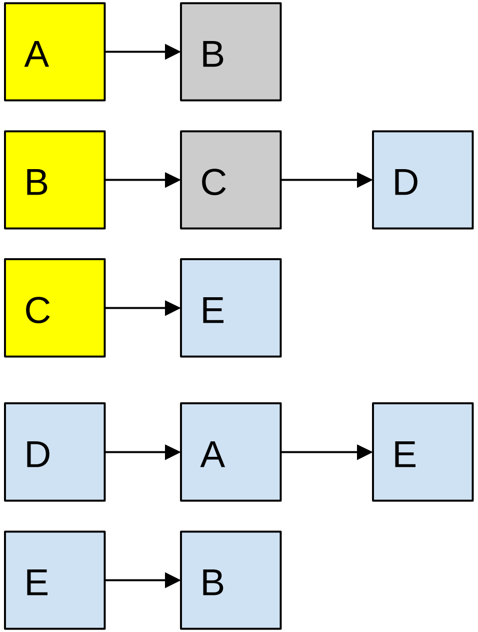
- **...E o colocamos na pilha. Marcando-o como visitado.**



Grafo



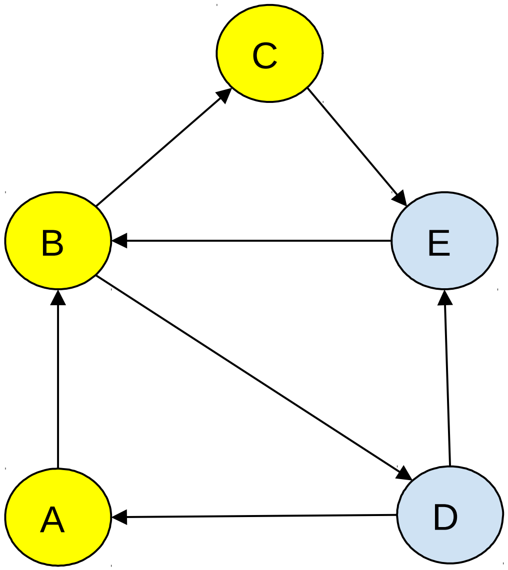
Pilha



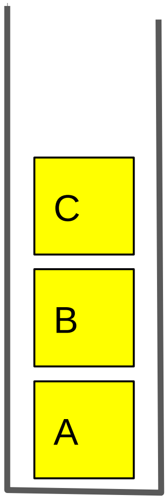
Lista de Adjacências

Busca em Profundidade

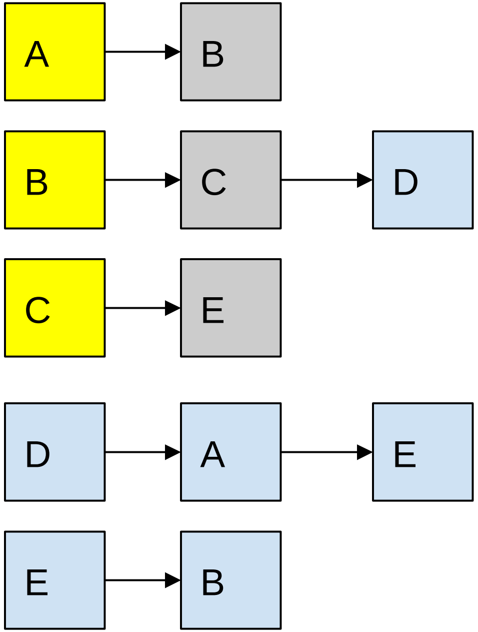
- **Olhamos o item no topo da pilha e pegamos a letra mais baixa próxima a ele que ainda NÃO foi visitada...**



Grafo



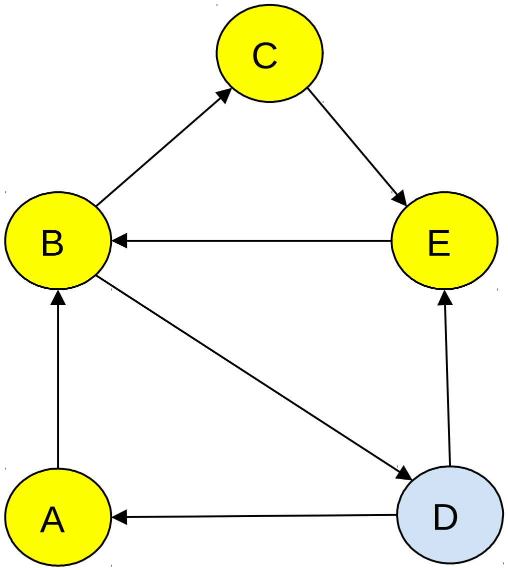
Pilha



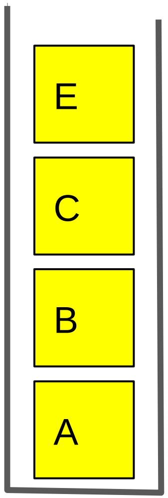
Lista de Adjacências

Busca em Profundidade

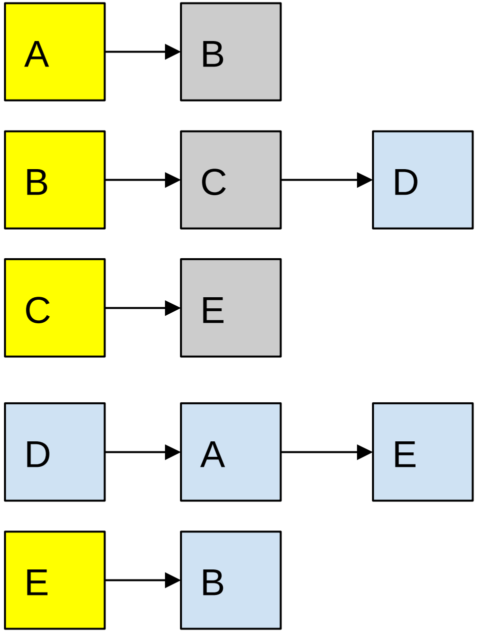
- **...E o colocamos na pilha. Marcando-o como visitado.**



Grafo



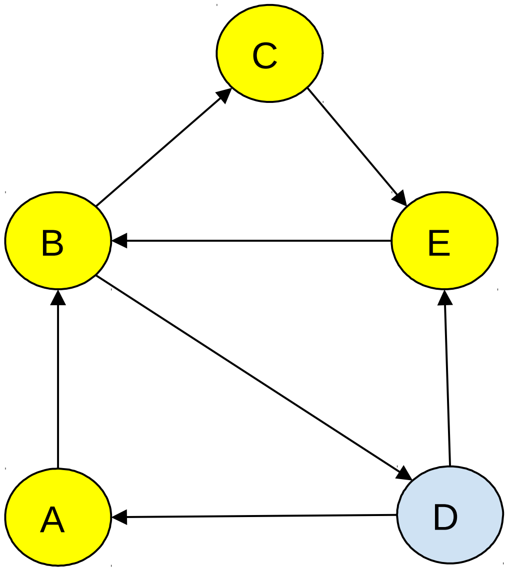
Pilha



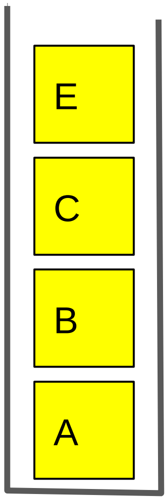
Lista de Adjacências

Busca em Profundidade

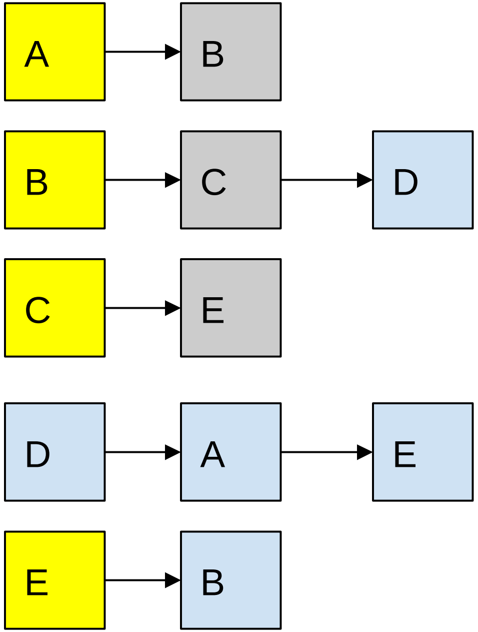
- **Olhamos o item no topo da pilha e pegamos a letra mais baixa próxima a ele que ainda NÃO foi visitada...**



Grafo



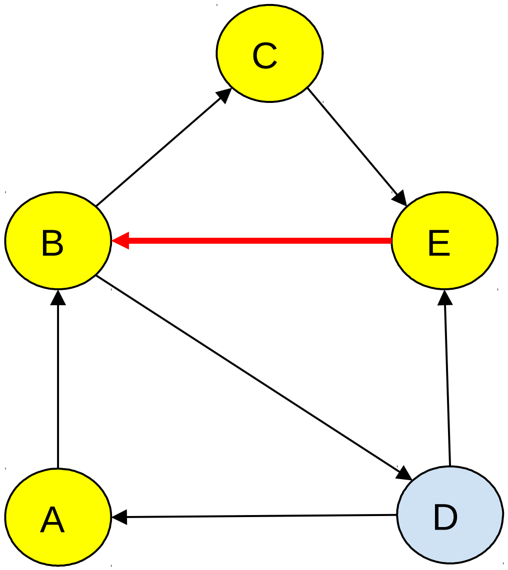
Pilha



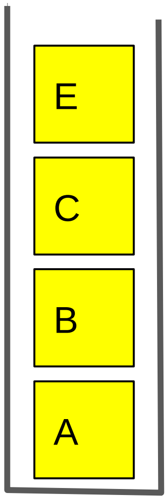
Lista de Adjacências

Busca em Profundidade

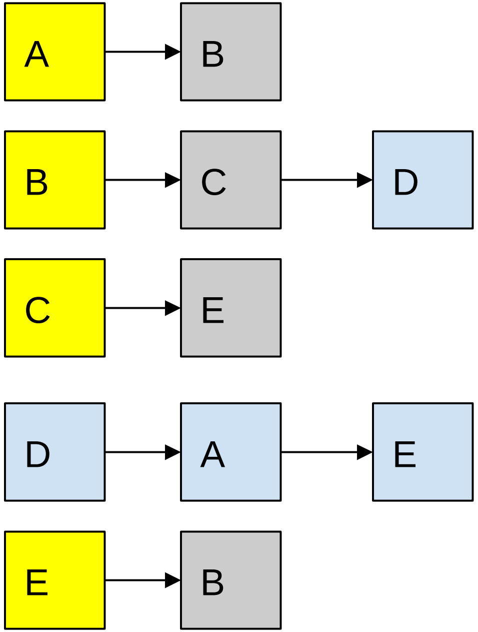
- **Observamos que a única ligação de “E” é “B”, o qual já foi visitado. Vamos retirar “E” do topo da pilha então.**



Grafo



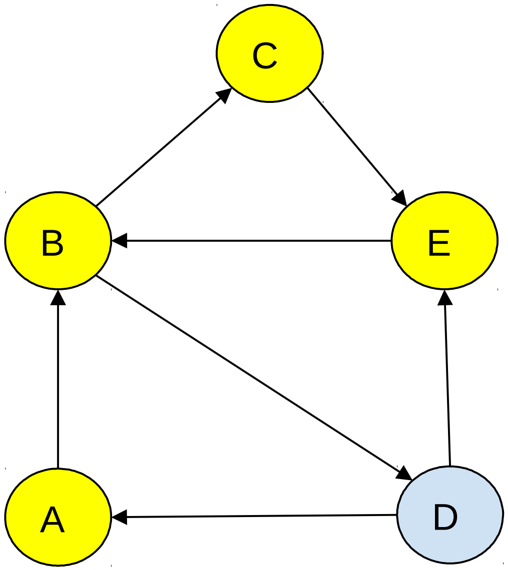
Pilha



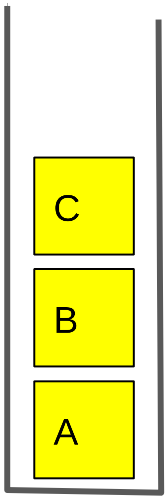
Lista de Adjacências

Busca em Profundidade

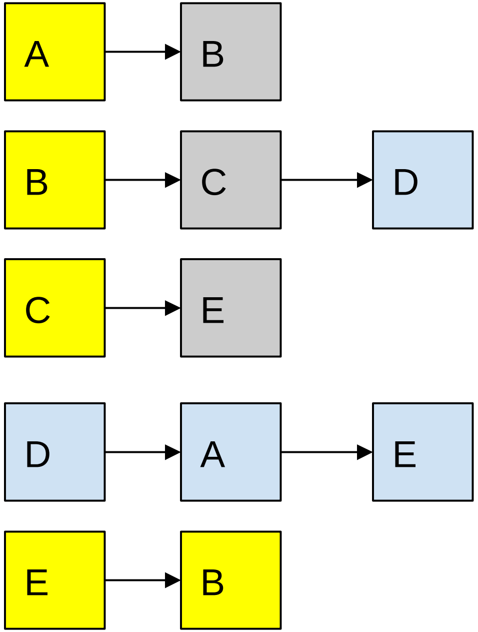
- **Retirando “E”... Estamos iniciando o chamado “Backtracking”.**



Grafo



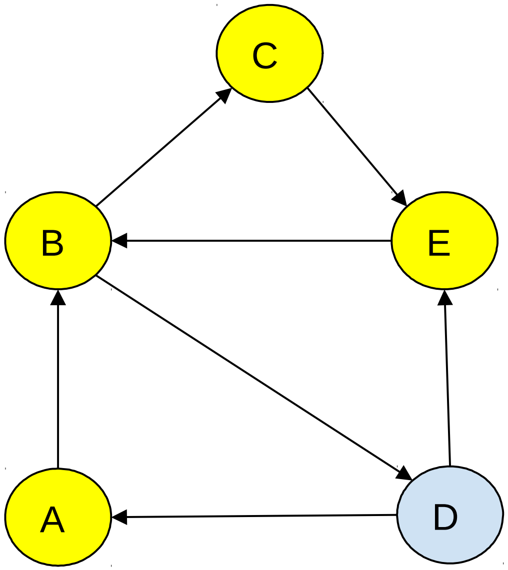
Pilha



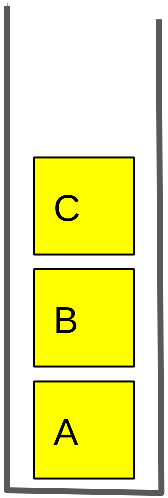
Lista de Adjacências

Busca em Profundidade

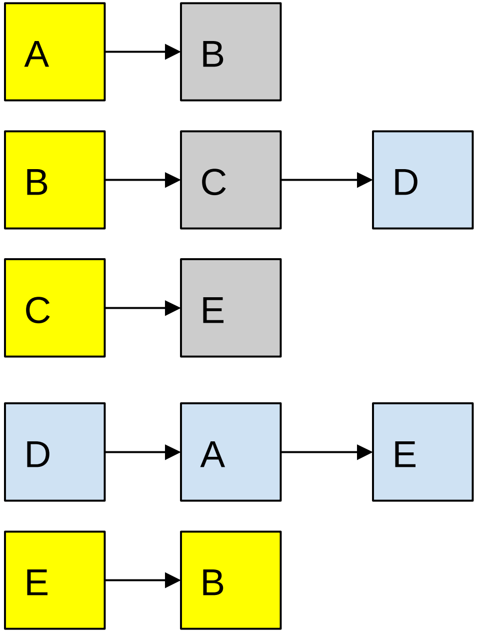
- **Olhamos o item no topo da pilha e pegamos a letra mais baixa próxima a ele que ainda NÃO foi visitada...**



Grafo



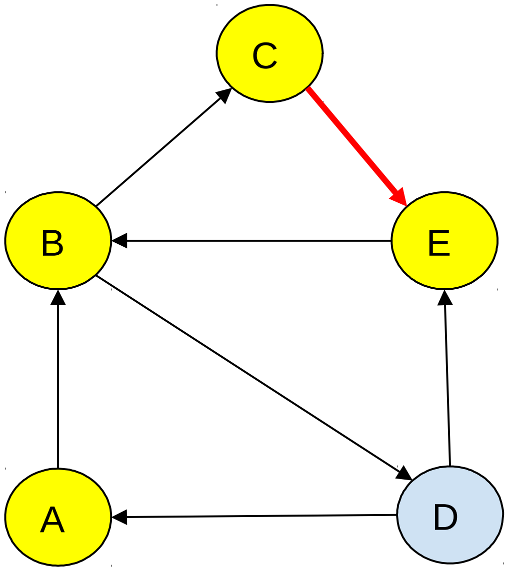
Pilha



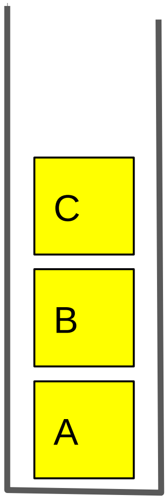
Lista de Adjacências

Busca em Profundidade

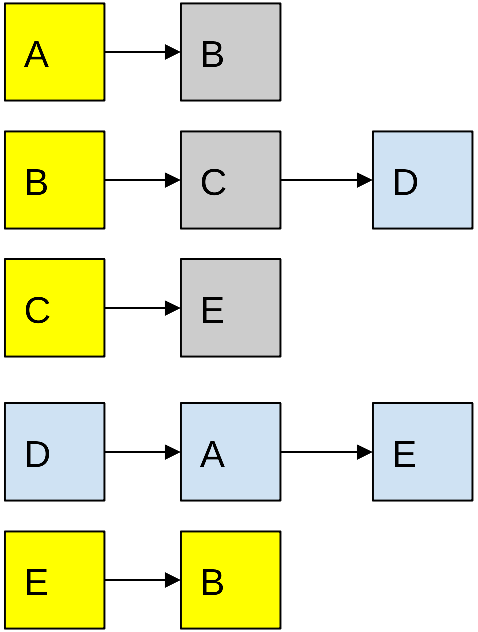
- **Observamos que a única ligação de “C” é “E”, o qual já foi visitado. Vamos retirar “C” do topo da pilha então.**



Grafo



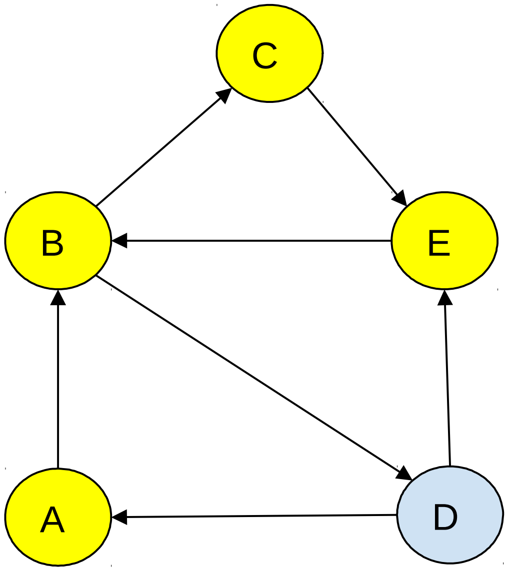
Pilha



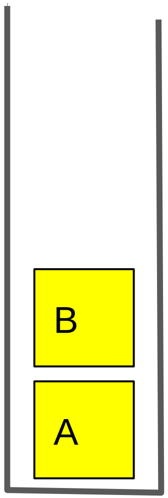
Lista de Adjacências

Busca em Profundidade

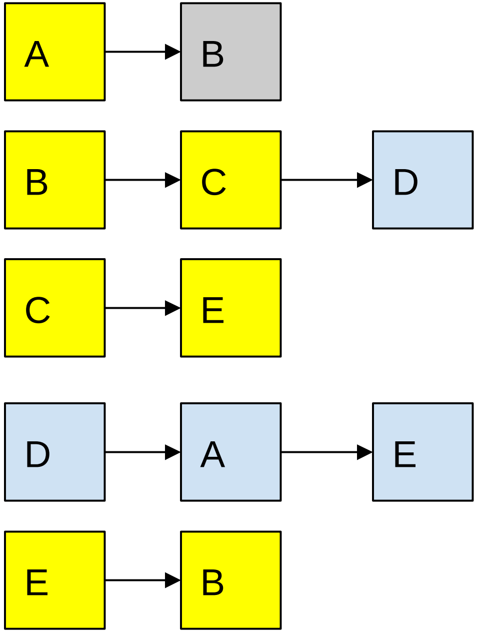
- **Retirando “C” e mais Backtracking...**



Grafo



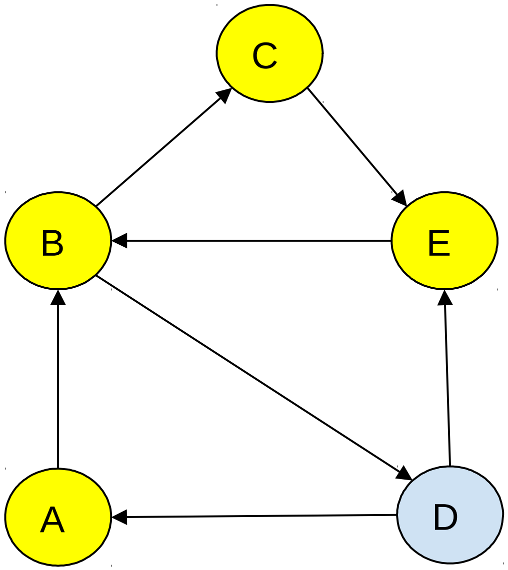
Pilha



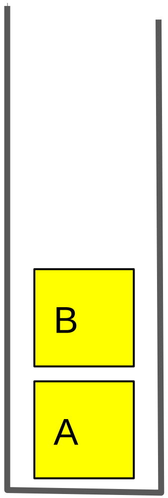
Lista de Adjacências

Busca em Profundidade

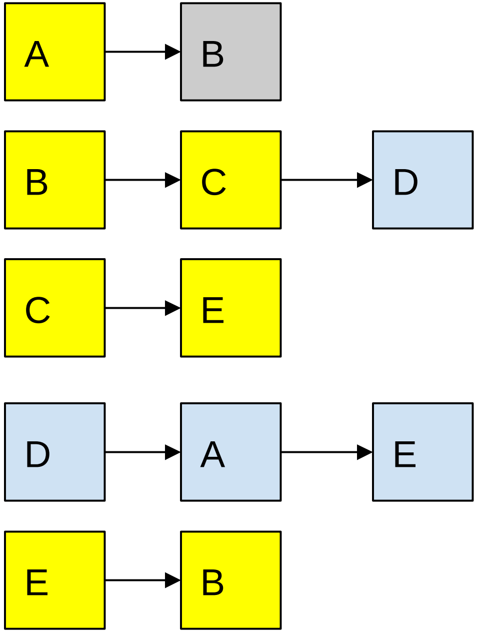
- **Olhamos o item no topo da pilha e pegamos a letra mais baixa próxima a ele que ainda NÃO foi visitada...**



Grafo



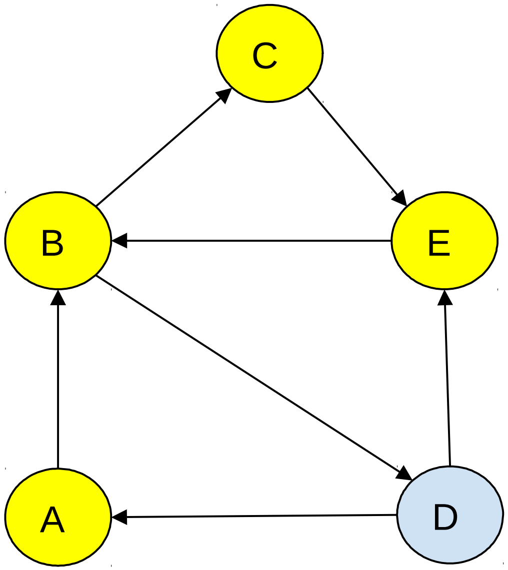
Pilha



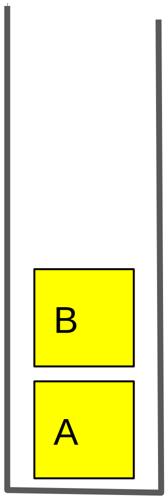
Lista de Adjacências

Busca em Profundidade

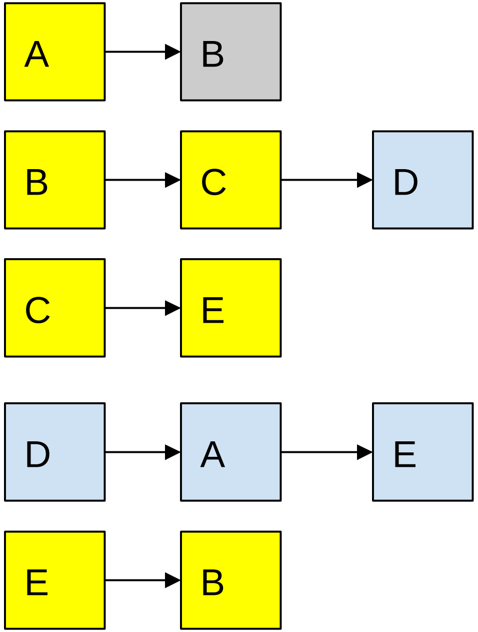
- **“C” já foi visitado, mas “D” ainda não foi.**



Grafo



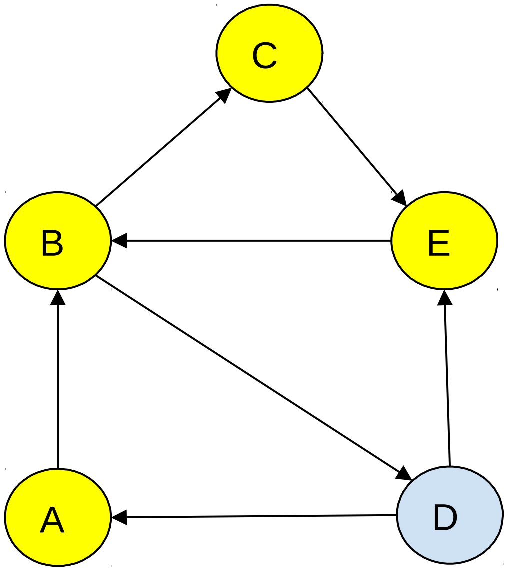
Pilha



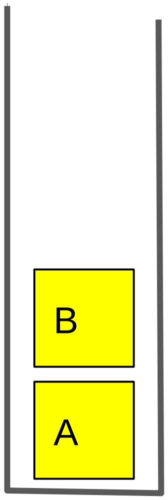
Lista de Adjacências

Busca em Profundidade

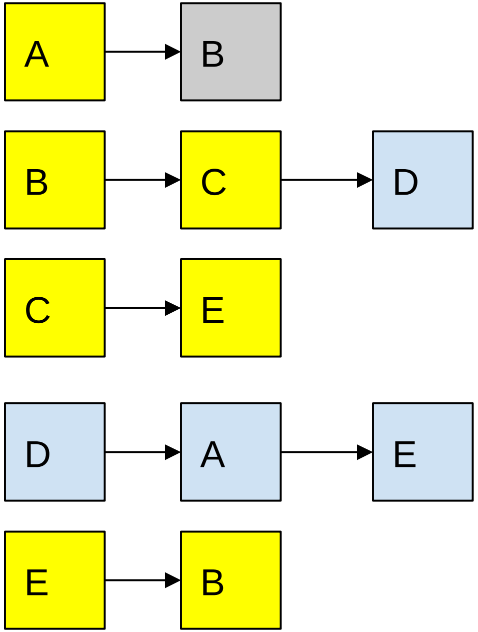
- **Olhamos o item no topo da pilha e pegamos a letra mais baixa próxima a ele que ainda NÃO foi visitada… “C” já foi visitado, mas “D” ainda não foi.**



Grafo



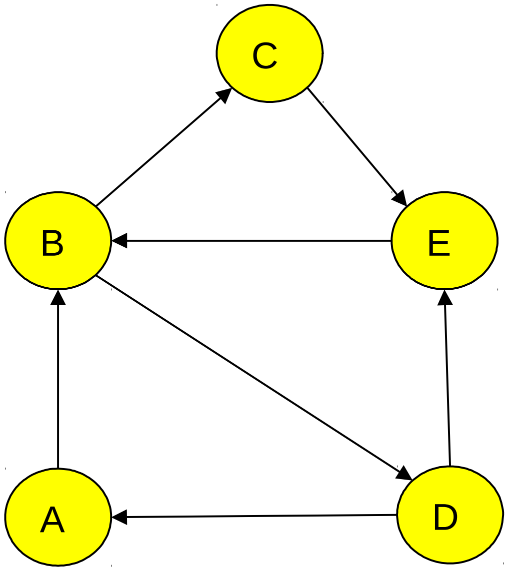
Pilha



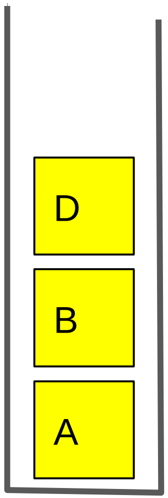
Lista de Adjacências

Busca em Profundidade

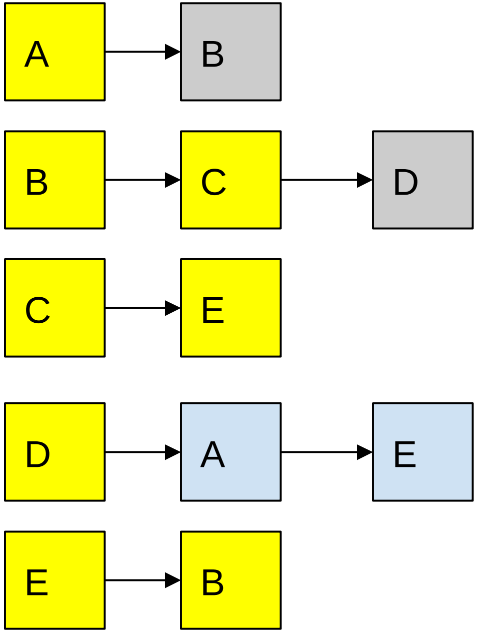
- **...E o colocamos na pilha. Marcando-o como visitado.**



Grafo



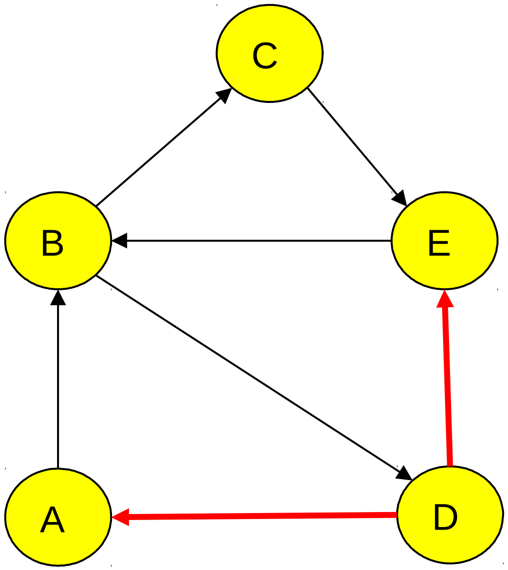
Pilha



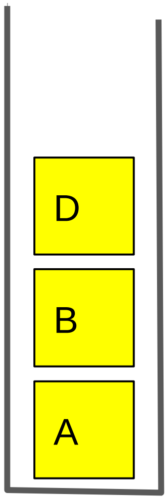
Lista de Adjacências

Busca em Profundidade

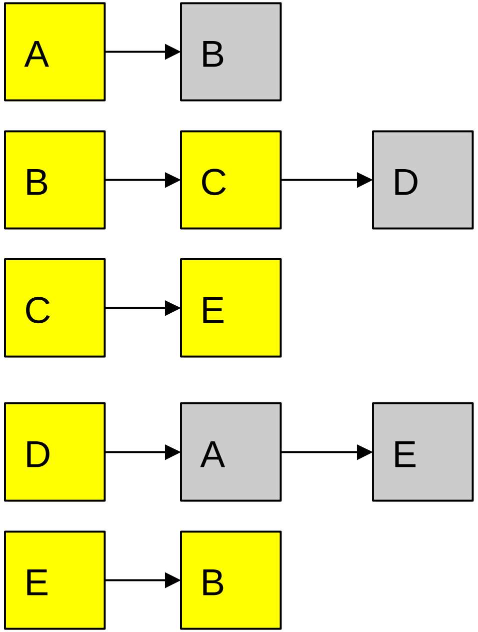
- **Olhamos o item no topo da pilha e pegamos a letra mais baixa próxima a ele que ainda NÃO foi visitada… “A” já foi visitado, “E” também já foi..**



Grafo



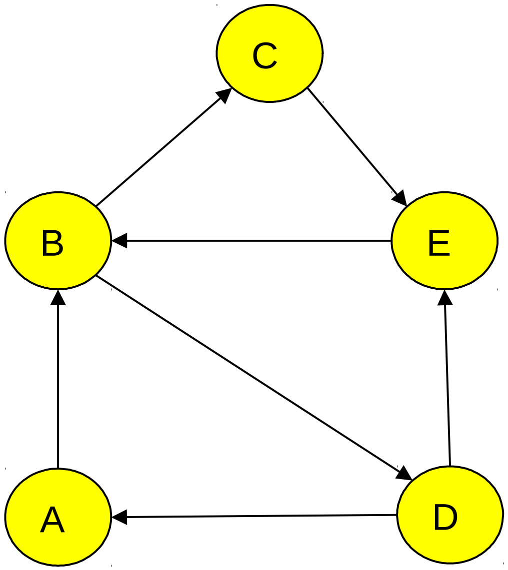
Pilha



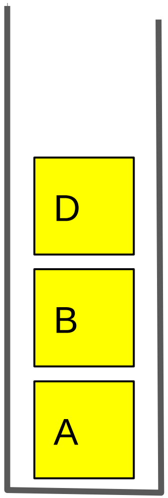
Lista de Adjacências

Busca em Profundidade

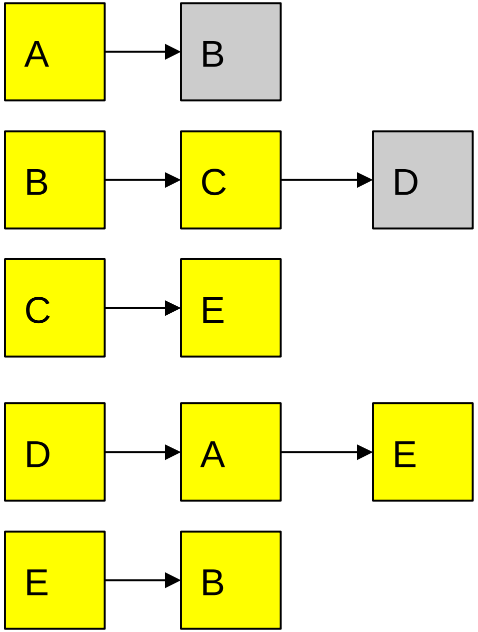
- **Observamos que não há mais ligações em “D”. Vamos retirar “D” do topo da pilha então.**



Grafo



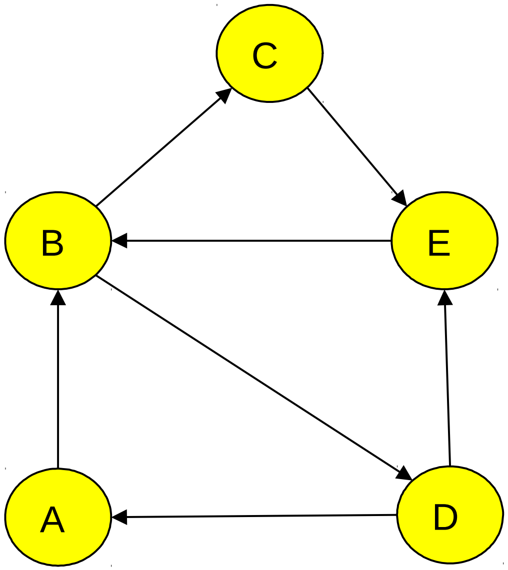
Pilha



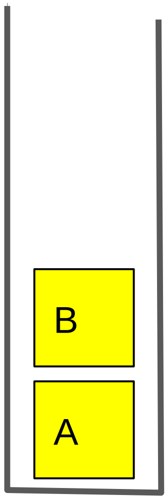
Lista de Adjacências

Busca em Profundidade

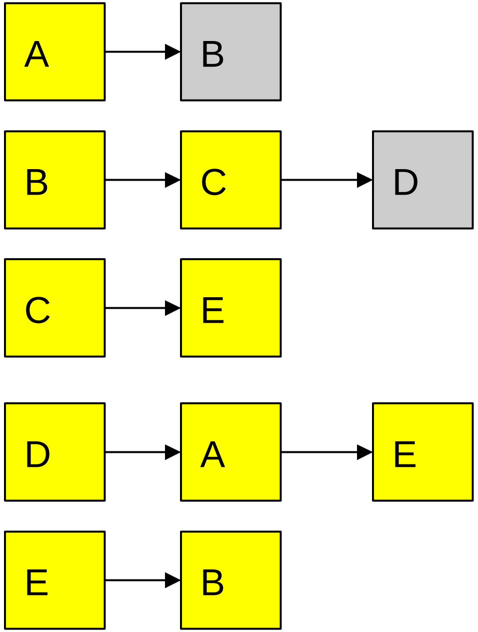
- **Retirando “D” e mais Backtracking...**



Grafo



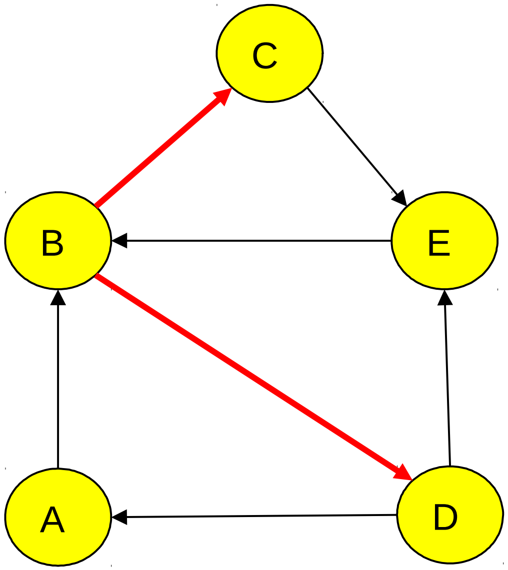
Pilha



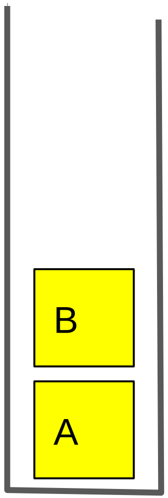
Lista de Adjacências

Busca em Profundidade

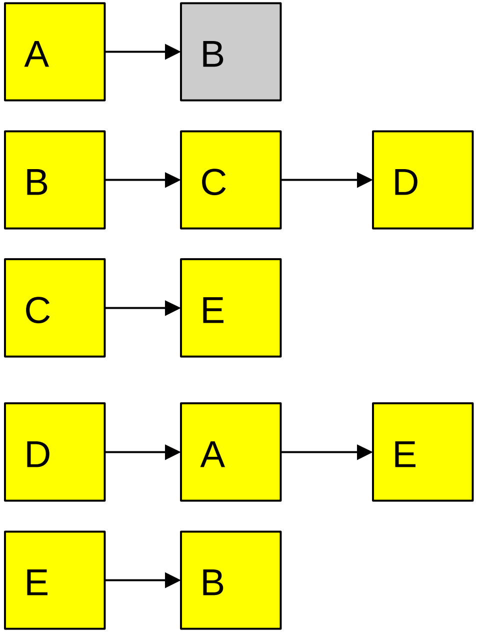
- **Olhamos o item no topo da pilha e pegamos a letra mais baixa próxima a ele que ainda NÃO foi visitada… “C” e “D” já foram visitados.**



Grafo



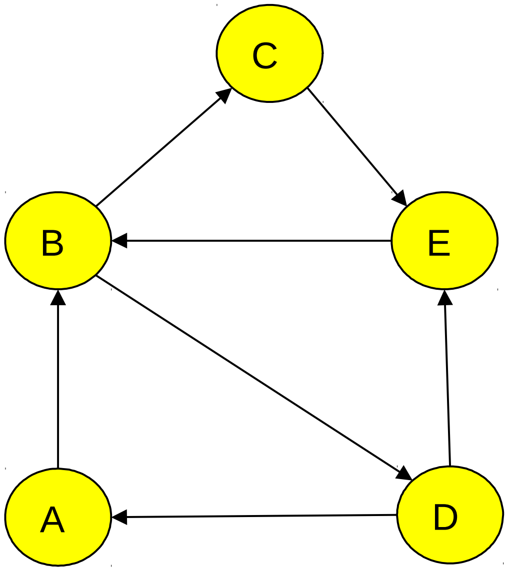
Pilha



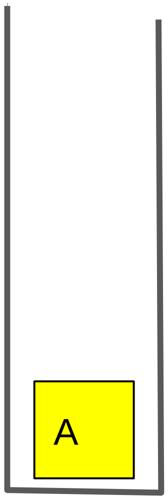
Lista de Adjacências

Busca em Profundidade

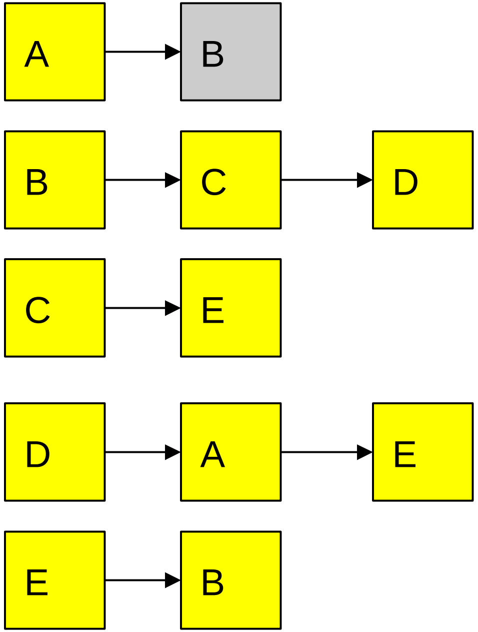
- **Retirando “B” e mais Backtracking...**



Grafo



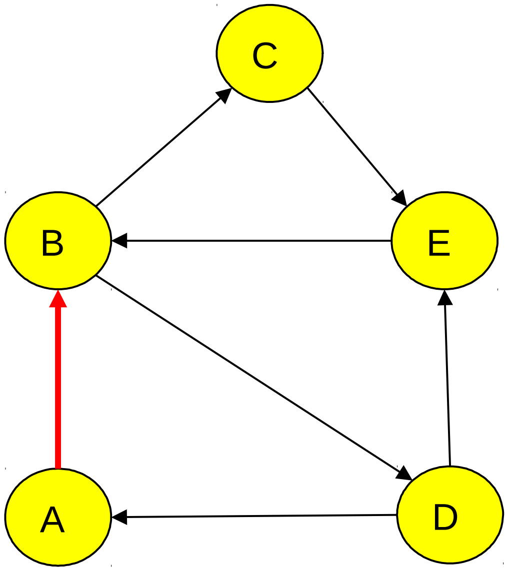
Pilha



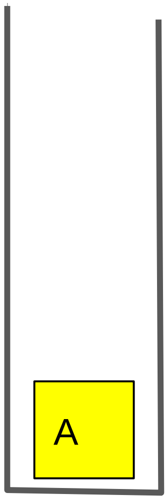
Lista de Adjacências

Busca em Profundidade

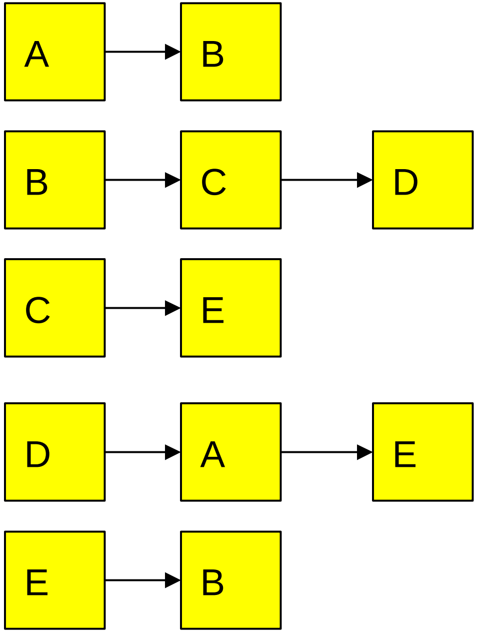
- **Olhamos o item no topo da pilha e pegamos a letra mais baixa próxima a ele que ainda NÃO foi visitada… “B” já foi visitado.**



Grafo



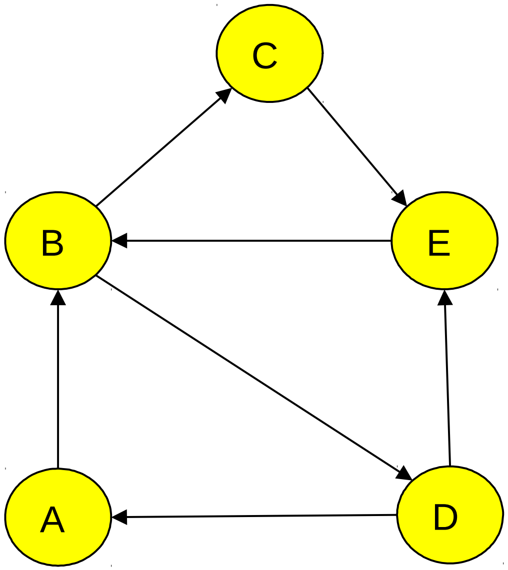
Pilha



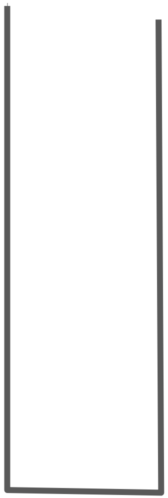
Lista de Adjacências

Busca em Profundidade

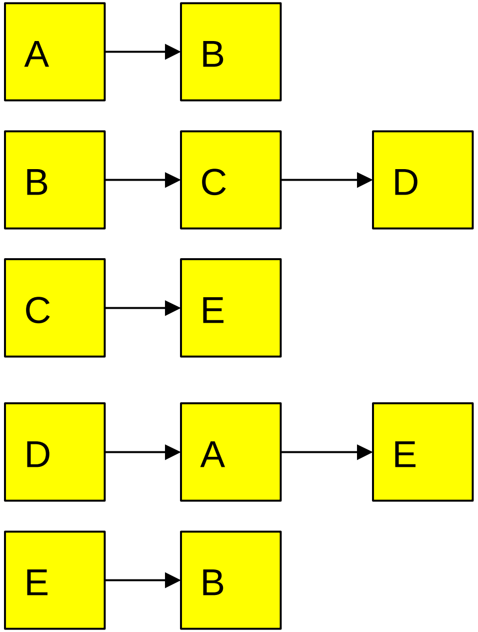
- **Removemos “A” e fim do passeio.**



Grafo



Pilha



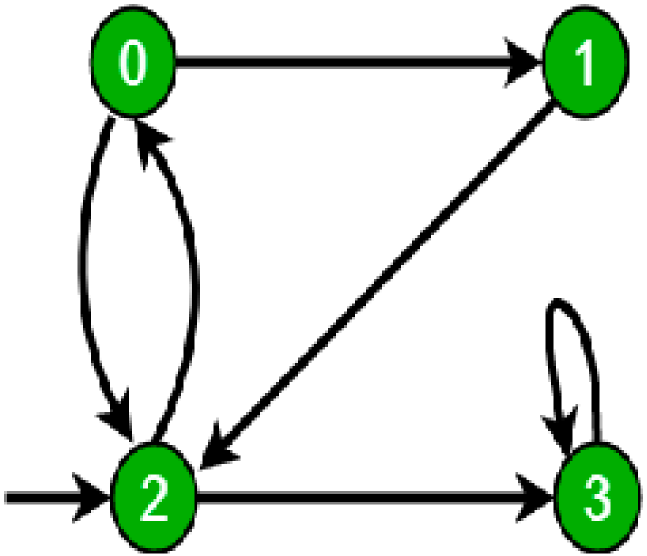
Lista de Adjacências

Busca em Profundidade

- Qual foi o caminho percorrido?

Parte Prática

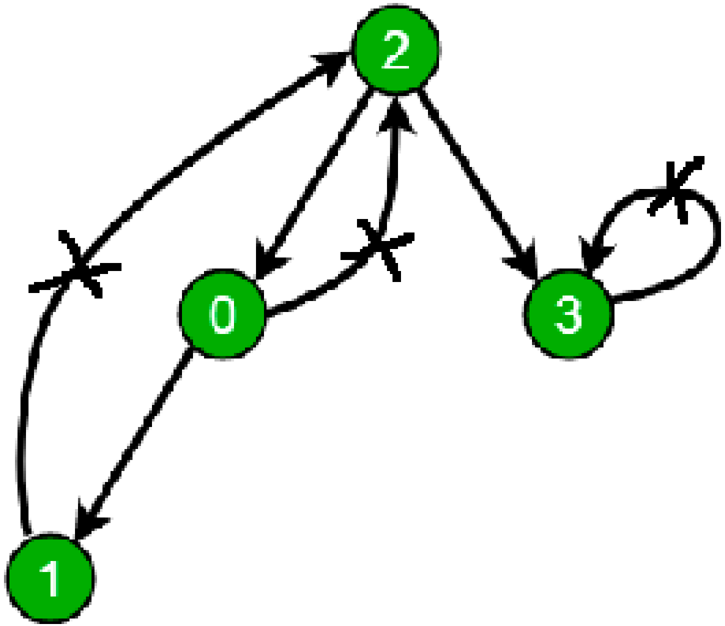
- Já implementamos o grafo anteriormente...
- Vamos implementar a Busca em Profundidade do Grafo abaixo?
- **Obs:** Precisamos de um array para marcar os vértices visitados.




```
adicionaAresta( 0, 1);
adicionaAresta( 0, 2);
adicionaAresta( 1, 2);
adicionaAresta( 2, 0);
adicionaAresta( 2, 3);
adicionaAresta( 3, 3);
```

Qual o caminho percorrido?

2 => 0 => 1 => 3



Perguntas

- Qual tipo de problemas podem ter o apoio de uma busca em profundidade?
  - Detectar ciclos em um grafo;
  - Encontrar caminhos (É possível, saindo do vértice X, chegarmos no vértice Y?)
  - Descobrir se um grafo é fortemente conexo
    - Um grafo fortemente conexo é aquele onde existe um caminho de todos os vértices um ao outro.
  - Resolver problemas de labirintos de solução única
  - ...E o problema de Euler em Königsberg, claro!


Thank You!