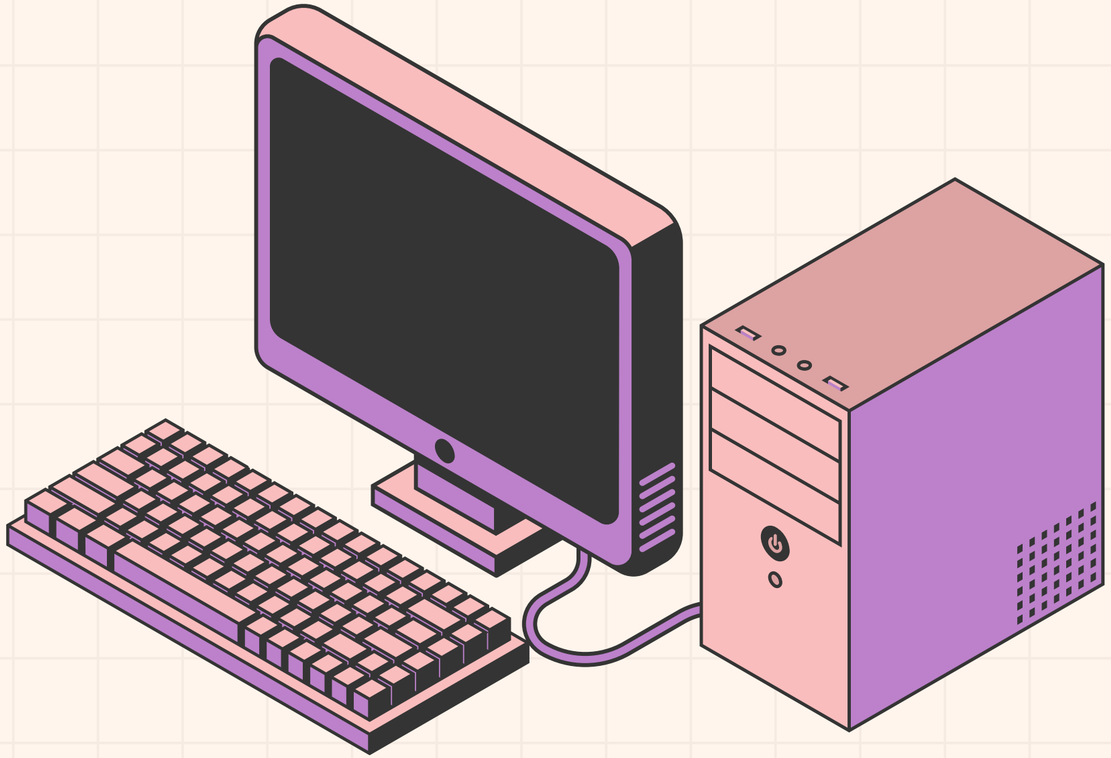
GRAFOS

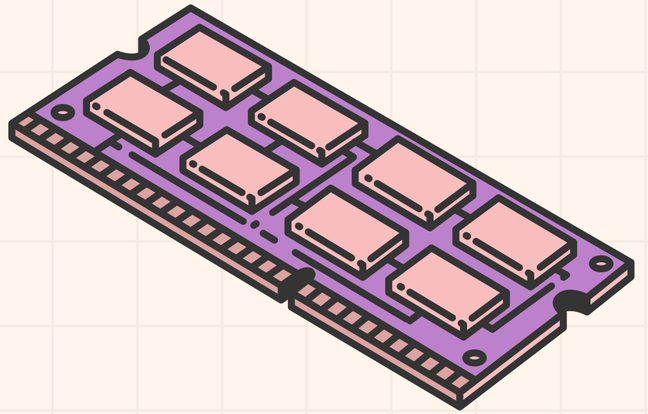
BUSCA EM PROFUNDIDADE

Autor: Lucas Santos (CJ3032116)



INSTITUTO FEDERAL
SÃO PAULO
Campus Campos do Jordão

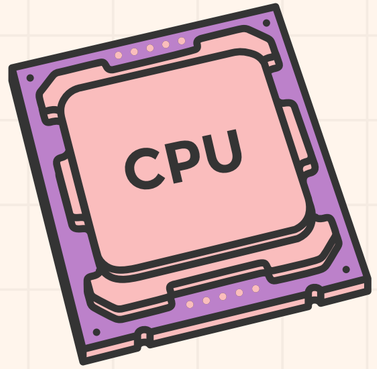
O QUE É A BUSCA EM PROFUNDIDADE?



A Busca em Profundidade (DFS - Depth-First Search) é um algoritmo de travessia de grafos que explora o grafo "profundamente" antes de voltar atrás e explorar outros caminhos. Ou seja, a DFS começa em um vértice e explora o máximo possível ao longo de um caminho antes de retroceder e explorar caminhos alternativos.

Segue a lógica de procurar um objeto em algum ambiente, geralmente começamos a procurar em um ponto inicial e parte para o próximo local mais próximo do que estamos, se o objeto não for encontrado passa para o proximo local, se mesmo procurando em todos os lugares nesse ambiente, normalmente voltamos para o ponto inicial e procuramos em outro ambiente até que o objeto seja encontrado (ou não).

PASSO A PASSO



Escolher um Vértice de Partida: Começa a partir de um vértice inicial. Esse vértice é chamado de origem ou inicial.

Visitar o Vértice: O vértice atual é marcado como visitado. O objetivo é garantir que o algoritmo não visite o mesmo vértice mais de uma vez.

Explorar os Vizinhos: O algoritmo então examina os vizinhos do vértice atual. Para cada vizinho que ainda não foi visitado, o algoritmo faz uma chamada recursiva para visitar o vizinho.

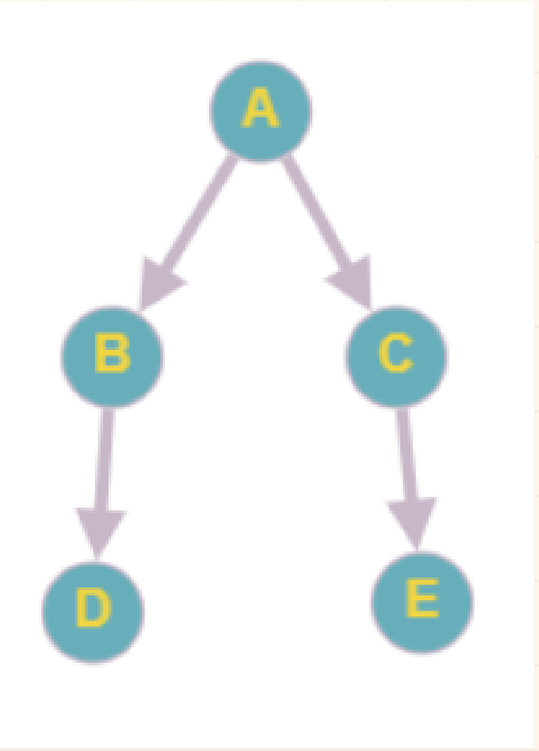
Retroceder Quando Não Houver Mais Vizinhos Não Visitados: Quando um vértice não tem mais vizinhos não visitados ou quando todos os seus vizinhos já foram visitados, o algoritmo retorna para o vértice anterior na pilha de chamadas recursivas (volta para o vértice pai).

Repetir Até Todos os Vértices: O processo continua até que todos os vértices acessíveis a partir do vértice inicial sejam visitados.

Começar em A: O vértice A é visitado e marcado.

Explorar os Vizinhos de A: A DFS explora os vizinhos de A, que são B e C. Suponha que a DFS escolha explorar B primeiro.

Ir para B: O vértice B é visitado e marcado. O algoritmo então explora os vizinhos de B. O único vizinho não visitado de B é D, então a DFS vai para D.



Ir para D: O vértice D é visitado. Como D não tem vizinhos não visitados, a DFS volta para B.

Voltar para B: Não há mais vizinhos não visitados de B, então a DFS volta para A.

Explorar C: Agora a DFS explora o vizinho C de A. C é visitado e o algoritmo explora seus vizinhos.

Ir para E: O único vizinho não visitado de C é E, então a DFS vai para E.

Ir para E: O vértice E é visitado. Como E não tem vizinhos não visitados, a DFS volta para C, e depois volta para A.

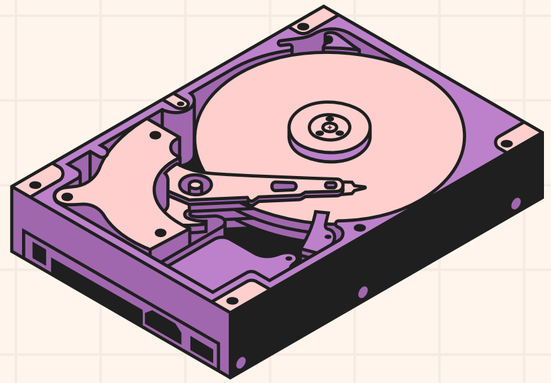
Conclusão: Todos os vértices acessíveis foram visitados. A DFS termina.

APLICAÇÕES DA DFS

Detecção de Ciclos: DFS pode ser usada para detectar ciclos em grafos direcionados e não direcionados.

Busca de Caminho: Pode ser utilizada para verificar se há um caminho entre dois vértices.

Componentes Conexos: Em grafos não direcionados, a DFS pode ajudar a encontrar componentes conexos, ou seja, subgrafos onde todos os vértices estão conectados.



EXEMPLO NA PRÁTICA

```
Grafo:
0 -> 1, 2
1 -> 2, 3
2 -> 0, 5
3 -> 2, 4, 5
4 -> 1, 3
5 -> 2, 3
```

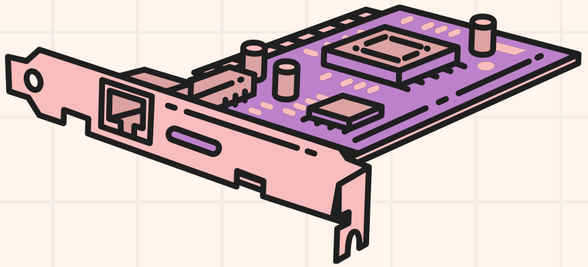
Consideremos a lista de adjacências grafo ao lado:
Vamos usar o algoritmo para buscar um caminho entre o vértice 0 e o vértice 4.

Passo 1: Início da Busca

- Início da busca a partir do vértice 0.
- O vértice 0 é marcado como visitado: visitado[0] = true.
- Fila de Processados: [0]
- Os vizinhos de 0 são 1 e 2. O algoritmo escolhe explorar o vértice 1 primeiro.

Passo 2: Explorando o Vértice 1

- O algoritmo visita o vértice 1.
- O vértice 1 é marcado como visitado: visitado[1] = true.
- Fila de Processados: [0, 1]
- Os vizinhos de 1 são 2 e 3. O algoritmo escolhe explorar o vértice 2 primeiro.



EXEMPLO NA PRÁTICA

```
Grafo:
0 -> 1, 2
1 -> 2, 3
2 -> 0, 5
3 -> 2, 4, 5
4 -> 1, 3
5 -> 2, 3
```

Passo 3: Explorando o Vértice 2

- O algoritmo visita o vértice 2.
- O vértice 2 é marcado como visitado: visitado[2] = true.
- Fila de Processados: [0, 1, 2]
- Os vizinhos de 2 são 0 (já visitado) e 5. O algoritmo escolhe explorar o vértice 5.

Passo 4: Explorando o Vértice 5

- O algoritmo visita o vértice 5.
- O vértice 5 é marcado como visitado: visitado[5] = true.
- Fila de Processados: [0, 1, 2, 5]
- Os vizinhos de 5 são 2 (já visitado) e 3. O algoritmo escolhe explorar o vértice 3.

Passo 5: Explorando o Vértice 3

- O algoritmo visita o vértice 3.
- O vértice 3 é marcado como visitado: visitado[3] = true.
- Fila de Processados: [0, 1, 2, 5, 3]
- Os vizinhos de 3 são 2 (já visitado), 4 e 5 (já visitado). O algoritmo escolhe explorar o vértice 4.

EXEMPLO NA PRÁTICA

```
Grafo:
0 -> 1, 2
1 -> 2, 3
2 -> 0, 5
3 -> 2, 4, 5
4 -> 1, 3
5 -> 2, 3
```

Passo 6: Encontrando o Destino (Vértice 4)

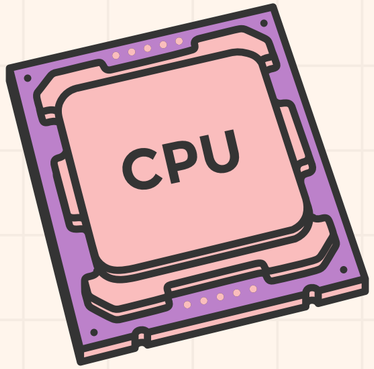
- O algoritmo visita o vértice 4, que é o destino.
- O vértice 4 é marcado como visitado: visitado[4] = true.
- Fila de Processados: [0, 1, 2, 5, 3, 4]
- Como o vértice 4 é o destino, a função retorna true imediatamente.

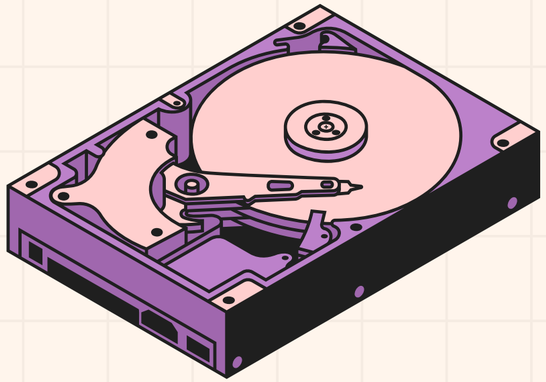
Resultado Final:

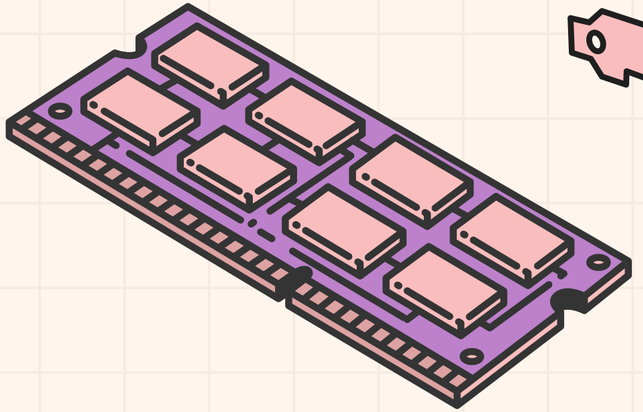
- O algoritmo encontrou o caminho do vértice 0 até o vértice 4, passando pelos vértices 0 → 1 → 2 → 5 → 3 → 4.
- O algoritmo retorna "Caminho Existe!".

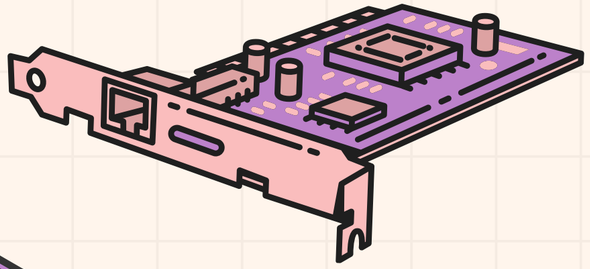
IMPLEMENTAÇÃO RECURSIVA

```
1  bool buscaProfundidadeRec(No **grafo, int atual, int destino, bool *visitado) // retorna true se caminho encontrado
2  {
3      visitado[atual] = true; // marca o vértice atual como visitado
4      if (atual == destino)   // verifica se o vértice atual é o destino // caso base da recursão
5          return true;        // retorna true imediatamente se destino alcançado
6
7      No *vizinho = grafo[atual]; // obtém a cabeça da lista de adjacência do vértice atual
8      while (vizinho != nullptr)  // percorre cada vizinho do vértice atual
9      {
10         if (!visitado[vizinho->vertice]) // se o vizinho ainda não foi visitado
11         {
12             if (buscaProfundidadeRec(grafo, vizinho->vertice, destino, visitado)) // chamada recursiva para o vizinho
13                 return true; // se a chamada recursiva encontrar o destino, propaga true para cima
14         }
15         vizinho = vizinho->proximo; // avança para o próximo vizinho na lista
16     }
17
18     return false; // se nenhum vizinho levou ao destino, retorna false
19 }
```









OBRIGADO :)

Autor: Lucas Santos (CJ3032116)

REFERÊNCIA

UNIVESP. Estrutura de Dados - Aula 26 - Grafos - Busca em profundidade. Disponível em: <https://www.youtube.com/watch?v=doH9o1sO-Cw>. Acesso em: 24 out. 2025.

Ferramenta de desenho de grafo: https://graphonline.top/pt/

INSTITUTO FEDERAL
SÃO PAULO
Campus Campos do Jordão