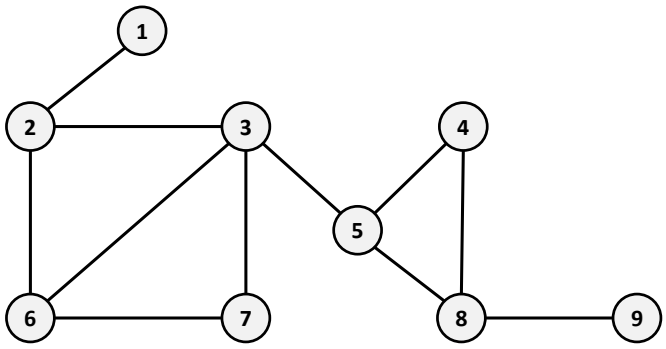
Universidade do Estado de Santa Catarina (UDESC)
Curso: Tecnologia em Análise e Desenvolvimento de Sistemas
Disciplina: Estrutura de dados (EDA0001)
Prof. Rui Jorge Tramontin Jr.

Trabalho 1: Implementando DFS com uma Pilha

Grafos são estruturas matemáticas utilizadas para modelar problemas em diversas áreas. Na Ciência da Computação, são aplicados em diversos problemas nas áreas de redes da computadores, mídias sociais, organização de informação, modelagem de circuitos, entre outras aplicações.

Grafos são definidos na forma *G(V, A)*, onde *V* é um conjunto não vazio de *vértices* (ou *nós*) e *A* é um conjunto de *arestas*. Uma aresta representa a conexão entre dois vértices. A figura a seguir apresenta um exemplo de grafo com 9 vértices.

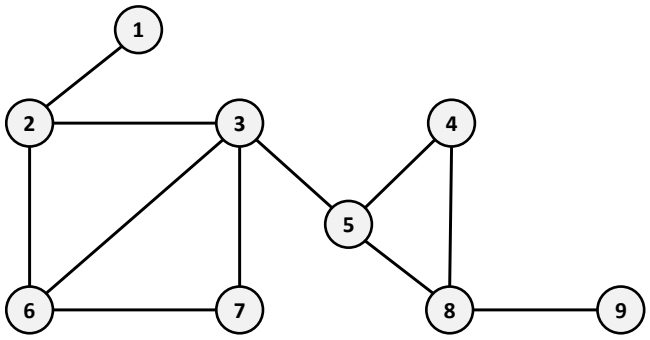


O objetivo deste trabalho é implementar o algoritmo DFS (*Depth-first Search*) ou ***Busca em Profundidade*** em um grafo. É um algoritmo usado para realizar uma busca ou travessia num grafo. Esse tipo de busca progride através da expansão do primeiro vértice (estabelecido como entrada), e se aprofunda cada vez mais, até que o alvo da busca seja encontrado ou até que não haja mais vértices a serem visitados. Então a busca retrocede (*backtrack*) e começa no próximo vértice. Numa implementação não-recursiva, todos os vértices expandidos recentemente são adicionados a uma **pilha**, para realizar a exploração.

O objetivo do algoritmo neste trabalho não é implementar uma busca em si, mas sim fazer a **travessia pelo grafo**, mostrando na tela todos os vétices encontrados. Esse tipo de algoritmo tem aplicação em problemas relacionados a encontrar os componentes conectados em um grafo ou resolução de quebra-cabeças, tal como um labirinto.

Implementação

Um grafo pode ser representado por uma matriz de adjacência, na qual cada linha representa um vértice e cada vértice adjacente a ele (coluna) possui valor 1, ou 0 para os que não são adjacentes (adjacente aqui signica estar conectado). Considerando o grafo apresentado como exemplo, a matriz de adjacência que o representa seria a seguinte:



	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	0	0	0	0
2	1	0	1	0	0	1	0	0	0
3	0	1	0	0	1	1	1	0	0
4	0	0	0	0	1	0	0	1	0
5	0	0	1	1	0	0	0	1	0
6	0	1	1	0	0	0	1	0	0
7	0	0	1	0	0	1	0	0	0
8	0	0	0	1	1	0	0	0	1
9	0	0	0	0	0	0	0	1	0

A matriz pode ser facilmente implementada utilizando a biblioteca de matrizes dinâmicas feita como exercício em aula (Lista 2). Repare que, para o usuário, os vértices são numerados a partir de 1, enquanto que na linguagem C, os índices começam em 0.

Adicionalmente, o algoritmo necessita de um vetor de inteiros, que marca o *status* de cada vértice, indicando se o mesmo já foi visitado (1), ou se ainda não foi (0).

Temos, portanto, três estruturas necessárias para o algorirmo:

1. Matriz de adjacência: alocada dinamicamante, conforme entrada do usuário;
2. Vetor de *status* dos vértices: também alocado dinamicamente; indica se cada vértice foi ou não visitado;
3. Pilha de inteiros: bilioteca de pilhas usando vetor dinâmico.

O algoritmo é implementado conforme o seguinte pseudocódigo:

```
 1. ENTRADA DE DADOS:
     - Matriz de adjacência (sua dimensão e seu conteúdo);
     - Vértice inicial da busca (INI);
 2. INICIALIZAÇÃO DO VETOR DE STATUS (VS): todos os índices com zero;
 3. INICIALIZAÇÃO DA PILHA P;
 4. EMPILHA INI em P;              // Empilha para começar o algoritmo.
 5. ENQUANTO P NÃO ESTIVER VAZIA FAÇA
 6.    DESEMPILHA O VÉRTICE X DE P;   // Desempilha vértice.
 7.    SE VS[X] = 0 ENTÃO             // Se ainda não foi visitado...
 8.       MOSTRA X NA TELA;           // mostra vértice...
 9.       VS[X] <- 1;                 // e marca como visitado.
10.       PARA CADA VÉRTICE I ADJACENTE A X FAÇA
11.          SE VS[I] = 0 ENTÃO       // Se ainda não foi visitado...
12.             EMPILHA I EM P;       // empilha para análise futura.
13.          FIMSE
14.       FIMPARA
15.    FIMSE
16. FIMENQUANTO
```

Entrada

A primeira linha da entrada contém um inteiro *N*, que representa o número de vértices do grafo. A matriz de adjacência terá, portanto, dimensão *N*. Em seguida, são lidas cada uma das *N* linhas da matriz de adjacência. Finalmente, temos como entrada o vértice inicial.

Saída

A lista de vértices visitados, um em cada linha.

Exemplo

Exemplo de entrada	Exemplo de saída
9 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1	1 2 6 7 3 5 8 9 4

Critérios de avaliação

- Execução correta e alinhamento com o que foi solicitado neste enunciado;
- Uso apropriado das funções dos *tipos abstratos de dados* (matriz e pilha). Respeite o encapsulamento!

Informações importantes

- **Equipe:** 1 ou 2 alunos.
- **Entrega via Moodle.**