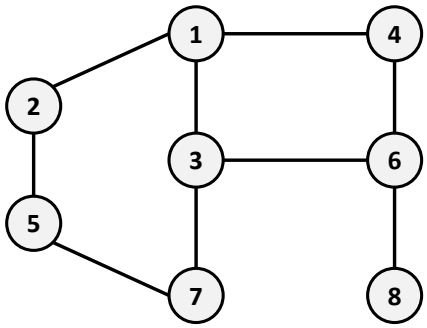
Universidade do Estado de Santa Catarina (UDESC)
Curso: Tecnologia em Análise e Desenvolvimento de Sistemas
Disciplina: Estrutura de dados (EDA0001)
Prof. Rui Jorge Tramontin Jr.

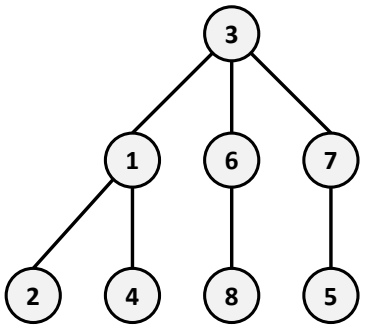
Trabalho 4: Árvore de Distâncias Mínimas

O objetivo deste trabalho é implementar o algoritmo BFS (*Breadth-first Search*) ou ***Busca em Largura*** em um grafo. Tal como a busca em profundidade, é um algoritmo usado para realizar uma busca ou travessia num grafo. Ele se inicia em algum vértice arbitrário do grafo e explora todos os vértices vizinhos (no mesmo nível), antes de se mover para os vértices no próximo nível de profundidade. Para isso, a implementação desse algoritmo utiliza uma **fila**.

O objetivo do algoritmo neste trabalho é, dado o vértice inicial A, realizar a **travessia em largura no grafo**. O objetivo é gerar uma **Árvore de Distâncias Mínimas** para o vértice A. Tal estrutura permite determinar **menor caminho entre A e os demais vértices**, em termos de número de arestas percorridas. Nesse sentido, após a aplicação da busca, a saída do algoritmo deve ser a **árvore gerada**. Um exemplo é apresentado na figura a seguir.



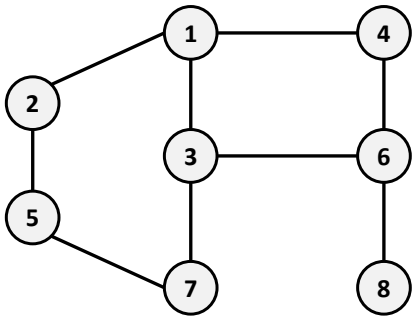
Grafo



Árvore de Distâncias Mínimas para o vértice 3

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 significa estar conectado). Por exemplo, considere o grafo a seguir e sua respectiva matriz de adjacência:



	1	2	3	4	5	6	7	8
1	0	**1**	**1**	**1**	0	0	0	0
2	**1**	0	0	0	**1**	0	0	0
3	**1**	0	0	0	0	**1**	**1**	0
4	**1**	0	0	0	0	**1**	0	0
5	0	**1**	0	0	0	0	**1**	0
6	0	0	**1**	**1**	0	0	0	**1**
7	0	0	**1**	0	**1**	0	0	0
8	0	0	0	0	0	**1**	0	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.

As seguintes estruturas são necessárias para o algoritmo:

1. Matriz de adjacência: alocada dinamicamante, conforme entrada do usuário;
2. Vetor de *status* dos vértices: também alocado dinamicamente; indica se cada vértice foi visitado (1), ou se ainda não foi (0);
3. Fila de inteiros: biblioteca de filas (vetor dinâmico);
4. Árvore n-ária: biblioteca usada na representação da árvore de distâncias mínimas.

O algoritmo é implementado conforme o seguinte pseudocódigo:

```
 1. ENTRADA DE DADOS:
     - Matriz de adjacência (sua dimensão e seu conteúdo);
     - Vértice inicial do percurso (A);
 2. INICIALIZA VETOR DE STATUS (VS): todos os índices com zero;
 3. INICIALIZA FILA F;
 4. INICIALIZA ÁRVORE T COM A; // Vértice A é o nó raiz de T
 5. VS[A] <- 1;               // Marca A como visitado,
 6. INSERE A em F;            // e insere em F para começar o algoritmo.
 7. ENQUANTO F NÃO ESTIVER VAZIA FAÇA
 8.    REMOVE O VÉRTICE X DE F;       // Remove vértice.
 9.    PARA CADA VÉRTICE I ADJACENTE A X FAÇA
10.       SE VS[I] = 0 ENTÃO              // Se ainda não foi visitado:
11.          VS[I] <- 1;                     // - marca como visitado;
12.          INSERE I EM T TENDO X COMO PAI; // - insere na árvore;
13.          INSERE I EM F;                  // - insere na fila.
14.       FIMSE
15.    FIMPARA
16. FIMENQUANTO
17. GERA XML DA ÁRVORE T;
```

A implementação desse algoritmo é similar ao do Trabalho 2, exceto pelos seguintes aspectos:

1. Não necessita do vértice final B, pois a travessia é em todo o grafo;
2. Não precisa do vetor de antecessores e nem da pilha;
3. Utiliza uma árvore n-ária para registrar a travessia.

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 A (inicial).

Saída: a representação em linguagem XML da árvore gerada.

Exemplo

Exemplo de entrada	Exemplo de saída
8 0 1 1 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 3	<3> <1> <2/> <4/> </1> <6> <8/> </6> <7> <5/> </7> </3>

Critérios de avaliação

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

Informações importantes

- **Equipe:** 1 ou 2 alunos.
- **Entrega**: no Moodle.