UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Prof. Dr. Ivan Carlos Alcântara de Oliveira
Teoria dos Grafos

Implementação

Busca em Largura e Profundidade (Grupo do Projeto)

Nome do Integrante (ordem alfabética)	RA
Pedro Padilha Farroco	10338552

Tendo por base a implementação da classe TGrafo para matriz de adjacência ou lista de adjacência e o estudo realizado sobre busca em profundidade e largura em aula, realizar a implementação dos dois algoritmos estudados na classe TGrafo. Utilizar como subsídios os métodos constantes no material da aula.

Os resultados de pelo menos dois dos seus testes devem ser colocados no relatório a seguir. Para os testes, fazer uso dos dois exemplos do material da aula.

Além disso, ao final do relatório, colocar um Apêndice com o código fonte dos métodos implementados para esta atividade.

Encaminhar o relatório e todos os arquivos com código fonte compactados ao enviar pela área do Ambiente Virtual na data limite programada.

Relatório

```
System.out.println("DFS:");
boolean[] visitedDFS = new boolean[8];
grafo.dfs(0, visitedDFS);
print(visitedDFS);
```

```
DFS:
0: visitado
1: visitado
2: visitado
3: visitado
4: visitado
5: visitado
6: visitado
7: visitado
```

```
System.out.println("\nBFS:");
boolean[] visitedBFS = new boolean[8];
ArrayList<Integer> order = grafo.bfs(0, visitedBFS);
print(visitedBFS);
System.out.println(order);
```

```
BFS:
0: visitado
1: visitado
2: visitado
3: visitado
4: visitado
5: visitado
6: visitado
7: visitado
[0, 1, 2, 4, 3, 5, 6, 7]
```

Apêndice

```
public void dfs(int start, boolean[] visited) {
    visited[start] = true;

    for (int i = 0; i < this.n; i++) {
        if (this.adj[start][i] == 1 && !visited[i]) {
            this.dfs(i, visited);
        }
    }
}
```

```
public ArrayList<Integer> bfs(int start, boolean[] visited) {
    Queue<Integer> q = new LinkedList<>();

    visited[start] = true;
    q.add(start);

    ArrayList<Integer> order = new ArrayList<>();

    while (!q.isEmpty()) {
        int current = q.poll();
```

```
        order.add(current);

        for (int i = 0; i < this.n; i++) {
            if (this.adj[current][i] == 0) {
                continue;
            }

            if (!visited[i]) {
                visited[i] = true;
                q.add(i);
            }
        }
    }

    return order;
}
```