Centro Federal de Educação Tecnológica de Minas Gerais - Departamento de Computação

Laboratório de Algoritmos e Estruturas de Dados II (2ECOM.025)

Lista de Exercícios 6

Professores: Amadeu Almeida e Thiago Rodrigues **Data de entrega:** 30 de Setembro de 2019

Instruções: leia com atenção **todas** as orientações abaixo antes de começar a lista.

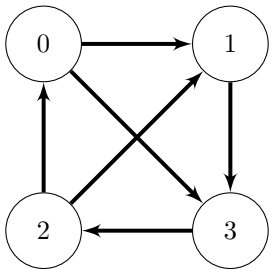
- Os exercícios deverão ser feitos individualmente e valem 5 pontos.
- Submeta no SIGAA um arquivo compactado chamado **Lista6.zip**. Ele deverá conter:
  - As classes Java que implementam as soluções das tarefas.
  - Um relatório em **PDF** com os gráficos dos experimentos e suas respectivas explicações.
- O prazo de entrega é até às 23:59 do dia 30 de setembro.
- Códigos fonte em PDF e submissões em atraso serão desconsiderados.
- O enunciado desta lista possui 3 exercícios e 3 páginas.

Exercício 1: classe Grafo

Esta tarefa objetiva implementar uma classe que manipula um grafo orientado que é representado por uma matriz de adjacência.

Um exemplo de grafo orientado com 4 vértices e 6 arestas e sua matriz de adjacência é apresentado na figura abaixo. É importante salientar que todas as arestas deste grafo tem peso 1.

Grafo:



Matriz de adjacência:

Vértice	0	1	2	3
0	0	1	0	1
1	0	0	0	1
2	1	1	0	0
3	0	0	1	0

1. Crie um projeto chamado **ListasDeGrafos** e um arquivo com o nome Grafo.java.
2. Implemente os atributos da classe **Grafo**:
   - private final int numeroVertices;
   - private final int[][] matrizAdjacencia;

3. Elabore os métodos a seguir:
    - public Grafo(int vertices) - método construtor. Inicializa todas as posições da matriz de adjacência com valores iguais a 0.
    - public void insereAresta(int vertice1, int vertice2, int peso) - insere uma aresta no grafo.
    - public boolean existeAresta(int vertice1, int vertice2) - verifica se uma aresta existe no grafo.
    - ArrayList<Integer> listaDeAdjacencia(int vertice1) - retorna a lista de vértices adjacentes a um determinado vértice v. Ou seja, todos os vértices u cujos quais existe uma aresta que sai de v e chega em u.
    - public int getPeso(int vertice1, int vertice2) - devolve o peso de uma aresta.
4. Explique o objetivo dos métodos implementados no item 3 por meio de comentários no próprio código.

Exercício 2: classe AlgoritmosEmGrafos

O objetivo deste exercício é iniciar a criação de uma biblioteca que irá conter diversos algoritmos em grafos. Ao longo desta lista, será implementado um procedimento de busca em profundidade. Já nas seguintes, esta biblioteca será incrementada com novos algoritmos.

1. Crie um arquivo chamado AlgoritmosEmGrafos.java.
2. Implemente os atributos da classe **AlgoritmosEmGrafos**:
    - private final int[] distanciaProfundidade; - guarda a distância, em vértices, entre o nó inicial da busca em profundidade e cada um dos outros vértices do grafo.
    - private final int[] verticePredecessor; - armazena o vértice predecessor de cada nó descoberto durante a busca em profundidade.
3. Elabore os métodos a seguir:
    - public AlgoritmosEmGrafos(int vertices) - método construtor.
    - public void iniciaBuscaEmProfundidade(int vertice) - inicia o algoritmo de busca em profundidade. Neste método, a distância relativa entre o vértice de partida e os demais deve ser iniciada com o valor $V + 1$, onde V é o número de vértices do grafo, enquanto a distância entre o vértice inicial e ele mesmo se inicia com 0. Já o valor de todos os vértices predecessores é inicialmente igual a -1.
    - private void buscaProfundidade(int vertice) - implementa o algoritmo de busca em profundidade.
    - public int[] getDistanciaProfundidade()
    - public int[] getVerticePai()
4. Clarifique o proposito de cada método elaborado no item 3 por meio de comentários no código.

Exercício 3: experimentos computacionais

Este exercício visa testar a eficiência do algoritmo de busca em profundidade para diferentes tipos de grafos.

1. Antes de começar os experimentos, faça o download dos dois grafos que estão anexados junto a esta lista no SIGAA. A primeira linha de cada arquivo possui, respectivamente, a quantidade de vértices e arestas dos grafos, enquanto as demais contém uma das arestas do grafo. O vértice de saída e entrada são retratados, respectivamente, no primeiro e no segundo número de cada linha.

Por exemplo, o arquivo relativo ao grafo mostrado na página 1 deste documento é apresentado da seguinte forma:

```
4 6
0 1
0 3
1 3
2 1
2 2
3 1
```

2. Experimento 1:
   (a) Execute o algoritmo de busca em profundidade partindo do vértice 0 de cada um dos dois grafos.
   (b) Para cada grafo, retorne:
      i. As distâncias entre o vértice 0 e cada um dos outros vértices.
      ii. A lista de nós precedentes.
3. Experimento 2:
   (a) Rode procedimento de busca em profundidade nos dois grafos partindo de um vértice a sua escolha.
   (b) Para cada grafo, compute:
      i. As distâncias entre o vértice selecionado e todos os outros vértices.
      ii. A lista de nós precedentes.
4. Crie quatro tabelas que retratam os resultados dos dois experimentos acima para ambos os grafos. Cada tabela possui três colunas contendo respectivamente:
   - O número de cada vértice.
   - A distância, em nós, entre o vértice atual e o inicial. Caso um nó não seja descoberto durante a busca, o valor da distância é igual a $V + 1$, onde V é o número de vértices do grafo.
   - O vértice precedente ao atual. Caso um nó não seja encontrado pela busca, o valor de seu precedente é igual a -1.

   Modelo de Tabela:

Vértice	Distância	Precedente

5. Faça dois gráficos:
   (a) Um gráfico que apresente, para o primeiro grafo, a distância, em vértices, entre os nós iniciais dos experimentos 1 e 2 e cada um dos demais vértices. O eixo X deste gráfico contém os valores relativos às distâncias entre os nós, variando-se de uma em uma unidade. Já o eixo Y é composto pela quantidade de vértices que estão a uma distância de no máximo K nós, onde K é um número entre 0 e $V + 1$. O eixo Y cresce a medida que o eixo X avança. Depois, explique as razões pelas quais os crescimentos das duas curvas do gráfico são diferentes.
   (b) Um que compare, para o segundo grafo, a distância, em nós, entre os vértices iniciais dos experimentos 1 e 2 e cada um dos demais nós. Neste gráfico, o eixo X refere-se aos valores relativos às distâncias entre os vértices, variando-se de uma em uma unidade. Enquanto a quantidade de nós que estão a uma distância de no máximo de K vértices, onde K é um número entre 0 e $V + 1$, é retratada pelo eixo Y. O eixo Y cresce a medida que o eixo X avança. Por fim, cite o motivos pelos quais alguns vértices estão a uma distância de $V + 1$ em relação ao nó inicial.

Guarde o projeto contendo as classes Grafo e AlgoritmosEmGrafos para as aulas seguintes, pois ele será utilizado nas próximas listas de exercícios.