Engenharia da Computação
Departamento de Engenharia
Centro Universitário da Fundação Herminio Ometto

Trabalho prático I
Análise e Projeto Orientado a Objetos I

Maurício Acconcia Dias[1]

[1]Engenharia da Computação, macdias@fho.edu.br

1. INTRODUÇÃO

O trabalho proposto será avaliado como entrega para composição da nota N1 do período valendo 20% da nota total. As instruções e partes necessárias para a execução do trabalho estão descritas abaixo.

2. INSTRUÇÕES

O trabalho deve ser desenvolvido em grupos de no máximo 3 pessoas e no mínimo 1. A linguagem a ser utilizada é o C# e o paradigma deve ser orientado a objetos em sua forma pura como apresentado em sala de aula. Qualquer modificação proposta deve ser justificada e a não utilização de orientação a objetos implicará em nota final 0.

Qualquer indício de plágio, seja de ferramentas computacionais ou dos colegas, irá resultar em nota 0. Assim como entregas que não executarem/rodarem. Vocês terão toda a primeira aula do dia da prova para acertar a entrega então acredito que não haverá problema.

3. DESCRIÇÃO DO TRABALHO

O trabalho prático consiste na implementação de um TAD Grafo. Seu TAD deve possuir necessariamente o seguinte conjunto mínimo de operações (funções):

- endVertices(G, e): retorna referências para os dois vértices finais da aresta e.
- opposite(G, v, e): retorna uma referência para o vértice oposto a v na aresta e.
- areAdjacent(G, v, w): verdadeiro se os vértices v e w forem adjacentes, falso caso contrário.
- replaceEdge(G, e, o): substitui o elemento da aresta e por o.
- replaceVertex(G, v, o): substitui o elemento do vértice v por o.
- insertVertex(G, o): insere um novo vértice isolado, armazenando nele o elemento o, e retorna uma referência para esse vértice.

- insertEdge(G, v, w, o): insere uma aresta (v,w), armazenando nela o elemento o, e retorna uma referência para essa aresta.
- removeVertex(G, v): remove o vértice v (e suas arestas) e retorna o elemento armazenado nele.
- removeEdge(G, e): remove a aresta e, retornando o elemento armazenado nela.
- edgeValue(G, e): retorna o elemento armazenado na aresta e.
- vertexValue(G, v): retorna o elemento armazenado no vértice v.

Após a implementação do TAD, Implementar uma rotina chamada Dijkstra(G, o, d) que determina o menor caminho (caminho mínimo) entre o vértice o de origem e o vértice d de destino do grafo G. Ao final da execução da rotina, seu programa deve imprimir na tela o caminho mínimo que leva de o até d, bem como seu custo, como será especificado adiante.

3.1. Entrada de dados

Seu programa deve ser capaz de ler um grafo de um arquivo texto. O formato do grafo no arquivo será o seguinte. A primeira linha informa o número de vértices do grafo. Cada linha subsequente informa as arestas. Um exemplo de um grafo e seu respectivo arquivo texto é dado na Figura 1.

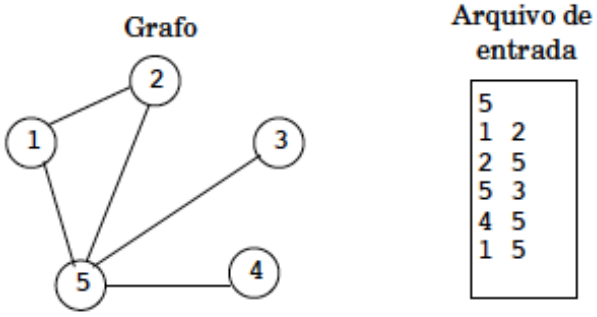

Figura 1: Exemplo de grafo e formato de arquivo de entrada.

3.2. Saída de dados

A saída do seu programa deve ser feita toda em prompt, permitindo que um menu acesse as funções propostas.

3.3. Representação

O grafo deve ser representado com lista de adjascências, sendo que a lista pode utilizar bibliotecas prontas do C#.

4. PERGUNTAS FREQUENTES (FAQ)

Pergunta 1: O trabalho pode ser feito em grupo?
Resposta: O trabalho pode ser feito em trio, dupla ou individualmente.

Pergunta 2: O que será levado em conta na correção?
Resposta: Na correção serão levados em conta (entre outros) os seguintes elementos.
– Conformidade com a especificação.
– Correção da implementação.
– Eficiência da implementação.
– Organização e clareza do código (nomes de funções e variáveis, comentários)

Pergunta 3: Meu trabalho tem um bug. O que vai acontecer com minha nota?
Resposta: Haverá algum desconto, dependendo da gravidade do bug. O desconto será menor se o bug for relatado num arquivo readme.txt, indicando que você estava ciente do problema quando entregou.