LISTA 04 – Classe Grafo

Professor: Valério Aymoré Martins

- *Tenho conhecimento que devo realizar todos os exercícios aqui contidos INDIVIDUALMENTE. Quaisquer códigos considerados como cópia serão invalidados para todas as cópias.*
- *Tenho conhecimento que os exercícios aqui contidos em parte ou no todo poderão ser objeto da 1a e 2 a Provas de Avaliação da disciplina ALGORITMOS E ESTRUTURA DE DADOS.*

1) Implementar uma classe Grafo (não direcionado) com:
    a. Os atributos:
        i. Lista de Adjacencia: array de listas encadeadas de arestas e vértices sendo que as arestas tem um valor inteiro e os vértices tem um flag inteiro chamado “visitado” (veja um exemplo em um código de Dijkstra).
        ii. Matriz de Adjacencia: matriz bidimensional quadrada

        Tanto a lista quanto a matriz tem no seu conteúdo um valor inteiro que representa a função custo do grafo.

    b. Um Construtor :
        i. que receba uma matriz[][] e carregue o atributo acima
        ii. que receba uma array de listas encadeadas e carregue a lista de adjacência.
    c. Dois métodos de sincronismo:
        i. TransformaMatrizEmLista()
        ii. TransformaListaEmMatriz()

        Obs: ao final do construtor que carrega a matriz use o método (i) para gerar a lista correspondente e ao final do construtor que carrega a lista use o método (ii).

d. Outros métodos básicos como abaixo (e já definidos em sala de aula):
    i. cardinalidadeVertices()
    ii. cardinalidadeArestas()
    iii. (array) grauVertices()
    iv. diferencaCardinalidadesVA() = (i)-(ii)
    v. CaminhoMaisCurto (shortestPath)
    vi. ehConexo (tem caminho entre todos os pares – 2 loops for e sai na primeira desconexão)
    vii. ehArvore (se é conexo, se diferencaCardinalidadesVA()==1 e se não tem loop – não tem caminho de V à V)
    viii. ehAnel()
    ix. ehEstrela()
    x. ehTotalmenteConexo()

e. Os métodos fundamentais em grafos:
    i. DFS()
    ii. BFS()

f. Os algoritmos de:
    i. Dijikstra()
    ii. Prim()
    iii. Krukal()
    iv. Bellman-Ford()
    v. Floyd-Warshall()

g. Adicionalmente fazer um algoritmo para resolver soluções por algoritmo:
    i. A* (A-star)

    Obs: compare de acordo com PathFind (http://cs.indstate.edu/hgopireddy/algor.pdf)

LISTA 04 (final) – Classe Grafo

1) Utilizar a classe classe Grafo (não direcionado) para resolver:
    a. Adicione na classe métodos para resolver os problemas comuns de:
        i. Problema de único destino: consiste em determinar o menor caminho entre cada um dos nós do grafo e um nó de destino dado.
        ii. Problema de único origem: determinar o menor caminho entre um nó dado e todos os demais nós do grafo.
        iii. Problema de origem-destino: determinar o menor caminho entre nós dados.
        iv. Problemas de todos os pares: determinar o menor caminho entre cada par de nós presentes no grafo.
        v. Caminho mínimo: dado um conjunto de cidades, as distâncias entre elas e duas cidades A e B, determinar um caminho (trajeto) mais curto de A até B.
        vi. Emparelhamento máximo: dado um conjunto de pessoas e um conjunto de vagas para diferentes empregos, onde cada pessoa é qualificada para certos empregos e cada vaga pode ser ocupada por uma pessoa, encontrar um modo de empregar o maior número possível de pessoas.
        vii. Problema do Caixeiro Viajante: dado um conjunto de cidades, encontrar um passeio que sai de uma cidade, passa por todas as cidades e volta para a cidade inicial tal que a distância total a ser percorrida seja menor possível.
        viii. Problema Chinês do Correio: dado o conjunto das ruas de um bairro, encontrar um passeio que passa por todas as ruas voltando ao ponto inicial tal que a distância total a ser percorrida seja menor possível.

b. Por fim:
    i. Adicione os códigos de OSPF e RIP (http://algs4.cs.princeton.edu/44sp/)
    Obs.: poderão ser cobrados o conhecimento nesses códigos e a capacidade e implementação de “melhorias” nos mesmos.