CCT/UFCA
Algoritmos em Grafos
1º semestre de 2023 - Turma CC0017

Prof. Carlos Vinicius G. C. Lima

Projeto de Implementação

03/10/2023

Informações:

i. O projeto consiste na implementação de propriedades e operações sobre grafos ponderados.

ii. A linguagem de programação utilizada pode ser a de sua preferência.

iii. Os algoritmos implementados devem seguir aqueles com a **menor complexidade possível** em termos assintóticos.

iv. Somente serão considerados para avaliação as operações que foram solicitadas explicitamente.

v. A clareza e concisão das implementações também são objetos de avaliação.

vi. As implementações devem seguir as especificações do projeto.

vii. Comentários de ajuda no código também serão levados em consideração.

viii. O projeto pode deve ser implementado de forma individual.

ix. O prazo máximo para entrega é o dia 03/11/2023.

x. A nota do projeto corresponderá a 30% da nota final.

xi. A entrega do projeto consinstirá em um arquivo compactado contendo: os arquivos de código; um arquivo contendo instruções para a execução do seu código; links para vídeos contendo a explicação da implementação, **um para cada participante da equipe**; no vídeo, deverá ser explicado qual parte você foi responsável por implementar.

xii. Considere o grafo $G = (V, E)$ com $V(G) = \{1, 2, \ldots, n\}$ e os algoritmos e definições vistos em sala. Considere também G representado por listas de adjacências, a menos que explícito o contrário.

Especificações:

1. **(10 pontos) - Implementar a leitura de um grafo dado como entrada através de arquivo de texto:** A entrada do programa principal consiste na leitura de um arquivo de texto, onde a primeira linha contém três campos: uma *string* para indicar o nome do grafo, por exemplo, G, H, G_1, etc; o número de vértices (n); o número de arestas (m) do grafo. Os valores serão separados por um espaço simples e devem ser lidos na ordem descrita anteriormente. A partir da terceira linha, as demais linhas consistem nas especificações das arestas (direcionadas), uma por linha. Cada linha contém apenas uma aresta (v_iv_j) começando com o valor (vértice v_i) e, após um espaço, o valor da outra extremidade v_j. Além das extremidades, cada linha terá um terceiro campo indicando o peso da aresta v_iv_j, especificado após um espaço em relação ao vértice v_j. O peso de cada aresta pode ser um número real positivo ou negativo, ou mesmo infinito, indicado como o valor máximo representável pela linguagem de programação escolhida.

2. **(10 pontos) - Representação do Grafo:** O arquivo, uma vez lido, deverá gerar uma estrutura de representação do grafo: uma lista de adjacências ou uma matriz de adjacências.

3. **(40 pontos) - Operações Básicas:** As seguintes operações sobre o grafo deverão ser implementadas:

   a) **NovoGrafo()**: Retorna um grafo vazio, ou seja, com apenas um novo vértice;

   b) **Grafo(G)**: Retorna uma representação por listas de adjacências de G;

   c) **EVertice(G, v)**: Verifica se $v \in V(G)$ ou não;

   d) **AddAresta(G, v_i, v_j, ω)**: Adiciona uma aresta em G entre os vértices v_i e v_j com peso ω. Deve verificar se $v_i, v_j \in V(G)$, caso contrário a operação não poderá ser efetuada;

   e) **RemoveAresta(G, v_i, v_j, ω)**: Remove uma aresta em G entre os vértices v_i e v_j com peso ω. Deve verificar se $v_i, v_j \in V(G)$ e se existe uma tal aresta, caso contrário a operação não poderá ser efetuada;

f) **ExisteAresta**(G, v_i, v_j, ω): Verifica se existe uma aresta em G entre os vértices v_i e v_j com peso ω.

g) **MudaPeso**(G, v_i, v_j, ω, ω'): Modifica valor de peso de uma aresta em G entre os vértices v_i e v_j de valor ω para o valor ω'. Deve verificar se $v_i, v_j \in V(G)$ e se existe uma tal aresta, caso contrário a operação não poderá ser efetuada;

h) **ImprimeGrafo**(G): Imprime todos os vértices e arestas de G;

i) **RemoveGrafo**(G): Libera todo o espaço utilizado pela representação de G;

j) **RecuperaPeso**(G, v_i, v_j): Devolve a lista de pesos de todas as arestas entre os vértices v_i e v_j em $V(G)$. Deve verificar se $v_i, v_j \in V(G)$, caso contrário a operação não poderá ser efetuada;

k) **GrafoSimples**(G): Retorna se o grafo G é um grafo simples ou não;

l) **EArvore**(G): Retorna se o grafo G é uma árvore ou não;

m) **EBipartido**(G): Retorna se o grafo G é bipartido ou não;

n) **Complemento**(G): Retorna o grafo complementar $\overline{G}$ de G;

o) **EAdj**(G, v_i, v_j): Verifica se $v_iv_j \in E(G)$;

p) **Adjacencia**(G, v): Devolve a lista de adjacência de v em G. Deve verificar se $v \in V(G)$, caso contrário a operação não poderá ser efetuada;

q) **Incidencia**(G, v): Devolve as arestas incidentes a v em G. Deve verificar se $v \in V(G)$, caso contrário a operação não poderá ser efetuada;

r) **MatrizAdj**(G): Constrói a matriz de adjacência de G, onde a posição $a_{i,j}$ corresponde ao peso da aresta v_iv_j, para todo $1 \leq i \leq j \leq n$;

s) **ImprimeMatrizAdj**(G): Imprime a matriz de adjacência de G;

t) **Conexo**(G): Retorna se G é conexo ou não.

4. **(20 pontos) - Percursos:** As seguintes operações sobre o grafo deverão ser implementadas:

   a) **DFS**(G, v_i): Percorre os vértices de G em profundidade, iniciando a busca em $v_i \in V(G)$. Imprime a árvore de busca gerada pelo percurso, ou seja, um vetor que indica, para cada vértice $v_i \in V(G)$, o vértice a partir do qual v_i foi alcançado pela primeira vez na busca;

   b) **BFS**(G, v_i): Percorre os vértices de G em largura, iniciando a busca em $v_i \in V(G)$. Imprime a árvore de busca gerada pelo percurso, ou seja, um vetor que indica, para cada vértice $v_i \in V(G)$, o vértice a partir do qual v_i foi alcançado pela primeira vez na busca.

5. **(30 pontos) - Caminhos mínimos:** Os seguintes algoritmos sobre o grafo deverão ser implementadas:

   a) **CaminhoMinimo**(G, v_i, v_j): Devolve um caminho mínimo (sequência de vértices) entre v_i e v_j no grafo G;

   b) **CustoMinimo**(G, v): Devolve os custos dos caminhos mínimos entre $v_i \in V(G)$ e todos os demais vértices de G;

   c) **CaminhoMinimo**(G, v): Devolve os caminhos mínimos entre $v_i \in V(G)$ e todos os demais vértices de G;