ALGORITMOS E ESTRUTURAS DE DADOS II

Trabalho 1 – Primeiro Semestre – 2010

Profa. Rosane Minghim

O problema do caminho mínimo consiste em encontrar o caminho de menor custo que começa em um vértice ***o*** e termina em um vértice ***d***. Uma vez definido tal problema, o presente trabalho consiste em:

- Implementar uma **Estrutura de Dados Dinâmica** para grafos não-direcionados, na forma de *ista de adjacências.*
- Usar a estrutura de dados implementada para implementar um **TAD Grafo**. Seu TAD deve possuir necessariamente o seguinteconjunto 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 c.c.
  - **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**.

    Notas: (i) Tanto arestas quanto vértices devem armazenar elementos com valores inteiros positivos; (iii) As "referências" a vértices e arestas são *ponteiros* para os respectivos nós das listas de arestas e vértices

  - 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 específicado adiante.

Entrada e Saída de Dados

O grafo a ser manipulado será especificado ao seu programa a partir da entrada de dados padrão (teclado). Para isso, implemente um procedimento que leia dados a partir do teclado e utilize as funções do seu TAD Grafo para construir e armazenar o grafo correspondente. O número máximo de vértices que seu programa deve tratar é igual a 100. O número máximo de arestas é igual a 200.

Os comandos permitidos para operar sobre seu TAD são representados por duas letras maiúsculas. Toda linha de entrada obrigatoriamente inicia com um comando. Só serão fornecidos como entrada comandos aqui especificados (não há necessidade de tratar comando inválidos pois só serão fornecidos comandos válidos). Os únicos comandos permitidos como entrada são apresentados a seguir. O símbolo ¶ denota um único espaço em branco.

- **Cria vértice**

CV ¶ *x*

CV cria um vértice contendo o valor *x*. Vértices só armazenarão números inteiros positivos.

- **Deleta vértice**

DV ¶ *v*

DV deleta o vértice identificado por *v* (veja abaixo a nota a respeito de identificadores). Não serão fornecidos como entrada identificadores inexistentes.

- **Cria aresta**

CA ¶ v_1 ¶ v_2 ¶ *x*

CA cria uma aresta entre os vértices de identificador v_1 e v_2. O valor armazenado na aresta é um número inteiro especificado por *x*. Não serão fornecidos identificadores de vértices inexistentes.

- **Deleta aresta**

DA ¶ *u*

DA deleta a aresta de identificador *u*. Não serão fornecidos identificadores de arestas inexistentes.

- **Troca valor do vértice**

TV ¶ *v* ¶ *x*

TV troca o valor armazenado no vértice de identificador *v* pelo valor *x*. Não serão fornecidos identificadores de vértices inexistentes.

- **Troca valor da aresta**

TA ¶ *u* ¶ *x*

TA troca o valor armazenado na aresta de identificador *u* pelo valor *x*. Não serão fornecidos identificadores de arestas inexistentes.

- **Imprime grafo**

IG

IG imprime o grafo na tela. Para definir a forma que seu programa deve imprimir o grafo na tela, considere um grafo com ***n*** vértices e ***m*** arestas, em que $identifica\,dor(v_i)$ é o identificador de um vértice ***i*** e em que $valor(v_i)$ é o valor armazenado no vértice ***i***. Considere ainda que $identifica\,dor(a_i)$ é o identificador da aresta ***i***, $identifica\,dor(v_{ai})$ e $identifica\,dor(v_{bi})$ são os identificadores de seus dois vértices finais. Finalmente, $valor(a_i)$ é o valor armazenado na aresta ***i***. Seu programa deve imprimir o grafo exatamente na seguinte forma:

n
$identifica\,dor(v_1)$ ¶ $valor(v_1)$
$identifica\,dor(v_2)$ ¶ $valor(v_2)$
...
$identifica\,dor(v_n)$ ¶ $valor(v_n)$
m
$identifica\,dor(a_1)$ ¶ $identifica\,dor(v_{a1})$ ¶ $identifica\,dor(v_{b1})$ ¶ $valor(a_1)$
$identifica\,dor(a_2)$ ¶ $identifica\,dor(v_{a2})$ ¶ $identifica\,dor(v_{b2})$ ¶ $valor(a_2)$
…
$identifica\,dor(a_m)$ ¶ $identifica\,dor(v_{am})$ ¶ $identifica\,dor(v_{bm})$ ¶ $valor(a_m)$

O símbolo ¶ representa um único espaço em branco. É claro que a impressão gerada, em geral, não é única (as linhas correspondentes aos diferentes vértices e arestas podem estar em qualquer ordem). Para tornar única a impressão gerada pela função **IG**, seu programa deve proceder da seguinte forma:

1. Imprima os pares $identifica\,dor(v_i)$ ¶ $valor(v_i)$ de forma que:
   $$identifica\,dor(v_1) < identifica\,dor(v_2) < ... < identifica\,dor(v_n)$$
   Ou seja, os vértices devem ser impressos por ordem de identificador.
2. Imprima as informações das arestas de forma que seus identificadores sejam apresentados ordenadamente, ou seja:
   $$identifica\,dor(a_1) < identifica\,dor(a_2) < ... < identifica\,dor(a_m)$$
   Além disso, imprima os identificadores dos vértices finais de arestas de forma que:
   $$identifica\,dor(v_{ai}) < identifica\,dor(v_{bi})$$
   Ou seja, para uma mesma aresta, imprima os identificadores de seus dois vértices de forma que o menor apareça antes.

- **Caminho mínimo**

CM ¶ v_1 ¶ v_2

Determina e imprime o caminho mínimo entre o vértice de identificador v_1 e o vértice de identificador v_2. Seu programa deve imprimir na tela as seguintes informações:

c $identifica\ dor(v_1)$ ¶ $identifica\ dor(v_i)$ ¶ $identifica\ dor(v_j)$ ¶ ... ¶ $identifica\ dor(v_2)$

Sendo c o custo do caminho mínimo entre os vértices v_1 e v_2, e $identifica\ dor(v_1)$, $identifica\ dor(v_i)$, $identifica\ dor(v_j)$, $identifica\ dor(v_2)$ os identificadores dos vértices que compõem o caminho mínimo entre v_1 e v_2, incluindo v_1 e v_2. Só serão fornecidos vértices de identificadores que contenham um **único** caminho mínimo.

- **Termina a execução**

FM

FM termina a execução do seu programa. Todas as estruturas dinâmicas devem ser desalocadas e seu programa deve encerrar.

Repare que não existem entradas para todas as operações especificadas para o seu TAD (endVertices, opposite, areAdjacent, edgeValue, vertexValue não possuem uma operação específica para que possam ser testadas), porém, essas operações serão utilizadas como operações auxiliares para as funções **IG** e **CM**.

*** NOTA**: Observe que o usuário não tem como inserir "referências" (ponteiros) para nós e arestas conforme estas são especificadas no TAD. Para interagir com o usuário, seu programa deverá se referir a vértices e arestas através de um **identificador numérico único**. Os identificadores para vértices e arestas devem ser gerados automaticamente durante a entrada de dados, com o cuidado de não gerar identificadores já existentes. Veja, para o primeiro vértice gerado, o identificador 1 deve ser atribuído; para o segundo vértice, o identificador 2; e assim sucessivamente. O mesmo vale para arestas: para a primeira aresta gerada, o identificador 1 deve ser atribuído; para a segunda aresta, o identificador 2; e assim sucessivamente. Identificadores são **únicos** e **não** devem ser reutilizados, ou seja, caso uma aresta seja removida, seu identificador não pode ser utilizado para outra aresta. Os identificadores NÃO devem ser armazenados na estrutura de dados do grafo, pois este será utilizado como TAD. Não confunda o identificador de um vértice ou aresta com o conteúdo que poderia ser armazenado naquele vértice ou aresta. Para associar os identificadores às referências dos vértices e arestas requeridos pelas operações do TAD, você deverá utilizar uma **estrutura de dados auxiliar** (um mapa), bem como as devidas operações nessa estrutura, de forma que a partir de um identificador você obtenha uma referência e vice-versa.

Exemplo de Entrada e Saída

A seguir é apresentada uma possível entrada para seu programa e a saída esperada (que deve ser produzida pelo seu programa). A coluna “Identificadores gerados” é apresentada por motivos didáticos e deve ser gerenciada internamente pelo seu programa.

Linhas	Entrada	Identificadores gerados		Saída
		Vértices	Arestas	
1.	CV 6	1		6
2.	CV 9	2		1 6
3.	CA 1 2 20		1	2 9
4.	CV 0	3		3 0
5.	CV 0	4		4 0
6.	CV 5	5		5 5
7.	CV 8	6		6 8
8.	CA 1 6 15		2	7
9.	CA 2 3 10		3	1 1 2 20
10.	CA 3 4 5		4	2 1 6 15
11.	CA 3 5 23		5	3 2 3 10
12.	CA 5 4 12		6	4 3 4 5
13.	CA 6 5 2		7	5 3 5 23
14.	IG			6 4 5 12
15.	CM 1 3			7 5 6 2
16.	FM			30
17.				1 2 3

- Linhas 1 a 15 correspondem à saída do comando **IG**.
- Linhas 16 e 17 correspondem à saída do comando **CM 1 3**.

OBSERVAÇÕES IMPORTANTES

- Implemente usando a linguagem C padrão ANSI (caso estiver usando o compilador GCC, utilize a *tag* -ansi);
- Vazamento de memória, referência a valores de variáveis não inicializadas e outros defeitos serão levados em conta na avaliação;
- Os trabalhos deverão ser feitos **individualmente**;
- Serão anulados aqueles trabalhos nos quais forem detectados quaisquer tipos de cópia ou plágio, não importa a origem;
- O período de entrega do trabalho é de 12/04 a 07/05;
- Os trabalhos devem ser entregues através do seguinte endereço eletrônico:
  http://www.lcad2.icmc.usp.br/cgi-bin/gfandery/2010/alg2/sqtpm.pl
  Selecione a opção Trabalho1;
- Seu nome de usuário para entrega é seu Número USP. Sua senha deve ser cadastrada no próprio sistema antes da utilização.