

Laboratório 1 - TAD Grafo
SCC0216 - Modelagem computacional em grafos
Professor: Alneu de Andrade Lopes
Estagiário PAE: Alan Valejo

LAB 1: TAD e operações comuns

1. Especificação

O programa principal deverá ler da entrada padrão a lista de arestas do grafo e representá-lo em memória usando matriz e lista de adjacência. Em seguida, o programa deverá ler instruções, realizar as operações correspondentes e, quando necessário, imprimir o resultado na saída padrão.

2. Descrição da entrada

Na primeira linha haverá o descritor do grafo contendo 2 letras e 2 números separados por espaço. A primeira letra, D ou G, indica se o grafo é direcionado (dígrafo) ou não, respectivamente. A segunda letra, M ou L, caracteriza a representação interna que deverá ser utilizada, matriz ou lista de adjacência. Os números indicam, nesta ordem, o número de vértices e de arestas do grafo. Nas linhas seguintes, as arestas do grafo serão representadas por triplas contendo os vértices incidentes e seu peso. No caso do dígrafo, a ordem dos vértices é a direção da aresta.

3. Operações

O programa deverá ler uma série de instruções (uma por linha) e realizar a operação correspondente. As instruções são:

- IG: imprimir grafo respeitando a representação;
- VA X: imprimir vértices adjacentes ao vértice X (em ordem crescente dos índices dos vértices, por exemplo, “1 5 7” e não “7 1 5”);
- AA X Y P: adicionar aresta entre os vértices X e Y com peso P (ATENÇÃO: OS PESOS SÃO MAIORES OU IGUAIS A 0);
- RA X Y: remover aresta entre os vértices X e Y;
- IT: imprimir transposto (caso não seja um dígrafo, ignore o comando);
- MP: imprimir aresta com menor peso (haverá apenas uma aresta com menor peso). Caso não seja um dígrafo, imprima a aresta com os índices dos vértices em ordem crescente, por exemplo, “3 7” e não “7 3” para a aresta entre os vértices 3 e 7.

4. Submissão

O exercício deverá ser entregue pelo sistema run.codes . Todos os alunos deverão submeter seus códigos no 'Exercício 1: TAD e operações comuns' até o final da aula. Somente a última submissão será

considerada. Todas as demais submissões serão desconsideradas, incluindo aquelas dentro do período normal de submissão. Os exercícios deverão submetidos em um arquivo .zip contendo código-fonte do programa e um Makefile para compilação e teste do trabalho (verificar com o estagiário PAE, caso não saiba escrever um Makefile). Se necessário, incluam no .zip um arquivo chamado readme com informações que julgarem necessárias.

Os códigos deverão ser compilados pelo compilador gcc com a flag -std=c99. A não conformidade das implementações com a versão C estabelecida acarretará em nota zero. Atenção! Todos os códigos enviados passarão pelo sistema de verificação de plágio. Se forem identificados códigos duplicados, todos os alunos envolvidos receberão nota zero.

5. Correção e Avaliação

As implementações serão avaliadas por meio de casos de testes, com peso 7, e pela legibilidade e boas práticas de programação, com peso 3.

Os seguintes casos implicarão em nota zero:

- Não conformidade com a versão C99;
- Programas não estruturados como um TAD;
- Exercícios plagiados.

Apêndice: Exemplos

Os comentários são apenas descritivos. Estes não existirão nas entradas e nem deverão ser impressos como saída.

Exemplo 1

```
G M 5 6           // Grafo não-direcionado representado por matriz de adjacência
0 1 1
1 2 1
1 3 1
2 3 1
2 4 1
3 4 1
IG               // Imprimir grafo (matriz de adjacência)
VA 1             // Imprimir vértices adjacentes ao vértice 1 (em ordem)
AA 0 2 5         // Adicionar aresta entre os vértices 0 e 2 com peso 5
RA 0 1           // Remover aresta entre os vértices 0 e 1
IG
```

Saída 1

```
.1...
1.11.
.1.11
.11.1
..11.           // Matriz de adjacência (pesos)
0 2 3           // Vértices adjacentes a 2
..5..
..11.
51.11
.11.1
..11.           // Matriz de adjacência com as alterações
```

Exemplo 2

```
D L 5           // Grafo direcionado representado por lista de adjacência
0 1 1
1 2 2
1 3 0
2 3 5
2 4 1
4 3 3
IG             // Imprimir grafo (lista de adjacência)
IT             // Imprimir transposto
MP             // Imprimir aresta com menor peso
```

Saída 2

```
0. 1(1)
1. 2(2) 3(0)
2. 3(5) 4(1)
3.           // Vértice 3 não possui arestas saindo
4. 3(3)      // Lista de adjacência
0.
1. 0(1)
2. 1(2)
3. 1(0) 2(5) 4(3)
4. 2(1)      // Transposto
1 3          // Aresta com menor peso
```