Laboratório 1 - TAD Grafo

SCC0216 - Modelagem computacional em grafos

Professor: Alneu de Andrade Lopes

Estagiário PAE: Filipe Alves Neto Verri

1 Especificação

O laboratório consiste na implementação em `C` do TAD grafo e operações comuns.

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

Na primeira linha da entrada, 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.

Por fim, 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.

Por exemplo, a entrada[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
```

terá como saída

```
. 1 . . .
1 . 1 1 .
. 1 . 1 1
. 1 1 . 1
. . 1 1 . // matriz de adjacência (pesos)
0 2 3     // vértices adjacentes a 2
. . 5 . .
. . 1 1 .
5 1 . 1 1
. 1 1 . 1
. . 1 1 . // matriz de adjacência com as alterações
```

Outro exemplo de entrada é

```
D L 5 6 // 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
```

que tem como saída esperada

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

```
1. 0(1)
2. 1(2)
3. 1(0) 2(5) 4(3)
4. 2(1) // transposto
1 3     // aresta com menor peso
```

2 Submissão

O exercício deverá ser entregue pelo sistema `run.codes`[2]. Todos os alunos deverão submeter seus códigos no exercício **1 - TAD Grafos** 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.

3 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.

[2] `https://run.codes/`