Universidade Federal da Fronteira Sul - Ciência da Computação

Disciplina GEN254: Grafos

2024.2

Tarefa Nº 01 - Grafo - Listas de Adjacência

Prazo de entrega: **Consultar a página da tarefa.**

Linguagem para implementação: **C++**.

Professor: Andrei Braga

Grafo - Listas de Adjacência

Uma forma comum de representar computacionalmente um grafo é como *listas de adjacência*. Nesta tarefa, você deve implementar uma classe que use esta representação para armazenar um grafo **simples**. Você deve fazer isso utilizando **listas encadeadas** e escrevendo os seguintes métodos, que devem executar no grafo as operações indicadas:

- construtor `Grafo`: constrói o grafo com o número de vértices recebido por parâmetro e sem arestas;
- método `insere_aresta`: insere uma aresta no grafo caso a aresta ainda não exista no grafo e não seja um laço;
- método `remove_aresta`: remove uma aresta do grafo caso a aresta exista no grafo;
- método `num_arestas`: retorna o número de arestas do grafo;
- método `num_arestas_subgrafo_induzido`: dado um conjunto de vértices do grafo, retorna o número de arestas do subgrafo induzido pelos vértices do conjunto;
- método `imprime_complemento`: imprime o complemento do grafo conforme especificado na **Seção Saída** abaixo;
- (se necessário) destruidor `~Grafo`: libera a memória alocada para o grafo.

Você deve escrever um programa que constrói um grafo, executa operações no grafo e depois, se necessário, explicitamente o destrói. O seu programa deve processar informações que determinarão as operações a serem executadas no grafo, o que deve ser feito de acordo com as **Seções Entrada** e **Saída** abaixo.

Entrada

A primeira linha da entrada contém dois inteiros **V** (**V** > 0) e **O** (**O** > 0), sendo **V** o número de vértices do grafo a ser construído e **O** o número de operações a serem executadas no grafo. Cada uma das **O** linhas seguintes consiste em uma das opções abaixo:

- O caractere `I`, um inteiro **X** e um inteiro **Y** separados por espaços em branco;
- O caractere `R`, um inteiro **X** e um inteiro **Y** separados por espaços em branco;
- O caractere `E`;
- O caractere `S`, um inteiro **N** e **N** inteiros separados por espaços em branco;
- O caractere `X`.

Estas opções representam o seguinte, de acordo com o primeiro caractere da linha:

- Se é `I`, então deve ser executada a operação de inserir a aresta **X Y** no grafo (método `insere_aresta` – veja a **descrição acima**);
- Se é `R`, então deve ser executada a operação de remover a aresta **X Y** do grafo (método `remove_aresta` – veja a **descrição acima**);

- Se é `E`, então deve ser executada a operação de retornar o número de arestas do grafo (método `num_arestas`);
- Se é `S`, então, após **N**, é dado um conjunto de **N** vértices do grafo. Deve ser executada a operação de retornar o número de arestas do subgrafo induzido pelos vértices do conjunto dado (método `num_arestas_subgrafo_induzido`);
- Se é `X`, então deve ser executada a operação de imprimir o complemento do grafo (método `imprime_complemento`).

Saída

A saída deve consistir no seguinte:

- Para cada execução da operação de retornar o número de arestas do grafo, o seu programa deve imprimir uma linha contendo o número de arestas retornado.
- Para cada execução da operação de retornar o número de arestas do subgrafo induzido pelos vértices de um dado conjunto, o seu programa deve imprimir uma linha contendo o número de arestas retornado.
- Para cada execução da operação de imprimir o complemento do grafo, o seu programa deve imprimir **V** linhas, uma para cada um dos vértices 0, 1, …, **V** - 1, em sequência. Cada uma destas linhas deve conter
  - o índice do vértice seguido do caractere `:` e
  - os índices dos vizinhos do vértice no complemento do grafo, cada um antecedido por um espaço em branco.

  Os índices dos vizinhos do vértice no complemento do grafo devem ser impressos **em ordem crescente**.

Exemplos de execução

Entrada	Saída
4 8 I 0 2 I 0 1 I 2 1 I 3 1 S 3 0 2 1 R 2 0 E X	3 3 0: 2 3 1: 2: 0 3 3: 0 2

Entrada	Saída
4 10 I 2 3 I 0 1 R 2 0 E S 3 0 3 2 I 0 1 I 2 2 S 1 2 E X	2 1 0 2 0: 2 3 1: 2 3 2: 0 1 3: 0 1

Observações:

- Para a realização dos testes automáticos, a compilação se dará da seguinte forma:
  `g++ -pedantic -Wall *.cpp -o main -lm -lutil`
