MC558 - Projeto e Análise de Algoritmos II
Instituto de Computação

Prof. Flávio Keidi Miyazawa

PED
Hismael Costa (hismael.costa@gmail.com)

TRABALHO PRÁTICO 2

Objetivo

Este trabalho tem por objetivo a auxiliar no entendimento de conceitos relacionados a *caminhos mínimos* através da implementação de algoritmos. Os principais conceitos abordados nesse trabalho são:

- Árvore de Precedência;
- Algoritmo de Dijkstra.

Descrição

O problema tem como entrada um grafo dirigido $G = (V, A)$, um inteiro W, tal que todos os pesos de G devem estar entre 0 e W, e um vértice inicial s. O seu programa deve realizar as seguintes verificações:

- Se existir $u \in V(G)$, tal que, $w(u) < 0$ ou $w(u) > W$, deve ser mostrada a seguinte mensagem: "*Erro: Existe peso fora do intervalo.*" e a execução deve ser interrompida;
- Se existir $u \in V(G)$, tal que, $w(u)$ não é inteiro, deve ser mostrada a seguinte mensagem: "*Erro: Existe peso não inteiro.*" e a execução deve ser interrompida.

Após o seu programa fazer todas as verificações acima e caso não seja finalizado por nenhuma, ele deve imprimir a *árvore de precedência a partir de um vértice s* e também apresentar *o caminho mínimo entre um vértice e outro*.

Implementação e Execução

Entrada e Saída

Como descrito acima, você receberá um arquivo de entrada contendo um grafo, com o seguinte formato:

A saída impressa da solução deve ser:

# Vértices	*# Arestas*	*W*
Origem aresta 0	*Destino aresta 0*	*Peso aresta 0*
⋮	⋮	⋮
Origem aresta m-1	*Destino aresta m-1*	*Peso aresta m-1*

Vértice 0	*Distância de s para 0*
Vértice 1	*Distância de s para 1*
⋮	⋮
Vértice n-1	*Distância de s para n-1*

Para o vértice inicial, utilizaremos o *vértice 0* como padrão. Caso não exista caminho entre s e um vértice, o valor do caminho deve ser mostrado com INF, utilizando a biblioteca *math.h* ou através do *DEFINE*.

Implementação

Será disponibilizada uma rotina para a leitura do grafo.

A rotina de leitura retornará um objeto do tipo Grafo, caso nenhum erro de leitura seja disparado. Caso exista esse erro, o mesmo deve ser mostrado.

Após a leitura, deve-se implementar uma rotina com o seguinte cabeçalho:

bool arvore_precedencia (int n, int m, int W, Grafo g, string mensagem, int RA, int[] pred, int[] dist)

- **n** e **m** são os números de vértices e arestas, respectivamente;
- **g** objeto do tipo Grafo, retornado a partir da leitura do arquivo;
- **W** é o valor máximo dos pesos em g;
- A rotina deve retornar **Verdadeiro** ou **Falso**, conforme não tenha erro na execução do algoritmo;
- Caso não seja possível criar uma árvore de precedência para o grafo informado, escrever o motivo do erro na variável **mensagem** e escrever no programa principal;
- A variável **RA** serve para que você armazene seu RA, facilitando assim a correção nos testes que serão feitos no seu algoritmo;
- O vetor **pred**, onde $pred[v]$ é o vértice antes de v na árvore de predecessores;
- O vetor **dist**, onde $dist[v]$ representa a distância de s para v

Submissão

O trabalho deverá ser enviado por por e-mail com o assunto "Entrega do Teste Prático 2 - RA 999999", onde 999999 deve ser trocado pelo número do seu RA, para o endereço *h265684@dac.unicamp.br* e até as **23:59 do dia 26/11**. Antes de submeter seu trabalho, verifique se ele atende todas as especificações de submissão descritas abaixo:

- Dois arquivos devem ser submetidos:
  - *arvore_precedencia_principal.cpp*: Neste arquivo deve conter a leitura, chamada para a função de geração da árvore de precedência, caminhos mínimos e impressão dos mesmos. O aluno pode criar a função para o cálculo dos caminhos mínimos entre s e t, entretanto não deve alterar as demais funções deste arquivo, com exceção da *main*;
  - *arvore_precedencia_ra999999.cpp*: Neste arquivo você deve implementar sua rotina de criação da trilha. Você também pode incluir outras rotinas, caso ache necessário. Deve-se trocar o *999999* no nome do arquivo, pela sua matrícula.
- O seu algoritmo deve ter complexidade de tempo linear no tamanho do grafo $(O(WV + E))$. Caso seja feito em tempo $(O((V + E)logW))$ terá acrescimo de um ponto na nota final do trabalho;
- O código deve estar comentado, de forma a facilitar o entendimento e correção;
- A implementação deve ser feita utilizando C++.

Avaliação

A avaliação será da seguinte forma:

- **70%** da nota será baseada na correta implementação do algoritmo;
- E os outros **30%** restantes serão baseados nos comentários e facilidade de leitura do programa.