Lab: Algoritmo de Prim

Descrição

Dado um grafo ponderado e não-direcionado encontre sua a árvore geradora mínima. Considere a visitação por ordem crescente de índice, caso contrário os casos de teste não irão funcionar corretamente.

O aluno deverá implementar um programa que leia a ordem do grafo, uma sequência de arestas e imprima as arestas contidas na árvore como no exemplo abaixo.

Exemplos:

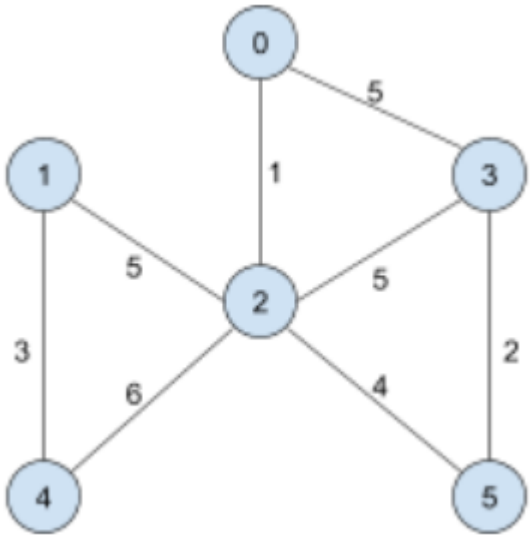

Figure 1: Grafo1

Entrada

```
6
0 2 1
0 3 5
1 2 5
1 4 3
2 3 5
2 4 6
2 5 4
3 5 2
```

Saída

```
2 0 1
5 2 4
3 5 2
1 2 5
4 1 3
```

Observação: Perceba que as arestas da árvore são listadas na ordem em que são descobertas, e que cada aresta é escrita na ordem de visitação dos vértices: *destino origem*, seguida de seu peso associado.

Importante: Você deverá submeter ao run.codes um arquivo .zip com os arquivos .c e .h do template alterados.

Como de costume, você utilizará o template para implementação do exercício. O template é fornecido para que você treine a implementação de um código organizado, separando as declarações no .h e as definições nos .c. Facilita a organização da estrutura lógica do programa, pois já está parcialmente resolvido e acelera o tempo de resolução, por ter parte já escrito.

A princípio, você poderia implementar uma solução a partir do zero: sem nenhum modelo .c e .h fornecido. Apenas com a descrição acima do problema: entrada e saída esperada. Contudo, neste caso, seriam descontados pontos pela má organização do código. Caso não compreenda alguma parte da organização, não tem problema de alterar, desde que mantenha seu código organizado. As funções da TAD apresentadas em sala e descritas nos .h não podem ser alteradas, apenas quando indicado, pois testam se conseguem implementar a TAD como solicitado.

Informativo sobre as avaliações no geral

Para a avaliação final de cada exercício, serão considerados diversos aspectos do código, como a formatação, a lógica e o uso correto das estruturas de dados. Já o Run.Codes servirá apenas de apoio para automatização de alguns testes mais simples.

Importante: O Run.Codes ficará aberto para submissão por apenas 1 ou 2 dias, forçando com que os testes e depurações do código sejam realizados antes da submissão à plataforma. Veja o Exercício 0 (Hello World), para mais detalhes.

Importante: Apenas o último código-fonte enviado na plataforma, após o término do prazo de submissão, será considerado para avaliação.

O trabalho que não cumprir qualquer item da seção 'Requisitos exigidos' receberá **nota zero**, mesmo que a funcionalidade principal esteja implementada corretamente.

Fique atento aos prazos de submissão e envie antes do horário máximo permitido. A plataforma pode sobrecarregar, fazendo com que a submissão seja impedida.

Alguns dos itens avaliados

A seguir é listado alguns dos itens avaliados que reduzem a nota:

- Compilando o código com as flags `-Wall -Wextra -std=c11 -pedantic` nenhuma mensagem de `warning` deve ser retornada
- Nome de variáveis não condizentes com seu uso
- Boas práticas de programação e formatação
- Falta do 'cabeçalho' no início do código-fonte, como o modelo abaixo
- *Memory leak*

```
/***********************************/
/* Aluno: Fulano de Tal            */
/* CES-XX: Nome da disciplina      */
/* Turma Tx                        */
/***********************************/
```

Requisitos exigidos

1. Código-fonte com encoding UTF-8 sem BOM;
2. Código-fonte compilando com sucesso;
3. Indentação;
4. Seguir estritamente o solicitado, incluindo as entradas e saídas para os casos de teste;
5. Código-fonte no padrão C11 da linguagem C.

Exemplo de item (4): enunciado informa que o exercício deve ser resolvido usando laços, porém o aluno implementa a solução com recursão.