Trabalho 2 de Estrutura de Dados

Um problema clássico da computação é o de encontrar a *Minimal Spanning Tree* (MST) de um grafo. A MST é definida como o subconjunto de arestas que conecta todos os vertices do grafo (conexo) que tem a soma de suas arestas mínima. Ou seja, não existe outro subconjunto conexo de arestas cuja soma de pesos seja menor que a da MST (ver Figura 1).

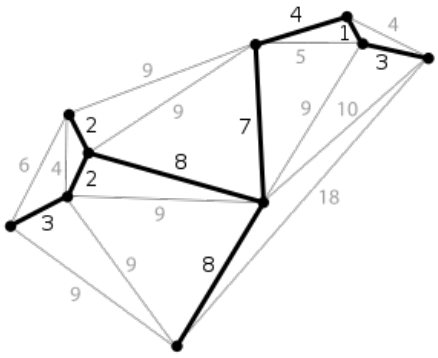


Figura 1: Um exemplo de MST gerada a partir de um grafo planar. O custo dessa MST é de 38.

1 Motivação

Uma concessionária de energia elétrica está com o projeto de trazer luz para pequenas cidades e vilarejos no interior do país. É do interesse deles gastar o minimo possível nessas instalações. Para tal, antes de implementar o novo sistema de luz é feito um estudo no local, levantando todas as casas que devem ter acesso a energia e o custo de subir postes ligando duas casas próximas. O último responsável pela tarefa de planejar o *layout* da rede elétrica de cada cidade não conseguiu manter o passo do progresso (dado que ele fazia tudo "na mão") e foi demitido, sendo vocês contratados para automatizar esse processo para qualquer nova cidade.

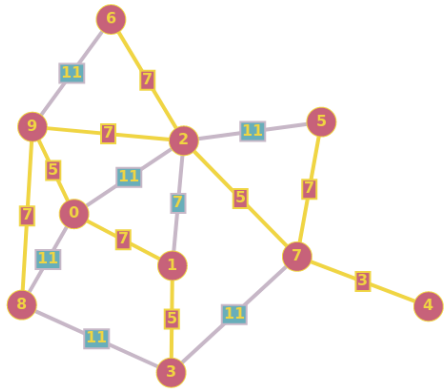
Para resolver esse problema vocês irão implementar um algoritmo de MST de sua preferência (alguns clássicos incluem os algoritmos de Kruskal, de Prim e o *reverse-delete*) e retornar as seguintes estatísticas:

- A MST com o seu custo;

- O quanto é economizado se fosse feita uma rede com conexa na cidade;
- Se a solução é única ou se podem existir mais de uma MST (nota-se que não é pedido que mais de uma MST seja encontrada, se existente, somente reconhecida sua existência).

2 Observações de Programação

Os dados de entrada serão passados para vocês por um arquivo de texto contendo a matriz de adjacência do grafo a ser analisado, que será plano e não-direcionado, após o número de vertices do grafo (ver Figura 2c). A saída do programa deve ser um arquivo de texto contendo todas as arestas da MST, ordenado de acordo com o nó de menor ID. E.g., para o grafo da Figura 2a a entrada é vista na Figura 2c e a saída esperada está na Figura 2b. Deverá também ser impresso as estatísticas na tela (ver Figura 2d).



(a) Grafo exemplo com a MST ressaltada em amarelo.

```
1  0,1
2  0,4
3  1,3
4  2,5
5  2,6
6  2,7
7  2,9
8  4,7
9  5,7
10 8,9
```

(b) Saída de texto da MST. Nota-se a ordenação das arestas.

```
1  10
2  0  7  11 0  0  0  0  0  11 5
3  7  0  7  5  0  0  0  0  0  0
4  11 7  0  0  0  11 7  5  0  7
5  0  5  0  0  0  0  0  11 11 0
6  0  0  0  0  0  0  0  3  0  0
7  0  0  11 0  0  0  0  7  0  0
8  0  0  7  0  0  0  0  0  0  11
9  0  0  5  11 3  7  0  0  0  0
10 11 0  0  11 0  0  0  0  0  7
11 5  0  7  0  0  0  11 0  7  0
```

(c) Representação textual do grafo em uma matriz de adjacência, e entrada do programa. Também contem o número de vertices na primeira linha.

```
willian@willian:~$ ./mst grafo.txt
MST com custo: 53
Economia vs malha totalmente conexa: 73
MST não é única
```

(d) Saída em terminal após a execução do algoritmo, mostrando as estatísticas pedidas. Uma outra MST pode ser alcançada pela troca da aresta $2, 9$ pela aresta $1, 2$.

Figura 2: Exemplo de MST.

Para esse exercício deverá ser utilizada a estrutura de lista de adjacência obrigatoriamente. **Algoritmos que usem matriz de adjacência serão desconsiderados.** O algoritmo pode ser uma implementação de qualquer algoritmo clássico de MST (obedecendo a restrição de usar lista de adjacência) ou pode ser original. Algoritmos originais ganharão pontos extras se devidamente explicados. Co-descobertas (algoritmos já existentes que foram implementados sem o estudo dos mesmos) não serão consideradas como algoritmos originais. A escolha do algoritmo deve ser fundamentada de alguma forma, assim recomenda-se que ao menos os três algoritmos clássicos mencionados sejam pesquisados.

O programa deverá ser escrito em C e dividido em três módulos: (i) o módulo contendo todas as definições de grafo, e qualquer função desse tipo de dado, (ii) o módulo de IO, responsável por ler o grafo do arquivo de entrada e gerar o grafo na estrutura interna, e (iii) um *main* que usa os demais módulos e implementa o algoritmo de MST. Deverá ser entregue também um Makefile, que será a única forma usada para compilar o programa em um executável de nome "mst.out". **Programas gerados com o nome diferente ou cuja compilação não consiga ser feita pela execução do comando "make" serão desconsiderados.**

3 Observações do Relatório

O relatório deverá ser entregue com nome e matrícula de todos integrantes do grupo em formato PDF, e deverá ser composto dos seguintes itens:

- Introdução ao problema resolvido;
- Descrição do algoritmo usado;
- Listagem de **todas** rotinas/funções implementadas;
- Complexidade algorítmica de **todas** rotinas/funções listadas;
- Complexidade algorítmica final do algoritmo de MST;
- Ao menos dois casos de testes explicados;
- Análise do tempo de execução em relação à complexidade do MST.

A seção de descrição deve falar em alto nível do algoritmo usado (dica: usem figuras de grafos exemplos para mostrar como deve funcionar o seu algoritmo). Na listagem das rotinas seria interessante colocar os pseudo-códigos onde for pertinente. Embora isso não seja obrigatório, será considerado como ponto extra se feito.

Os dois casos de testes explicados devem ser um com o MST único, e outro com a existência de mais de um MST. Para a análise de tempo de execução recomenda-se o uso do número de vertices como parâmetro. Vale ressaltar que os resultados podem ser estranhos se não houver uma constância na forma dos diferentes grafos testados (número de arestas por vértice por exemplo). Dica:

uma boa análise observaria o impacto do número de vertices no tempo de execução. Uma ótima análise observaria este impacto para diferentes densidades dos grafos (número de arestas).

Para a descrição dos testes de tempo de execução não esquecer de colocar a escala de tempo e de usar uma escala apropriada. Para a explicação dos dois casos de teste espera-se que seja mostrado como seria possível encontrar outra MST quando existente.

4 Avaliação

A avaliação do trabalho será dividida da seguinte forma:

- Código (50/100)
  - Legibilidade e comentários (10/100)
  - Testes (40/100)
    - Validação dos testes no relatório (20/100)
    - Testes de instâncias desconhecidas (20/100)
- Relatório (50/100)
  - Formatação (10/100)
  - Introdução e descrição de alto nível (10/100)
  - Análise de complexidade (15/100)
  - Análise de testes (15/100)

Deverão ser entregues os códigos (.c e .h), o makefile, o relatório em pdf e os testes executados (arquivos .txt com as matrizes de adjacência) em um diretório de testes. Todos esses deverão ser zipados em um arquivo contendo a matrícula dos componentes do grupo. Não deverão ser mandados arquivos binários.

Haverá pontuação extra por zelo no relatório pela adição de figuras e/ou diagramas pertinentes, pela estruturação do texto, pelo uso de *latex*, etc. Haverá pontuação extra de código para aqueles que implementarem um algoritmo de MST diferente dos algoritmos clássicos e/ou que consiga encontrar mais de uma MST em um grafo, se houver.