Gabriel Chaves Ferreira
2018046700

Trabalho Pratico 2

Modelagem computacional do problema

O problema tem como objetivo solucionar um problema enfrentado pelo prefeito da cidade de Belleville, que deseja construir uma ciclovia que passa por todos os pontos turísticos da cidade de modo a melhorar o turismo local.

Para construir esta ciclovia, existem diversos trechos que devem ser construídos, e o prefeito deseja encontrar aqueles que passam por todos os pontos turísticos mas com o menor custo possível.

Temos como entrada N pontos de interesse na cidade que devem ser conectados pela ciclovia. Cada um destes pontos possui uma atratividade agregada, e além de ter o menor custo, a ciclovia também deve ter a maior atratividade agregada.

Baseado nisso podemos construir um grafo para representar o problema, onde os nossos nós representam os pontos turísticos e as arestas com pesos representam os trechos com seus respectivos custos. Após construir o grafo, fica claro que se acharmos a árvore geradora mínima considerando o peso e a atratividade como peso para cada aresta do nosso grafo, teremos solucionado o problema.

Estrutura de dados e algoritmos utilizados:

Meu algoritmo tem três estruturas de dados (podem ser encontradas nos arquivos **estruturas.h** e **grafos.h**:

1. PontoDeInteresse: Uma classe para representar os pontos de interesse, com seus respectivos códigos e valor turístico.
2. Trecho: Uma classe para representar os trechos e facilitar a modelagem do problema, nela armazeno o trecho de origem, trecho de destino, custo do trecho e a atratividade agregada do trecho.
3. Grafo: Uma classe para representar um grafo e verificar se ele é cíclico ou não.

O algoritmo utilizado para resolver o problema segue da seguinte forma:

1. Ordeno o vetor de trechos em ordem crescente de custo e caso dois trechos tenham o mesmo custo, aquele com maior atratividade agregada vem primeiro.
2. Pego o trecho com o menor custo e maior atratividade agregada (que é o primeiro trecho do meu vetor ordenado).
3. Verifico se a adição dele no grafo gera um ciclo.
4. Se não gera um ciclo, adiciono no grafo e no resultado. Se gera um ciclo, não adiciono no resultado.
5. Removo o trecho atual do meu vetor de trechos.
6. Repito a etapa 2 até o meu vetor de trechos ficar vazio.

Análise de complexidade assintótica do problema:

1. Para o preenchimento da atratividade agregada, para cada trecho nosso algoritmo executa um **for** percorrendo todos os pontos de interesse. Ou seja, temos uma complexidade **n * m** onde **n** e o numero de trechos e **m** o numero de pontos de interesse.
2. Para a ordenação dos trechos por custo e atratividade agregada utilizamos o método sort, que executa em **n log n**, onde **n** e o numero de trechos.
3. Para verificar se a adição do trecho no grafo causa um ciclo, nosso algoritmo roda em **V + E**, onde **V** é o número de vértices e **E** o número de arestas.
4. Para calcular a rota, utilizamos um loop do-while, que executa **n** operações**,** onde **n** é o número de trechos. Para cada trecho, temos que verificar se a adição dele no grafo do resultado gera um ciclo, deste modo temos **n * (V+E)** ao final.

Como compilar e executar:

O compilador é o g++.

1. make clean
2. make build
3. ./tp02