TP 02 - Trabalho Prático 02
Algoritmos I

Samuel Fantini Braga
2018046637

Modelagem computacional do problema

O problema tem como objetivo suprir as necessidades da cidade de Belleville. A cidade é um polo turístico e que vem sofrendo impactos com o aumento da emissão de poluentes e poluição sonora proveniente do aumento da frota de veículos movidos a combustão.

Com a redução do fluxo de turistas devido a poluição e engarrafamentos a economia local tem sido afetada significativamente. Dessa forma, o novo prefeito implementou políticas de incentivo ao uso de bicicletas. Portanto, uma das principais iniciativas é a construção de uma ciclovia pela cidade, porém, o prefeito deseja que a ciclovia seja construída de forma inteligente.

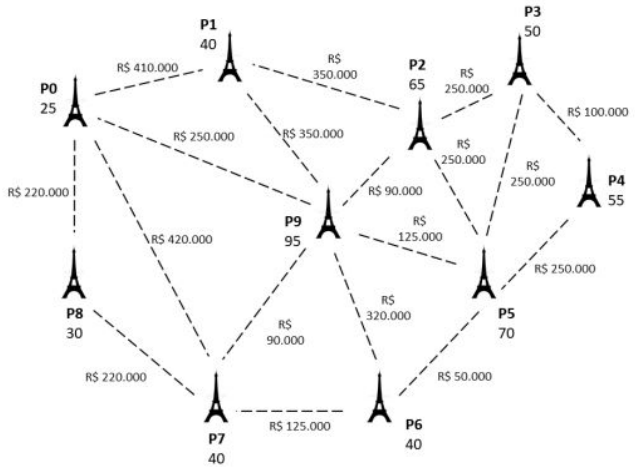
Com isso, ele solicitou o desenvolvimento de um sistema que oriente a construção da ciclovia. A ciclovia deve interligar os principais pontos da cidade gastando o menor montante de recursos públicos possível.

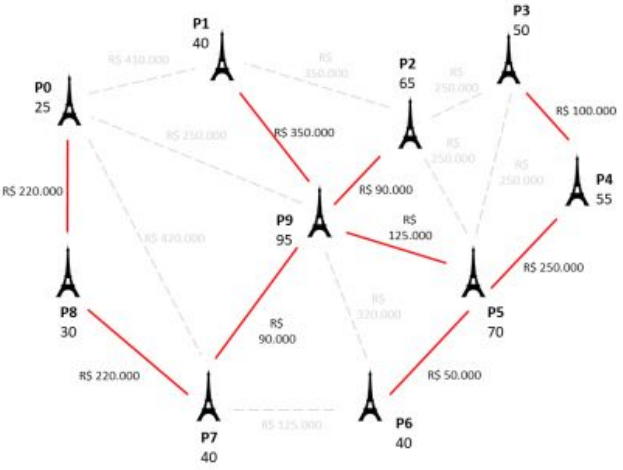
Será levantado N pontos de interesse na cidade que deveriam ser conectados pela ciclovia. Baseado na quantidade de turistas que visitam estes pontos de interesse, foi definida para cada ponto de interesse uma métrica denominada valor turístico. Quanto maior o valor turístico de um ponto de interesse, mais turistas visitam aquele local.

De maneira formal, foram apresentados T trechos possíveis para a ciclovia, ligando um ponto de interesse (Pi) a outro ponto de interesse (Pj), com custo de construção igual a ct. Além disso, definiu-se a atratividade de um trecho da ciclovia como sendo a soma dos valores turísticos dos dois pontos de interesse conectados por este trecho.

Portanto, o programa solicitado pelo prefeito deve sugerir o trecho T onde terá o menor orçamento possível para a construção da ciclovia e atingir o maior valor turístico possível.

Exemplo de ciclovia





Entrada e saída de dados

Exemplo das linhas da entrada

```
N T // Qtde de pontos de interesse, Qtde de trechos possiveis.
VT_1 VT_2 VT_3 ... // Valores turísticos de cada ponto de interesse
p_i p_j c_t // Proposta de trechos interligando o ponto de interesse i \\
               ao ponto de interesse j com custo c_t
...
```

Exemplo da saída

```
C A // Custo mínimo para construção da ciclovia e atratividade agregada
qtde_trechos_P1 qtde_trechos_P2 qtde_trechos_P3 qtde_trechos_P4 ...
// sequência da quantidade de trechos de cada ponto de interesse para configuração
    da ciclovia selecionada
// Relação dos trechos selecionados para ciclovia no mesmo formato da entrada
```

Exemplo prático

Entrada do exemplo da seção 3

```
10 18
25 40 65 50 55 70 40 40 30 95
1 0 410000
2 1 350000
3 2 250000
4 3 100000
5 2 250000
5 3 250000
5 4 250000
6 5 50000
7 0 420000
7 6 125000
8 0 220000
8 7 220000
9 0 250000
9 1 350000
9 2 90000
9 5 125000
9 6 320000
9 7 90000
```

Saída do exemplo da seção 3 - Configuração R2

```
1495000 1060 // Custo mínimo para construção da ciclovia e atratividade agregada
1 1 1 1 2 3 1 2 2 4 // Qtde de trechos da ciclovia partindo/chegando no ponto de interesse
4 3 100000
5 4 250000
6 5 50000
8 7 220000
8 0 220000
9 2 90000
9 7 90000
9 5 125000
9 1 350000
```

Estruturas de dados e algoritmos utilizados:

Eu separei a lógica do trabalho em três etapas: ingestão, árvore de consumo mínimo e apresentação dos resultados.

Como o tp tem regras claras e específicas para entrada e saída de dados foi importante atender esse fluxo para atender a todos os requisitos.

Primeiro, realizei o desenvolvimento do algoritmo de Kruskal para a realização da lógica da árvore de consumo mínimo. Com isso, identificando as estruturas de dados para realização do trabalho

Identifiquei quatro estruturas:

- Spot: que representa os pontos turísticos e suas características - índice e valor turístico
- Edge: que representa o vínculo entre dois spots
- Graph: onde é armazenado todas os vínculos (edges)
- Subset: para auxiliar na lógica do algoritmo de Kruskal na parte de ranqueamento e vínculos que atendem as regras de negócio

Com o algoritmos de Kruskal e as estruturas implementadas fizemos apenas um ajuste no algoritmo. No momento onde verificamos a primeira métrica de ranqueamento - custo do trecho - analisamos também, em caso de empate na primeira métrica, o valor turístico do trecho. Assim, atendendo a regra de negócio para o algoritmo de árvore de consumo mínimo.

Em seguida, foi feito implementações para captura da entrada de dados de forma correta e a saída de dados de forma correta.

Para a saída de dados foi utilizado um map contendo como chave o par xy que compõe o trecho. Dessa forma, simplifica as lógicas de ordenação e contagem de vínculos por ponto turístico.

Análise de complexidade de tempo assintótica da solução:

1. Construção dos spots O(n) onde n é o número de pontos turísticos;
2. Atribuição dos possíveis trechos, vínculo entre um ponto turístico e outro, para a construção da ciclovia O(m) onde m é o número de possíveis trechos;
3. Execução do algoritmo de Kruskal é O(m log n) onde n é o número de pontos turísticos e m é o número de possíveis trechos;

Para um grafo com E arestas e V vértices, o algoritmo de Kruskal pode ser mostrado para rodar no tempo O (E log E), ou equivalentemente, no tempo O (E log V), todos com estruturas de dados simples. Esses tempos de execução são equivalentes porque

- E é no máximo V^2 e log V^2, dessa forma, 2 log V que é igual a log V
- Cada vértice isolado é um componente separado da floresta de abrangência mínima. Se ignorarmos vértices isolados, obtemos V <= 2 E , então log V é O(log E)

Primeiro vamos classificar as arestas por peso usando uma classificação por comparação em tempo O (E log E). Isso faz com que a etapa “remova uma aresta com peso mínimo de S” para operar em tempo constante.

Agora vamos usar uma estrutura de dados com conjuntos separados para controlar quais vértices estão em quais componentes. Colocando cada vértice em seu próprio conjunto disjunto, que leva operações O (V).

Dessa forma, no pior caso, precisamos iterar por todas as arestas e, para cada aresta, precisamos fazer duas operações de localização e possivelmente uma união. Mesmo uma estrutura de dados com conjuntos separados simples, com união por classificação pode realizar O(E) operações em tempo O(E log V). Assim, o tempo total é:

O(E log E) = O(E log V).

Compilação e execução:

Compilador: g++

Make:

make - build
make test - execução de testes
make clean - remoção dos binários gerados

Execução:

./tp02