Relatório Trabalho de Desenvolvimento

Programação Linear e Grafos

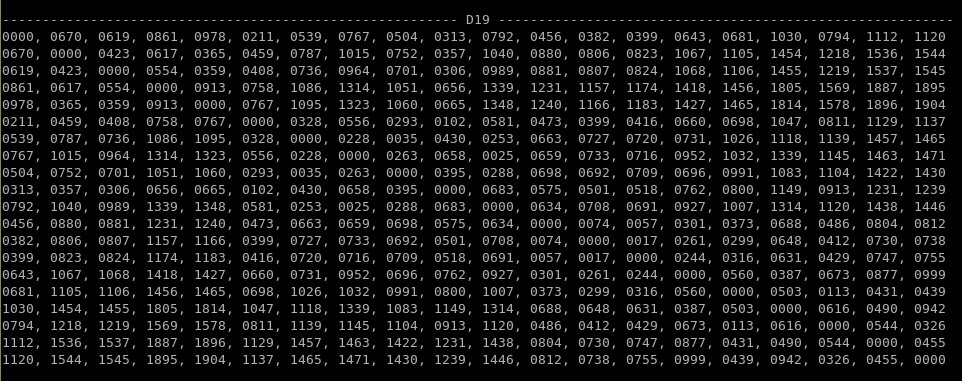
Geison Machado da Silva

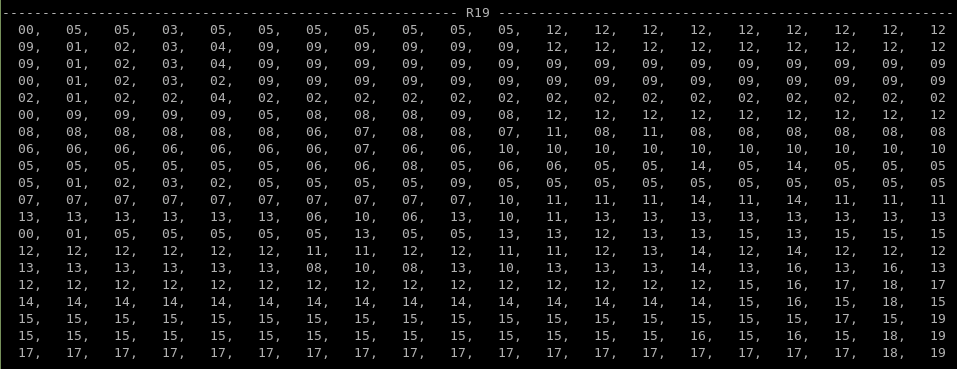
**1 Análise do Problema**

Podemos considerar cada cidade como um vértice e o caminho entre elas como as arestas. Assim para calcular a menor distância entre duas cidades basta aplicar o algorítimo de Floyd na matriz iterando 19 vezes, porque são 20 cidades.

**2 Resultados Obtidos**

A matriz de até 19 passos ficou da seguinte maneira:

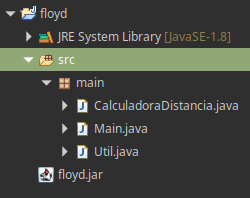
****

A matriz de roteamento de até 19 passos ficou da seguinte maneira:

Assim, por exemplo, pode-se perceber que que para ir da cidade 1 até a 15 o custo é de 643 e deve-se passar pelas cidades 1 → 13 → 14 → 15. Olhando visualmente no mapa dá a impressão que não é o caminho mais curto, mas se somar os valores na D1 de 1 a 13, 13 a 14 e 14 a 15 pode-se ver que o custo é o menor.

**3 Construção do Código**

A estrutura do projeto ficou simples:



Utilizou-se apenas 3 classes dentro de um pacote:

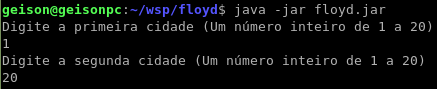
* Main.java: Lê a entrada das duas cidades pelo usuário.
* Util.java: Classe utilitária que contém as matrizes *hardcoded* no código, métodos de impressão, etc.
* CalculdadoraDistancia.java: Classe princípal com o *core* do programa que contém o método principal *calculaDistancia()* que emula o algorítimo que vimos em aula para calcular Floyd.

No código fonte do programa foi todo documentado cada ação do algorítimo explicando como funciona.

**3 Execução do programa**

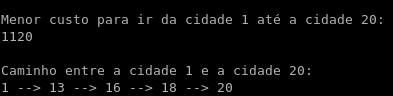
Na pasta raiz do software tem um arquivo chamado **floyd.jar.**

Ele pode ser executado com o comando **java -jar floyd.jar.**

****

Após pedir para digitar as duas cidades, o programa vai imprimir na tela todas as matrizes, de distância e roteamento, de D1 até D19, de R1 até R19.

No final ele diz o custo e o menor caminho para navegar entre as cidade que o usuário digitou:

****