Documentação do Trabalho Prático da disciplina de Algoritmos

Hector Julio Pardinho Soares

hectorjpsoares@gmail.com

1. Introdução

Este trabalho visa criar um sistema para alocação de clientes em lojas mais próximas, dado algumas condições específicas. Esta situação caracteriza-se como um problema de casamento estável, e os algoritmos utilizados visam o melhor desempenho para resolução de tal problema. A implementação foi baseada em criar entidades para clientes e lojas, além de uma classe para auxiliar na leitura do arquivo de entrada. O código foi implementado em C++ em um ambiente Linux

O código está organizado na seguinte estrutura:

/bin

/include

/infra

/src

Pasta **bin**: Onde esta localizado o arquivo executável gerado ao rodar o MakeFile;

Pasta **include**: Onde estão os .h com a estrutura de cada classe criada no projeto;

Pasta **src**: Onde estão todas as implementações das classes definidas na pasta include;

Pasta **test**: Onde estão todos os inputs e outputs disponibilizados para teste;

2. Modelagem computacional

O projeto foi dividido em três classes. Duas delas são entidades que caracterizam elementos do problema, logo foram criadas classes para cliente e loja onde ambos possuem coordenadas como um dos seus atributos. A outra classe atua como serviço, uma para auxiliar na alocação dos dados da entrada. O arquivo main conta apenas com a função main e a lógica de alocação.

3. Instruções de compilação e execução

Para a execução do programa, basta ter o MingW instalado e extrair os arquivos do zip disponibilizado em um diretório qualquer e ,em seguida, rodar o comando “make” na raiz do diretório. Ao executar o comando, irá criar um executável na pasta ‘bin’. Basta executar este arquivo criado passando como parâmetro o caminho do arquivo de entrada e o número de linhas (na pasta raiz, ficaria: ./bin/tp01.exe ./test/caso_teste_01.txt).

4. Implementação

Sobre os algoritmos implementados, temos a seguinte estrutura e lógica: Duas estruturas são inicializadas e populadas com os dados lidos do documento input selecionado, são elas: Client e Store. Cada uma dessas estruturas são Arrays que guardam as informações da loja(store) e as informações dos clientes(clients). Em seguida, os clientes são ordenados

pelas especificação ditas no trabalho utilizando o método sort e uma condição especial enviada no terceiro parâmetro para que as pessoas sejam ordenadas por prioridade. Dessa forma, temos um fila de pessoas ordenadas pela prioridade, independente das distâncias das lojas. Com os clientes ordenados e cada pessoa possuindo uma lista de prioridade, o restante do código vai alocando a pessoa mais prioritária na loja mais próxima e assim por diante.

5. Análise de complexidade

- getStores: Utilizado para a interpretação das linhas das lojas percorre todo o arquivo de texto de entrada e extrai as lojas cadastradas de acordo com as regras especificadas no roteiro do trabalho. Sua complexidade é O(n) onde n é o número de linhas do arquivo de entrada.
- getClients: Utilizado para a interpretação das linhas das lojas percorre todo o arquivo de texto de entrada e extrai as lojas cadastradas de acordo com as regras especificadas no roteiro do trabalho. Sua complexidade é O(n) onde n é o número de linhas do arquivo de entrada.
- sort: Essa função organiza os clientes de acordo com a prioridade. A operação é um laço que percorre n vezes e sua complexidade é O(n *logn).
- removeClient: Esse método serve para desalocar o cliente de uma loja para que um cliente com uma prioridade maior seja alocado no lugar dele. Como esse método retira o cliente da lista e reorganiza o vetor de clientes dentro da loja de forma que a última posição do

vetor fique disponível, sua complexidade é de O(n) onde n é o número de clientes que a loja consegue alocar.