FUNDAÇÃO HERMÍNIO OMETTO

Engenharia da Computação – 7º Período

Guilherme Cavenaghi – RA 109317

Rafael Pereira de Souza - RA 109680

Vinicius Rossi – RA 110273

RELATÓRIO SISTEMA GRAFO METROVIARIO

Araras – SP

2023

1. Objetivo

O objetivo do trabalho é implementar um código que controle as linhas de um metrô, para isso a melhor maneira é representar essa linha através de um Grafo, basicamente um modelo que representa relações entre objetos e permite ao programador solucionar problemas e desenvolver códigos mais eficientes.

2. O código

Definimos uma classe chamada MetroSystem que representa um sistema metroviário. Essa classe encapsula todas as funcionalidades relacionadas ao sistema, como a criação do grafo, adição e remoção de estações, busca em largura, busca em profundidade, busca de caminho, busca do menor caminho, busca da árvore geradora mínima e impressão do sistema.

As funcionalidades definidas nessa classe foram:

addStation(): Permite adicionar uma nova estação ao sistema. O usuário informa o nome da estação e suas conexões com as estações existentes.

removeStation(): Permite remover uma estação do sistema. O usuário informa o nome da estação a ser removida.

breadthFirstSearch(): Realiza uma busca em largura a partir de uma estação inicial, exibindo as estações visitadas.

depthFirstSearch(): Realiza uma busca em profundidade a partir de uma estação inicial, exibindo as estações visitadas.

findPath(): Encontra um caminho entre duas estações no sistema metroviário.

findShortestPath(): Encontra o menor caminho entre duas estações no sistema metroviário.

findMinimumSpanningTree(): Encontra a árvore geradora mínima do sistema metroviário.

printMetroSystem(): Permite escolher a forma de impressão do sistema metroviário, podendo ser em lista de adjacência ou matriz de adjacência.

printAdjacencyList(): Imprime o sistema metroviário em forma de lista de adjacência.

printAdjacencyMatrix(): Imprime o sistema metroviário em forma de matriz de adjacência.

Seguindo para a struct Station onde representa uma estação, contendo o nome da estação e uma lista de conexões com outras estações, representadas por pares (índice da estação, tempo necessário para ir de uma estação para outra).

stations é um vetor de Station, que armazena todas as estações do sistema.

adjacencyList é um vetor de vetores de pares, que representa a lista de adjacência do grafo do sistema metroviário. Cada índice do vetor representa uma estação, e cada elemento do vetor interno representa uma conexão com outra estação.

adjacencyMatrix é uma matriz de inteiros, que representa a matriz de adjacência do grafo do sistema metroviário. O valor -1 indica que não há uma conexão direta entre as estações.

Foi criado o grafo das linhas metroviárias usando a função createGraph(), solicitando ao usuário o número de estações, seus nomes e as conexões entre elas, e preenchendo os membros de dados stations, adjacencyList e adjacencyMatrix com as informações fornecidas.

3. Manipulação de dados

O código solicita ao usuário que digite o número de estações do sistema metroviário. Essa abordagem permite criar um grafo com o tamanho adequado para acomodar todas as estações e suas conexões.

O uso do loop while com a função cin -> numStations garante que apenas um número inteiro válido seja aceito como entrada. Caso o usuário insira um valor inválido e solicita uma nova entrada.

O uso de cin.clear() e cin.ignore() garante que quaisquer caracteres inválidos sejam limpos.

Inicialização das estruturas de dados:

Após obter o número de estações, o código redimensiona os vetores stations, adjacencyList e adjacencyMatrix para terem o tamanho correto.

Obter o nome de cada estação:

O código solicita ao usuário que digite o nome de cada estação individualmente e armazena os nomes no vetor stations.

A iteração ocorre de 0 a numStations - 1, onde i representa o índice da estação sendo lida.

Para se adicionar as conexões é iterado sobre cada estação e solicita ao usuário o tempo necessário para a viagem entre a estação atual e todas as outras estações do sistema.

A verificação (if (i != j)) e garante que não haja uma conexão entre uma estação e ela mesma.

O uso de um loop while com a função cin -> weight garante que apenas um número inteiro válido seja aceito como entrada para representar o tempo de viagem.

As informações das conexões são armazenadas tanto na lista de adjacência, quanto na matriz.

4. O sucesso

Ao finalizar a criação do grafo, o código exibe uma mensagem informando que o grafo foi criado com sucesso.

Essas decisões de implementação foram tomadas para garantir a integridade e a consistência dos dados inseridos pelo usuário. Além disso, a utilização de estruturas de dados adequadas (vetor de estações, lista de adjacência e matriz de adjacência) permite representar de forma eficiente as informações do sistema metroviário e suas conexões.