UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UNIVERSIDADE FEDERAL DE MINAS GERAIS

Instituto de Ciências Exatas | Departamento de Ciência da Computação

Sistemas de Informação 2023/1

João Vitor Soares Santos

Documentação do Trabalho Prático Google Maps

Belo Horizonte

2023

Introdução

Em uma cidade existem restaurantes e ruas, as ruas são representadas por coordenadas (x,y) que podem ser pavimentadas (1) ou não-pavimentados (0), e também há de se considerar que se duas coordenadas adjacentes são pavimentadas, então o trecho entre elas também é. O arquivo "ruas.txt" detém as informações de todas as ruas pavimentadas ou não pavimentadas (x,y,pavimentado). Além disso, todos os restaurantes são acessíveis por um único caminho, eles podem ser baratos ou caros e possuem apenas um entregador. O arquivo "restaurantes.txt" possui todas as informações sobre os restaurantes (x,y,nome,custo,velocidade). A distância entre duas coordenadas adjacentes é de um "zambs" e a velocidade dos motoboys é medida em "zambs/minuto", e com todas essas informações, objetivo do programa é calcular a distância de uma coordenada escolhida pelo usuário e fornecer por ordem de agilidade quais restaurantes entregam mais rápido, além do usuário poder escolher preferencias de preço e de tempo máximo, tudo isso considerando que a coordenada fornecida pelo usuário está em um trecho pavimentado.

Proposta de solução

A princípio, o programa começa com a leitura do arquivo **"ruas.txt"**, que são armazenadas na estrutura de dados **Cidade** que nela contêm a uma matriz da estrutura de dados **Rua,** cada rua é representa por uma posição na matriz bidimensional **r[ ][ ]**, contendo informações como coordenadas (x,y) e se possui pavimento. A seguir é feita a leitura dos restaurantes através do arquivo **"restaurantes.txt"** que são armazenados em um arranjo da estrutura de dados **Restaurante**, que compreende informações como nome, preço, coordenadas (x,y), velocidade do motoboy e distância, sendo todas elas extraídas do arquivo, com exceção da última que será calculada posteriormente.

Após a leitura dos arquivos e a estruturação dos dados, o programa entra em um loop condicional usando a estrutura de controle **"do-while"**. Essa estrutura garante que o usuário tenha a oportunidade de interagir com o laço no mínimo uma única vez. Sendo assim, por questões de praticidade, o mapa da cidade é impresso em formato de matriz pela função do tipo **void imprimirPlanoCartesiano(Cidade c)**, que recebe como **parâmetro** um registro do tipo **Cidade**. Dentro do loop, o programa solicita ao usuário que informe as coordenadas da sua casa, representada pelas variáveis inteiras **x** e **y**. Essas coordenadas são usadas para verificar se o endereço apresentado corresponde a alguma rua pavimentada a matriz **Cidade**, caso o trecho apresentado seja inexistente ou de alguma rua não-pavimentada o programa exibe uma mensagem de erro e solicita ao usuário outro endereço válido.

Após a validação de endereço, o programa utiliza o laço "**for"** que se repete o número armazenado na variável **qtd_restaurantes**, assim, calculando a distância entre cada restaurante e a localização do usuário. O programa calcula a distância utilizando um algoritmo baseado no conceito de empilhamento, implementado pela estrutura de dados **Caminho** e seu arranjo **esquinas[ ]**. Este arranjo representa as esquinas da cidade, e em cada elemento está armazenado coordenadas (x,y), e informações sobre possíveis novos caminhos, que são representados pela rosa dos ventos (norte, sul,

leste, oeste), cada direção recebendo o caractere **'S'** se aquela direção já foi explorada ou se ela é impossível, e o caractere **'N'** se é uma direção inexplorada. Com essa estrutura de dados o programa calcula todas as rotas possíveis entre a casa do usuário e o restaurante, a cada movimento para uma nova esquina a variável **distancia** é incrementada e a rua é adicionada ao arranjo **esquinas[ ],** e caso o programa encontre em um beco sem saída a distância é decrementada e o elemento atual é desempilhado do arranjo **esquinas[**], esse processo se repete até o programa encontrar um caminho apropriado ou em último caso, não encontrar nenhum. Quando o algoritmo alcança o restaurante almejado, a distância final é armazenada na estrutura de dados **Restaurante** correspondente, caso o algoritmo tenha percorrido todo mapa e não tenha achado o respectivo restaurante e sua única alternativa tenha sido voltar para a coordenada original, o valor –**1** armazenado, representando que alcançar tal restaurante é impossível.

Após o cálculo de todas as distancias, o programa solicita ao usuário sua preferência de custo, que pode ser **"BARATO"** ou **"CARO"**, e com base nessa preferência e com a ajuda da função do tipo **int preferenciaCusto (Restaurante r[],int n, char *p, Restaurante copia[])** que recebe como parâmetros um arranjo de Restaurante, o tamanho do respectivo arranjo, uma string contendo a preferência, e um arranjo que receberá por referência todos os restaurantes correspondentes a preferência de preço e retorna o tamanho do arranjo filtrado, logo em seguida com o auxílio da função tipo **void organizarPorRapidez(Restaurante r[], int n)** que recebe como parâmetros um arranjo do tipo **Restaurante** e o tamanho do mesmo, e organiza por ordem de rapidez (do menor ao maior), sendo a rapidez calculada através da divisão entre distância e velocidade do motoboy, vale ressaltar que restaurantes com distancia –**1**, ou seja impossíveis de alcançar a partir do endereço do usuário, são desconsiderados pela função **preferenciaCusto** e pela função que veremos a seguir **preferenciaTempo**, e como consequência não serão impressos . Após a filtragem, os restaurantes são impressos pela função do tipo **void imprimirRestaurantes(Restaurante r[], int n)**, que recebe como parâmetros um arranjo do tipo Restaurante, e o tamanho do respectivo arranjo. Em seguida o programa pede ao usuário que informe novamente a preferência de custo e o tempo limite que ele está disposto a esperar em minutos, e com base nessa informação os restaurantes são novamente filtrados pela função **preferenciaCusto** e agora pela função do tipo **int preferenciaTempo(Restaurante r[], int n, float limite, Restaurante copia[]),** que recebe como parâmetros um arranjo de Restaurante, o tamanho do respectivo arranjo, o limite de tempo, e um arranjo que receberá por referência todos os restaurantes que estão dentro das preferencias e retorna a dimensão do arranjo filtrado, e após isso, com a ajuda da função **imprimirRestaurantes**, os restaurantes filtrados são impressos.

Ao final de cada iteração do loop, o programa pergunta ao usuário se ele deseja continuar interagindo com o programa. Se o usuário digitar **"1"**, o loop é reiniciado e o usuário pode fornecer novas coordenadas, preferências de custo e limite de tempo. Caso contrário e o programa é encerrado.

Discussão dos resultados

Durante a execução do programa desenvolvido, foram obtidos resultados satisfatórios em relação ao cálculo da distância entre determinados restaurantes e a casa do usuário. Porém, existem

casos específicos em que não existe caminho até determinado restaurante, nesses casos o valor retornado é **-1**, que representa a impossibilidade de alcançar determinado restaurante.

Questão 1 – Para calcular a distância em zambs da casa do usuário a todos os restaurantes, foi usado um algoritmo de empilhamento. O algoritmo começa na casa do usuário e verifica as possíveis direções para se mover. Em seguida, ele escolhe uma direção não explorada, marca essa direção como explorada e avança para a próxima esquina. O processo continua até que o algoritmo alcance o restaurante desejado ou chegue a uma rua sem saída. Durante o percurso, o algoritmo mantém uma pilha (implementada pela estrutura de dados Caminho) para rastrear as esquinas visitadas e calcular a distância percorrida. Quando o algoritmo alcança o restaurante desejado, a distância percorrida é armazenada na estrutura de dados Restaurante correspondente. Em caso de um caminho até certo restaurante não existir, o algoritmo percorrerá todas as ruas e quando perceber que não há caminhos restantes ele removera todos as ruas da pilha, só sobrando a coordenada original, nesse caso ele armazena o valor –**1** na distância do restaurante correspondente.

Questão 2 – Para retornar um arranjo de acordo com uma preferência de preço (Caro ou barato) foram utilizadas duas funções, primeiro a **int preferenciaCusto(Restaurante r[],int n, char *p, Restaurante copia[])**, que retorna por referência o arranjo filtrado de acordo com o preço escolhido e também o seu tamanho. Por fim, a função **void organizarPorRapidez(Restaurante r[], int n)** organiza por ordem de rapidez do menor ao maior um determinado arranjo do tipo **Restaurante**.

Questão 3 – Com o uso das funções usadas na questão dois (preferenciaCusto e organizarPorRapidez), o diferencial é a função **int preferenciaTempo(Restaurante r[], int n, float limite, Restaurante copia[])**, que filtra um arranjo do tipo **Restaurante** de acordo com um limite de tempo em minutos, selecionando apenas tempos menores que o máximo, e desta forma retornando por referência o arranjo filtrado e seu tamanho.