PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

CEATEC – CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIA

ENGENHARIA DE SOFTWARE

RELATÓRIO

Projeto Grafos

CAMPINAS

2022

APRESENTAÇÃO DO PROBLEMA

O código é um ajudante para planejar os caminhos mais rápidos e melhores entre os prédios da PUCC (Pontifícia Universidade Católica de Campinas). O usuário digita os o prédio de origem e o prédio de destino, com base nisso o app calcula o tempo que leva para ir de um ponto a outro com base no números de prédios cadastrados e gera o menor caminho com base no algoritmo Dijkstra para descobrir os caminhos mais rápidos.

Quando o usuário clica no botão "Calcular", o código pega as informações digitadas, cria um mapa de conexões entre os prédios e roda o algoritmo de Dijkstra para encontrar as distâncias mínimas de um prédio inicial para todos os outros prédios.

Os resultados são mostrados numa caixa de texto, com o tempo dos caminhos mais rápidos entre os prédios. Assim, o usuário pode ver quais são os caminhos mais rápidos para se locomover entre os prédios.

PONTOS DE MAIOR DIFICULDADE

Algumas partes do código como adicionar, buscar e percorrer esses blocos foi um desafio, e a implementação e adaptação do algoritmo de Dijkstra e mexer com essas estruturas de dados nos trouxe um pouco de dificuldade. Mas, no geral após um tempo de aprendizagem conseguimos implementar o algoritmo.

DISCUSSÃO DE SOLUÇÕES

A solução utilizada é conhecida como "Algoritmo de Dijkstra". Essa solução consiste em um algoritmo para encontrar o caminho mais curto entre dois pontos em um grafo ponderado. No contexto específico do problema proposto, o grafo representa os prédios, e as arestas possuem pesos associados aos tempos de percurso entre os prédios.

O algoritmo de Dijkstra começa atribuindo um valor inicial de distância infinita para todos os prédios, exceto o prédio de partida, que recebe uma distância inicial de zero. Em seguida, ele itera sobre os prédios, sempre selecionando o prédio com a menor distância atual para explorar seus vizinhos.

Para cada prédio selecionado, o algoritmo atualiza as distâncias dos seus vizinhos se encontrar um caminho mais curto do que o atualmente registrado. Essa atualização é feita comparando a distância atual do prédio vizinho com a soma da distância atual do prédio selecionado e o peso da aresta que conecta os dois.

Esse processo de atualização das distâncias é repetido até que todos os prédios tenham sido visitados ou até que a distância para o prédio de destino seja mínima.

A solução implementada no Android Studio utiliza o algoritmo de Dijkstra para encontrar os caminhos mais rápidos entre os prédios com base nos tempos de percurso fornecidos pelo usuário. A implementação foi adaptada para permitir a interação com a interface de usuário e exibir os resultados de forma adequada no aplicativo móvel. O algoritmo de Dijkstra foi escolhido porque é muito bom em encontrar os caminhos mais curtos em um mapa. Ele leva em conta os tempos de percurso entre os prédios e consegue determinar a rota mais rápida.

IMPLEMENTAÇÃO DO CÓDIGO

Para implementar o código, foi definido as estruturas de dados necessárias. Foi criada uma classe chamada "Graph" para representar o grafo e armazenar as informações sobre os prédios e os tempos de percurso entre eles. Usamo um HashMap para mapear cada prédio a uma lista de seus vizinhos e os respectivos tempos de percurso.

Em seguida, foi implementado a função do algoritmo de Dijkstra dentro da classe Graph. Essa função recebeu como parâmetros do prédio de partida e retornou um HashMap contendo os tempos dos caminhos mais rápidos para cada prédio.

No Android Studio, foi implementado uma interface e suas funções (EditText, TextView, Button)

Para lidar com o evento de clique do botão, implementamos uma callback no código. Quando o botão é clicado, recupera os dados inseridos pelo usuário nos campos de entrada. Em seguida, a função do algoritmo de Dijkstra é chamada passando o prédio de partida e o prédio de chegada os dados do grafo.

Após a execução do algoritmo de Dijkstra, formatamos os tempos dos caminhos mais rápidos em uma string. Então, atribuímos essa string ao TextView para que os resultados fossem exibidos na interface do usuário.

Referência bibliográfica:

Basic graph algorithms in Kotlin - Piyush Chauhan (2023)

Kodeco data-structures-algorithms-in-kotlin - v1.0

Blog DevGenius Kotlin-graphs