**Atividade Avaliativa Google Maps**
**Aluno: Pedro Muhamad Suleiman Craveiro**
**RA: 193934**
**Curso: Engenharia de Computação**
**Semestre: 4° Semestre**

**Processo de Desenvolvimento:**
O projeto consiste na criação de um programa para encontrar o menor caminho entre dois pontos de um mapa utilizando o algoritmo de busca pelo menor caminho em grafos com pesos. Dessa forma, a ideia foi inspirada por meio de um vídeo do canal do professor Hemerson Pistori “Caminho de custo mínimo usando Dijkstra”, juntamente com os conhecimentos da biblioteca Folium, somado a aula ministrada no dia 26/09/2023 a respeito da criação de interfaces gráficas em Python. Ademais, o cenário escolhido foi a cidade de Araranguá, Santa Catarina, considerando a rota entre a UFSC Campus Araranguá, minha antiga casa e a cidade em si. Além disso, um importante fator é que os pontos de decisão foram previamente traçados, levando em consideração características como o tipo de caminho (a pé ou de carro) e os pesos associados a cada rota.

Em seguida, detalharei as seguintes funções desenvolvidas no código:
Função calcular_rota:
Esta função é responsável por calcular a rota com base no modo de transporte escolhido (carro ou a pé). Primeiro, cria um grafo usando a biblioteca networkx, adiciona vértices e arestas com pesos correspondentes às coordenadas dos pontos da cidade. As coordenadas foram retiradas do Google Maps, onde todos os pontos de tomada de decisão foram traçados e definidos em torno do trajeto com objetivo de sair do ponto A (minha antiga casa) e o ponto I (a universidade). Dependendo do modo de transporte escolhido, define diferentes arestas com diferentes pesos, o detalhamento dos pesos foi pensado com base na estrutura das ruas, como por exemplo: se a rua é adequada para o transporte de carro, se possuía calçada para os pedestres, entre outros. Em seguida, utiliza a função encontrar_menor_caminho para encontrar o menor caminho entre os dois pontos (casa e UFSC). Posteriormente, cria um mapa interativo usando a biblioteca folium e exibe os pontos e a rota do menor caminho. O mapa é salvo em um arquivo HTML que permite o usuário ter essa visualização.

Função encontrar_menor_caminho:
Essa função implementa um algoritmo de busca em largura (BFS) personalizado para encontrar o menor caminho entre dois pontos (origem e destino) em um grafo ponderado. O grafo é representado por um dicionário G, onde as chaves são os nós e os valores são dicionários que representam as arestas com os pesos associados.

A função utiliza uma fila de prioridade para explorar os nós de forma ordenada, priorizando os caminhos de menor peso acumulado. A tupla na fila contém o nó atual, o caminho percorrido até esse nó e o peso total acumulado até esse ponto.

Aqui estão as principais etapas da função:

Inicialização da fila:
- fila: Uma lista que contém tuplas no formato nó, caminho, peso total). Inicialmente, a fila contém uma única tupla representando a origem.

Exploração:
- A função entra em um loop enquanto a fila não estiver vazia.
- A fila é ordenada com base no peso total acumulado, para que os caminhos mais curtos sejam explorados primeiro.

Pop da fila:
- A tupla com o menor peso total acumulado é removida da fila, representando o próximo nó a ser explorado.

Verificação de visitados:
- O nó é marcado como visitado para evitar ciclos.

Verificação de destino:
- Se o nó atual for o destino, a função retorna o caminho encontrado.

Exploração de vizinhos:
- Os vizinhos do nó atual são explorados.
- Para cada vizinho não visitado, um novo caminho é formado adicionando o vizinho ao caminho existente.
- O peso da aresta entre o nó atual e o vizinho é considerado no cálculo do novo peso total acumulado.
- Esses novos caminhos são adicionados à fila para posterior exploração.

Conclusão:
- Se nenhum caminho for encontrado após explorar todos os nós possíveis, a função retorna None.

Essa função pode ser útil para encontrar o menor caminho em grafos ponderados, onde o peso das arestas é levado em consideração. O parâmetro modo pode ser utilizado para ajustar o peso das arestas de acordo com a necessidade específica do problema.

Função mostrar_grafo:
Esta função utiliza a biblioteca folium para criar um mapa interativo exibindo os pontos e conexões do grafo. Define coordenadas para os pontos e arestas para representar as conexões entre eles. Adiciona marcadores para os pontos e linhas

para as conexões. O mapa é salvo em um arquivo HTML chamado 'grafo-rotas.html'.

Interface Gráfica (tkinter):
O código cria uma interface gráfica simples usando a biblioteca tkinter. Permite ao usuário escolher entre os modos de transporte (carro ou a pé) por meio de botões de seleção. Além disso, há botões para calcular a rota e exibir o grafo.

Em resumo, o código integra funcionalidades de criação e visualização de grafos, cálculo de rotas mínimas e interação com o usuário por meio de uma interface gráfica.

**Materiais Consultados:**

- Biblioteca Python: networkx
- Biblioteca Python: folium
- Biblioteca Python: webbrowser
- Biblioteca Python: os
- Biblioteca Python: tkinter
- Vídeo do Professor Hemerson Pistori a respeito do conteúdo de Caminho de Custo Mínimo: Caminho de custo mínimo usando Dijkstra

**Ajudas Recebidas:**
Dúvidas específicas foram sanadas por meio de consultas ao professor e interações com colegas, discutindo as melhores formas de implementação, questionamento sobre bibliotecas e a linguagem de programação Python. Além do mais, foram realizados testes do funcionamento do código com os colegas de laboratório.

**Estimativa de Tempo de Desenvolvimento:**
A estimativa de tempo foi realizada com base nos commits no GitHub e o desenvolvimento em sala de aula, permitindo uma visão cronológica de todo o processo. Conclui-se que o tempo de desenvolvimento foi em torno de 25 horas.

**Código:**
Foi utilizado o modelo GPT-3.5 da OpenAI (ChatGPT) como ferramenta de auxílio na elaboração do código, correção de funções e lógicas do algoritmo.

É de suma importância que para utilizar o código você deve instalar as seguintes bibliotecas no terminal: ` pip install networkx folium tk `

Link para o Repositório:
O código está disponível em link para o repositório no GitHub (Github - Atividade Avaliativa Google Maps (Pedro M. S. Craveiro)).