Algerian People's Democratic
Republic Ministry of Higher Education
The Higher School of Computer and Digital Sciences and Technologies

Report about:

THE SHORTEST PATH
between two places in algeria

Prepared by:

Miss Nehili Hiba

Miss Ait Baziz Siham

Supervised by:

Dr Lekehali Somia

Year of Submission:

2023/2024

Table of Contents

1. Introduction

In our daily lives, and especially in the life of a programmer, search algorithms are among the most important and widely used algorithms. These algorithms are crucial for solving a variety of problems, from finding information on the internet to navigating through maps. Implementing search algorithms can transform simple Python code into powerful tools that create amazing solutions.

In this project, we focus on implementing the A* algorithm to find the shortest path between two places. The A* algorithm is a popular and efficient search algorithm that combines the strengths of Dijkstra's algorithm and Greedy Best-First-Search. By using the A* algorithm, we can efficiently find the optimal path on a map, considering both the cost to reach a node and the estimated cost to reach the goal from that node.

so, our goal is to demonstrate a real-world application of the A* algorithm by finding the shortest path between two places on a map in the Algerian region. We will utilize open-source tools and libraries to transform map data into a graph representation, apply the A* algorithm to find the shortest path, and visualize the result. This project showcases the practical importance of search algorithms and their implementation, highlighting how they can solve real-world navigation problems effectively and efficiently.

2. Tools and Libraries

In this project, we utilized several key tools and libraries to implement the A* algorithm and visualize the shortest path between two locations on a map. We used:

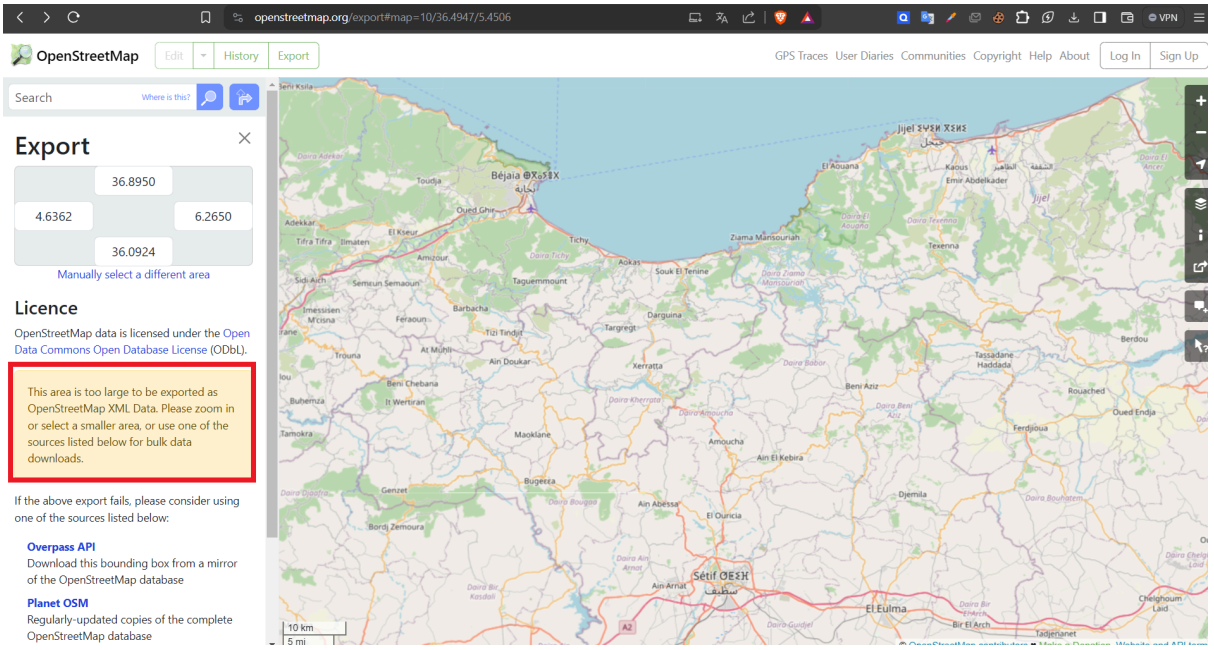
- **Python**: as the primary programming language for implementation.
- **Tkinter**: for creating the graphical user interface.
- **OSMnx**: to acquire and process map data from OpenStreetMap.
- **NetworkX**: for graph operations and implementing the A* algorithm.
- **Matplotlib**: for visualizing the graph and the shortest path.



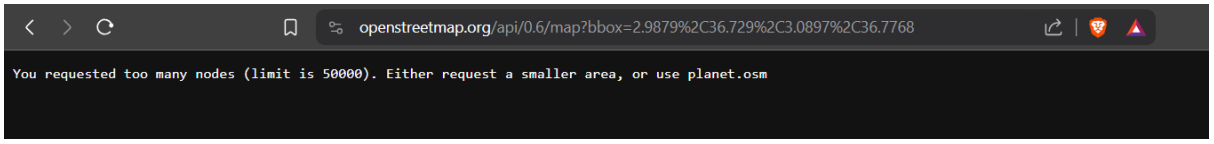
3. Methodology

3.1 Gather Map Data:

- OpenStreetMap (OSM): Use OpenStreetMap to obtain detailed map data for the selected region in Algeria.
- Issue Encountered: While attempting to extract the required map data, we encountered an issue with a message indicating that the selected area is too large to export in the OSM XML format, requiring us to either zoom in on a smaller area or use alternative sources for bulk data downloads.



- Adjustment: To address this issue, we adjusted our approach to calculate the shortest path between two places rather than two cities of a region. Specifically, we focused on the city of Algiers and Bejaia, However, despite this adjustment, we encountered another obstacle as we requested too many nodes, exceeding the limit of 50,000 nodes.
- Adjustment: To address the issue of exceeding the node limit, we refined our approach by requesting a smaller area for data extraction. Specifically, we narrowed our focus to key locations within Algiers and Bejaia, namely El Kasbah for Algiers and the port for Bejaia.



3.2 Convert Map Data into Graph Format:

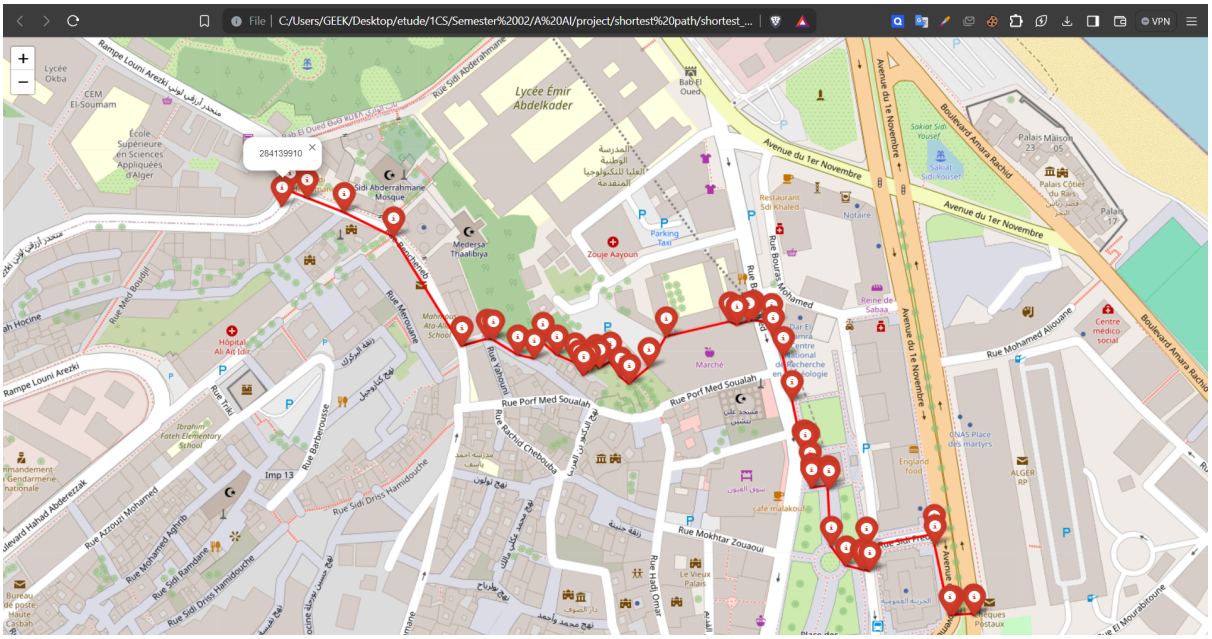
- OSMnx Library: Use OSMnx, a Python library, to convert the OSM data into a graph representation.
    - Nodes: Represent intersections, landmarks, or specific locations on the map.
    - Edges: Represent the roads or paths connecting these nodes, with attributes like distance (length).
- Save and Load: Save the graph representation for future use, typically in a GraphML format.

3.3 Implement the A Algorithm*:

We use the A* algorithm to find the shortest path between two points in the graph. In our implementation, we employ a heuristic function to estimate the distance from each node to the goal node. This function helps guide the algorithm towards the goal efficiently. By iteratively exploring neighboring nodes and updating the best path found, we navigate through the graph effectively. This process continues until we reach the goal node, ensuring that we find the shortest path.

3.4 Visualize the Shortest Path:

To visualize the shortest path, We retrieve the coordinates of each node along the shortest path and initialize a Folium map centered at the starting node's coordinates. Then, we iterate through each node along the path, adding a marker with a popup displaying the node's label. Additionally, we create a polyline connecting consecutive nodes to illustrate the shortest path. Finally, the function returns the Folium map, saved as an HTML file, for visualization in the default web browser.below you see the result of the visualization process:

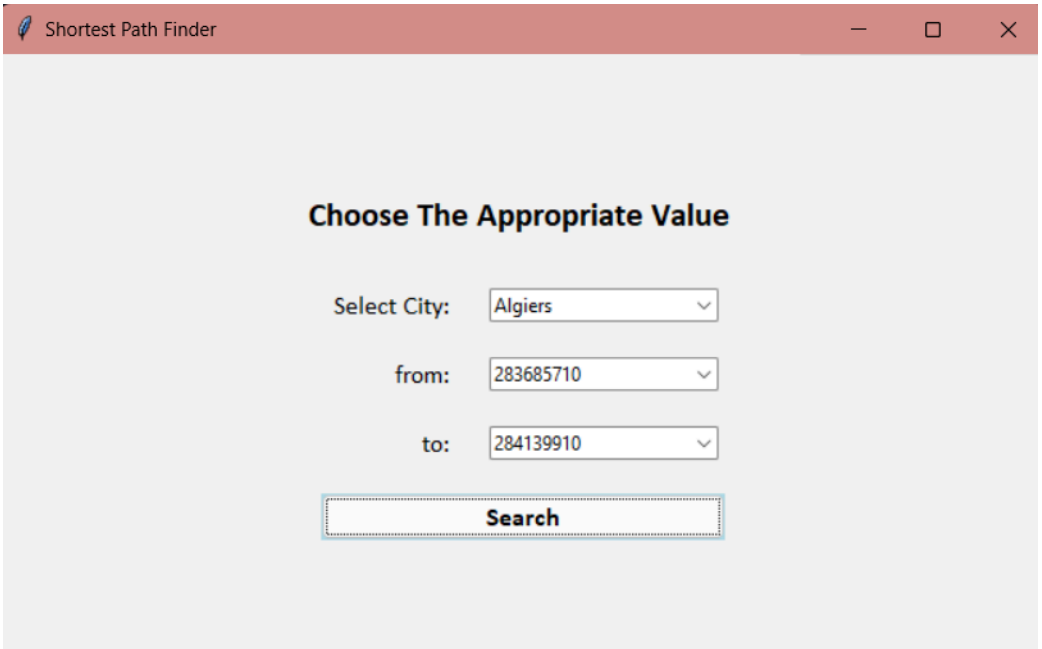


3.5 Create a User Interface (UI):

to create the UI of this project we use the following:

- Tkinter: We develop a GUI using Tkinter, a standard Python library for creating graphical interfaces.
- User Inputs: We provide dropdown menus or input fields for users to select the start and goal nodes.
- Buttons: We include a button to trigger the pathfinding process.

4. Conclusion



Through this project, we have gained valuable insights and skills in several areas. We learned how to implement the A* algorithm to find the shortest path between two locations on a map, using heuristic functions to guide the search efficiently. The project introduced us to graph representation and operations, allowing us to transform map data into a graph format using OSMnx and NetworkX. We also developed proficiency in visualizing geographic data and paths using Matplotlib, and we learned how to create a user-friendly graphical interface with Tkinter.
Overall, this project has equipped us with practical skills in algorithm implementation, data visualization, and GUI development, demonstrating the practical application of search algorithms in solving real-world navigation problems.

- The QR code below links to the source code of the project on Github.