Assignment 2: Real-World Application of Search Algorithms: Finding Shortest Paths Between Two Places on a Map (Bejaia Region)

Introduction

Search Algorithms is one of the approaches used to solve state-based problems. In this assignment, I've used it to solve the problem of finding the shortest path in the province of Bejaia. The process involves utilizing open-source tools and libraries to transform map data into a graph representation, applying a search algorithm A* to find the shortest path, and visualizing the result on the map.

Assignment Tasks:

Step 1: The process of getting the shortest path nodes cardinalities using python packages OSMNX and Flask for running python Backend

Process

1. **Obtaining Map Data:** I start by obtaining map data for the Bejaia region from OpenStreetMap (OSMNX). OSMNX provides free and detailed map data that can be exported in various formats. For this assignment, we will use the XML format, which is compatible with the OSMnx library.
2. **Transforming Map Data into a Graph:** Using the OSMnx library, we transform the map data into a graph representation. In this graph, intersections or landmarks are represented as nodes, and roads or paths between them are represented as edges.
3. **Applying the Search Algorithm:** To find the shortest path between two cities, I've used the A* search algorithm. A* is chosen for its efficiency in finding the shortest path in weighted graphs, which makes it suitable for our map data where edges have varying lengths. The output of this search algorithm is the cardinalities of nodes passed as the shortest path to the map.

---

this is the Python function for searching the shortest path between two places in bejaia that i've use it using A* search :

```
import osmnx as ox
import networkx as nx
import json
def search(start_place, end_place):
########################################
    #1:getting the map data of bejaia with ox
    ox.settings.max_query_area_size = 1e9
    region = "Bejaia, DZ"
```

```python
    map_data = ox.graph_from_place(region, network_type='drive')
    ox.save_graphml(map_data,'map.graphml')
    fig, ax= ox.plot_graph(map_data, show=False)
########################################
#########################################
    #2:transform this data to graph
    graph = ox.load_graphml('map.graphml')
#########################################
##########################################
    #3:apply A* search in this graph
    start_node = ox.distance.nearest_nodes(graph, start_place[1], start_place[0])
    end_node = ox.distance.nearest_nodes(graph, end_place[1], end_place[0])
    shortest_path = nx.astar_path(graph, start_place, end_place, weight='length')
    shortest_path_coords = [(graph.nodes[node]['y'], graph.nodes[node]['x']) for node in sho
###########################################
    return shortest_path_coords
```

This fonction return shortest_path_coords now i should pass it as an HTTPS Response from the Backend (Flask) to the Frontend (React.jsx) with this code as post request.

```python
############################################################################
app = Flask(__name__)
CORS(app, resources={r"/*": {"origins": "*"}})
api = Api(app)
class Map(Resource):
    def post(self):
        ############################
        #1:pass to this Api start and end place given by user as input from fontend passed
        data = request.get_json()
        start_place = data['start_place'][0]
        end_place = data['end_place'][0]
        ##############################
        #############################
        #2:pass them to the search fonction
        shortest_path_coords = search(start_place, end_place)
        ###############################
        ###################
        #3:pass the output to the frontend
        return jsonify({"shortest_path_coords": shortest_path_coords, 'METHOD': 'POST'})
        ####################
api.add_resource(Map, '/')
if __name__=='__main__':
    app.run(debug=True)
```

##

Step 2 : user interface

The process of Visualizing the Shortest Path in the frontend map using the cardinalities

For the frontend, I've used the package react-leaflet that displays the Map of Bejaia. Then I've got the shortest path cardinalities (x,y) list from the Backend (Flask) as an HTTPS response. Then I've used these cardinalities to highlight the path on the Map of React.jsx. The shortest path between the chosen cities is marked with a different color to distinguish it from other paths.

Challenges and Solutions

1. **Handling Large Datasets:** When I started the project, I tried to deal with the entire map of Algeria, but it was really large. It took more time in processing and was more memory consuming. So, I chose the map of the Bejaia Region to reduce the dataset.
2. **Ensuring Accurate Pathfinding:** When I tested the program a lot of times, I saw that in the places that are far from the center of the city, the path is not optimal. So, I tried to use a new version of the OSMnx library with more dataset than the older version, so the solution becomes more optimal.

Conclusion

The developed solution successfully implements the A* search algorithm for finding the shortest path between two cities in Algeria using OSM data. The use of OSMnx and React enables effective map and path visualization. The solution addresses challenges related to data handling, pathfinding accuracy, and visualization, providing a robust framework for similar real-world applications.