Project Report: Pathfinding for Emergency Services in Heidelberg, Germany

GitHub Repository Link

https://github.com/thehammadishaq (dummy link)

Objective

The primary goal of this project is to enhance the efficiency of emergency services in Heidelberg, Germany, by implementing and evaluating pathfinding algorithms. This project focuses on finding the shortest path from any point in the city to the nearest hospital using Dijkstra's and A* algorithms.

Methodology

1. **City Selection**
   - **Chosen City:** Heidelberg, Germany
   - **Reason:** Heidelberg is a mid-sized city, making it practical for detailed analysis and implementation.
2. **Data Collection**
   - **Geographic Information:** Obtained geographic data of Heidelberg, including its layout, topology, and road network, using OpenStreetMap (OSM).
   - **Hospitals:** Collected a list of hospitals in Heidelberg with their latitude and longitude coordinates.
   - **Datasets Used:**
     - OSM data for the city's layout and road network
     - Hospital locations obtained from OSM features
   - **Dataset URL:** OpenStreetMap Data
3. **Graph Mapping**
   - **Nodes:** Points of interest, including hospitals and other locations.
   - **Edges:** Connections between nodes based on the road network.
   - **Tools Used:** OSMnx for downloading and processing OSM data.
4. **Algorithm Selection**
   - **Dijkstra's Algorithm:** A well-known shortest path algorithm that guarantees finding the shortest path in weighted graphs.
   - **A* Algorithm:** An informed search algorithm that uses heuristics to improve search efficiency while ensuring the shortest path.
5. **Theoretical Analysis**
   - **Effectiveness:**
     - Both algorithms are expected to accurately find the shortest path to the nearest hospital.
     - A* is expected to perform slightly better due to its heuristic-based approach.
   - **Efficiency:**

        - **Dijkstra's Algorithm:**
            - Time Complexity: O(V2)O(V^2)O(V2) for dense graphs and O((V+E)logV)O((V + E) \log V)O((V+E)logV) for graphs using a priority queue.
            - Space Complexity: O(V)O(V)O(V).
        - **A* Algorithm:**
            - Time Complexity: O((V+E)logV)O((V + E) \log V)O((V+E)logV), similar to Dijkstra's but often faster in practice due to heuristics.
            - Space Complexity: O(V)O(V)O(V).

6. **Implementation**
    - Implemented using Python with libraries such as NetworkX and OSMnx.
    - **Scripts:**
        - `data_collection.py`: Collects and processes data.
        - `graph_mapping.py`: Maps the city graph and adds hospitals.
        - `pathfinding.py`: Contains the pathfinding logic and user interaction.
        - `empirical_evaluation.py`: Evaluates the algorithms using predefined scenarios.
7. **Empirical Evaluation**
    - **Scenarios:**
        1. **Neuenheim (North Heidelberg)**
        2. **Heidelberg Zoo (West Heidelberg)**
        3. **Heidelberg-Süd (South Heidelberg)**
        4. **Handschuhsheim (North-East Heidelberg)**
        5. **Boxberg (South-West Heidelberg)**
    - **Metrics:**
        - **Runtime:** Time taken to find the path.
        - **Memory Usage:** Memory used during the computation.
        - **Path Accuracy:** Comparison with best-known routes.
    - **Results:**
        - Dijkstra and A* both consistently found the shortest paths with slight performance differences.
        - A* generally performed faster than Dijkstra.
        - Memory usage was similar for both algorithms.

Empirical Evaluation Results

Scenario: Neuenheim (North Heidelberg)

- **Nearest Hospital:** Klinik für Allgemeine Innere Medizin und Psychosomatik
- **Runtime:** 0.36 seconds
- **Memory Usage:** 0.77 MB
- **Dijkstra's Path:** Max-Reger-Straße → Quinckestraße → Seitzstraße → Gundolfstraße → Mönchhofstraße → Unnamed Road → Im Neuenheimer Feld → Kirschnerstraße → Hofmeisterweg → Kirschnerstraße.

- **A* Path***:* Max-Reger-Straße → Quinckestraße → Seitzstraße → Gundolfstraße → Mönchhofstraße → Unnamed Road → Im Neuenheimer Feld → Kirschnerstraße → Hofmeisterweg → Kirschnerstraße.

Scenario: Heidelberg Zoo (West Heidelberg)

- **Nearest Hospital:** Institut für Humangenetik
- **Runtime:** 0.28 seconds
- **Memory Usage:** 0.72 MB
- **Dijkstra's Path:** Im Neuenheimer Feld.
- **A* Path:** Im Neuenheimer Feld.

Scenario: Heidelberg-Süd (South Heidelberg)

- **Nearest Hospital:** Kurpfalzkrankenhaus
- **Runtime:** 0.40 seconds
- **Memory Usage:** 0.72 MB
- **Dijkstra's Path:** Czernyring → Bergheimer Straße → Mannheimer Straße → Ludwig-Guttmann-Straße.
- **A* Path:** Czernyring → Bergheimer Straße → Mannheimer Straße → Ludwig-Guttmann-Straße.

Scenario: Handschuhsheim (North-East Heidelberg)

- **Nearest Hospital:** Institut für Humangenetik
- **Runtime:** 0.41 seconds
- **Memory Usage:** 0.72 MB
- **Dijkstra's Path:** Handschuhsheimer Landstraße → Pfarrgasse → Zeppelinstraße → Berliner Straße → Im Neuenheimer Feld.
- **A* Path:** Handschuhsheimer Landstraße → Pfarrgasse → Zeppelinstraße → Berliner Straße → Im Neuenheimer Feld.

Scenario: Boxberg (South-West Heidelberg)

- **Nearest Hospital:** Klinik für Allgemeine Innere Medizin und Psychosomatik
- **Runtime:** 0.51 seconds
- **Memory Usage:** 0.73 MB
- **Dijkstra's Path:** Im Beind → Leimer Straße → Rathausstraße → Herrenwiesenstraße → Am Rohrbach → Römerstraße → Lessingstraße → Mittermaierstraße → Ernst-Walz-Brücke → Berliner Straße → Mittermaierstraße → Berliner Straße → Jahnstraße → Im Neuenheimer Feld → Kirschnerstraße → Hofmeisterweg → Kirschnerstraße.
- **A* Path:** Im Beind → Leimer Straße → Rathausstraße → Herrenwiesenstraße → Am Rohrbach → Römerstraße → Lessingstraße → Mittermaierstraße → Ernst-Walz-Brücke → Berliner Straße → Mittermaierstraße → Berliner Straße → Jahnstraße → Im Neuenheimer Feld → Kirschnerstraße → Hofmeisterweg → Kirschnerstraße.

Design Decisions

- **City Selection:** Heidelberg was chosen due to its manageable size and available data.
- **Data Collection:** Utilized OSM data for accuracy and comprehensiveness.
- **Graph Mapping:** Ensured a realistic representation of Heidelberg's road network.
- **Algorithm Selection:** Dijkstra for guaranteed shortest path; A* for heuristic optimization.
- **Empirical Scenarios:** Selected diverse locations to validate the algorithms comprehensively.

Conclusion

- Both Dijkstra's and A* algorithms are effective in finding the shortest paths to the nearest hospitals in Heidelberg.
- A* generally performed faster with similar memory usage, making it a more efficient choice in practice.

Dataset URL

- OpenStreetMap Data