AI Project – Group 7

Title: Path Planning

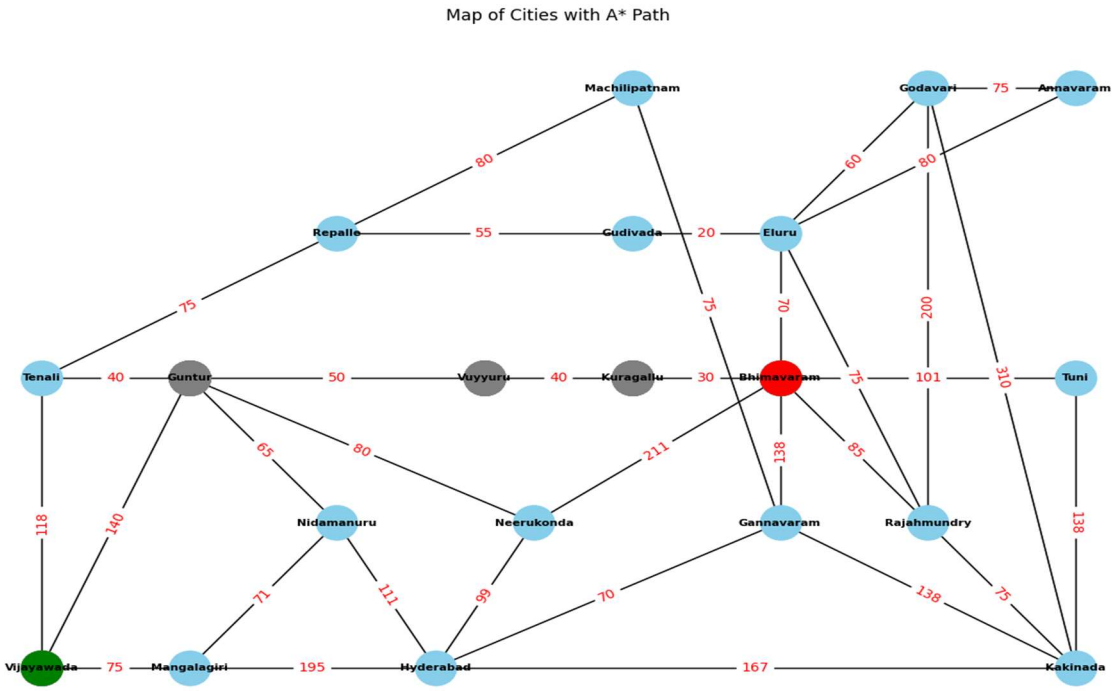
Problem Description:

The path planning problem in AI involves finding the most efficient route between two points in a given environment. Typically represented as a graph, nodes correspond to locations, and edges indicate possible paths. Common algorithms include Dijkstra's, A*, BFS, and DFS. Heuristics, estimates of costs to the goal, are often used to guide searches. Pathfinding is crucial in dynamic environments with changing conditions. Grid-based representations, especially in video games and robotics, simplify navigation. Real-time constraints are vital for applications requiring prompt responses. Efficient implementation is essential for large-scale environments. Applications range from robotics and autonomous vehicles to video game character navigation and network routing. The challenge extends to adapting to dynamic changes and optimizing paths for logistical efficiency.

Rationale:

We have chosen A* for the Path planning. The A* algorithm is favoured in path planning due to its completeness, ensuring a solution is found if it exists, and optimality, guaranteeing the shortest path. It uses a heuristic function to guide the search efficiently, focusing on promising paths. This adaptability makes A* suitable for various environments and graph representations, enhancing its versatility. The algorithm's ability to balance efficiency and optimality makes it well-suited for applications in robotics, gaming, logistics, and network routing.

Illustrations:



Let us take an example and try to understand it

Here, in the above graph the starting city is in green colour, destination is in red colour and the cities which are in the path are in grey colour.

1. Starting Point and Destination:

The A* algorithm is applied to find the shortest path from Vijayawada to Bhimavaram.

2. Cities Explored:

The cities explored during the A* algorithm execution are: 'Vijayawada', 'Mangalagiri', 'Nidamanuru', 'Guntur', 'Vuyyuru', 'Kuragallu', 'Bhimavaram'. These cities are considered at various stages of the search.

3. Number of Possible Cities:

There are 7 cities that the algorithm may explore during its search.

4. Shortest Path:

The shortest path from Vijayawada to Bhimavaram is found to be: 'Vijayawada', 'Guntur', 'Vuyyuru', 'Kuragallu', 'Bhimavaram'. This path is determined by tracing back through the optimal choices made during the A* algorithm execution.

5. Number of Cities Passed:

The number of cities passed in the shortest path is 5.

6. Total Distance:

The total distance along the shortest path is 260. This distance is calculated based on the sum of the weights between consecutive cities in the path.

Explanation of the A* Algorithm Steps:

- The A* algorithm starts at Vijayawada and explores neighbouring cities based on their total costs, considering both the actual distance travelled and the heuristic estimate (straight-line distance to the destination).
- It expands the search to cities like Mangalagiri, Nidamanuru, Guntur, Vuyyuru, Kuragallu, and Bhimavaram.
- The algorithm dynamically updates the distances and priority queue, choosing the most promising paths at each step.
- The final result is the shortest path from Vijayawada to Bhimavaram, considering both actual distances and the heuristic estimate.

In summary, the A* algorithm efficiently explored cities and determined the shortest path from Vijayawada to Bhimavaram, as reflected in the output.

Code Outputs:

```
A* algorithm program for the given problem
        Vijayawada => Bhimavaram
=====================================================
A city that may be explored             : ['Vijayawada', 'Mangalagiri', 'Nidamanuru', 'Guntur', 'Vuyyuru', 'Kuragallu', 'Bhimavaram']
Number of possible cities               : 7
=====================================================
The city passed with the shortest distance      : ['Vijayawada', 'Guntur', 'Vuyyuru', 'Kuragallu', 'Bhimavaram']
Number of cities passed                         : 5
Total distance                                  : 260
['Vijayawada', 'Guntur', 'Vuyyuru', 'Kuragallu', 'Bhimavaram']
```

```
  A* algorithm program for the given problem
          Guntur => Annavaram
  =====================================================
  A city that may be explored             : ['Guntur', 'Vuyyuru', 'Kuragallu', 'Bhimavaram', 'Eluru', 'Annavaram']
  Number of possible cities               : 6
  =====================================================
  The city passed with the shortest distance      : ['Guntur', 'Vuyyuru', 'Kuragallu', 'Bhimavaram', 'Eluru', 'Annavaram']
  Number of cities passed                         : 6
  Total distance                                  : 270
```

```
A* algorithm program for the given problem
        Mangalagiri => Kuragallu
=====================================================
A city that may be explored             : ['Mangalagiri', 'Nidamanuru', 'Hyderabad', 'Gannavaram', 'Guntur', 'Vuyyuru', 'Kuragallu']
Number of possible cities               : 7
=====================================================
The city passed with the shortest distance      : ['Mangalagiri', 'Nidamanuru', 'Guntur', 'Vuyyuru', 'Kuragallu']
Number of cities passed                         : 5
Total distance                                  : 226
```

```
A* algorithm program for the given problem
        Hyderabad => Godavari
=====================================================
A city that may be explored             : ['Hyderabad', 'Gannavaram', 'Machilipatnam', 'Bhimavaram', 'Eluru', 'Godavari']
Number of possible cities               : 6
=====================================================
The city passed with the shortest distance      : ['Hyderabad', 'Gannavaram', 'Bhimavaram', 'Eluru', 'Godavari']
Number of cities passed                         : 5
Total distance                                  : 338
```

These are some of the few example output snippets.