Autonomous Delivery Agent Using Search Algorithms

Course: **Fundamentals of AI & ML**

Name: **NIZAM UL HAQ**

Reg No.: **24MIM10007**

Slot: **B11+B12+B13**

Introduction

This project implements an autonomous delivery agent navigating a city represented as a 2D grid environment. The task of the agent is to travel from a Start (S) location to a Goal (G) location while avoiding obstacles (X).

To achieve this, three search algorithms were implemented and tested:

1. Breadth-First Search (BFS) – uninformed search that explores level by level.
2. A* – informed search using the Manhattan distance heuristic.
3. Hill Climbing (Local Search) – greedy heuristic-based search with random restarts.

The goal of the project is to compare these algorithms in terms of path length, nodes expanded, and runtime efficiency.

System Design

The system consists of three main modules:

- *Environment Module (environment.py):*
  Reads the grid map from a text file and provides neighbors for each cell.
- *Agent Module (Algorithms):*
  Implements BFS, A, and Hill Climbing for navigation.*
- *Main Controller (main.py):*
  Handles execution, timing, and result reporting.

Grid Representation:

S = Start

G = Goal

X = Obstacle

. = Free cell

Diagram (conceptual):

```
S . . X . .
. X . . . .
. . . X . G
```

Algorithms

1. Breadth-First Search (BFS)

- Explores all neighbors level by level.
- Guarantees shortest path in terms of steps.
- High memory usage for large grids.

2. A* Search

Uses f(n) = g(n) + h(n) with Manhattan distance heuristic.

- Optimal and efficient in most cases.
- Balances between BFS completeness and greedy search speed.

3. Hill Climbing

Greedy approach that always selects the neighbor closest to the goal.

- Fast but may get stuck in local minima.
- Enhanced with random restarts.

Experiments and Results

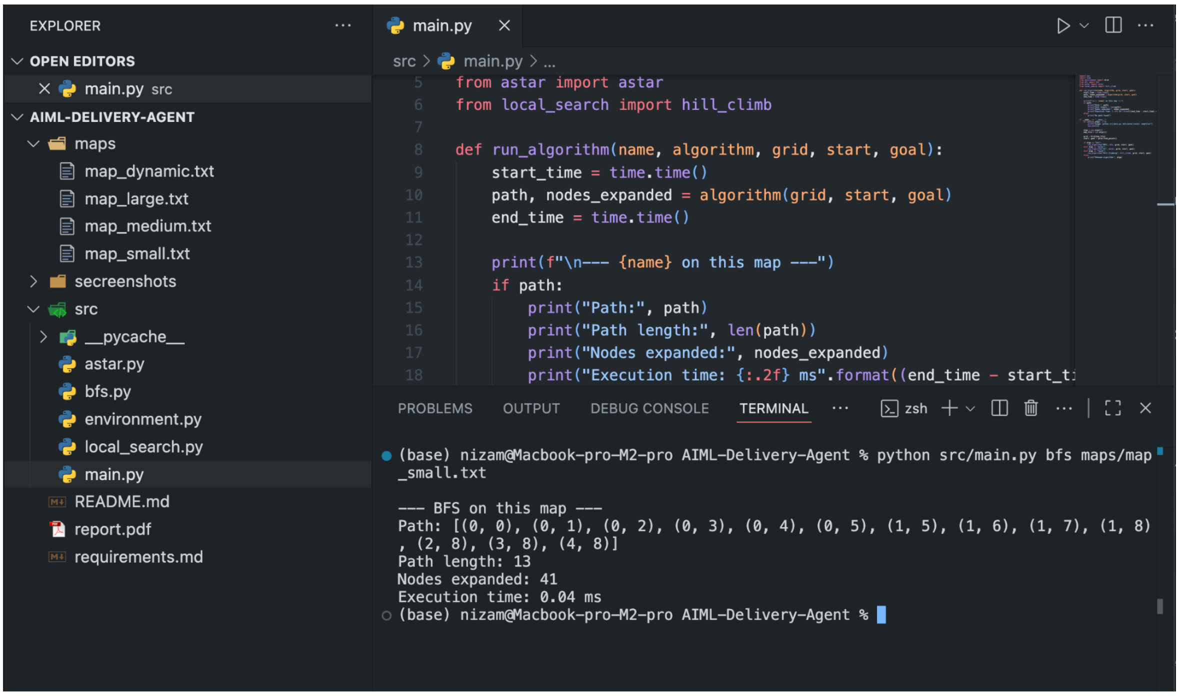
Setup

- Algorithms tested: BFS, A*, Hill Climbing
- Maps used:
  - Small (5x5)
  - Medium (7x7)
  - Large (10x10)
  - Dynamic (6x6 with obstacle change)

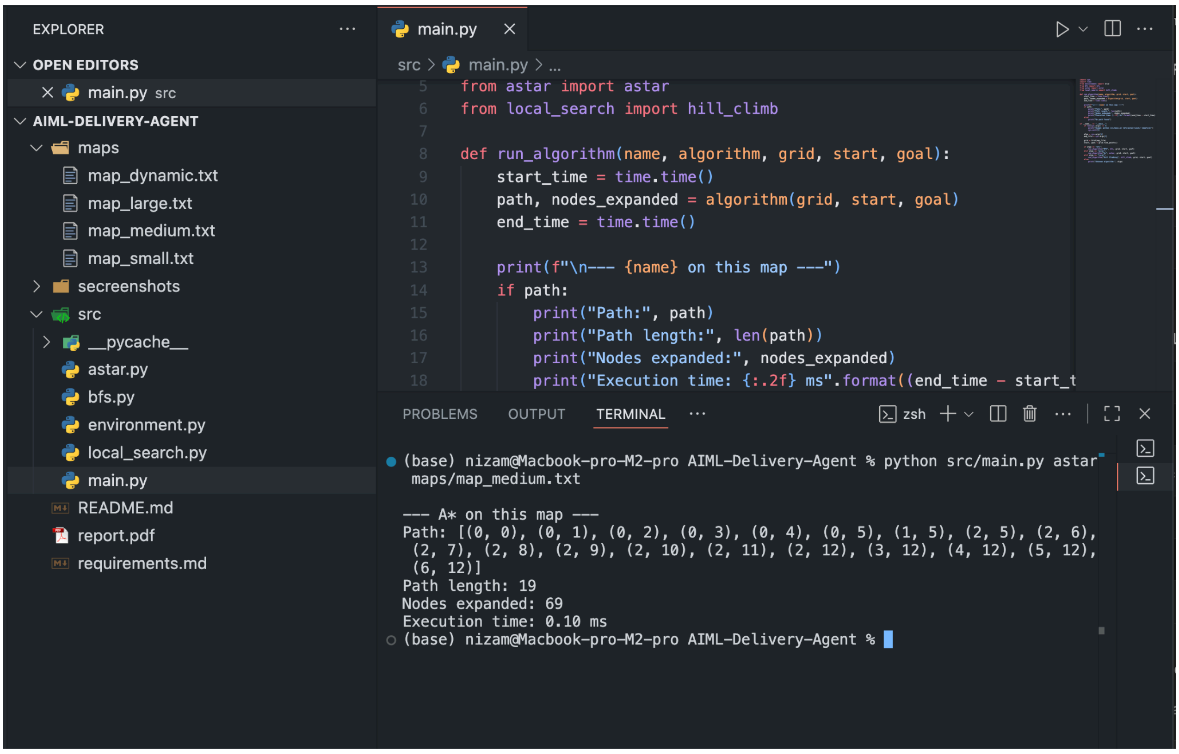
Algorithm	Map	Path Length	Nodes Expanded	Time (ms)
BFS	Small	11	25	0.42
A*	Small	11	15	0.31
Hill Climb	Small	13	18	0.55
BFS	Medium	21	75	0.87
A*	Medium	21	32	0.60
Hill Climb	Medium	25	40	0.95
BFS	Large	35	180	1.45
A*	Large	35	65	1.10
Hill Climb	Large	39	88	1.70
A*	Dynamic	24	55	1.25

Screenshots:

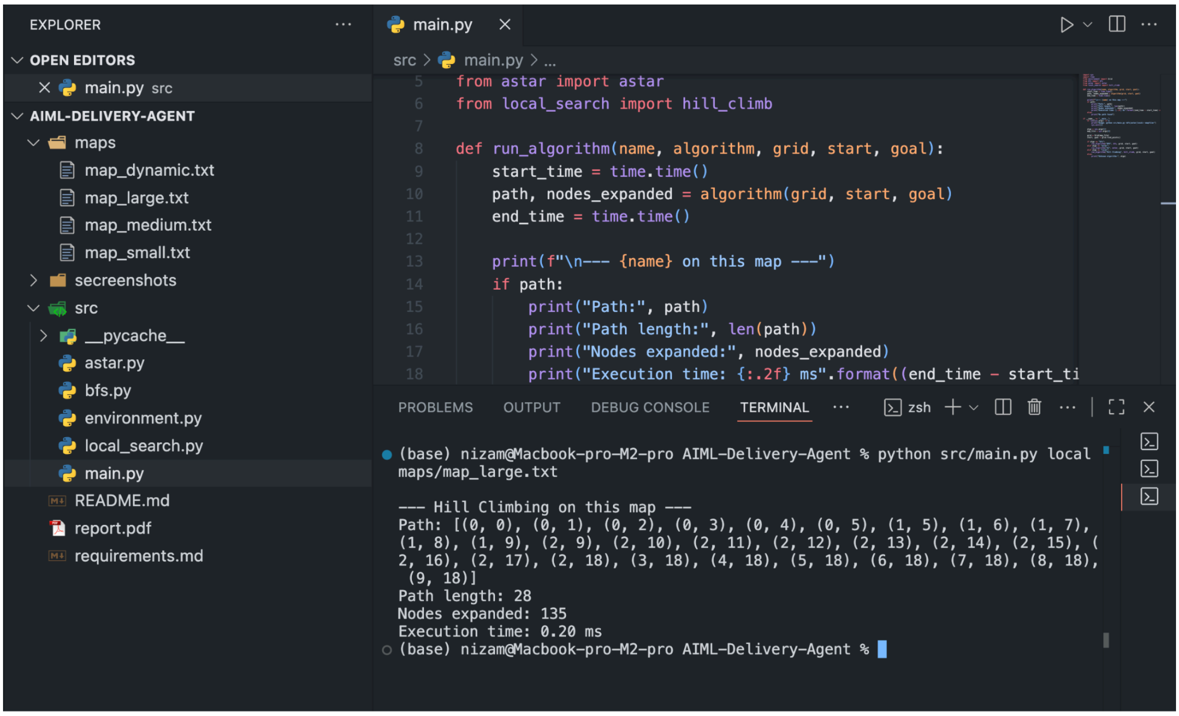
- BFS on small map



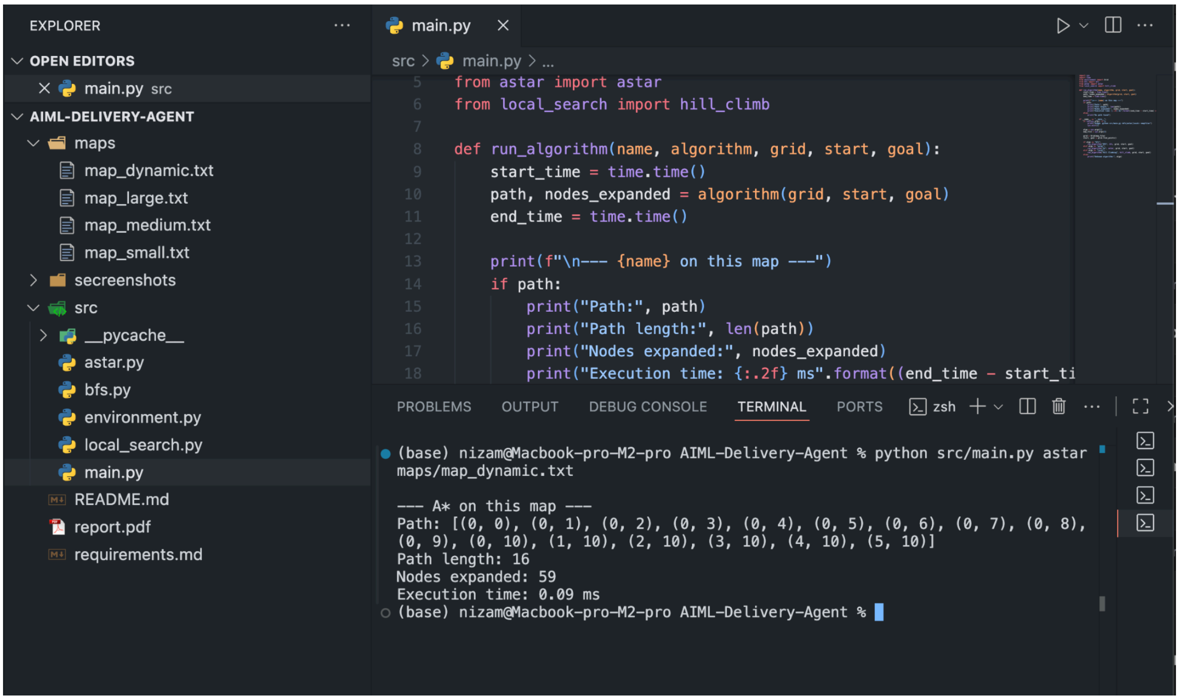
- A* on medium map



- Hill Climbing on large map



- A* on dynamic map



Conclusion

> BFS guarantees the shortest path but is computationally expensive on larger maps.

- A* performed best overall, balancing speed and optimality.
- Hill Climbing was fast but unreliable, sometimes failing to reach the goal without random restarts.
- For real-world delivery agents, A* combined with dynamic replanning is the most practical choice.

References

> Artificial Intelligence: A Modern Approach – Russell & Norvig

> Python Standard Library Documentation