**OopsiRoute: Phase 2: Proof of Concept Implementation**

Farheen Dhanani & Hajeera Hajeera

University of Cumberlands

MSCS-532-M80: Algorithms and Data Structures

Dr. Satish Penmatsa

10/25/2025

**OopsiRoute: Phase 2: Proof of Concept Implementation**

## Introduction

# This paper presents the Phase 2 proof of concept (PoC) for *OopsiRoute*, a Python-based navigation system developed as part of the “Developing and Optimizing Data Structures for Real-World Applications” project. The implementation focuses on building and testing the core data structures that enable efficient route finding and navigation. Using graph-based models and classical shortest-path algorithms such as Dijkstra’s and A\*, this phase validates the functional backbone of the application. Demonstrations of insertion, deletion, and traversal operations confirm the accuracy and efficiency of the approach. This PoC sets the foundation for scaling the system to handle real-world datasets and integrating it with mobile platforms.

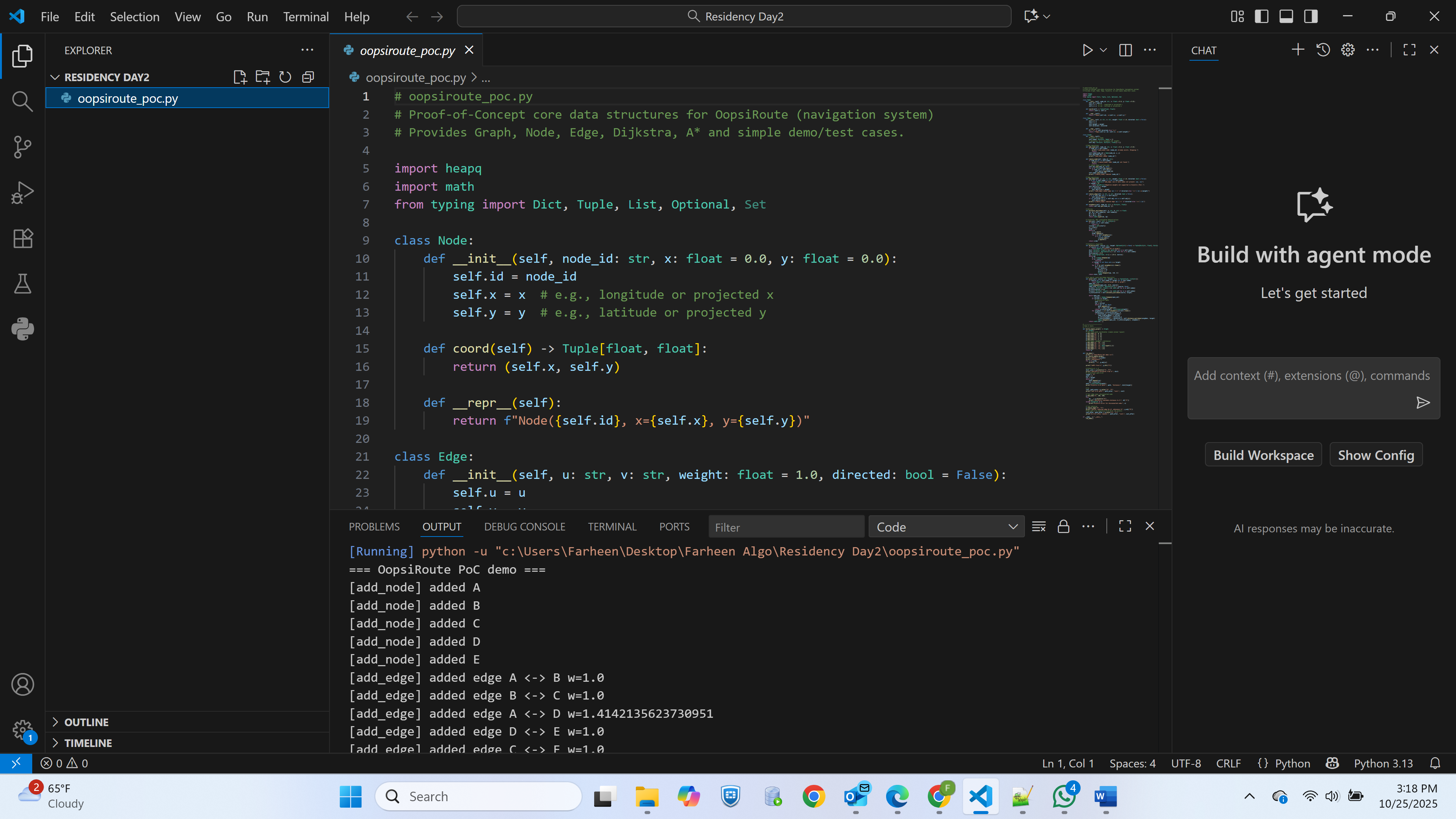
### ****Partial Implementation Overview****

The primary goal of Phase 2 was to implement the core data structures designed in Phase 1 of OopsiRoute. The chosen data structure is **a weighted graph,** where nodes represent intersections and edges represent roads with associated travel distances.

Each graph operation—node and edge insertion, deletion, and traversal was implemented using Python’s dictionaries for adjacency lists, offering O(1) average-time lookups and modifications. The algorithms implemented include:

* **Breadth-First Search (BFS)** for graph traversal and connectivity verification.
* **Dijkstra’s Algorithm** for shortest path computation in weighted graphs with non-negative edges.
* **A\*** Search for efficient route computation using Euclidean heuristics.

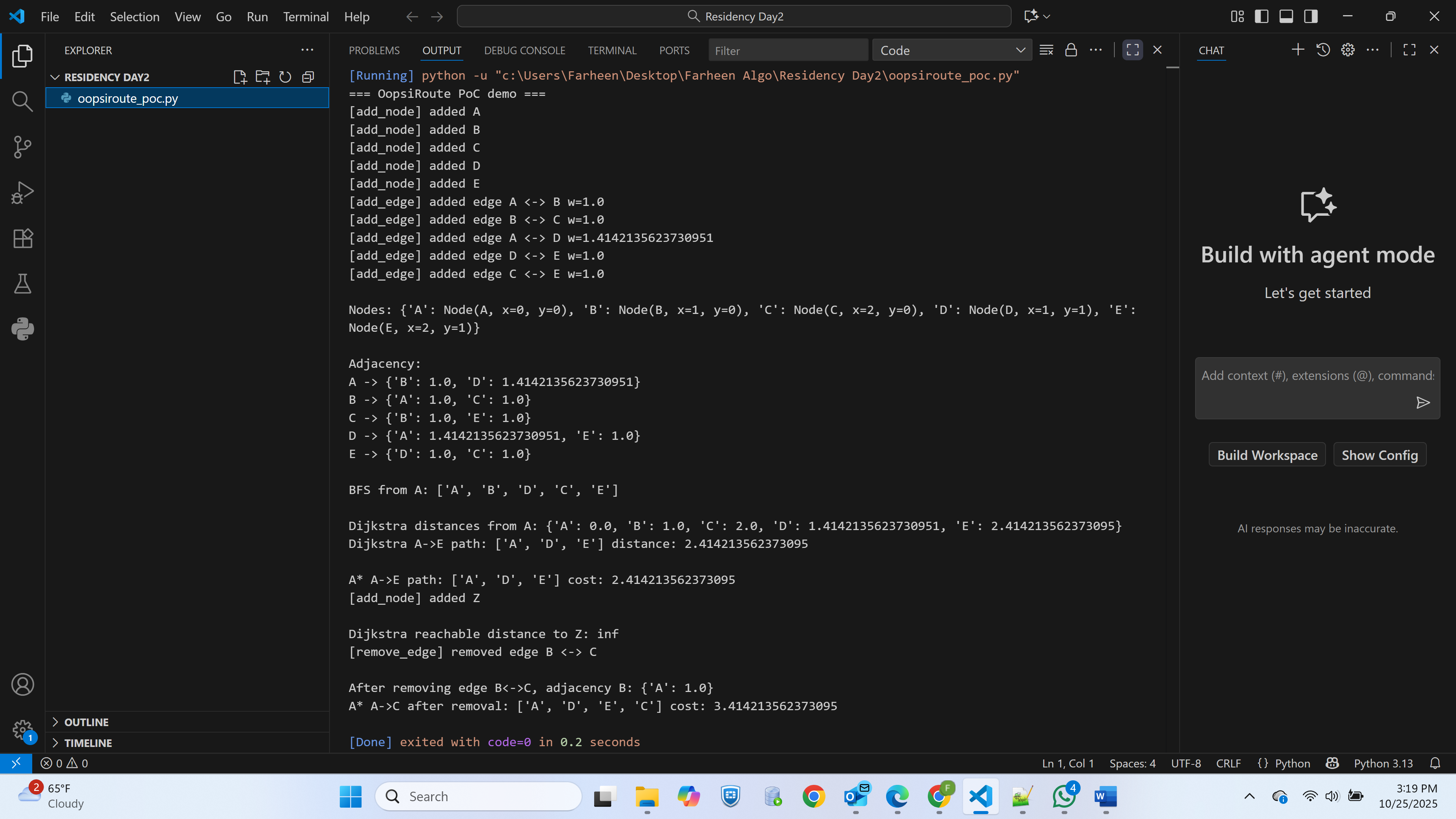
The graph is modular and supports real-time updates (adding or removing roads) and directional routes. This implementation forms the foundation for efficient route-finding across cities.

** ***Python code implementation in VS Code***

### ****Demonstration and Testing****

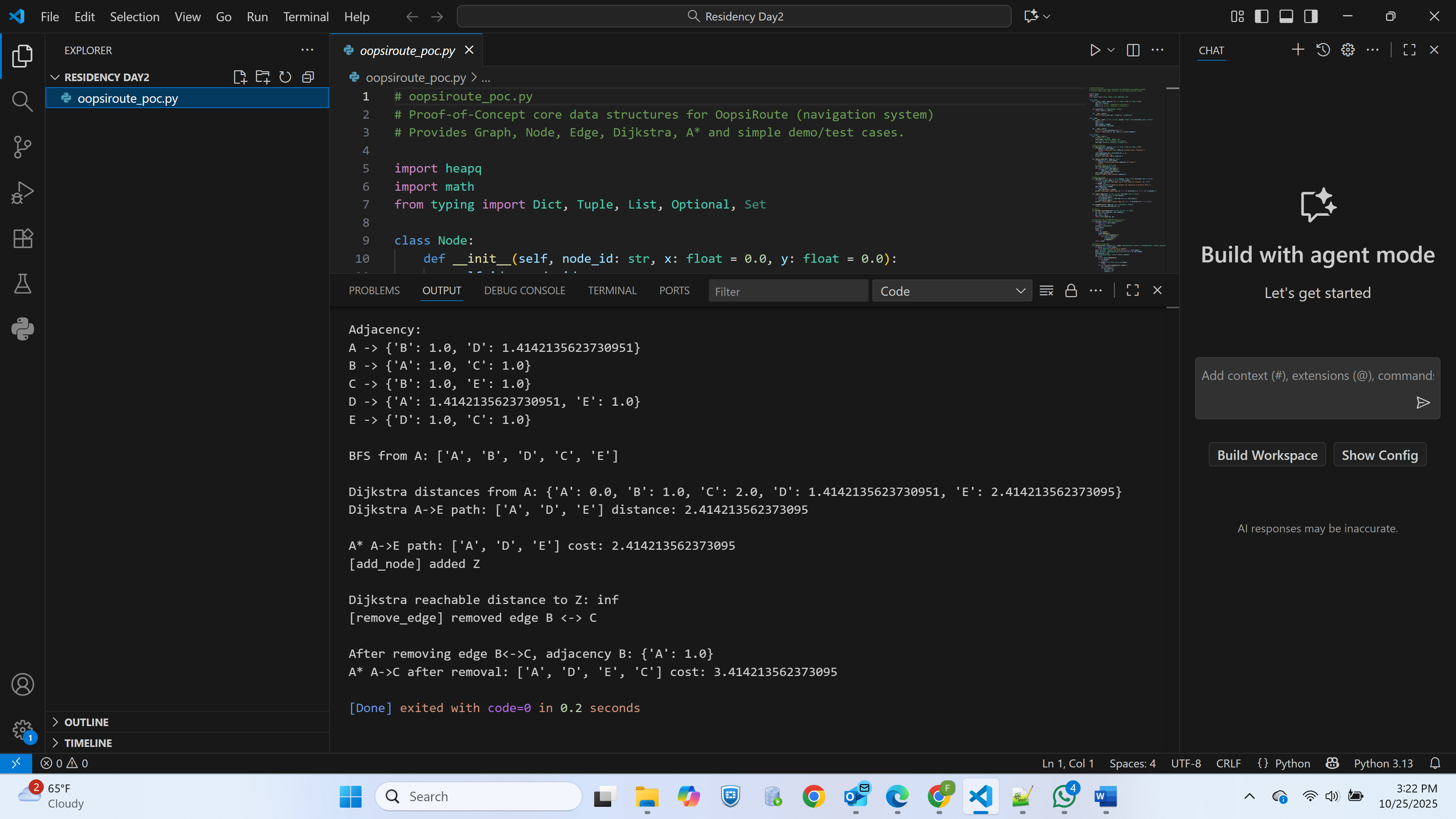
A demonstration script (oopsiroute\_poc.py) was created to test:

* **Node and edge insertion/deletion**
* **BFS traversal for connectivity**
* **Shortest path computation** (Dijkstra and A)
* **Edge cases** such as disconnected nodes and edge removals



Program output showing Dijkstra and A results

The program successfully demonstrated correct shortest-path results and dynamic graph updates.  
BFS confirmed graph connectivity, while Dijkstra and A\* computed paths with matching results, confirming heuristic admissibility.



Console output in VS Code terminal showing BFS and A results

### ****Implementation Challenges and Solutions****

| **Challenge** | **Solution** |
| --- | --- |
| Maintaining modularity while implementing complex graph algorithms | Encapsulated logic into separate Graph, Node, and Edge classes |
| Managing dynamic updates (edge/node deletions) | Implemented checks to remove all connected edges before deleting nodes |
| Ensuring heuristic admissibility in A\* | Used Euclidean distance based on node coordinates to maintain accuracy |
| Handling invalid operations | Added exception handling for missing nodes and negative weights |

These decisions ensure extensibility and allow integration with larger datasets in later phases.

### ****Next Steps****

The next development phase will extend this proof of concept to handle real-world data and performance optimization:

1. **Large-Scale Dataset Integration**
   * Import OpenStreetMap (OSM) data to represent real-world cities.
   * Implement data compression for scalability.
2. **Performance Optimization**
   * Introduce Contraction Hierarchies or ALT (A\*, Landmarks, Triangle inequality) methods for faster routing.
3. **Mobile Integration**
   * Connect Python-based routing engine with an Android front-end to visualize navigation paths on maps.
4. **Testing and Profiling**
   * Conduct stress tests on thousands of nodes and measure time and memory performance.

### ****Conclusion****

Phase 2 of OopsiRoute demonstrates the feasibility of using Python-based data structures to model a scalable navigation system. Through modular graph design and efficient pathfinding algorithms, this PoC provides a solid foundation for future integration with real-time navigation services and mobile deployment. The testing results validate the efficiency and accuracy of the implementation, paving the way for the final optimization and deployment phase.

# Reference

Dijkstra, E. W. (1959). *A note on two problems in connexion with graphs.* *Numerische Mathematik, 1*, 269–271. <https://doi.org/10.1007/BF01386390>

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). *A formal basis for the heuristic determination of minimum cost paths.* *IEEE Transactions on Systems Science and Cybernetics, 4*(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

Geisberger, R., Sanders, P., Schultes, D., & Delling, D. (2008). *Contraction hierarchies: Faster and simpler hierarchical routing in road networks.* *Experimental Algorithms, 5038*, 319–333. Springer. https://doi.org/10.1007/978-3-540-68552-4\_24