Libraries Used

- **SciPy**: For creating a KDTree.
- **NumPy**: For operating and modifying arrays.
- **OpenCV**: For image reading, writing, and processing.
- **NetworkX**: For graph creation.
- **Skimage**: For checking if a wall exists between two nodes.
- **Matplotlib**: For plotting the nodes, edges, and final path.

Methodology

The following steps were used to achieve PRM-based pathfinding:

1. Image Preprocessing

- The input maze image was loaded in **grayscale**.
- The start and end points were closed to prevent shortcuts in the **easy start-easy end** case.
- The image was converted into a **binary image** where black regions represented walls.

2. Node Generation

- **100 nodes** were generated for the first problem, and **300 nodes** for the second problem.
- Nodes were placed randomly in the **white (free space) regions**.

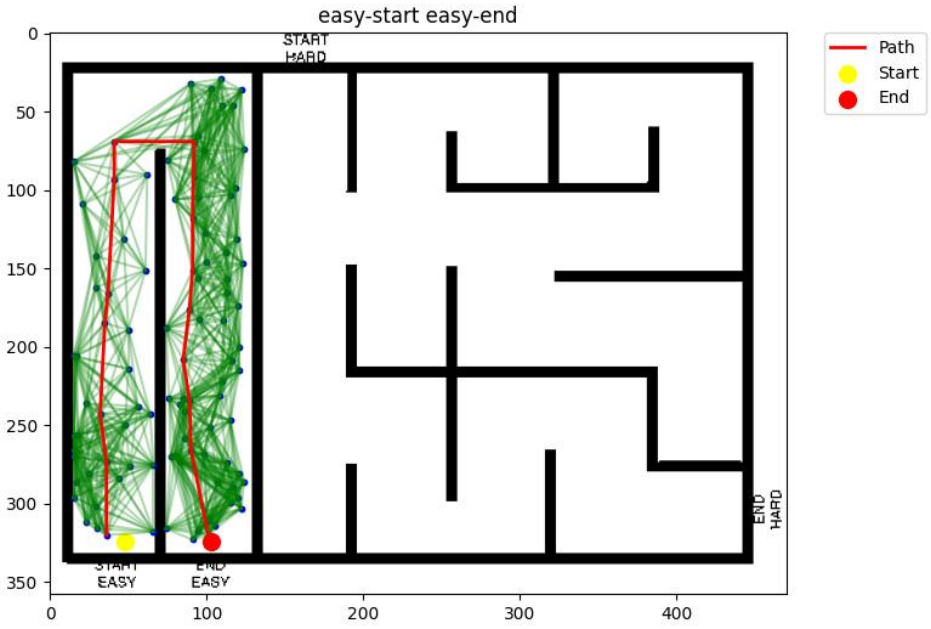
3. Graph Construction and Pathfinding

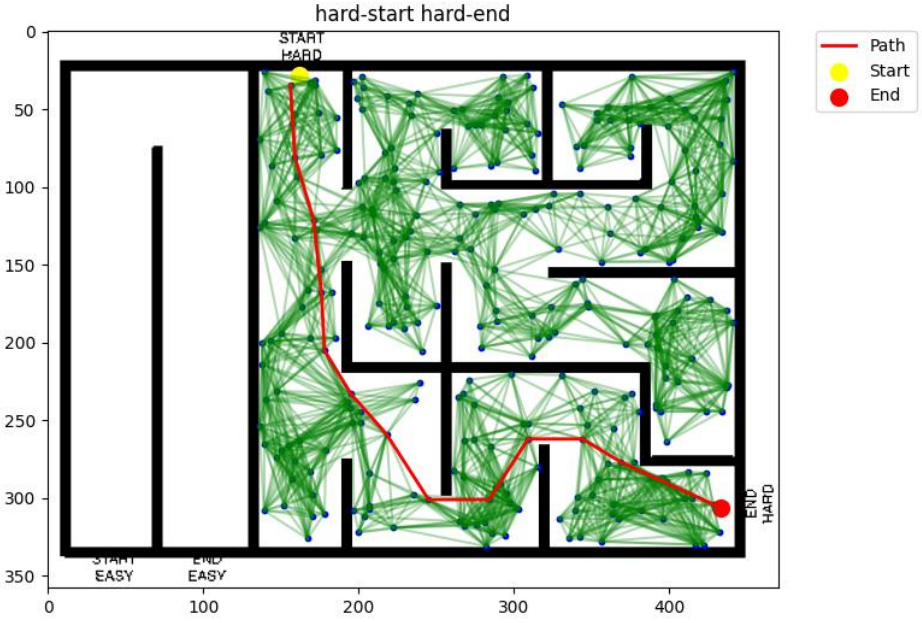
- All nodes and edges were stored in a **NetworkX Graph**.
- The shortest_path() function was used to find the **shortest path** from the starting point to the ending point through the nodes.

4. Visualization

- **Matplotlib** was used to plot:

    1. All nodes and edges.
    2. The starting and ending points.
    3. The optimal path.
    4. Two different plots were obtained for the two different problems.





Some constants:

Neighbors Considered (k): 20

For first problem:

- **Region:** (8,19) to (125,338)
- **Start coordinate** : (48,324)
- **end coordinate :** (103,324)

For second problem:

- **Region:** (132,20) to (444,338)
- **Start coordinate** : (162, 28)
- **end coordinate :** (433,306)