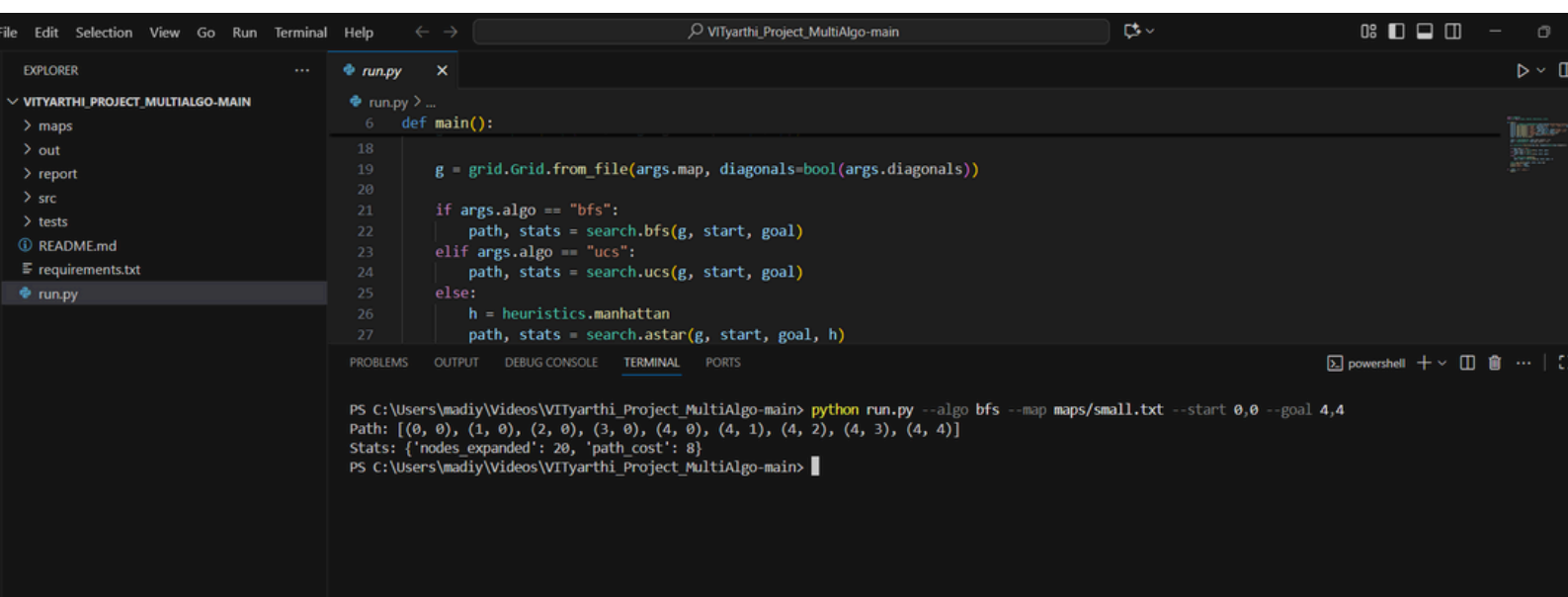


Sample screenshorts

TEST 1: BFS

- Small map

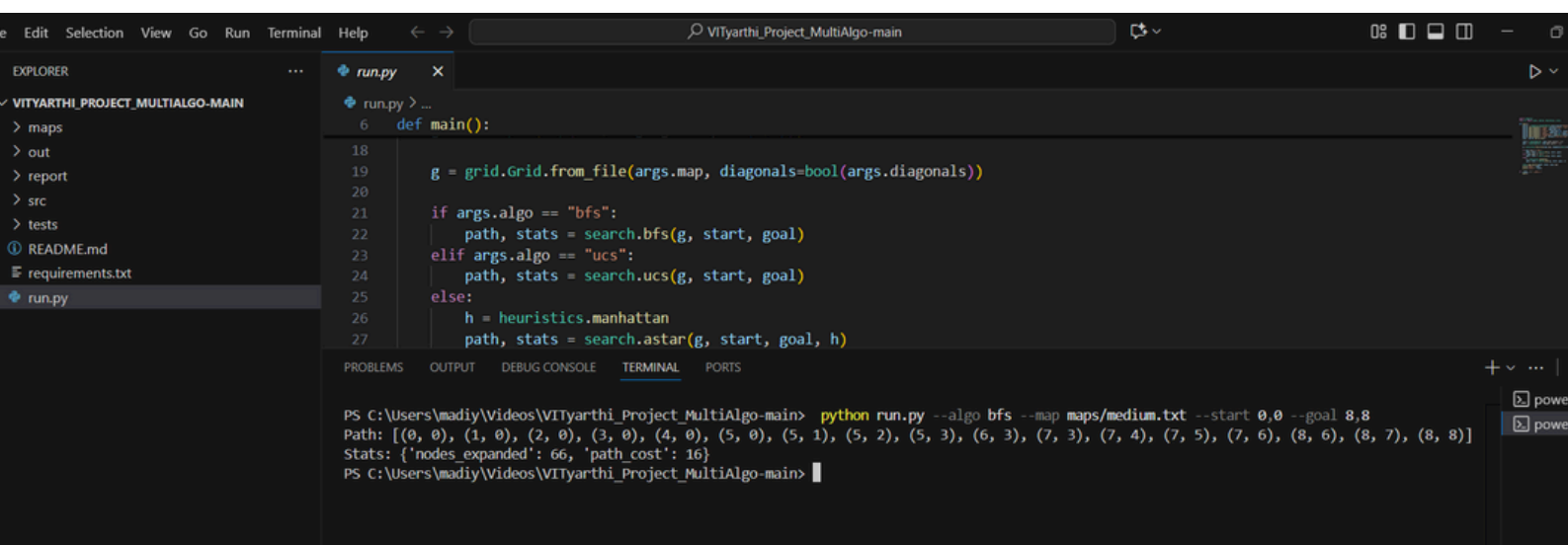


The screenshot shows the Visual Studio Code interface with the Explorer, Run and Debug, and Terminal panels. The Explorer panel on the left shows the project structure for 'VITYARTHI_PROJECT_MULTIALGO-MAIN', including folders 'maps', 'out', 'report', 'src', and 'tests', and files 'README.md', 'requirements.txt', and 'run.py'. The Run and Debug panel in the center shows the code for 'run.py', which defines a 'main()' function that uses a grid and search algorithms (bfs, ucs, astar) to find a path. The Terminal panel at the bottom shows the command 'python run.py --algo bfs --map maps/small.txt --start 0,0 --goal 4,4' being executed, resulting in a path of 10 nodes and a path cost of 8.

```
def main():
    6
    18 g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))
    19
    20
    21 if args.algo == "bfs":
    22     path, stats = search.bfs(g, start, goal)
    23 elif args.algo == "ucs":
    24     path, stats = search.ucs(g, start, goal)
    25 else:
    26     h = heuristics.manhattan
    27     path, stats = search.astar(g, start, goal, h)

PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo bfs --map maps/small.txt --start 0,0 --goal 4,4
Path: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]
Stats: {'nodes_expanded': 20, 'path_cost': 8}
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main>
```

- Medium map



The screenshot shows the Visual Studio Code interface with the Explorer, Run and Debug, and Terminal panels. The Explorer panel on the left shows the project structure for 'VITYARTHI_PROJECT_MULTIALGO-MAIN', including folders 'maps', 'out', 'report', 'src', and 'tests', and files 'README.md', 'requirements.txt', and 'run.py'. The Run and Debug panel in the center shows the code for 'run.py', which defines a 'main()' function that uses a grid and search algorithms (bfs, ucs, astar) to find a path. The Terminal panel at the bottom shows the command 'python run.py --algo bfs --map maps/medium.txt --start 0,0 --goal 8,8' being executed, resulting in a path of 25 nodes and a path cost of 16.

```
def main():
    6
    18 g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))
    19
    20
    21 if args.algo == "bfs":
    22     path, stats = search.bfs(g, start, goal)
    23 elif args.algo == "ucs":
    24     path, stats = search.ucs(g, start, goal)
    25 else:
    26     h = heuristics.manhattan
    27     path, stats = search.astar(g, start, goal, h)

PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo bfs --map maps/medium.txt --start 0,0 --goal 8,8
Path: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (5, 1), (5, 2), (5, 3), (6, 3), (7, 3), (7, 4), (7, 5), (7, 6), (8, 6), (8, 7), (8, 8)]
Stats: {'nodes_expanded': 66, 'path_cost': 16}
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main>
```

- Large map

The screenshot shows the Visual Studio Code interface with a Python file named `run.py` open. The code defines a `main()` function that uses a grid-based search algorithm. The terminal output shows the command `python run.py --algo bfs --map maps/large.txt --start 0,0 --goal 10,10` being executed. The output displays the path found and the statistics: `Path: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0), (8, 0), (9, 0), (10, 0), (11, 0), (12, 0), (12, 1), (12, 2), (12, 3), (12, 4), (12, 5), (12, 6), (12, 7), (12, 8), (11, 8), (10, 8), (10, 9), (10, 10)]` and `Stats: {'nodes_expanded': 139, 'path_cost': 24}`.

```
def main():
    g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))

    if args.algo == "bfs":
        path, stats = search.bfs(g, start, goal)
    elif args.algo == "ucs":
        path, stats = search.ucs(g, start, goal)
    else:
        h = heuristics.manhattan
        path, stats = search.astar(g, start, goal, h)
```

```
PS C:\Users\madiy\Videos\VITyarthi_Project_MultiAlgo-main> python run.py --algo bfs --map maps/large.txt --start 0,0 --goal 10,10
Path: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0), (8, 0), (9, 0), (10, 0), (11, 0), (12, 0), (12, 1), (12, 2), (12, 3), (12, 4), (12, 5), (12, 6), (12, 7), (12, 8), (11, 8), (10, 8), (10, 9), (10, 10)]
Stats: {'nodes_expanded': 139, 'path_cost': 24}
PS C:\Users\madiy\Videos\VITyarthi_Project_MultiAlgo-main>
```

TEST 2: UCS

- Small Map

The screenshot shows the Visual Studio Code interface with the same `run.py` file. The terminal output shows the command `python run.py --algo ucs --map maps/small.txt --start 0,0 --goal 4,4` being executed. The output displays the path found and the statistics: `Path: [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 4), (2, 4), (3, 4), (4, 4)]` and `Stats: {'nodes_expanded': 20, 'path_cost': 8}`.

```
def main():
    g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))

    if args.algo == "bfs":
        path, stats = search.bfs(g, start, goal)
    elif args.algo == "ucs":
        path, stats = search.ucs(g, start, goal)
    else:
        h = heuristics.manhattan
        path, stats = search.astar(g, start, goal, h)
```

```
PS C:\Users\madiy\Videos\VITyarthi_Project_MultiAlgo-main> python run.py --algo ucs --map maps/small.txt --start 0,0 --goal 4,4
Path: [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 4), (2, 4), (3, 4), (4, 4)]
Stats: {'nodes_expanded': 20, 'path_cost': 8}
PS C:\Users\madiy\Videos\VITyarthi_Project_MultiAlgo-main>
```

- Medium Map

The screenshot shows the Visual Studio Code interface with the Explorer, Run and Debug, and Terminal panels. The Explorer panel on the left shows the project structure for 'VITYARTHI_PROJECT_MULTIALGO-MAIN', including folders like 'maps', 'out', 'report', 'src', 'tests', and files like 'README.md', 'requirements.txt', and 'run.py'. The Run and Debug panel in the center shows the 'run.py' file with a 'def main()' function. The function initializes a grid from a file, then uses a search algorithm (bfs, ucs, or astar) to find a path from a start point to a goal point. The Terminal panel at the bottom shows the command 'python run.py --algo ucs --map maps/medium.txt --start 0,0 --goal 8,8' being executed. The output shows the path found: 'Path: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (5, 1), (5, 2), (5, 3), (6, 3), (7, 3), (7, 4), (7, 5), (7, 6), (8, 6), (8, 7), (8, 8)]' and the stats: 'Stats: {'nodes_expanded': 66, 'path_cost': 16}'.

```
def main():
    6
    18 g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))
    19
    20
    21 if args.algo == "bfs":
    22     path, stats = search.bfs(g, start, goal)
    23 elif args.algo == "ucs":
    24     path, stats = search.ucs(g, start, goal)
    25 else:
    26     h = heuristics.manhattan
    27     path, stats = search.astar(g, start, goal, h)

PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo ucs --map maps/medium.txt --start 0,0 --goal 8,8
Path: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (5, 1), (5, 2), (5, 3), (6, 3), (7, 3), (7, 4), (7, 5), (7, 6), (8, 6), (8, 7), (8, 8)]
Stats: {'nodes_expanded': 66, 'path_cost': 16}
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main>
```

- Large Map

The screenshot shows the Visual Studio Code interface with the Explorer, Run and Debug, and Terminal panels. The Explorer panel on the left shows the project structure for 'VITYARTHI_PROJECT_MULTIALGO-MAIN', including folders like 'maps', 'out', 'report', 'src', 'tests', and files like 'README.md', 'requirements.txt', and 'run.py'. The Run and Debug panel in the center shows the 'run.py' file with a 'def main()' function. The function initializes a grid from a file, then uses a search algorithm (bfs, ucs, or astar) to find a path from a start point to a goal point. The Terminal panel at the bottom shows the command 'python run.py --algo ucs --map maps/large.txt --start 0,0 --goal 10,10' being executed. The output shows the path found: 'Path: [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 6), (2, 6), (3, 6), (4, 6), (5, 6), (6, 6), (7, 6), (8, 6), (9, 6), (10, 6), (11, 6), (12, 6), (12, 7), (12, 8), (11, 8), (10, 8), (10, 9), (10, 10)]' and the stats: 'Stats: {'nodes_expanded': 138, 'path_cost': 24}'.

```
def main():
    6
    18 g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))
    19
    20
    21 if args.algo == "bfs":
    22     path, stats = search.bfs(g, start, goal)
    23 elif args.algo == "ucs":
    24     path, stats = search.ucs(g, start, goal)
    25 else:
    26     h = heuristics.manhattan
    27     path, stats = search.astar(g, start, goal, h)

PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo ucs --map maps/large.txt --start 0,0 --goal 10,10
Path: [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 6), (2, 6), (3, 6), (4, 6), (5, 6), (6, 6), (7, 6), (8, 6), (9, 6), (10, 6), (11, 6), (12, 6), (12, 7), (12, 8), (11, 8), (10, 8), (10, 9), (10, 10)]
Stats: {'nodes_expanded': 138, 'path_cost': 24}
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main>
```

TEST 3: A* with diagonals

- Small Map

The screenshot shows the VS Code editor with the file `run.py` open. The code defines a `main()` function that uses `argparse` to handle command-line arguments for a pathfinding algorithm. It supports `--start`, `--goal`, `--diagonals`, and `--log` options. The algorithm can be set to `bfs`, `ucs`, or `astar` (the default). The `astar` algorithm uses a heuristic (Manhattan distance) to find the shortest path. The terminal shows the execution of the program with the command `python run.py --algo astar --map maps/small.txt --start 0,0 --goal 4,4 --diagonals 1`. The output shows the path `[(0, 0), (0, 1), (1, 2), (2, 3), (3, 4), (4, 4)]` and the statistics `Stats: {'nodes_expanded': 6, 'path_cost': 5}`.

```
def main():
    parser.add_argument("--start", required=True, help="format: x,y")
    parser.add_argument("--goal", required=True, help="format: x,y")
    parser.add_argument("--diagonals", type=int, default=0)
    parser.add_argument("--log", default="out/run.log")
    args = parser.parse_args()

    start = tuple(map(int, args.start.split(",")))
    goal = tuple(map(int, args.goal.split(",")))

    g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))

    if args.algo == "bfs":
        path, stats = search.bfs(g, start, goal)
    elif args.algo == "ucs":
        path, stats = search.ucs(g, start, goal)
    else:
        h = heuristics.manhattan
        path, stats = search.astar(g, start, goal, h)
```

PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo astar --map maps/small.txt --start 0,0 --goal 4,4 --diagonals 1
Path: [(0, 0), (0, 1), (1, 2), (2, 3), (3, 4), (4, 4)]
Stats: {'nodes_expanded': 6, 'path_cost': 5}
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main>

• Medium Map

The screenshot shows the VS Code editor with the file `run.py` open. The code is identical to the previous one, but the terminal shows the execution of the program with the command `python run.py --algo astar --map maps/medium.txt --start 0,0 --goal 8,8 --diagonals 1`. The output shows the path `[(0, 0), (0, 1), (1, 2), (2, 2), (3, 3), (3, 4), (2, 5), (3, 6), (4, 7), (5, 8), (6, 8), (7, 9), (8, 8)]` and the statistics `Stats: {'nodes_expanded': 16, 'path_cost': 12}`.

```
def main():
    parser.add_argument("--start", required=True, help="format: x,y")
    parser.add_argument("--goal", required=True, help="format: x,y")
    parser.add_argument("--diagonals", type=int, default=0)
    parser.add_argument("--log", default="out/run.log")
    args = parser.parse_args()

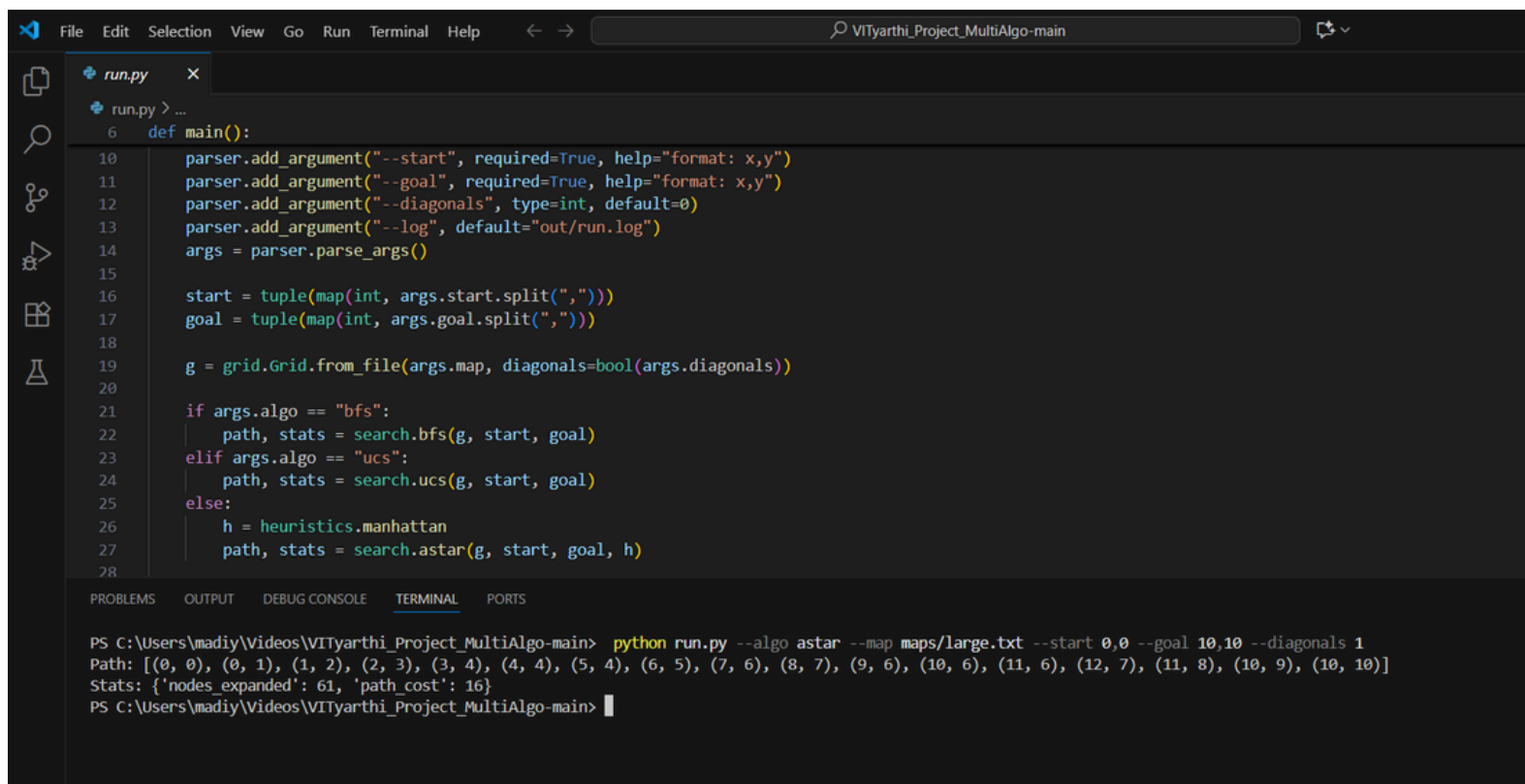
    start = tuple(map(int, args.start.split(",")))
    goal = tuple(map(int, args.goal.split(",")))

    g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))

    if args.algo == "bfs":
        path, stats = search.bfs(g, start, goal)
    elif args.algo == "ucs":
        path, stats = search.ucs(g, start, goal)
    else:
        h = heuristics.manhattan
        path, stats = search.astar(g, start, goal, h)
```

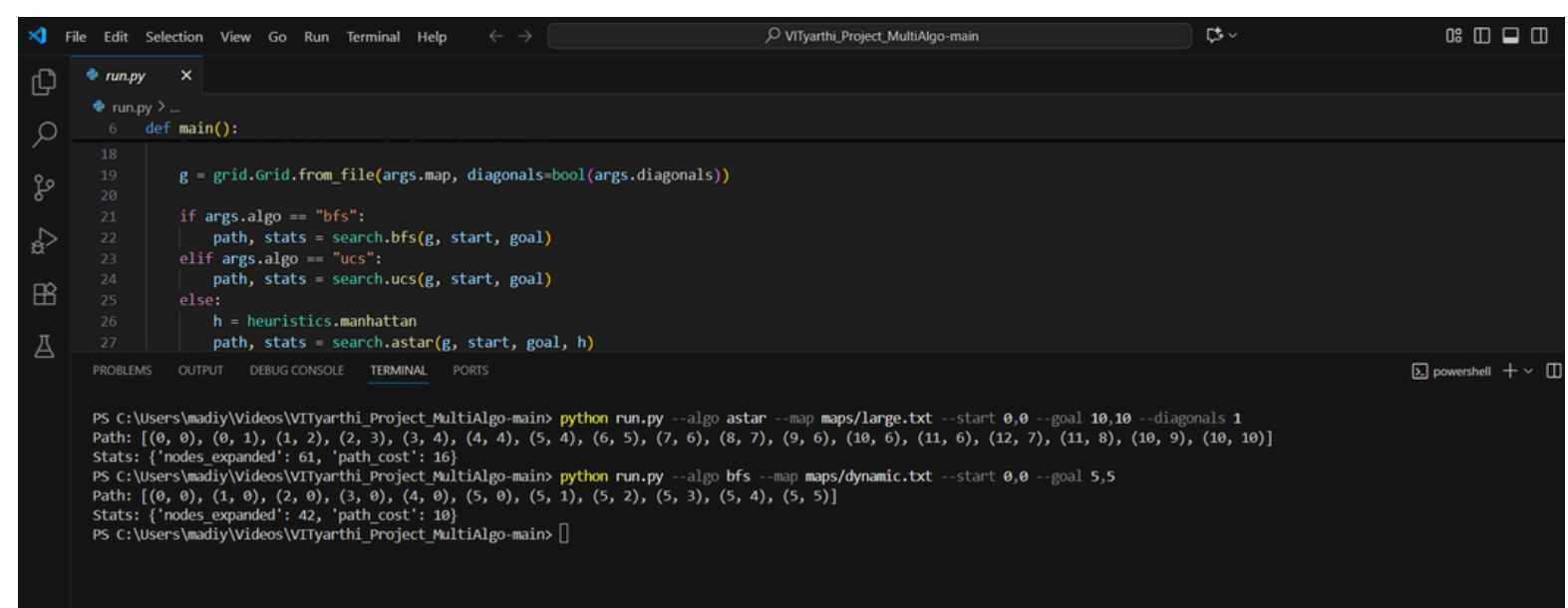
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo astar --map maps/medium.txt --start 0,0 --goal 8,8 --diagonals 1
Path: [(0, 0), (0, 1), (1, 2), (2, 2), (3, 3), (3, 4), (2, 5), (3, 6), (4, 7), (5, 8), (6, 8), (7, 9), (8, 8)]
Stats: {'nodes_expanded': 16, 'path_cost': 12}
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main>

• Large Map



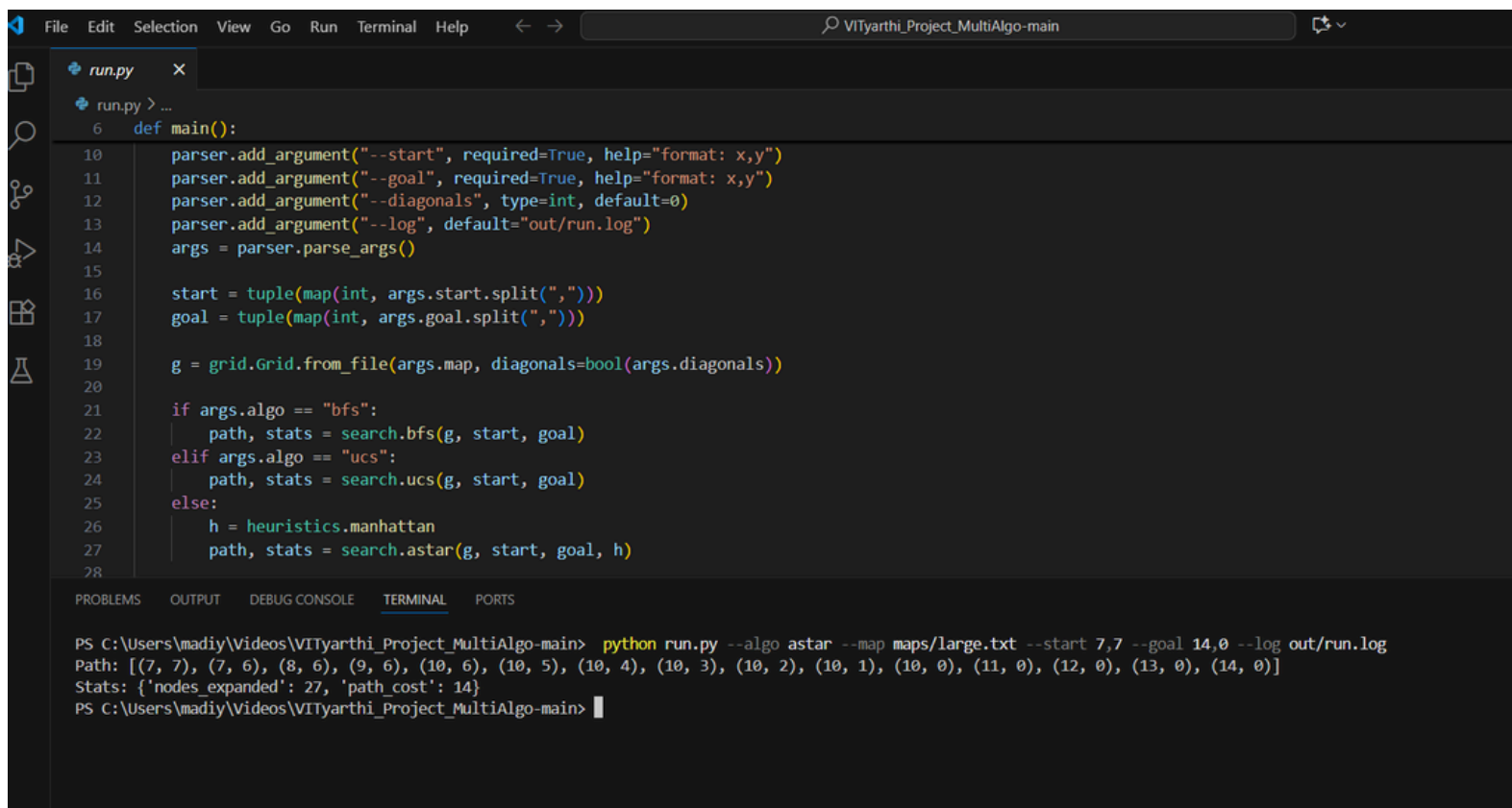
```
run.py
run.py > ...
6 def main():
10     parser.add_argument("--start", required=True, help="format: x,y")
11     parser.add_argument("--goal", required=True, help="format: x,y")
12     parser.add_argument("--diagonals", type=int, default=0)
13     parser.add_argument("--log", default="out/run.log")
14     args = parser.parse_args()
15
16     start = tuple(map(int, args.start.split(",")))
17     goal = tuple(map(int, args.goal.split(",")))
18
19     g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))
20
21     if args.algo == "bfs":
22         path, stats = search.bfs(g, start, goal)
23     elif args.algo == "ucs":
24         path, stats = search.ucs(g, start, goal)
25     else:
26         h = heuristics.manhattan
27         path, stats = search.astar(g, start, goal, h)
28
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo astar --map maps/large.txt --start 0,0 --goal 10,10 --diagonals 1
Path: [(0, 0), (0, 1), (1, 2), (2, 3), (3, 4), (4, 4), (5, 4), (6, 5), (7, 6), (8, 7), (9, 6), (10, 6), (11, 6), (12, 7), (11, 8), (10, 9), (10, 10)]
Stats: {'nodes_expanded': 61, 'path_cost': 16}
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main>
```

TEST 4: BFS on large map with logging



```
run.py
run.py > ...
6 def main():
18
19     g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))
20
21     if args.algo == "bfs":
22         path, stats = search.bfs(g, start, goal)
23     elif args.algo == "ucs":
24         path, stats = search.ucs(g, start, goal)
25     else:
26         h = heuristics.manhattan
27         path, stats = search.astar(g, start, goal, h)
28
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo astar --map maps/large.txt --start 0,0 --goal 10,10 --diagonals 1
Path: [(0, 0), (0, 1), (1, 2), (2, 3), (3, 4), (4, 4), (5, 4), (6, 5), (7, 6), (8, 7), (9, 6), (10, 6), (11, 6), (12, 7), (11, 8), (10, 9), (10, 10)]
Stats: {'nodes_expanded': 61, 'path_cost': 16}
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo bfs --map maps/dynamic.txt --start 0,0 --goal 5,5
Path: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5)]
Stats: {'nodes_expanded': 42, 'path_cost': 10}
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main>
```

TEST 5: A* on large map, log to custom file



The image shows a Visual Studio Code editor window with a file named `run.py` open. The editor is displaying the following Python code:

```
6 def main():
10     parser.add_argument("--start", required=True, help="format: x,y")
11     parser.add_argument("--goal", required=True, help="format: x,y")
12     parser.add_argument("--diagonals", type=int, default=0)
13     parser.add_argument("--log", default="out/run.log")
14     args = parser.parse_args()
15
16     start = tuple(map(int, args.start.split(",")))
17     goal = tuple(map(int, args.goal.split(",")))
18
19     g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))
20
21     if args.algo == "bfs":
22         path, stats = search.bfs(g, start, goal)
23     elif args.algo == "ucs":
24         path, stats = search.ucs(g, start, goal)
25     else:
26         h = heuristics.manhattan
27         path, stats = search.astar(g, start, goal, h)
28
```

Below the code editor, the **TERMINAL** tab is active, showing the command executed and its output:

```
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo astar --map maps/large.txt --start 7,7 --goal 14,0 --log out/run.log
Path: [(7, 7), (7, 6), (8, 6), (9, 6), (10, 6), (10, 5), (10, 4), (10, 3), (10, 2), (10, 1), (10, 0), (11, 0), (12, 0), (13, 0), (14, 0)]
Stats: {'nodes_expanded': 27, 'path_cost': 14}
PS C:\Users\madiy\Videos\VITYarthi_Project_MultiAlgo-main>
```

NAME: M.NEHITH

REG NO: 24MIM10107