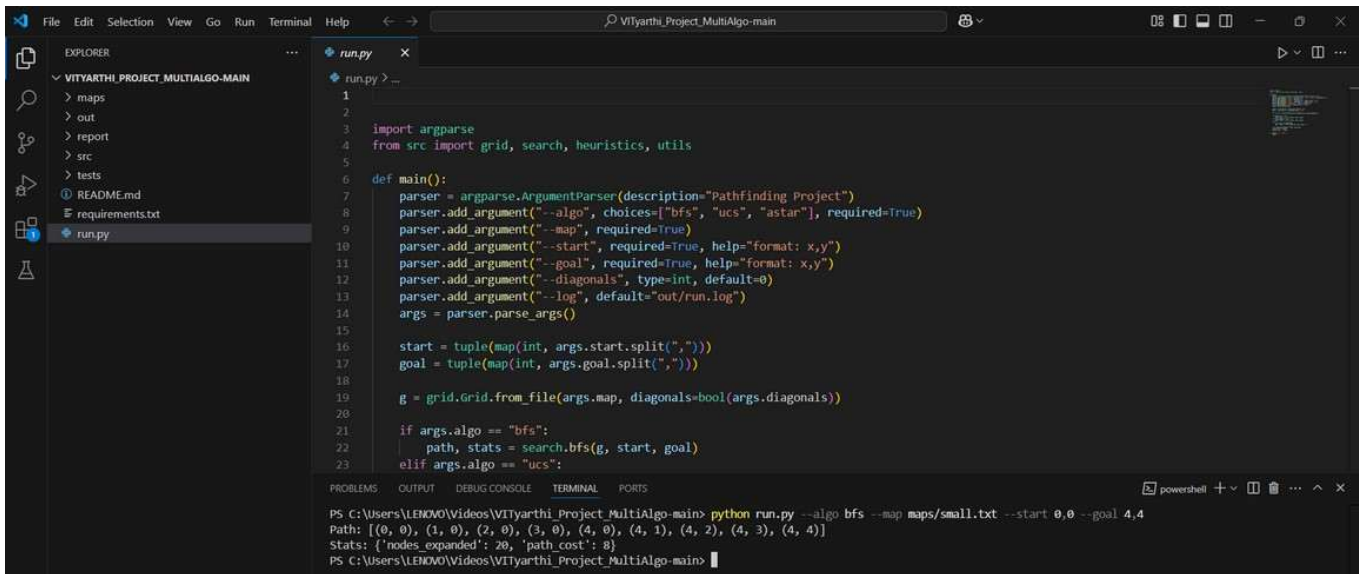


Sample Screenshots

TEST 1 : BFS

- SMALL MAP

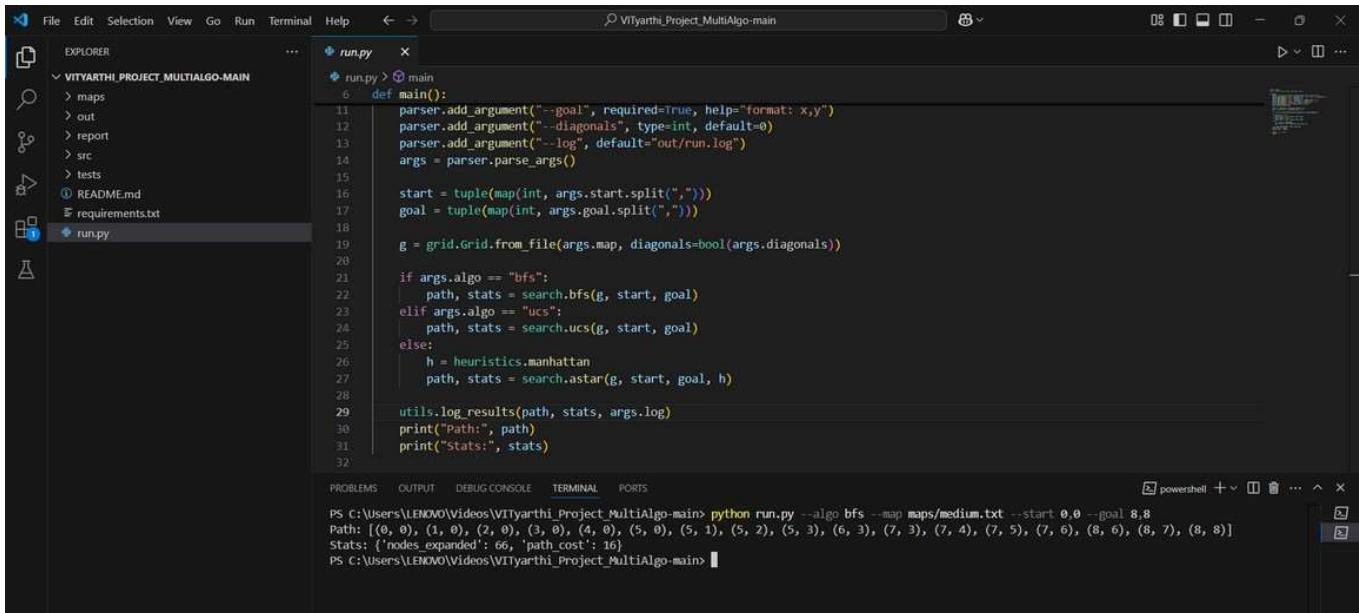


The screenshot shows the Visual Studio Code interface with the Explorer, Run and Debug, and Terminal panels. The Explorer panel on the left shows the project structure for 'VITYARTHI_PROJECT_MULTIALGO-MAIN', including folders like 'maps', 'out', 'report', 'src', and 'tests', and files like 'README.md', 'requirements.txt', and 'run.py'. The Run and Debug panel in the center shows the 'run.py' file with the following code:

```
1
2
3 import argparse
4 from src import grid, search, heuristics, utils
5
6 def main():
7     parser = argparse.ArgumentParser(description="Pathfinding Project")
8     parser.add_argument("--algo", choices=["bfs", "ucs", "astar"], required=True)
9     parser.add_argument("--map", required=True)
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":
```

The Terminal panel at the bottom shows the command prompt output for the command `python run.py --algo bfs --map maps/small.txt --start 0,0 --goal 4,4`. The output displays the path `Path: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]` and the statistics `Stats: {'nodes_expanded': 20, 'path_cost': 8}`.

- 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', and '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 the following code:

```
11 def main():
12     parser.add_argument("--goal", required=True, help="format: x,y")
13     parser.add_argument("--diagonals", type=int, default=0)
14     parser.add_argument("--log", default="out/run.log")
15     args = parser.parse_args()
16
17     start = tuple(map(int, args.start.split(",")))
18     goal = tuple(map(int, args.goal.split(",")))
19
20     g = grid.Grid.from_file(args.map, diagonals=bool(args.diagonals))
21
22     if args.algo == "bfs":
23         path, stats = search.bfs(g, start, goal)
24     elif args.algo == "ucs":
25         path, stats = search.ucs(g, start, goal)
26     else:
27         h = heuristics.manhattan
28         path, stats = search.astar(g, start, goal, h)
29
30     utils.log_results(path, stats, args.log)
31     print("Path:", path)
32     print("Stats:", stats)
```

The Terminal panel at the bottom shows the command prompt output for the command `python run.py --algo bfs --map maps/medium.txt --start 0,0 --goal 8,8`. The output displays the path `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 statistics `Stats: {'nodes_expanded': 66, 'path_cost': 16}`.

- LARGE MAP

The screenshot shows the Visual Studio Code interface with a project named 'Vityarthi_Project_MultiAlgo-main'. The Explorer panel on the left shows the project structure. The main editor displays the 'run.py' file, which contains a Python script for a pathfinding algorithm. The script uses argparse for command-line arguments and implements BFS, UCS, and A* search algorithms. The terminal at the bottom shows the command: `python run.py --algo bfs --map maps/large.txt --start 0,0 --goal 10,10`. The output shows the path and statistics for the BFS algorithm on a large map.

```
def main():
    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)

    utils.log_results(path, stats, args.log)
    print("Path:", path)
    print("Stats:", stats)
```

Terminal Output:

```
PS C:\Users\LENOVO\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), (12, 9), (12, 10)]
Stats: {'nodes_expanded': 130, 'path_cost': 24}
PS C:\Users\LENOVO\Videos\Vityarthi_Project_MultiAlgo-main>
```

TEST 2: UCS

- SMALL MAP

The screenshot shows the Visual Studio Code interface with the same project. The main editor displays the 'run.py' file. The terminal at the bottom shows the command: `python run.py --algo ucs --map maps/small.txt --start 0,0 --goal 4,4`. The output shows the path and statistics for the UCS algorithm on a small map.

```
def main():
    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)

    utils.log_results(path, stats, args.log)
    print("Path:", path)
    print("Stats:", stats)
```

Terminal Output:

```
PS C:\Users\LENOVO\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\LENOVO\Videos\Vityarthi_Project_MultiAlgo-main>
```

- MEDIUM MAP

The screenshot shows a Visual Studio Code window with a project named 'VITYarthi_Project_MultiAlgo-main'. The Explorer panel on the left shows the project structure with files like 'maps', 'out', 'report', 'src', 'tests', 'README.md', 'requirements.txt', and 'run.py'. The 'run.py' file is open in the editor, showing a Python script for a pathfinding algorithm. The script uses argparse for command-line arguments, a Grid class for the map, and search algorithms (bfs, ucs, and astar with a heuristic) to find a path from a start point to a goal point. The terminal at the bottom shows the command 'python run.py --algo ucs --map maps/medium.txt --start 0,0 --goal 8,8' and its output, which includes the path and statistics.

```
def main():
    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)

    utils.log_results(path, stats, args.log)
    print("Path:", path)
    print("Stats:", stats)
```

PS C:\Users\LENOVO\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\LENOVO\Videos\VITYarthi_Project_MultiAlgo-main>

- **LARGE MAP**

This screenshot is similar to the first one, showing the same VS Code environment and 'run.py' script. The terminal shows the command 'python run.py --algo ucs --map maps/large.txt --start 0,0 --goal 10,10'. The output displays a longer path and statistics for the 'large.txt' map.

```
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}
```

TEST 3 : A* (A STAR) WITH DIAGONALS

- **SMALL MAP**

The screenshot shows the Visual Studio Code interface with a Python file named `run.py` open. The file contains a script for a pathfinding algorithm. The terminal at the bottom shows the command `python run.py --algo astar --map maps/small.txt --start 0,0 --goal 4,4 --diagonals 1` and its output, which includes the path `[(0, 0), (0, 1), (1, 2), (2, 3), (3, 4), (4, 4)]` and statistics: `Stats: {'nodes_expanded': 6, 'path_cost': 5}`.

```
def main():
    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)

    utils.log_results(path, stats, args.log)
    print("Path:", path)
    print("Stats:", stats)
```

PS C:\Users\LENOVO\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\LENOVO\Videos\VITYarthi_Project_MultiAlgo-main>

• MEDIUM MAP

The screenshot shows the Visual Studio Code interface with the same `run.py` file. The terminal at the bottom shows the command `python run.py --algo astar --map maps/medium.txt --start 0,0 --goal 8,8 --diagonals 1` and its output, which includes 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 statistics: `Stats: {'nodes_expanded': 16, 'path_cost': 12}`.

```
def main():
    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)

    utils.log_results(path, stats, args.log)
    print("Path:", path)
    print("Stats:", stats)
```

PS C:\Users\LENOVO\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\LENOVO\Videos\VITYarthi_Project_MultiAlgo-main>

• LARGE MAP


```
File Edit Selection View Go Run Terminal Help
VITyarthi_Project_MultiAlgo-main

EXPLORER
VITYARTHI_PROJECT_MULTIALGO-MAIN
  > maps
  > out
  > report
  > src
  > tests
  @ README.md
  requirements.txt
  run.py

run.py
6 def main():
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
29     utils.log_results(path, stats, args.log)
30     print("Path:", path)
31     print("Stats:", stats)
32
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\LENOVO\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\LENOVO\Videos\VITyarthi_Project_MultiAlgo-main>
```

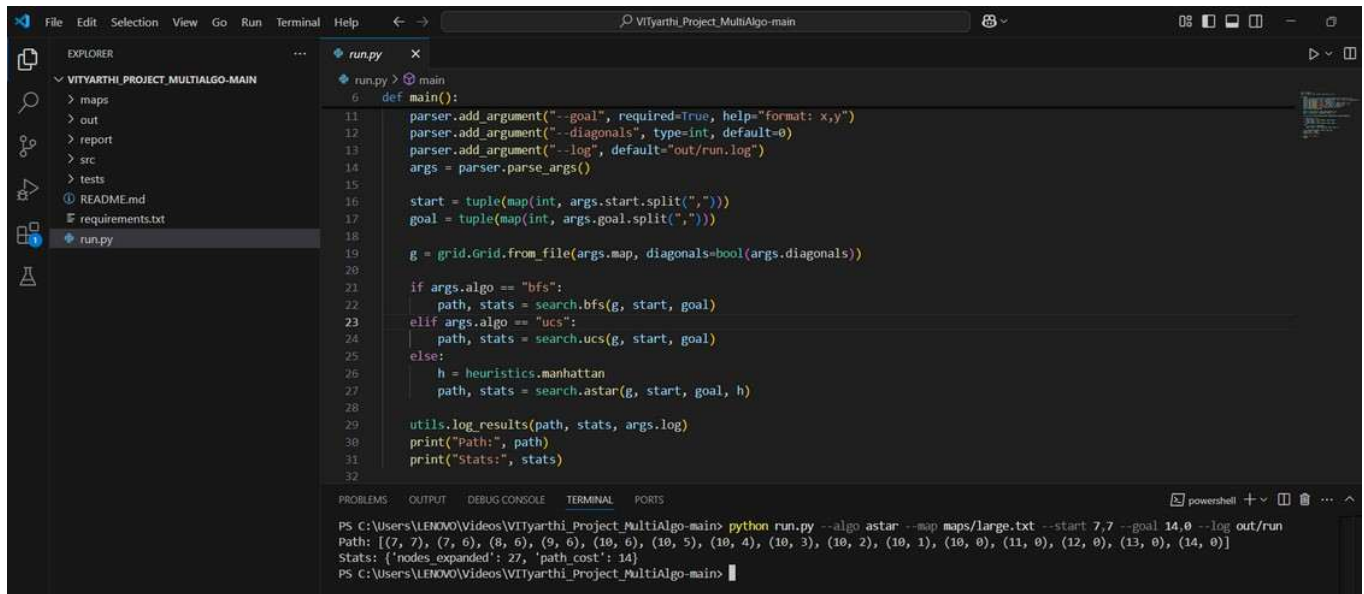
TEST 4 : BFS ON LARGE MAP WITH LOGGING

```
File Edit Selection View Go Run Terminal Help
VITyarthi_Project_MultiAlgo-main

EXPLORER
VITYARTHI_PROJECT_MULTIALGO-MAIN
  > maps
  > out
  > report
  > src
  > tests
  @ README.md
  requirements.txt
  run.py

run.py
6 def main():
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
29     utils.log_results(path, stats, args.log)
30     print("Path:", path)
31     print("Stats:", stats)
32
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\LENOVO\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\LENOVO\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 project named 'VITYarthi_Project_MultiAlgo-main'. The Explorer sidebar on the left shows the project structure with folders 'maps', 'out', 'report', 'src', and 'tests', and files 'README.md', 'requirements.txt', and 'run.py'. The 'run.py' file is open in the editor, showing a Python script that uses the argparse module to parse command-line arguments for a pathfinding algorithm. The script defines a main function that takes arguments for goal, diagonals, and log, and then uses a Grid class and search algorithms (bfs, ucs, astar) to find a path. The terminal at the bottom shows the command 'python run.py --algo astar --map maps/large.txt --start 7,7 --goal 14,0 --log out/run' and its output, which includes the path and statistics.

```
run.py x
run.py > main
6 def main():
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
29     utils.log_results(path, stats, args.log)
30     print("Path:", path)
31     print("Stats:", stats)
32
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\LENOVO\Videos\VITYarthi_Project_MultiAlgo-main> python run.py --algo astar --map maps/large.txt --start 7,7 --goal 14,0 --log out/run

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\LENOVO\Videos\VITYarthi_Project_MultiAlgo-main>

NAME : K.BALA YASWANTH

REG.NO : 24MIM10243