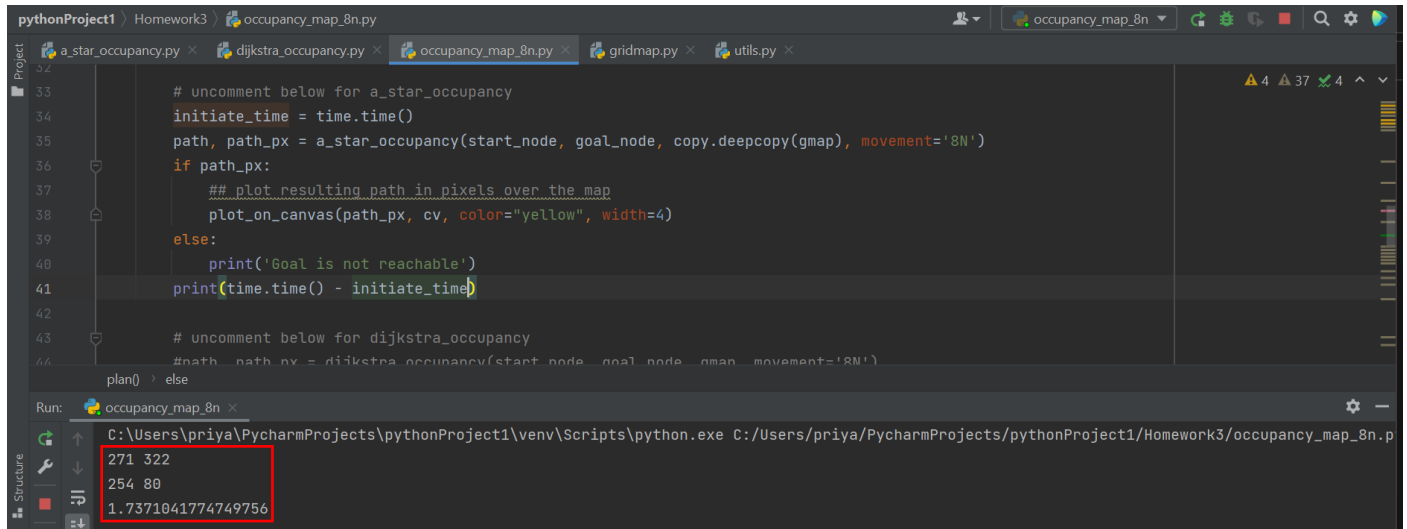


PART-B

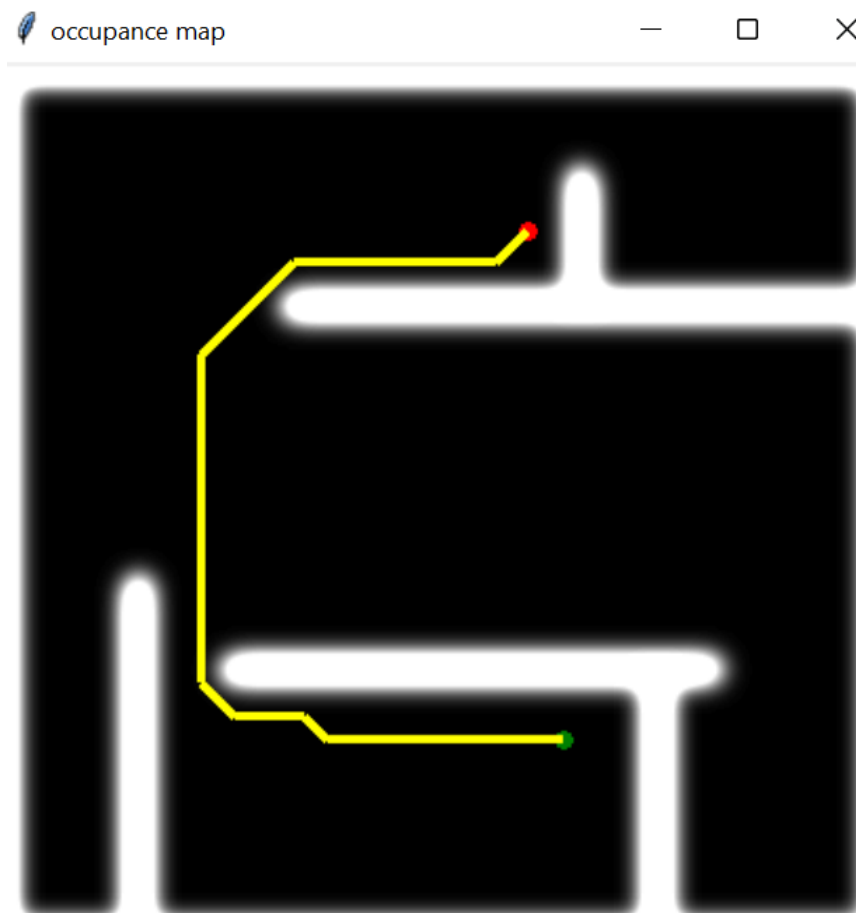
All codes were submitted previously and can be found in a zip folder on Canvas.

1.a. Demo existing reference and get to know the behaviour of its path search. (The demo run file is: examples/occupancy_map_8n.py)

Solution:



```
pythonProject1 > Homework3 > occupancy_map_8n.py
a_star_occupancy.py x dijkstra_occupancy.py x occupancy_map_8n.py x gridmap.py x utils.py x
32
33 # uncomment below for a_star_occupancy
34 initiate_time = time.time()
35 path, path_px = a_star_occupancy(start_node, goal_node, copy.deepcopy(gmap), movement='8N')
36 if path_px:
37     ## plot resulting path in pixels over the map
38     plot_on_canvas(path_px, cv, color="yellow", width=4)
39 else:
40     print('Goal is not reachable')
41 print(time.time() - initiate_time)
42
43 # uncomment below for dijkstra_occupancy
44 #path, path_px = dijkstra_occupancy(start_node, goal_node, gmap, movement='8N')
plan() > else
Run: occupancy_map_8n.py
C:\Users\priya\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/priya/PycharmProjects/pythonProject1/Homework3/occupancy_map_8n.p
271 322
254 80
1.7371041774749756
```



The figure represents the use of A* algorithm to find the path and traverse the graph from start to end node. The A* algorithm features the use of a heuristic function to estimate the location of the end node.

Also, It took around 1.73 seconds to execute this code.

1.b. Implement occupancy gridmap-based Dijkstra for same functionality as (1a)

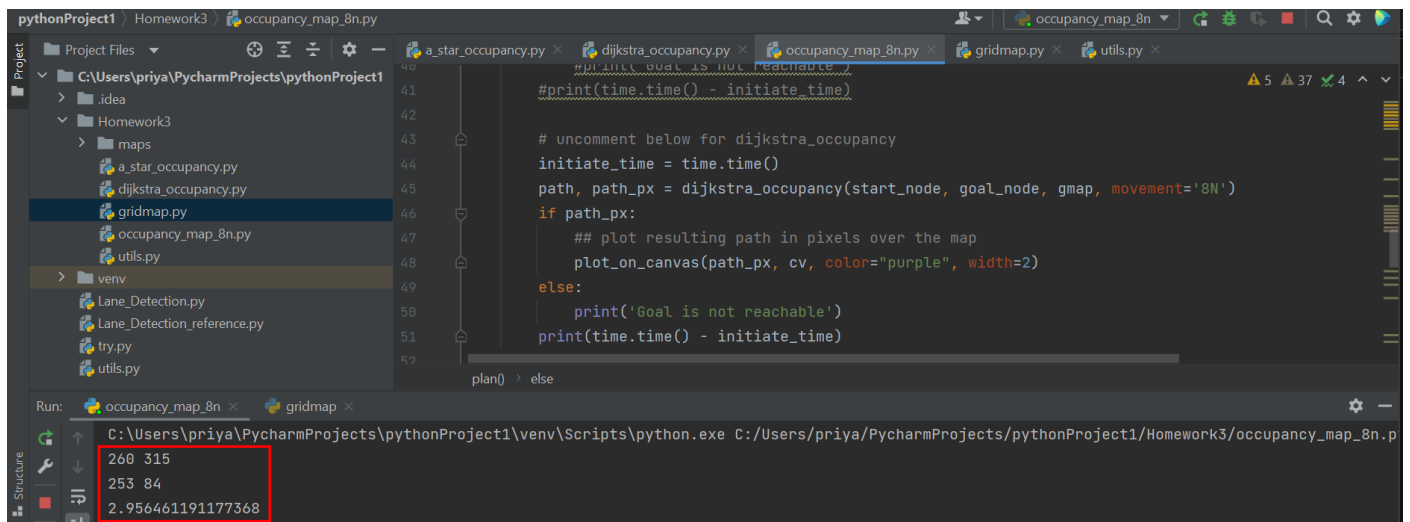
Solution:

The Dijkstra algorithm was defined as –

```
def dijkstra_occupancy(start_m, goal_m, gmap, movement='8N', occupancy_cost_factor=3):
```

The primary difference between the A* and the Dijkstra algorithm is the absence of the distance heuristic function in Dijkstra algorithm; which is present in the A* algorithm. This is the reason why Dijkstra algorithm takes more time in finding the goal node when compared to the A* algorithm.

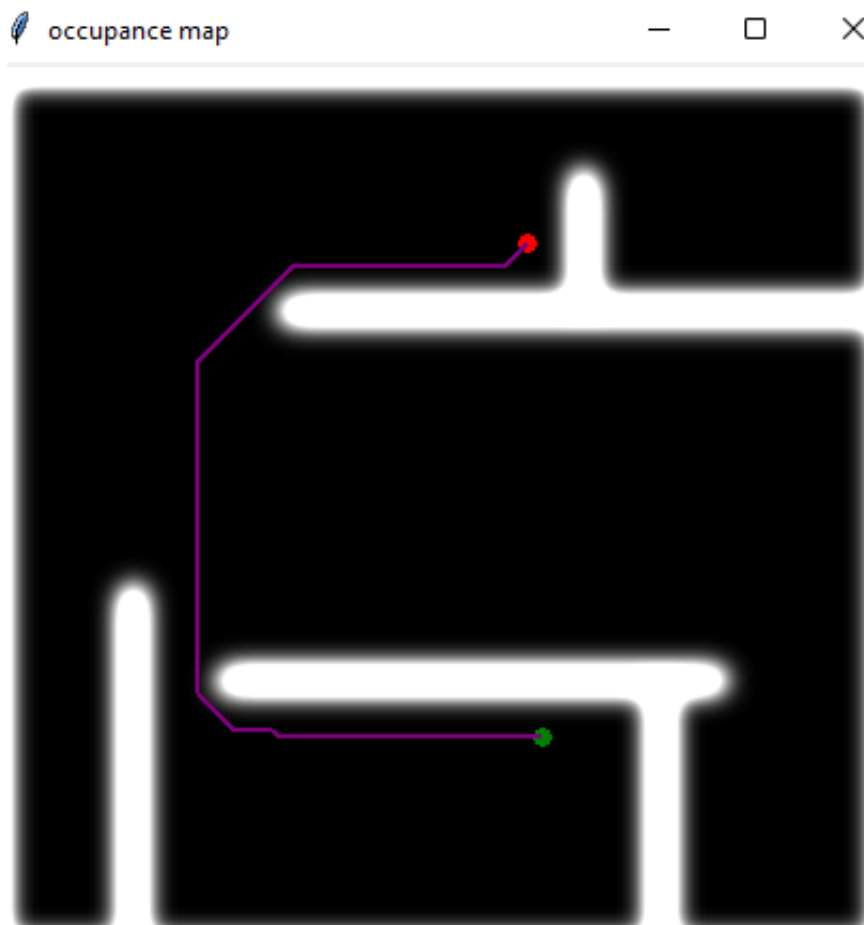
For instance, in the below example Dijkstra took around 2.95 seconds to find the goal node which is more than what A* algorithm took (i.e., 1.73 seconds).



The screenshot shows the PyCharm IDE with the following details:

- Project Files:** The left sidebar shows the project structure with files like `a_star_occupancy.py`, `dijkstra_occupancy.py`, `gridmap.py`, `occupancy_map_8n.py`, and `utils.py`.
- Code Editor:** The `dijkstra_occupancy.py` file is open, showing the function definition and its execution. The code includes comments and a `plot_on_canvas` function call.
- Run Console:** The bottom console shows the output of the program, including the coordinates of the start and goal nodes and the time taken to find the path.

```
Run: C:\Users\priya\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\priya\PycharmProjects\pythonProject1\Homework3\occupancy_map_8n.p
260 315
253 84
2.956461191177368
```

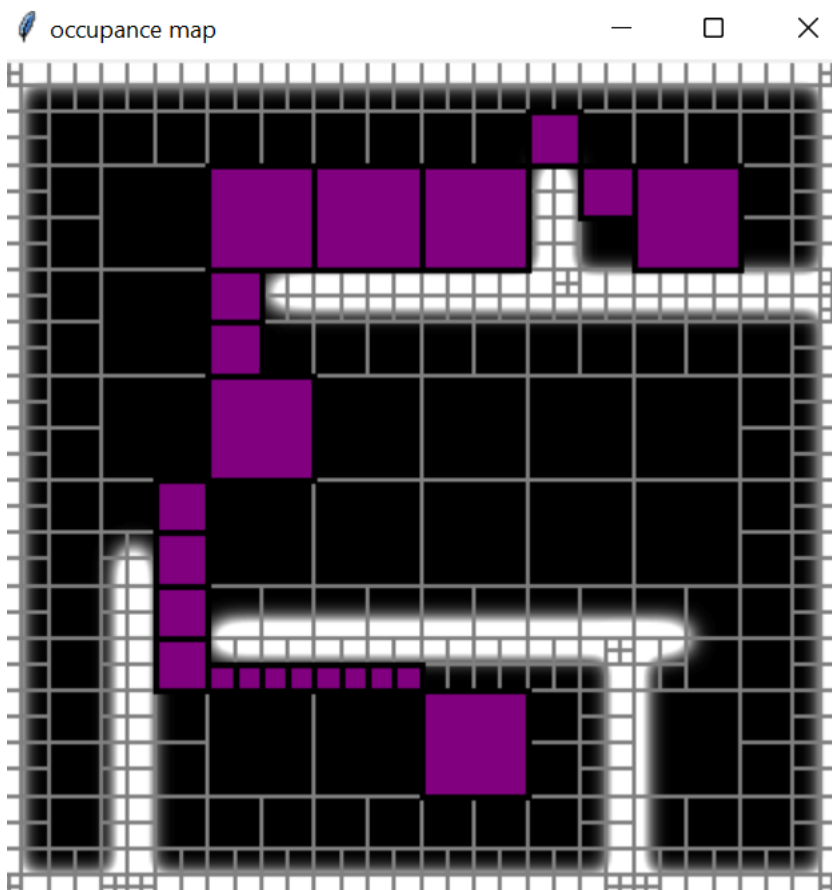


2.a. Demo existing reference and get to know the behaviour of its path search. (2') The demo run file is: examples/quadtrees_map_8n.py

Solution:

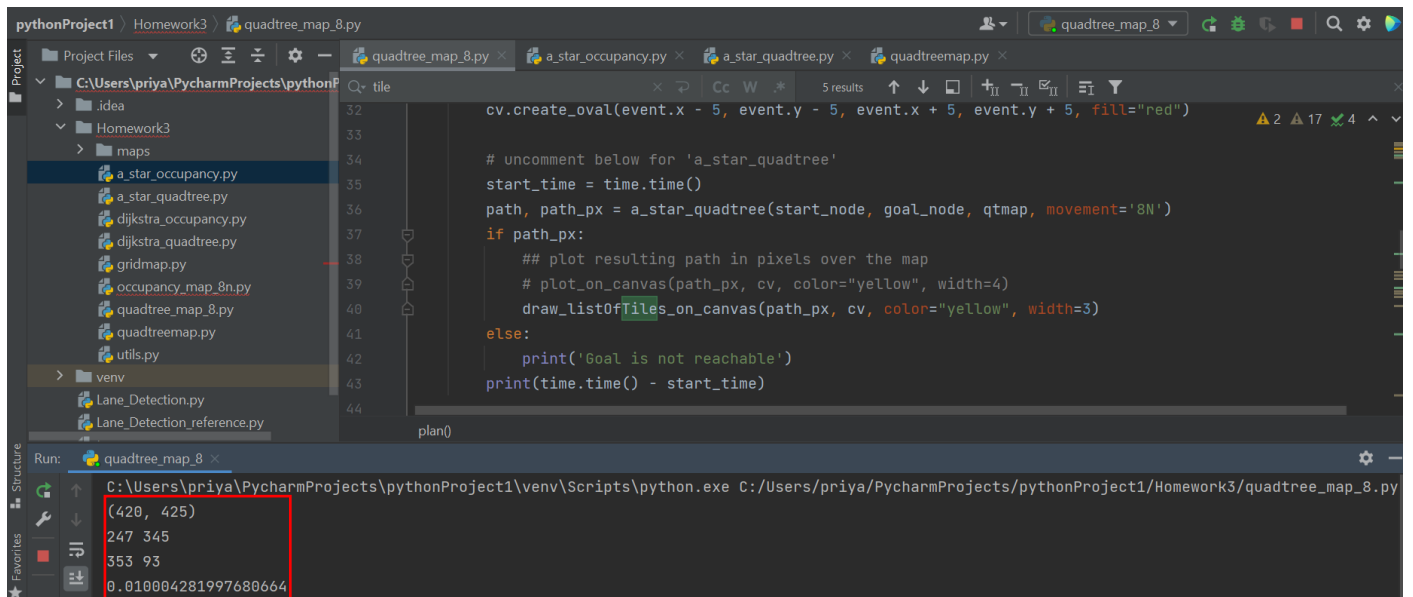
```
pythonProject1 > Homework3 > quadtree_map_8.py
quadtree_map_8.py x dijkstra_quadtree.py x quadtreemap.py x
44
45 # uncomment below for 'dijkstra_quadtree'
46 initiate_time = time.time()
47 path, path_px = dijkstra_quadtree(start_node, goal_node, qtmap, movement='8N')
48 if path_px:
49     # plot resulting path in pixels over the map
50     # plot_on_canvas(path_px, cv, color="purple", width=2)
51     draw_listOfTiles_on_canvas(path_px, cv, color="purple", width=3)
52 else:
53     print('Goal is not reachable')
54 print(time.time() - initiate_time)
55
if __name__ == '__main__':
    Run: quadtree_map_8 x gridmap x
    C:\Users\priya\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\priya\PycharmProjects\pythonProject1\Homework3\quadtree_map_8.py
    (420, 425)
    246 332
    326 72
    0.010030031204223633
```

The output below uses Dijkstra's algorithm and took around 0.01003 seconds to execute.



2.b. Implement Quadtree map-based A* for same functionality as (2a)

Solution:



```
pythonProject1 > Homework3 > quadtree_map_8.py
cv.create_oval(event.x - 5, event.y - 5, event.x + 5, event.y + 5, fill="red")

# uncomment below for 'a_star_quadtree'
start_time = time.time()
path, path_px = a_star_quadtree(start_node, goal_node, qtmap, movement='8N')
if path_px:
    ## plot resulting path in pixels over the map
    # plot_on_canvas(path_px, cv, color="yellow", width=4)
    draw_list_of_files_on_canvas(path_px, cv, color="yellow", width=3)
else:
    print('Goal is not reachable')
print(time.time() - start_time)

plan()
```

Run: quadtree_map_8

C:\Users\priya\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\priya\PycharmProjects\pythonProject1\Homework3\quadtree_map_8.py

(420, 425)
247 345
353 93
0.010004281997680664

The output below uses A* algorithm and took around 0.010004 seconds to execute. (For A* algorithm a heuristic distance function should be added.)

