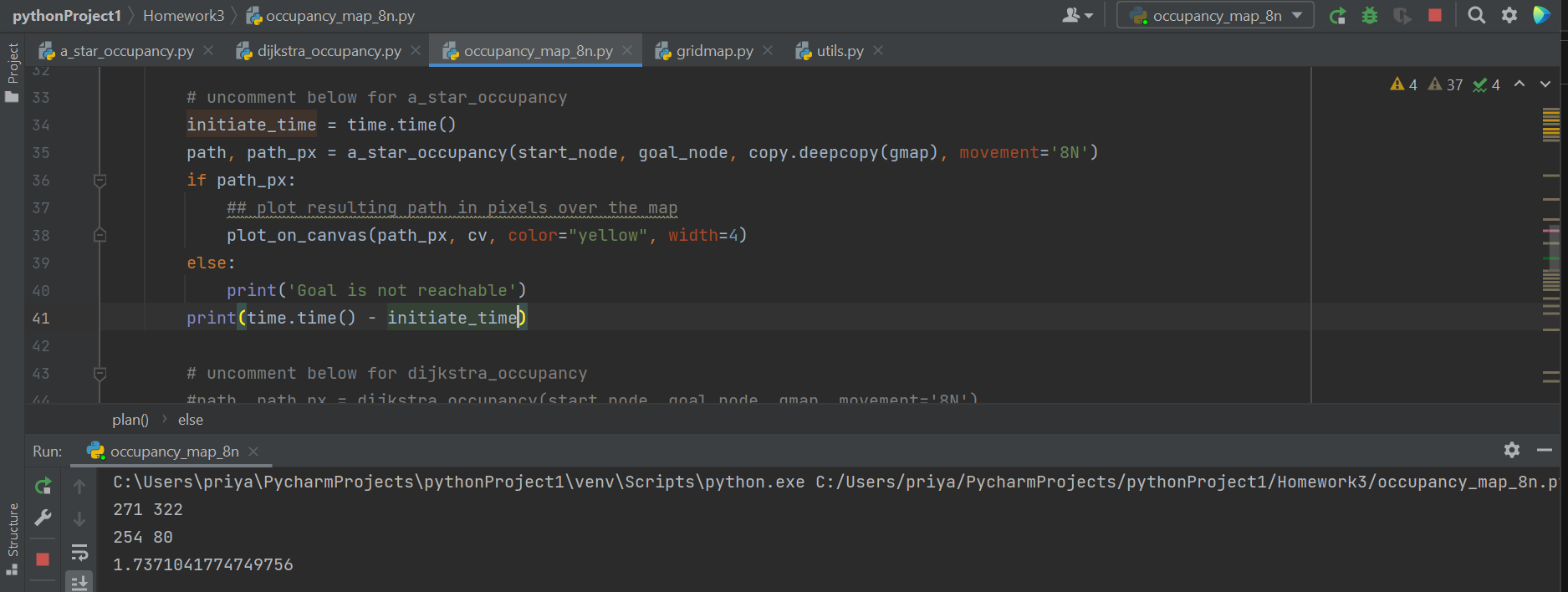
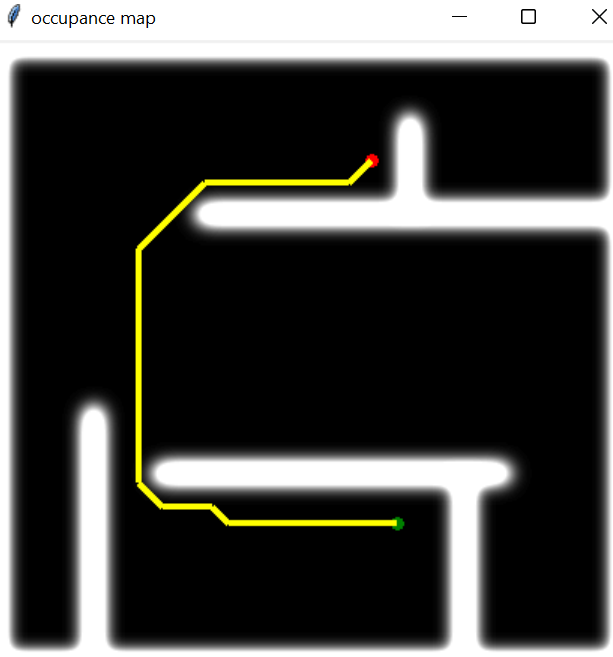
**PART-B**

*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:***





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)**

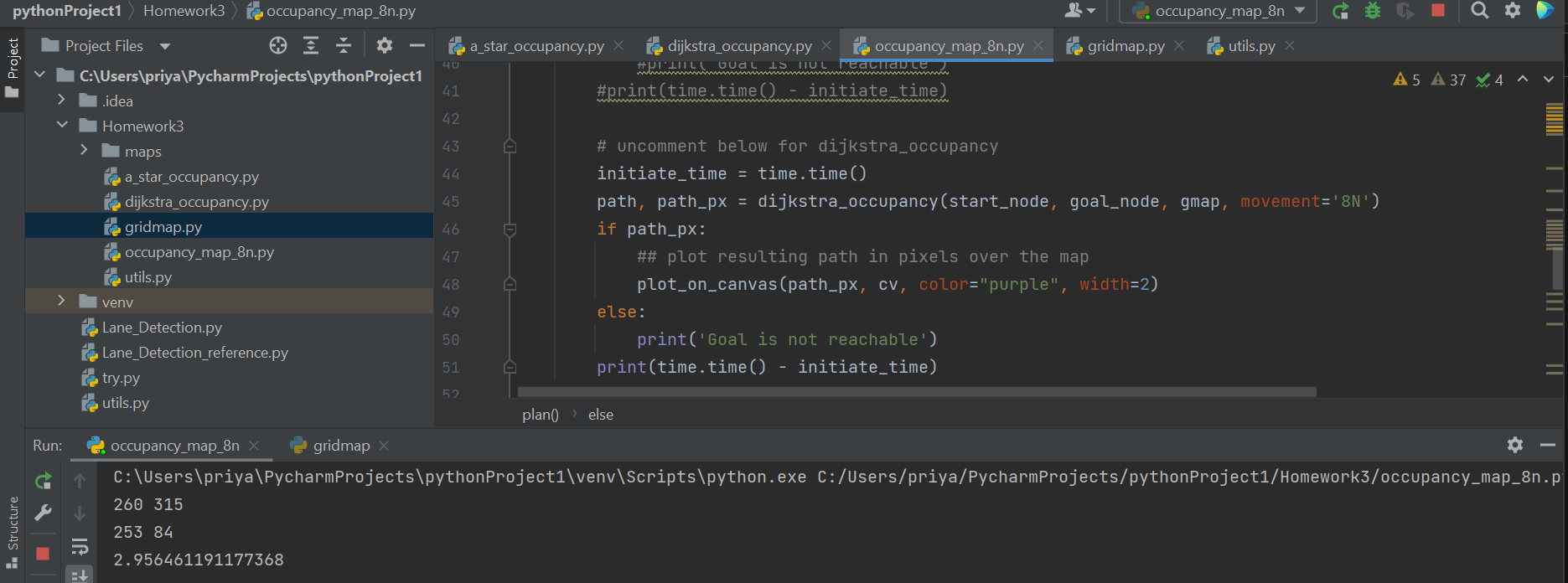
***Solution:***

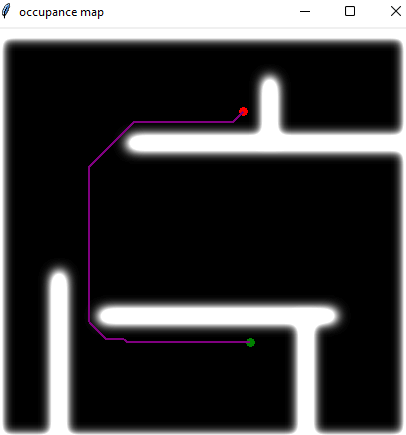
The Dijkstra algorithm was defined as –

**

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.

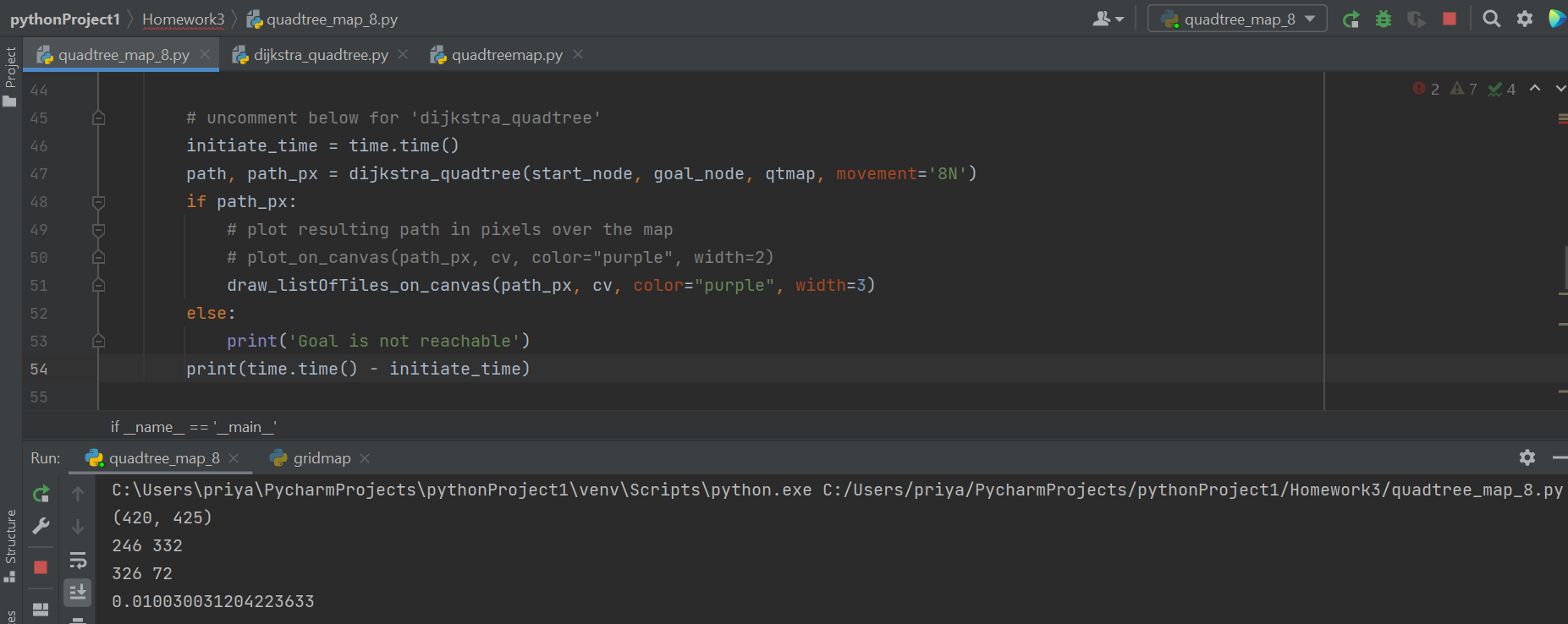
For instance, in the below example Dijkstra took around 2.95 seconds to find the goal node which is more than what A\* algorithm took (i.e., 1.73 seconds).

****

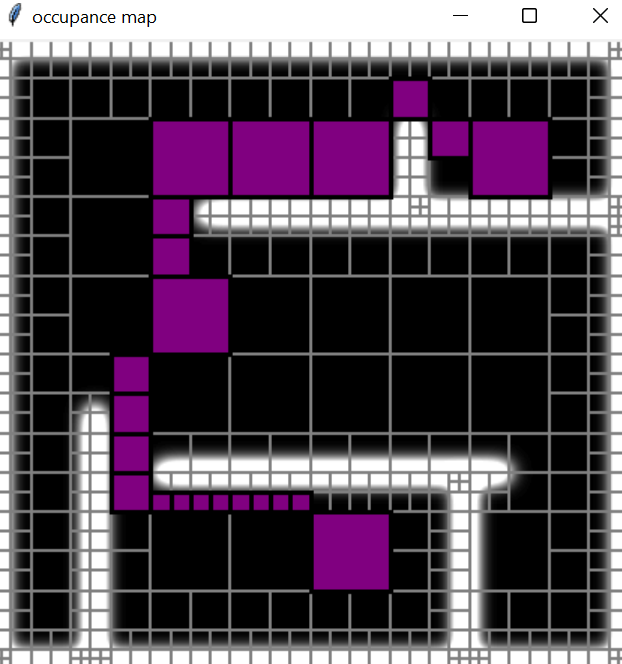


**2.a. Demo existing reference and get to know the behaviour of its path search. (2’) The demo run file is: examples/quadtree\_map\_8n.py**

***Solution:***

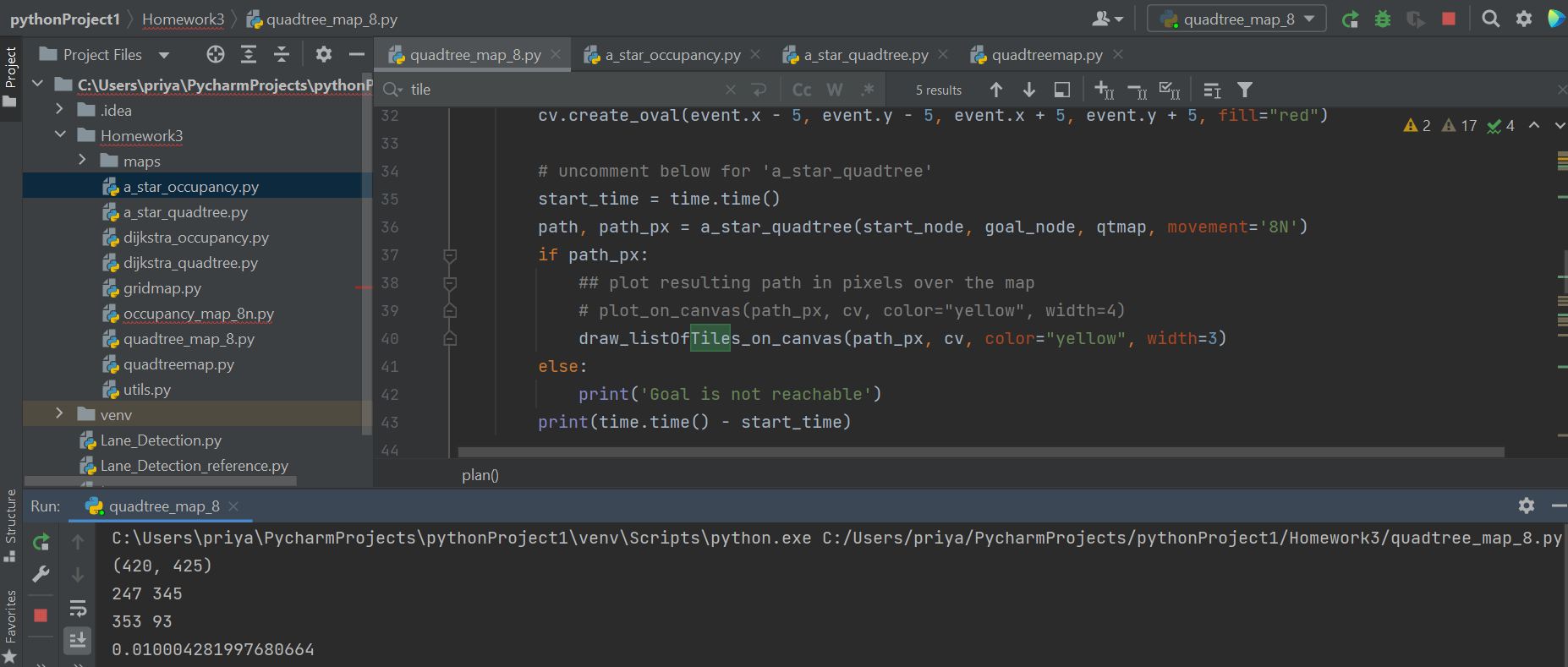
****

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:***

****

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

