Digital Design II
Mini Project
Dr Mohamed Shalan

Interactive A* Multi layer Maze Router

Ahmed Ashraf Taha 900152132

A* Algorithm

First i got the idea of the algorithm and that the BFS is a special case of the A* algorithm, as the A* is defined by the function f = g + h, h is the heuristic function which makes this algorithm smart and efficient. In the BFS h = 0. In the heurstic function i used the manhattan version of it **h** = abs (current_cell.x – goal.x) + abs (current_cell.y – goal.y) but with a modification that i added the cost of the vias for moving between layers which is adding the via count and cost to the heuristic.

Implementation

The 2 fundamental functions of the program is route and next_node.

1- route , this function adds to the path the source node, then calculate f = g + h which will be f = h then passes the current node to the function next_node, when next node returns the chosen node, route checks whether this node is the destination or not, if yes we add it to the path and return the g’s and h’s to the main , if not we just repeat the process until the destination is reached.

2 - next_node , this method is the most important method of them all, it receives the current node, then checks for the neighbours, if the neighbours are out of boundaries they’re rejected, also if they’re blocked they’re ignored, if they passed from this test, they enters a new one, i compute the value of f of the passed nodes and the lowest value of f node is the chosen one, in the computation of f vias are taken care of as their count and value are added to f when we are switching between layers. This function returns the chosen node in addition to the f value and the g value.

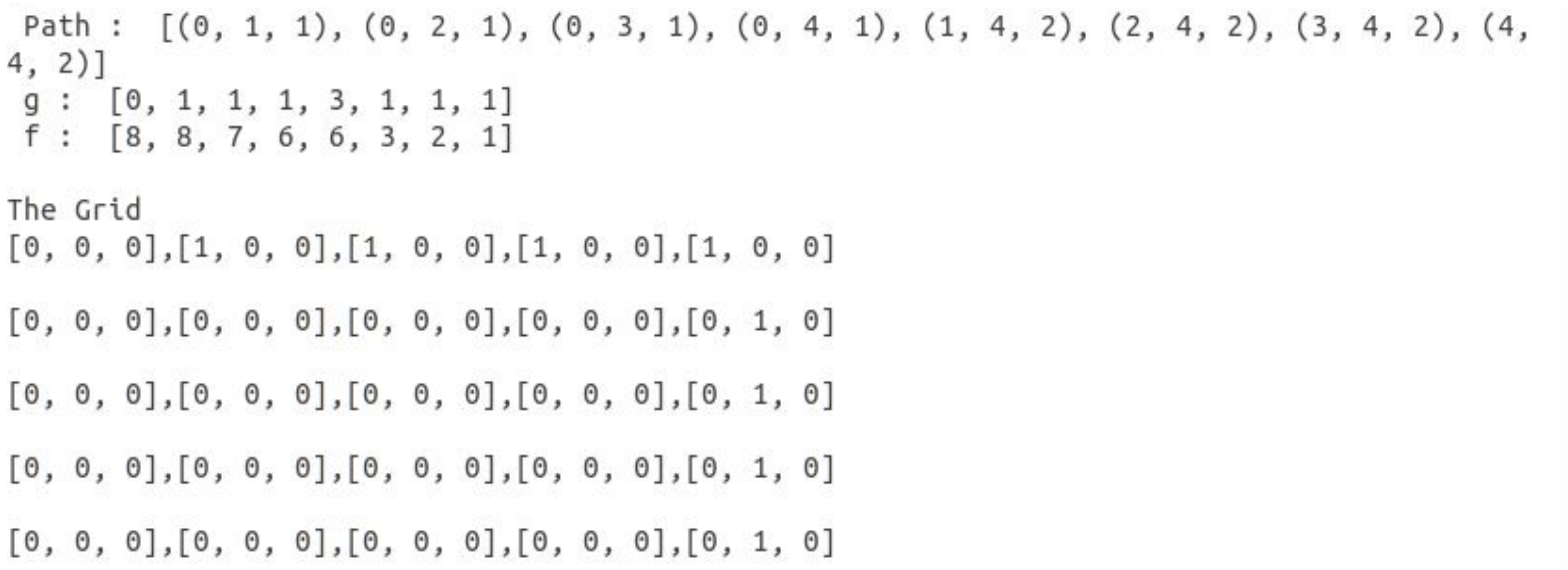
The grid

I initialized the grid at the begining with the set (0,0,0), then each time a cell in a specific layer is involved in a path i increment it by 1, for example if grid [0][0] is involved in layer 1 it's content become (1,0,0) then g[0][1] for example is involved the next step it becomes (1,0,0) then g[1][1] , this case will be moving from one layer to another do it will be (0,1,0). Those elements when passed by again in the same layer will be treated as obstacles.

Outputs g and h

Each one of them is an array, each value represent the cost at each step, in the final step h is Zero so f = h.

Sample Output of a Path from (011)->(4,4,2)



```
 Path :  [(0, 1, 1), (0, 2, 1), (0, 3, 1), (0, 4, 1), (1, 4, 2), (2, 4, 2), (3, 4, 2), (4,
4, 2)]
 g :  [0, 1, 1, 1, 3, 1, 1, 1]
 f :  [8, 8, 7, 6, 6, 3, 2, 1]

The Grid
[0, 0, 0],[1, 0, 0],[1, 0, 0],[1, 0, 0],[1, 0, 0]

[0, 0, 0],[0, 0, 0],[0, 0, 0],[0, 0, 0],[0, 1, 0]

[0, 0, 0],[0, 0, 0],[0, 0, 0],[0, 0, 0],[0, 1, 0]

[0, 0, 0],[0, 0, 0],[0, 0, 0],[0, 0, 0],[0, 1, 0]

[0, 0, 0],[0, 0, 0],[0, 0, 0],[0, 0, 0],[0, 1, 0]
```

Plotting the grid using Matplotlib

I Attached two pictures of plots i tried to do one in 2D and one in 3D, they are included in Test1.py, but i didn’t feel they’re that accurate.

CPU time

I used the following to measure the CPU time in Python.

```
import time.

start = time. time()

"The Program"

end = time. time()

print(end - start)
```

Limitations

I spent a lot of time until i figured out how to put everything together, but it was a good experience after all to be exposed to such routing technique which is so efficient and smart.