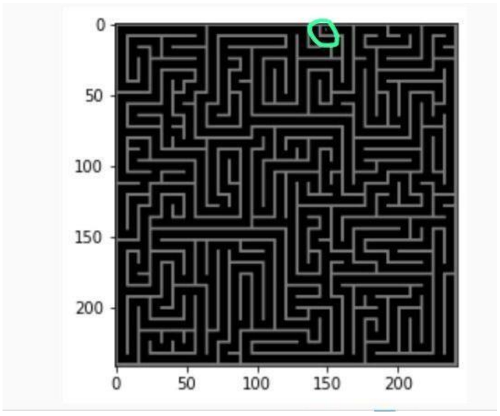
Cairo university

Faculty of engineering

Name	ID
دانيال رفيق حليم	1190263
مهرائيل ايمن مرقص	1190491
رامي هشام أحمد	1190008
يوسف سعد لطفي	4200340

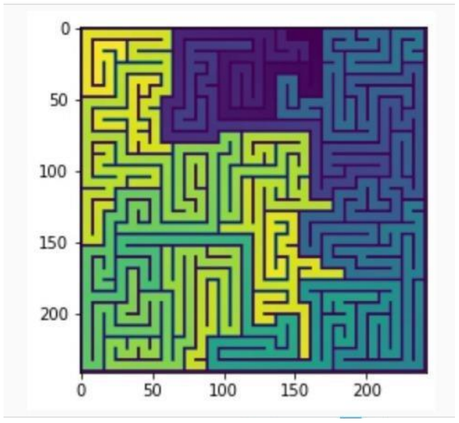
Function find_goal(map,width) Explanation:



First, we passed to this function our map and its width, and in order to decrease the complexity of this function from O(n**2) to O(n) we used the function NumPy. Flatten to change our array from 2D to 1 D then inside our loop we checked if the element equals 2 this function will return its index.

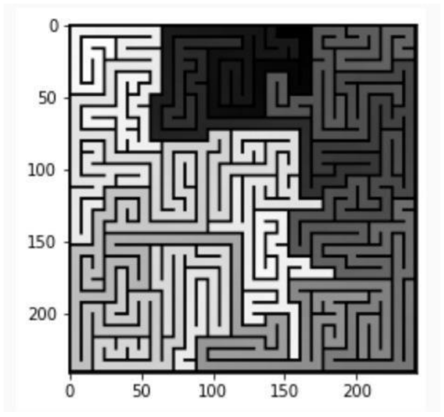
Function the_filling() Explanation:

This function is the main function responsible for filling the matrix from the goal having a value of “2” and increments its surroundings till it fills all the zeros of the map or “2” is surrounded by “1” s then there will be nothing to fill. It calls the “fill_the_matrix()” when we need to increment the value. It takes parameters of the last filled points indexes to know where to fill in the current call, and takes the map array, and takes the width and height to compensate for processing time. Then, we call the trace_back function to search for the goal. The opposite photo is the filled matrix in color scale. Where the lighter colors like the yellow are the ones with larges values. They have the largest distance from the goal.



Function fill_the_matrix() Explanation:

This function is responsible for looping over the matrix of indices. It is called by the “the_filling()” function when it is time to increment the next value. It loops over the matrix of indices to know the coordinates where we need to increment the values in the array then calls the “fill_one_point() ” function to fill each point which get called 8 times for each one of the adjacent of the point. The opposite photo is the filled matrix in gray scale.

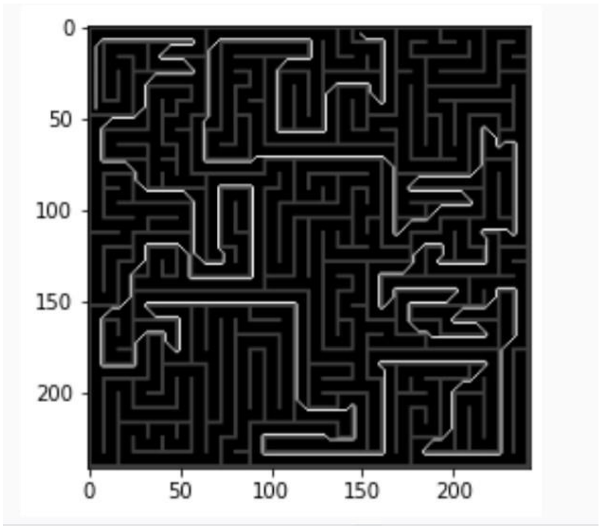


Function fill_one_point()Explanation:

This function is simple. It just changes the value of a certain point and stores its index in a new array. It takes the array and its width and height to check if the coordination in the limits, the coordinate of the points, the new value that should be inserted and the array where we store the coordinate of this new value.

Function trace_back() Explanation:

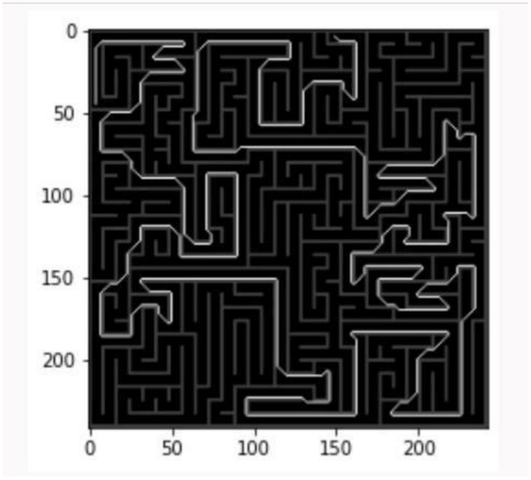
We implemented this function in order to get the shortest path from the starting point to the goal. Our implementation of this function was based on the priority that was given in the document which is as follows: upper, Right, Lower, Left, Upper right, Lower right, Lower left, Upper Left. In every if statement we append the index of the element according to our priority, then we return an array called arr_of_coordinates which has the index of our required path.



Problems that faced us:

One of our problems was that in filling the matrix that the map can be filled with numbers from 0 to 255. if we tried to fill the map with values greater than 256, it would make an overflow and the value would l become 0 instead of 256. We overcame this problem by making each pixel stored in 32 bits instead of 8 bits with the line astype('uint32'). This enabled us to store larger values in the map.
Another problem faced us that we needed a more powerful IDE other than VS code during recursion so we used spyder which was better, we searched more about this problem to understand the reason behind that and we found that the recursion limit was set by default to 1000 when using VScode while it was set to 3000 in spyder, so we managed to hard coded it at the beginning of the program to avoid such a problem and make the program independent on the IDE.

Here is the shortest path between point (45,4) and our goal


0
50
100
150
200
0
50
100
150
200