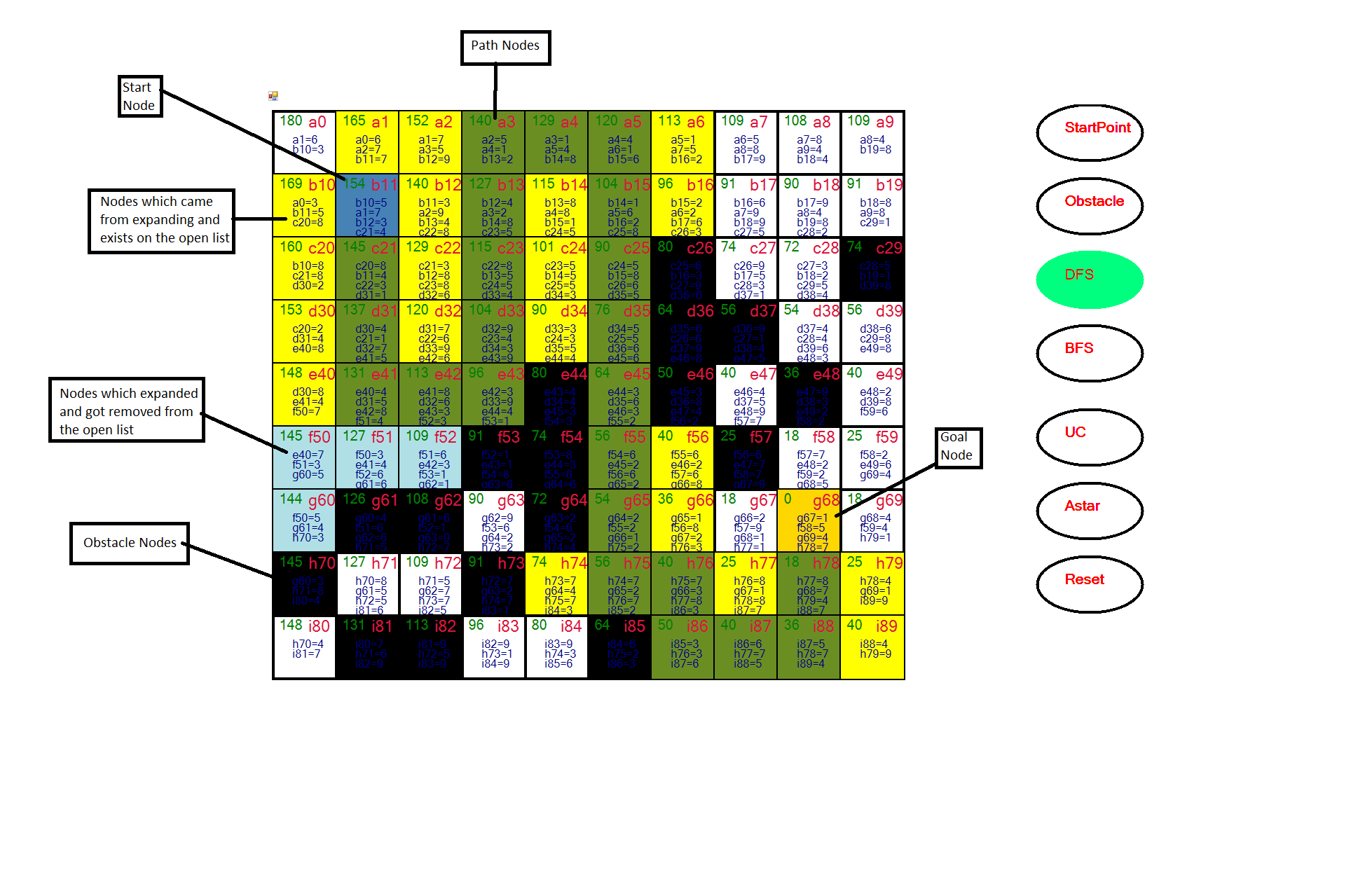
Project Description:

This project will be mainly about path finding and using the search algorithms (DFS , BFS , Uniform Cost , A\* , Greedy) to achieve reaching the goal from the start. This project adds interactivity for the player to adjust the starting position node and the goal node , also it adds freedom for the player to add as much obstacles as he wants in order to make it more challenging for the search techniques to reach the goal . Also when the player presses right click to create the goal node random costs are created between all nodes and the heuristics can be imagined as straight line distance between nodes by means of using Pythagorean theorem. The player can switch between algorithms on the project to compare the performances between them like showing the cost of the path , path drawing , number of iterations it took to find the goal.

Project parts overview:



DFS code Overview:

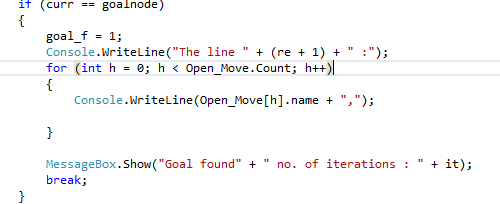
Looping until the open list becomes empty:



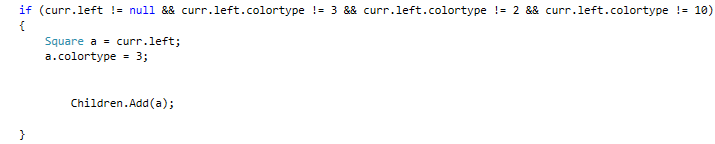
Curr will equal to the head of the open list:



If curr will equal to goalnode break from loop



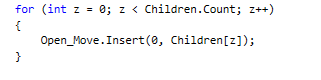
If curr.left exists and curr.left is not of colortype 3(nodes with colortype = 3 or yellow are nodes which already exist on the open list) and curr.left is not of colortype 2(nodes with colortype = 2 or black are obstacles nodes ) and curr.left is not of colortype 10(nodes with colortype = 10 or powderblue are nodes) then put that node into the children list and make it with color type = 3. The same logic will be applied to curr.up , curr.down , curr.right



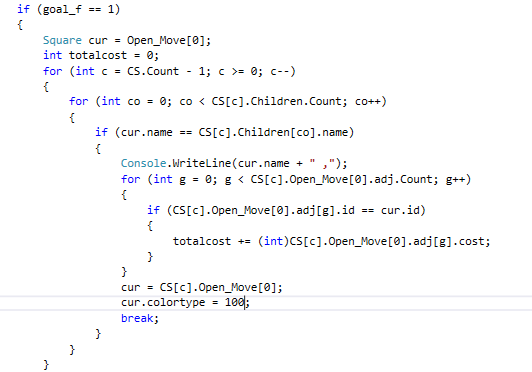
This code removes the current node(head of open list)



This for loop inserts the current node’s children into the open list at the begging

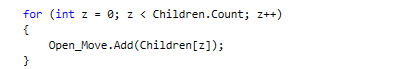


Once a goal is reached We will iterate through a list which stores the states of open list and it’s children so that if we can find child belonging to a parent we can also see that parent belonging to another open list as a child . This allows us to calculate the cost of the path attained by reaching the goal.



BFS code overview:

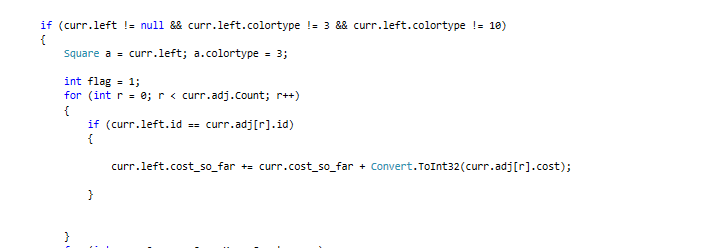
BFS : The only difference BFS has from DFS is that when the children are inserted into the open list they are inserted at the end.



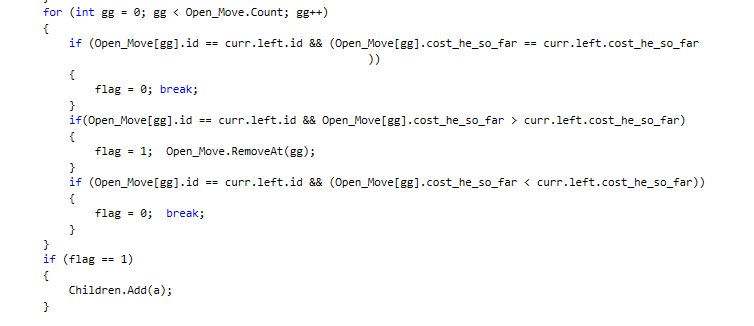
Uniform cost code overview:

Here it iterates on current’s node adjacent nodes to get the cost to reach to current’s left side. after the check it will make current’s left side cost equal to the cost so far by the current node + the cost to reach that side from the current node .

cost\_so\_far can be thought as a cumulative cost from the start node to the current node.



It will then check to not put the curr.left on the open list if there exists already a version of that node which has a less cost so far or equal to it’s cost so far on the list . It will also check that if our curr.left has a cost\_so\_far less than it’s same version on the open list because if that was the case the curr.left will be inserted in the open list and the previous version will be removed from the open list.



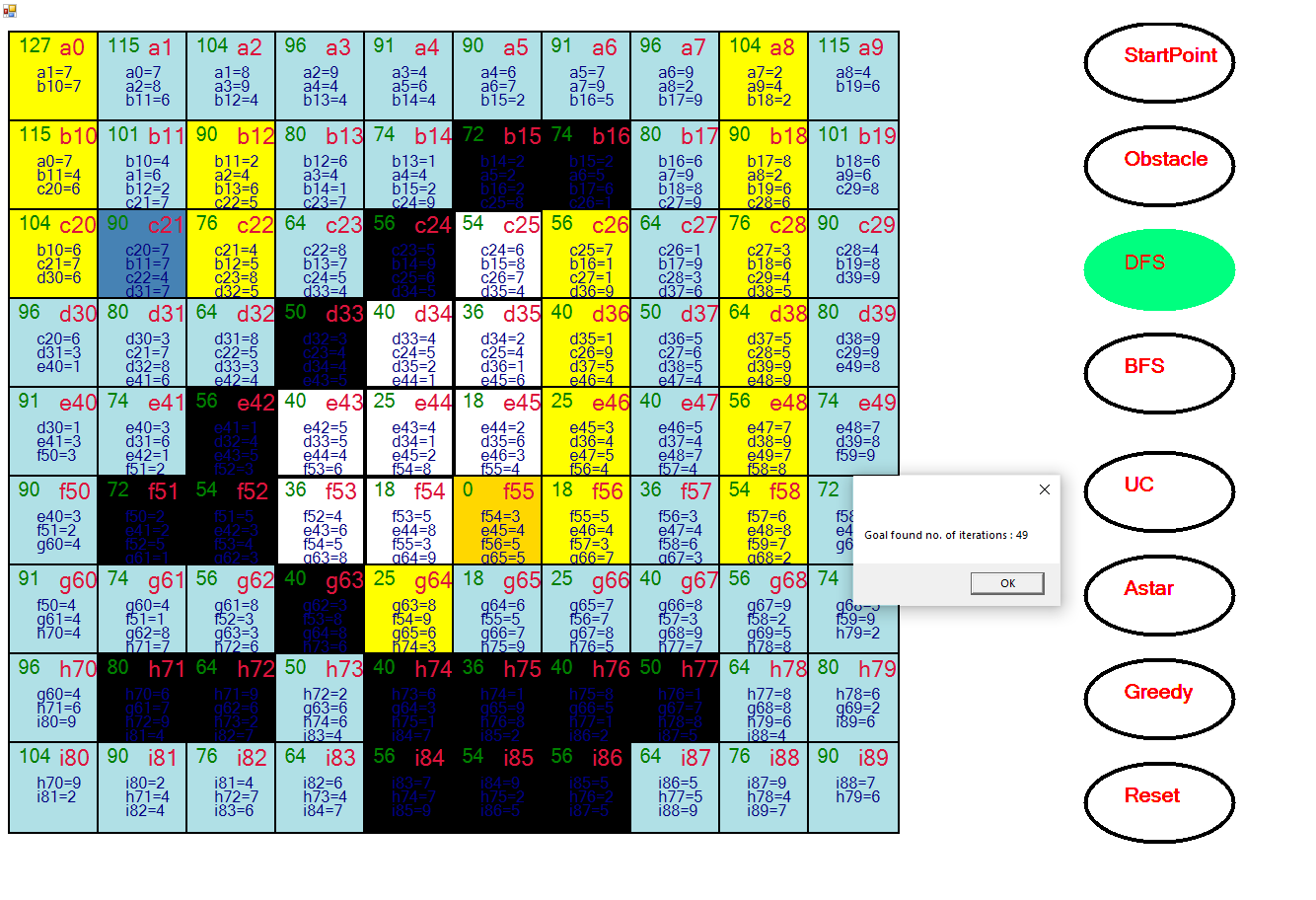
A\* Algorithm Code Overview :

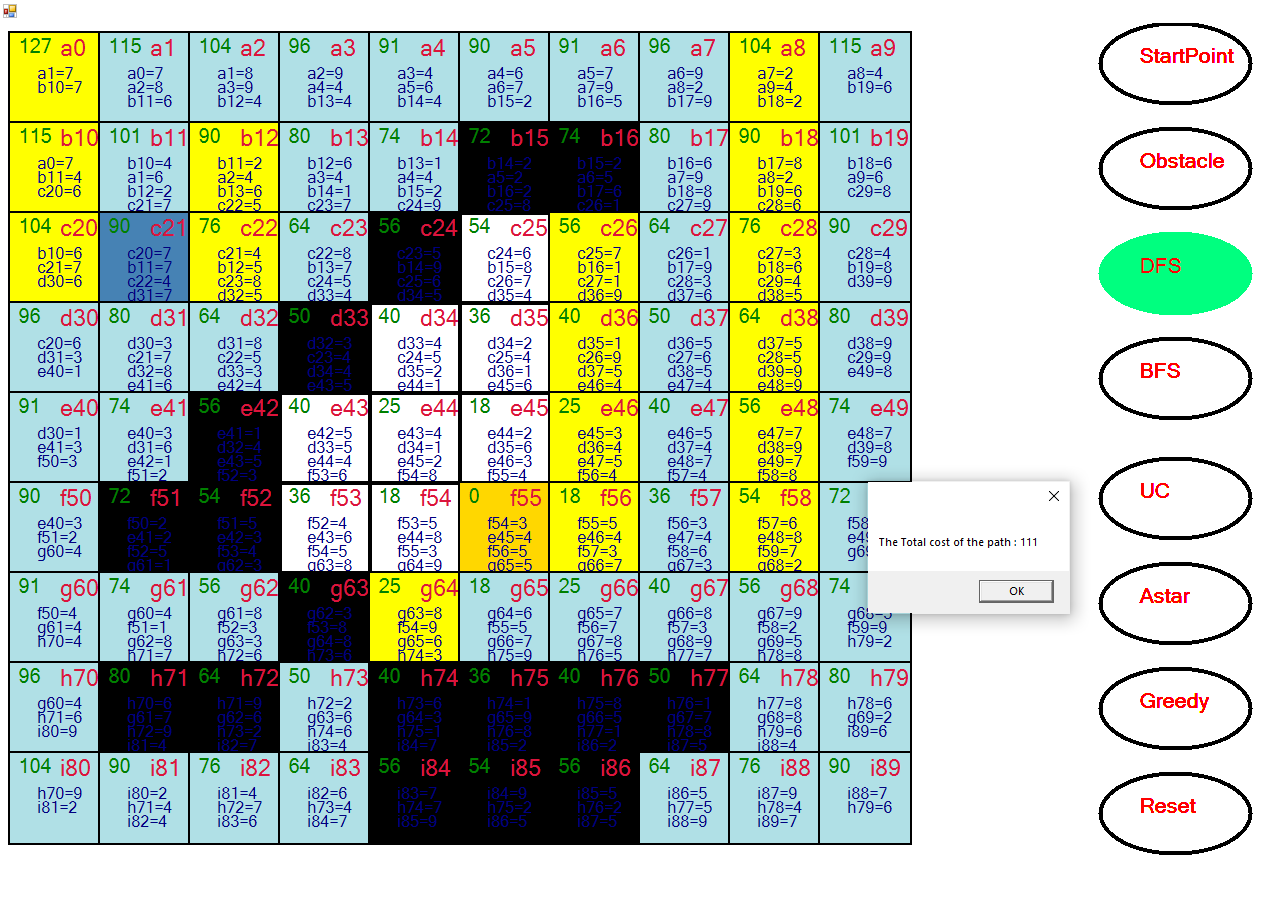
It’s code is pretty much like the Uniform cost the only addition it had was to add the heuristic cost on the total cost acquired from the start node to current node.

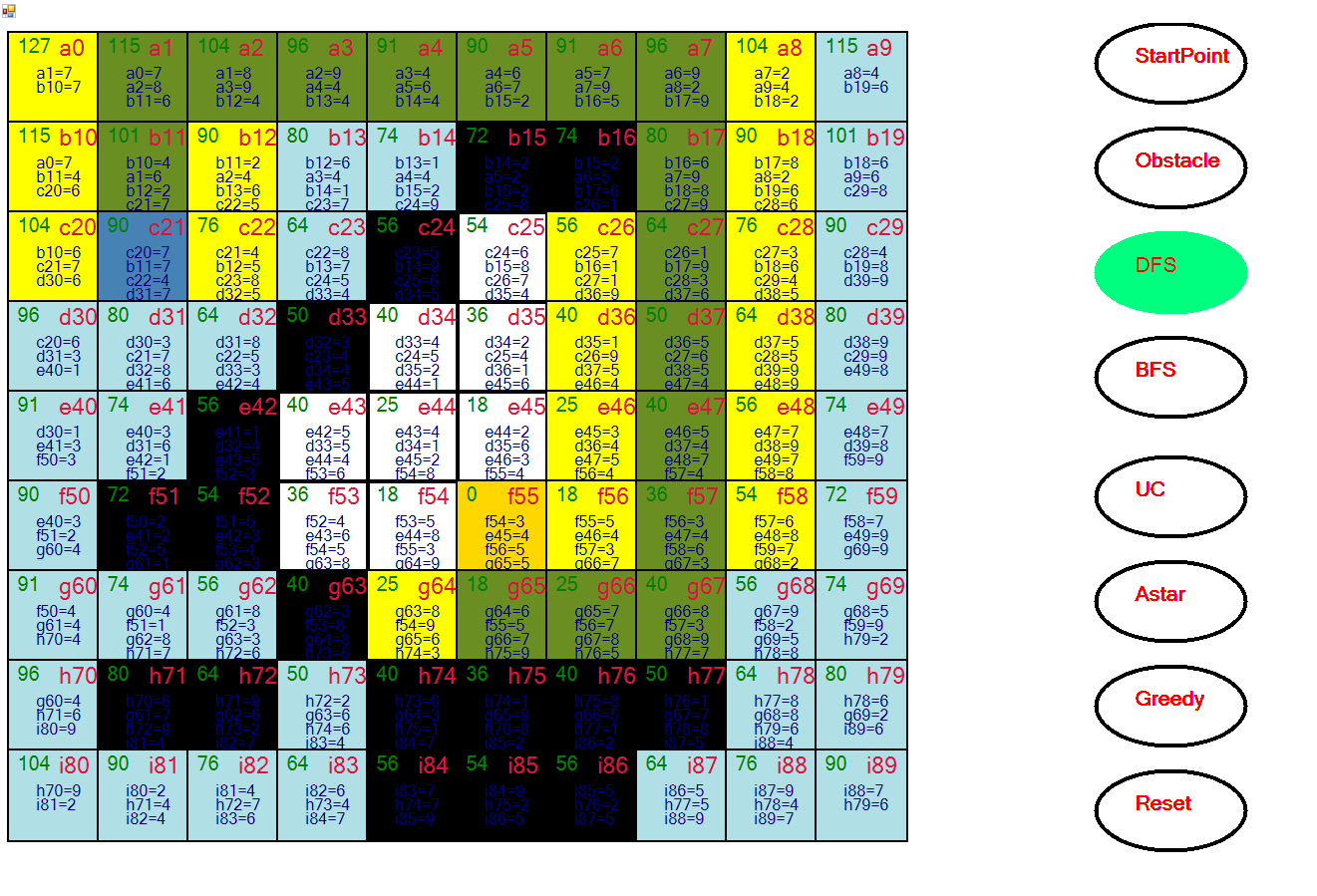


DFS implementation overview

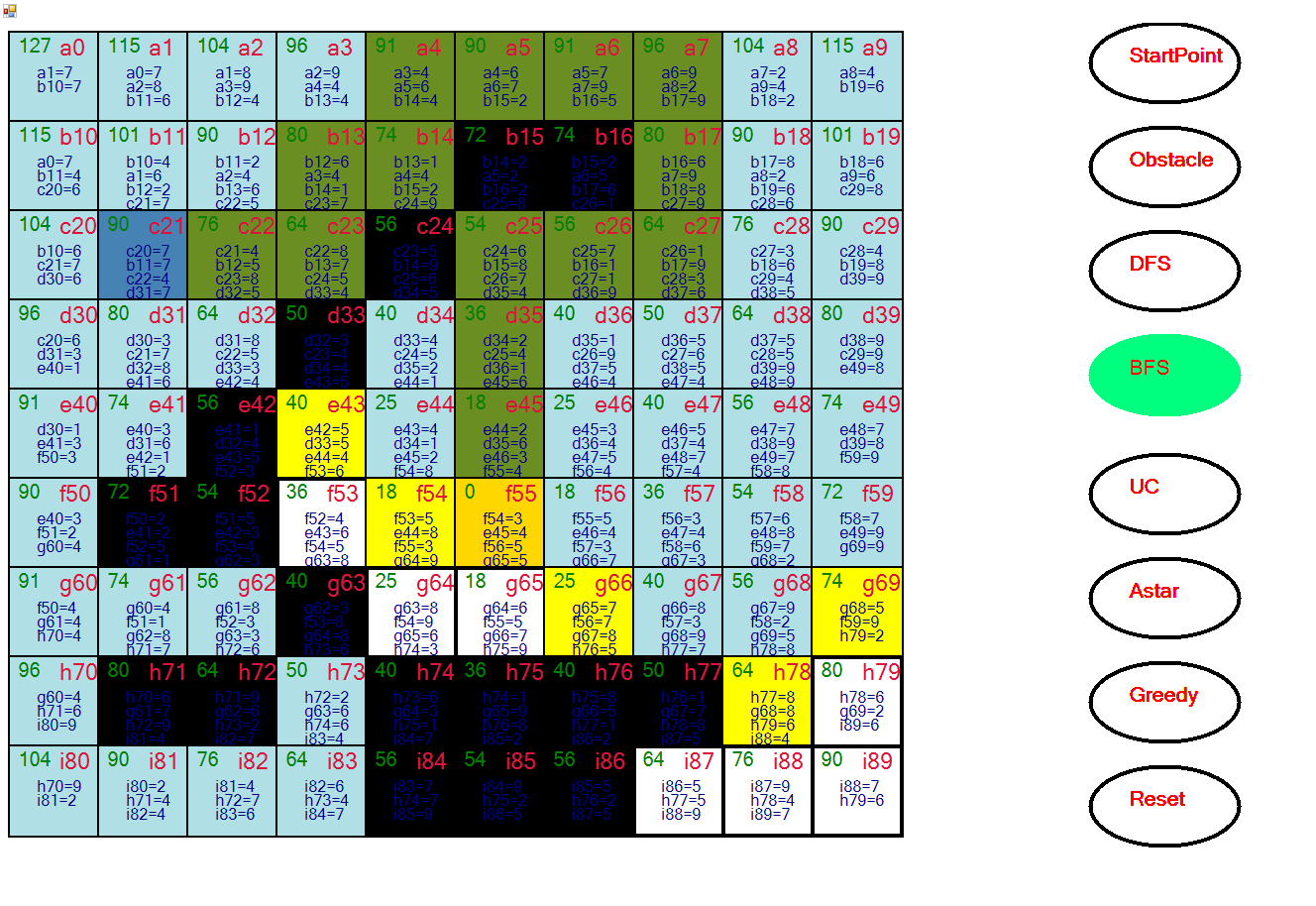
Get no. of iterations



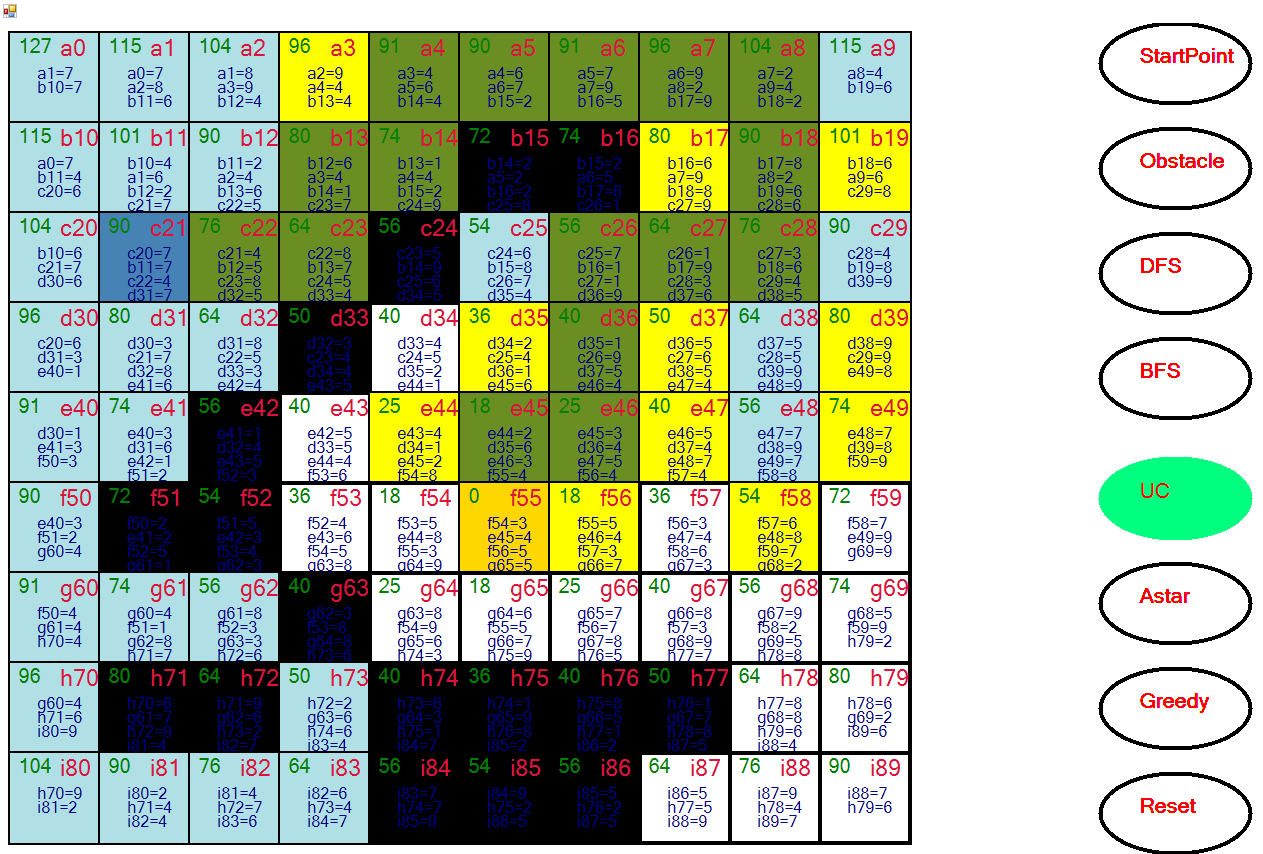




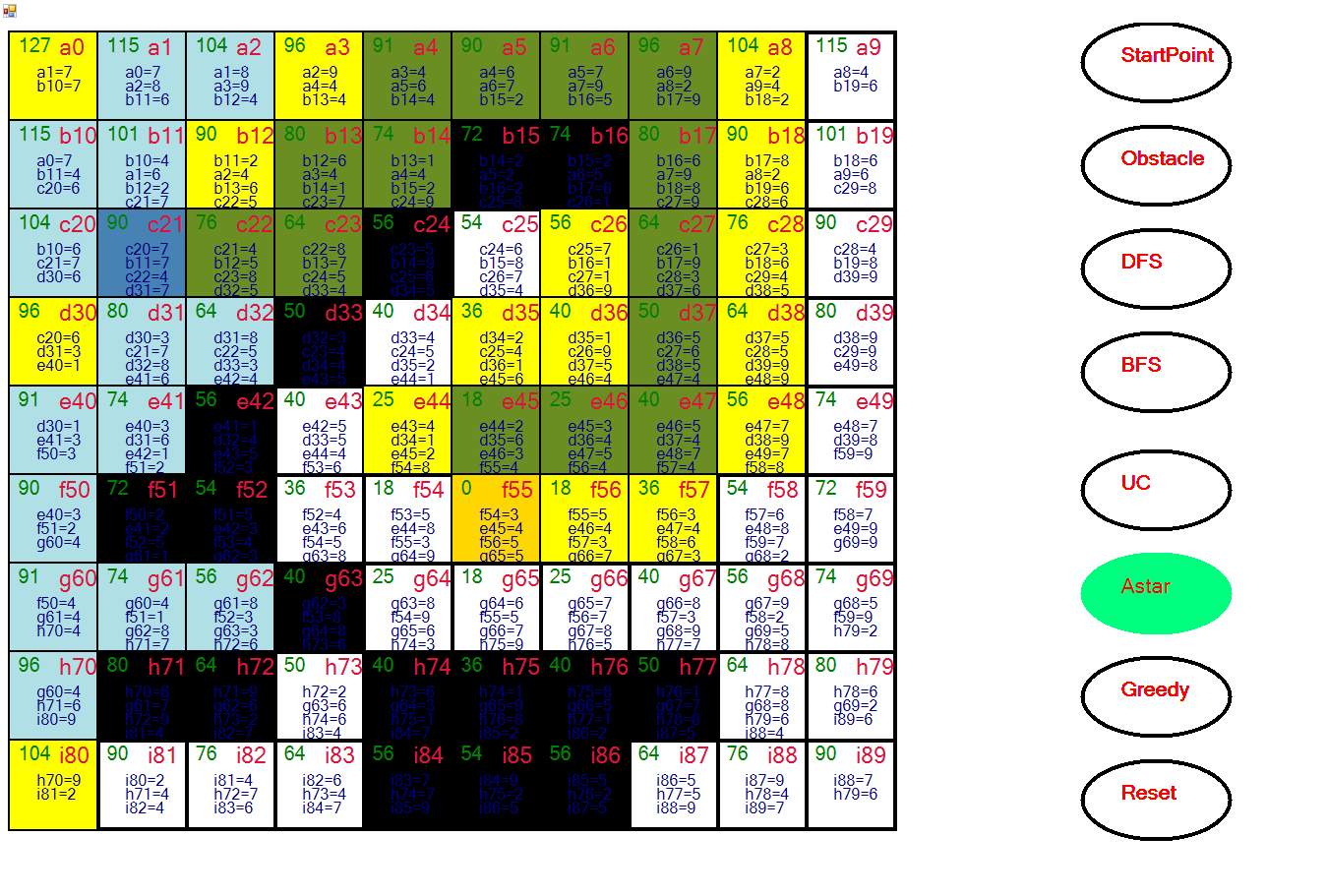
BFS path



Uniform cost path :



A\* algorithm path



Greedy path

