DAT405 Assignment 8 – Group 53

Venkata Sai Dinesh Uddagiri - (12 hrs)
Madumitha Venkatesan - (12 hrs)

January 4, 2023

Problem 1

The branching factor 'd' of a directed graph is the maximum number of children (outer degree) of a node in the graph. Suppose that the shortest path between the initial state and a goal is of length 'r'.

a) What is the maximum number of Breadth First Search (BFS) iterations required to reach the solution in terms of 'd' and 'r'?

In the worst case senario, we are searching for element present in last level right end, then the maximum number of BFS iterations required is total number of node. In BFS, the maximum number of iterations required to reach the solution in terms of 'd' and 'r' is actually determined by

$$\frac{1-d^{r+1}}{1-d}$$

Considering degree is homogeneous for all the nodes in the graph.

For example consider the level of nodes r=2 and degree of each node is d=2.

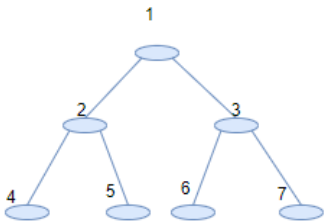

Figure 1: Graph with level 2 and degree 2

Maximum BFS iterations = $\frac{1-2^{2+1}}{1-2} = \frac{1-2^3}{2} = \frac{7}{1} = 7$

By looking at the graph of each node with degree d=3 and level of tree r=2 has 7 nodes. which equals to number of nodes occurred using the formula in terms of 'd' and 'r'.

b) Suppose that storing each node requires one unit of memory and the search algorithm stores each entire path as a list of nodes. Hence, storing a path with k nodes requires k

units of memory. What is the maximum amount of memory required for BFS in terms of 'd' and 'r' ?

The maximum amount of memory required for BFS in terms of 'd' and 'r' is as follows

$$\frac{1 - d^{r+1}}{1 - d}$$

because BFS keeps every node in memory.

Problem 2

Take the following graph where 0 and 2 are respectively the initial and the goal states. The other nodes are to be labelled by 1,3 and 4.

a) Suppose that we use the Depth First Search (DFS) method and in the case of a tie, we chose the smaller label. Find all labelling of these three nodes, where DFS will never reach to the goal! Discuss how DFS should be modified to avoid this situation?

We find the two cases of labeling for three nodes where DFS never reach to the goal, if we label top node as 1 and other 2 nodes as 3 and 4 or vice versa. In both the conditions algorithm will stuck in infinite branch. An infinite branch is a branch that never ends and does not take you to the target node. In DFS, you could keep moving through the same branch indefinitely and "miss" the good branch that leads to the target node.

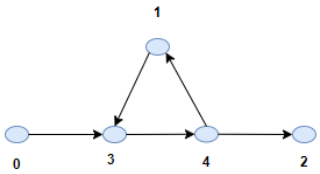

Figure 2: Case1

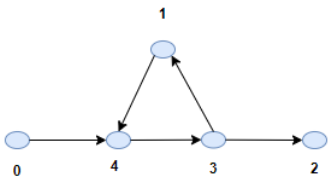

Figure 3: Case 2

To overcome this we can modify usage of DFS as BFS. However, if we do so, DFS loses its primary benefit, which is space efficiency. It is the primary benefit over BFS. Since you must store every visited node in memory if you maintain track of visited nodes, you lose this advantage. Remember that visited node sizes grow significantly with time and may not fit in memory, for very large or infinite graphs. One further thing to note is that the suggestion to choose the smallest label in a tie situation was made without checking if the other label was our target or not. If that check were included, we would avoid going down an endless branch.

Problem 3

a) Suppose a teacher requests a customised textbook that covers the topics

$$introduction\_to\_AI, regression, classification$$

and that the algorithm always selects the leftmost topic when generating child nodes of the current node. Draw (by hand) the search space as a tree expanded for a lowest cost-first search, until the first solution is found. This should show all nodes expanded. Indicate which node is a goal node, and which nodes are at the frontier when the goal is found.

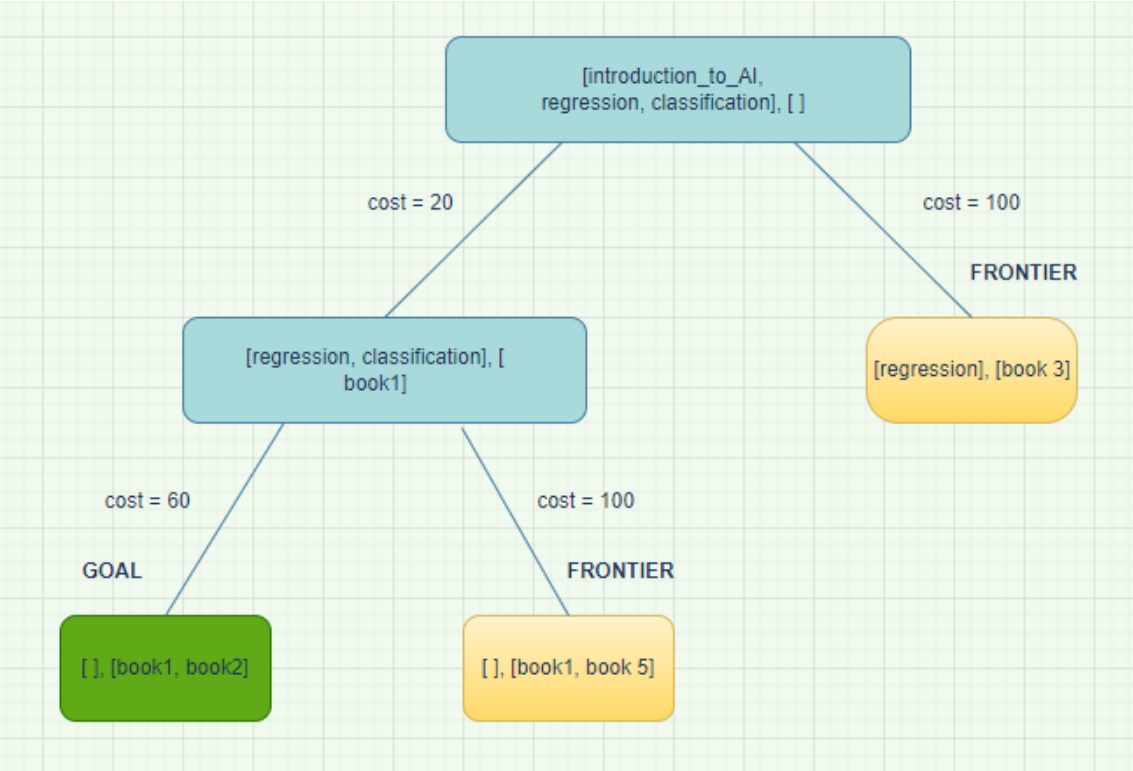

Figure 4: Search space as tree

As suggested in the question left most topic "introduction to AI" is considered as the first child node. The first child node is present in book 1 and 3. Book 1 is the one costs 20 pages and book 3 costs 100 pages. So Book1 is prefered and Book 3 is therefore a "frontier" node. The next two topics are "regression" and "classification", we need to consider regression as child topic at this point but both the topics together are present in Book 5 and Book2. Book 2 costs 60 pages and Book 5 costs 100 pages, So we choose Book 2 as our goal node and Book 5 as a "frontier" node.

b) Give a non-trivial heuristic function h that is admissible. [h(n)=0 for all n is the trivial heuristic function.]

A non-trivial, allowable heuristic function for the process of creating a customized textbook is to estimate the least number of pages that need to be added to the present list of books in order to cover all of the remaining topics. The following steps can be taken to accomplish this:

Find the book in the library's catalog that has the fewest pages that covers each of the topics listed in the current node.

Add the page counts of the books you've chosen, then give the total as the heuristic value.

This heuristic function assesses the cost of achieving the objective with the fewest number of pages needed, which makes it non-trivial because it takes into consideration of addressing all the remaining topics, which is the task's specific requirements. This function is admissible because as it only chooses the books with the fewest pages that cover each topic, it never overestimates the actual cost of achieving the goal.

Problem 4

a) Write the paths stored and selected in the first five iterations of the A* algorithm, assuming that in the case of tie the algorithm prefers the path stored first.

Iteration 1

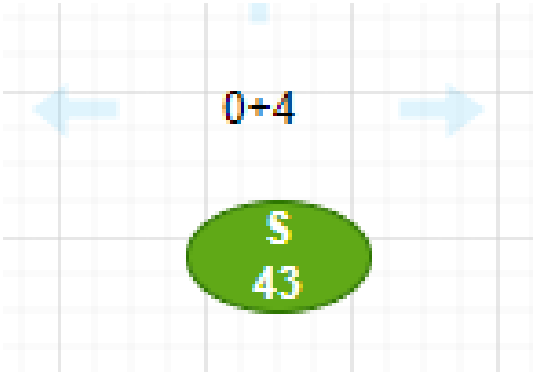

Figure 5: Iteration 1

The only option in iteration 1 is 'S' which is 43 because it is the stating point. Thus the path become S

Iteration 2

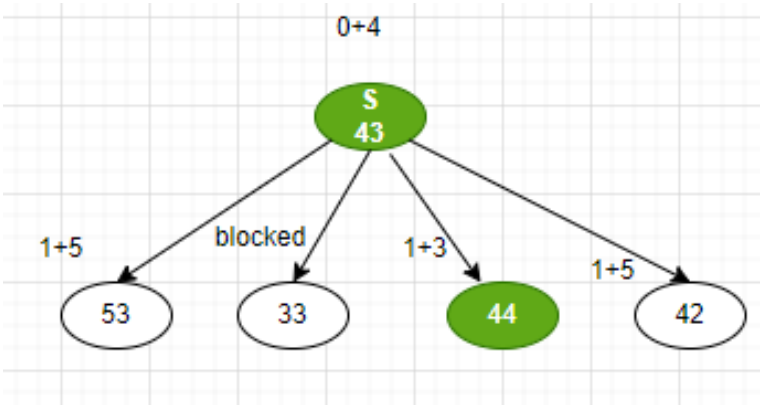

Figure 6: Iteration 2

Each point in the maze has 4 options but some of them are blocked. From point S we can move to 53, 33, 44 or 42. But 33 is already blocked, so not considering it. Out of other three options 44 has minimum heuristic value. So path becomes S, 44

Iteration 3

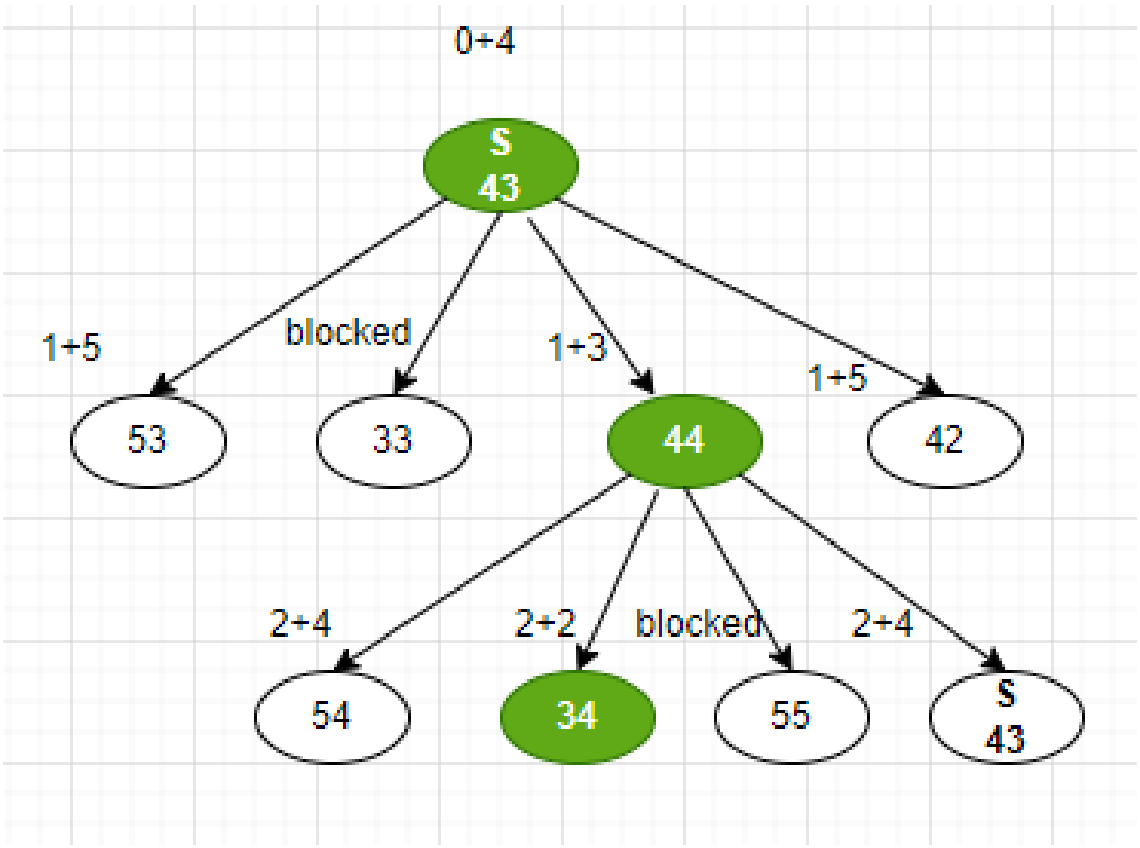

Figure 7: Iteration 3

From point 44 we can move to 54, 34, 55 or 43(S). But 55 is already blocked, so not considering it. Out of other three options 34 has minimum heuristic value. So path becomes S, 44, 34

Iteration 4

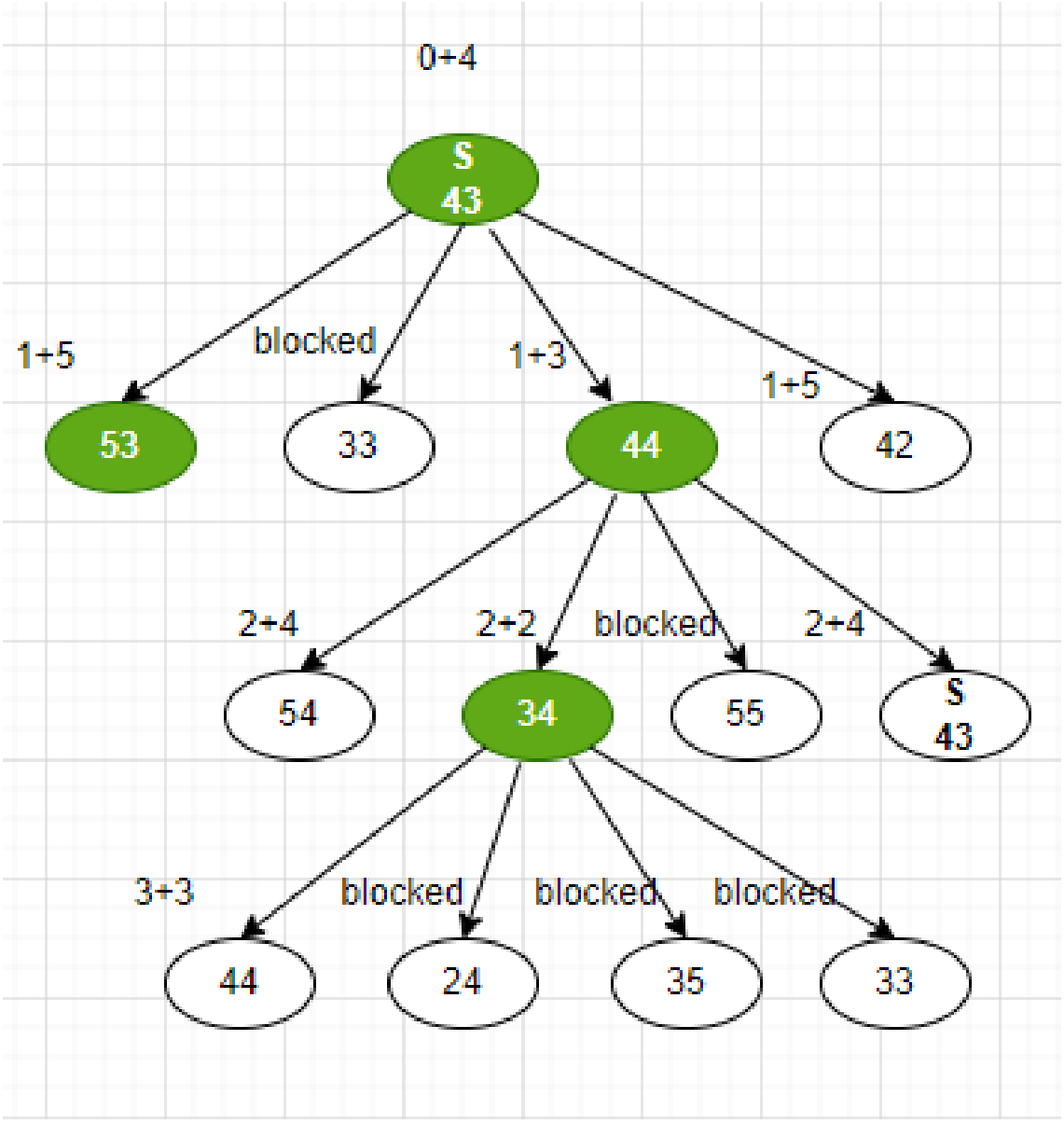

Figure 8: Iteration 4

From point 34 we can move to 44, 24, 35 or 33. But 24, 35 and 33 are already blocked, so not considering them. Only option now we have is 44 but its heuristic value ties up with other values stored which are 54, 53 and 42. Out of which 53 is the one which stored previous to all other points. So considering 53, then path becomes S, 53

Iteration 5

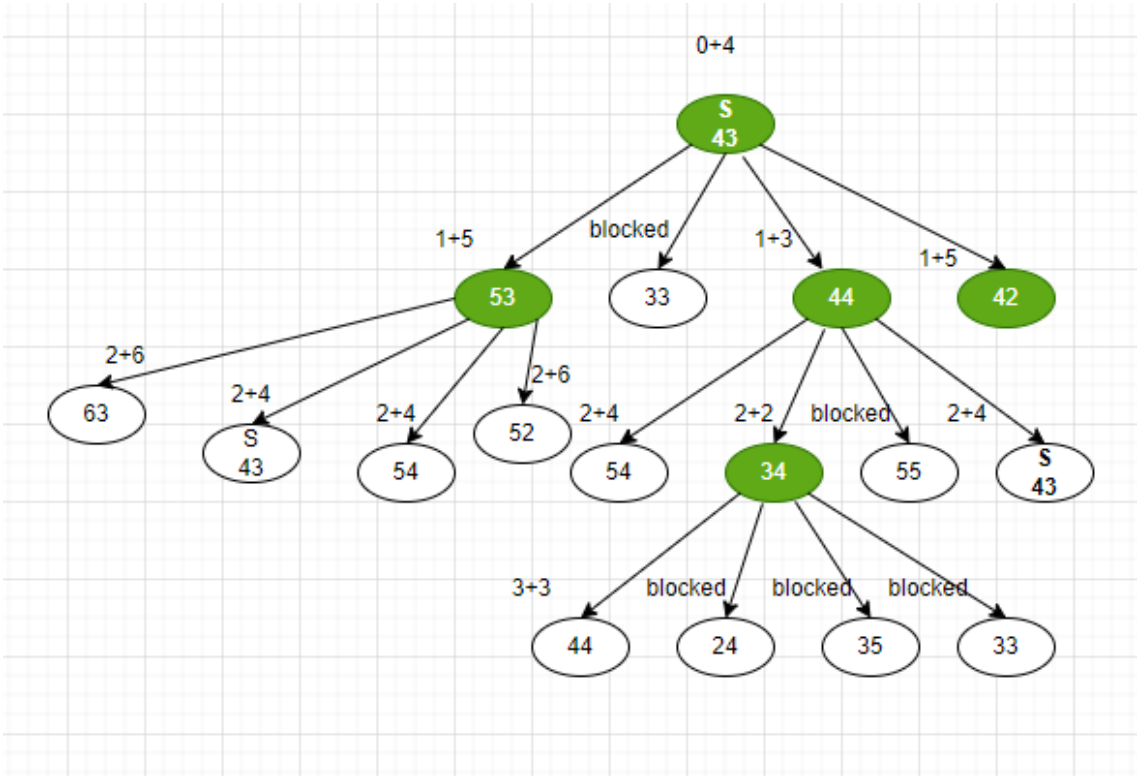

Figure 9: Iteration 5

From point 53 we can move to 63, 54 or 52. Now heuristic value ties up with all the four possible moves. So we choose 42 which is oldest value, then path becomes S, 42

b) Solve this problem using the software in http://qiao.github.io/PathFinding.js/visual/ Use Manhattan distance, no diagonal step and compare A*, BFS and Best-first search. Describe your observations. Explain how each of these methods reaches the solution. Discuss the efficiency of each of the methods for this situation/scenario.

We found number of iterarions required to move from start node to end node using different algorithms is as follows.

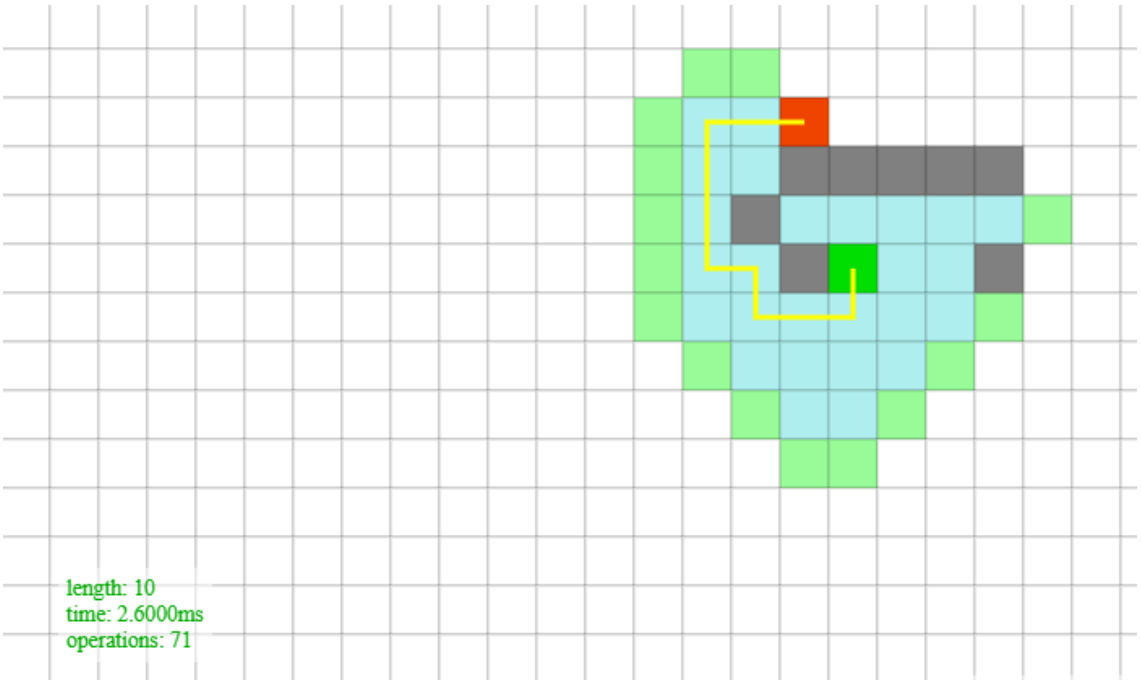

Figure 10: A* search

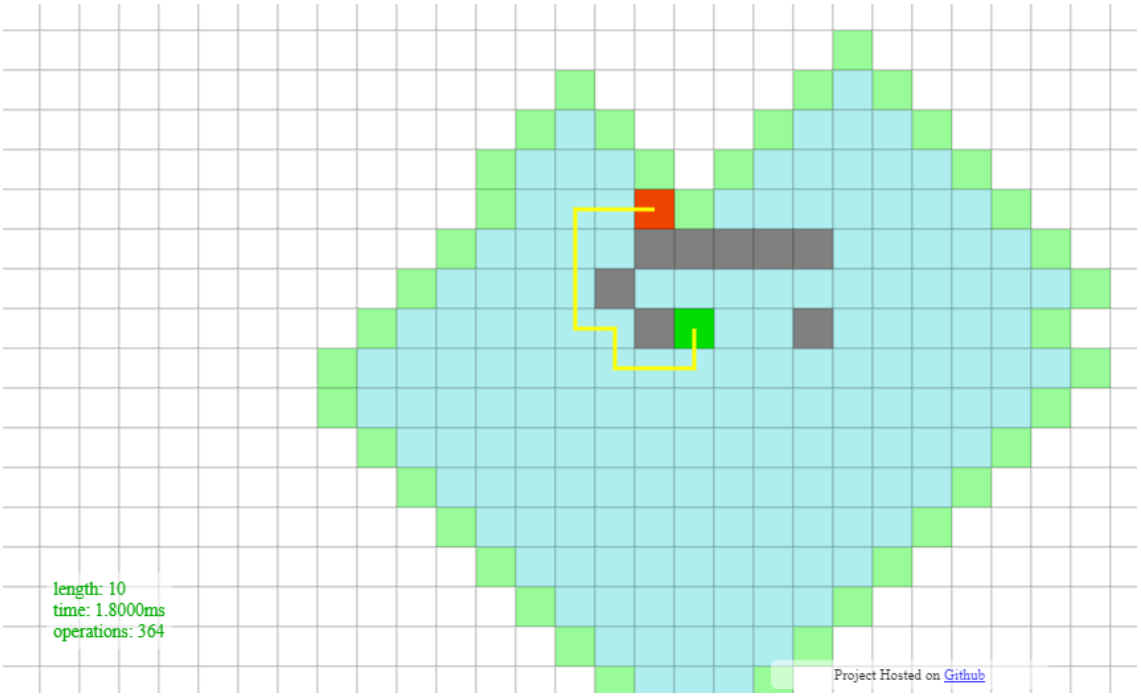

Figure 11: BFS search

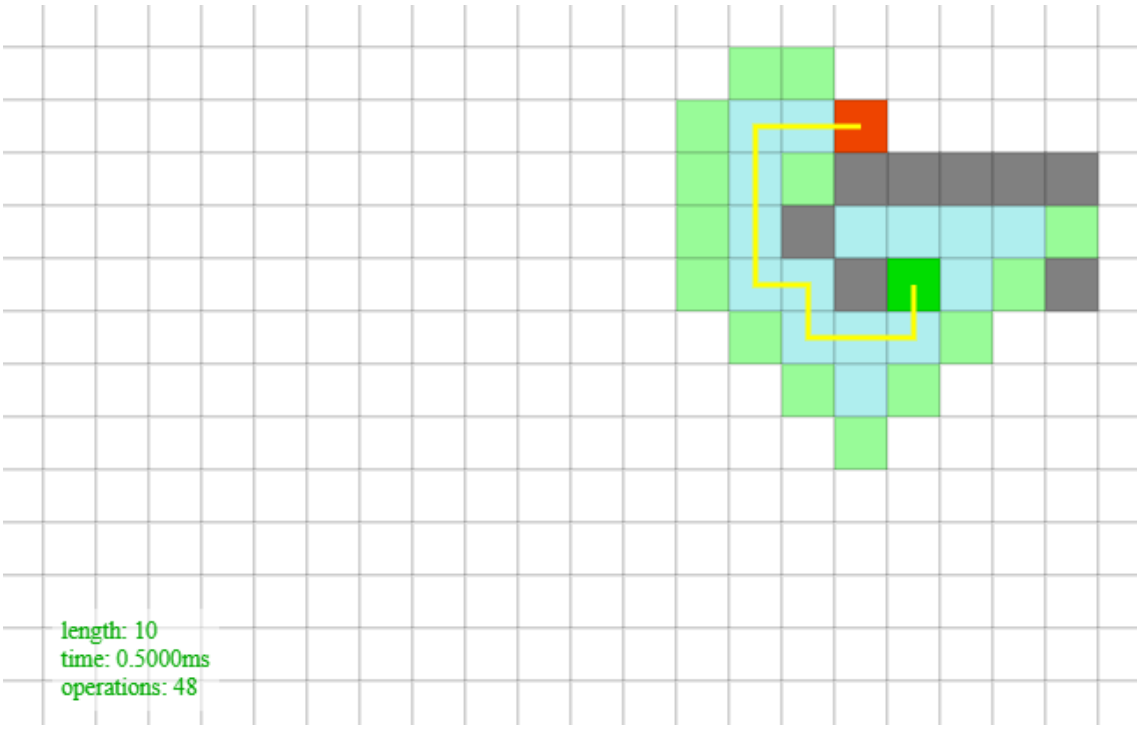

Figure 12: Best-first search

A* = length 10, 71 operations

BFS = length 10, 364 operations

BestFS = length 10, 48 operations

Out of the three algorithms BFS has not performed best, as it does not take into account the distance to the goal when making decisions about which path to explore next.

In general, $A*$ should perform well because it considers both the cost of the path and the estimated distance to the goal. However, in our case, Best First Search performs well, which may be because the estimated distance to the goal varies significantly but the cost of moving from one node to another does not. An algorithm that takes into account both the cost and the distance to the goal will struggle in this scenario because it will spend too much time following paths that are not the most direct route to the goal. However, Best-first search will prioritize routes according to estimated distance to the goal and could be able to locate the shortest path more fastly.

c) Using a board like the board used in question 4a) or in http://qiao.github.io/PathFinding.js/visual/, describe and draw a situation/scenario where Breadth-first search would find a shorter path to the goal compared to Greedy best-first search. Consider that a piece can move on the grid horizontally or vertically, but not diagonally. Explain why Breadth-first search finds a shorter path in this case.

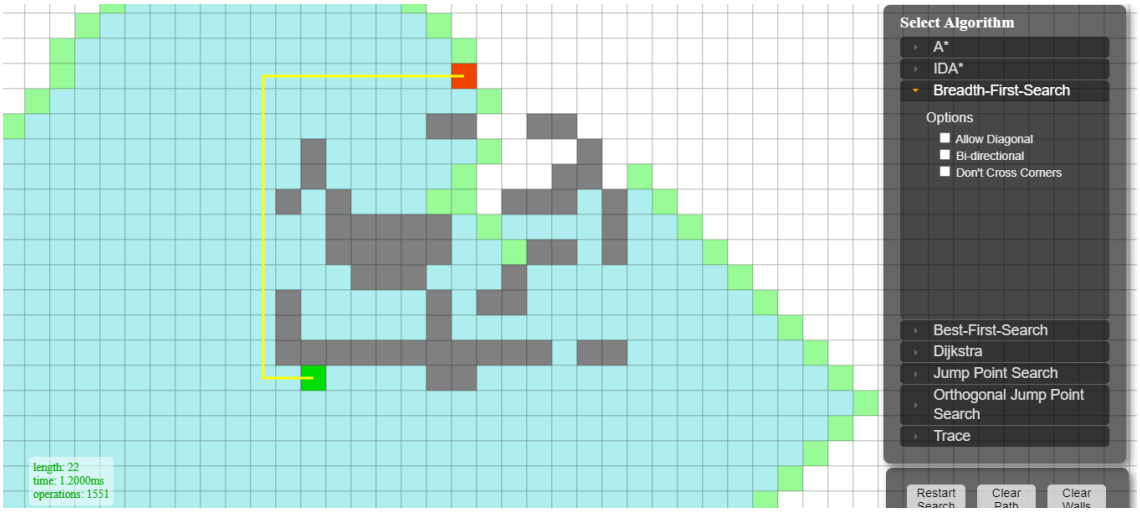

Figure 13: Breadth-first search

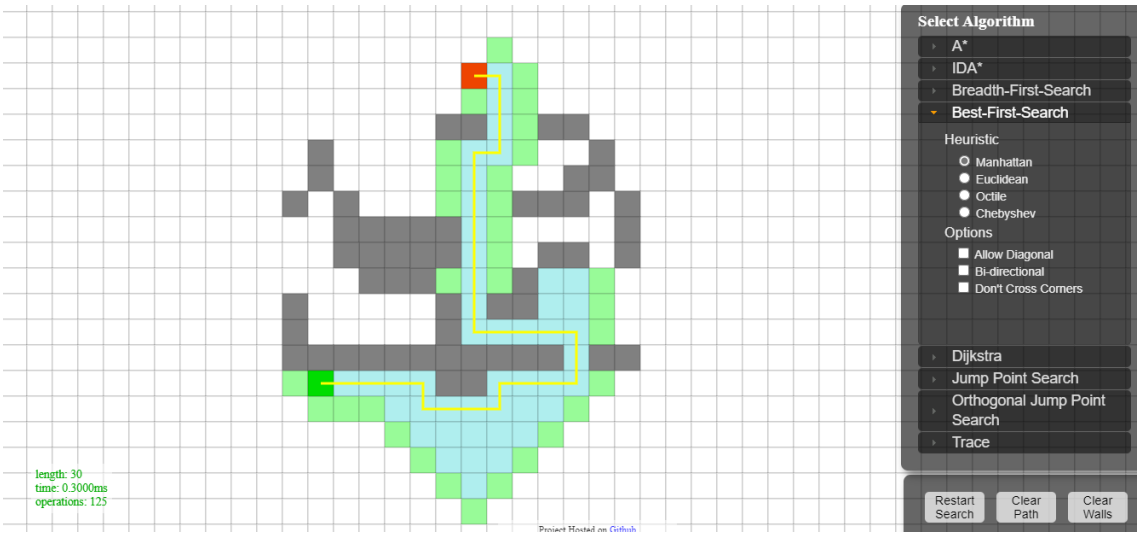

Figure 14: Greedy best-first search

The target was found by the Breadth first search in 1551 iterations with a path length of 22 and by the Greedy best first search in 125 iterations with a path length of 30. When compared to Greedy best first search, Breadth first search (BFS) finds the shortest path since it will first investigate all pathways of a given length before moving on to greater paths. However, Greedy best-first search gives pathways a higher priority based on the estimated distance to the goal.

Problem 5

a) Discuss when and how the generic search problem can be described as a Markov decision process (MDP).

By treating the search space as a state space and the actions that can be taken at each state as transitions between states, the general search problem can be treated as a markov decision process.

For example, to describe a maze search problem as MDP, consider states s as maze's cell and action a could represent four possible directions which are up, left, right and down to move. The probability of going from cell s to cell s' by performing action an is defined as the transition function T(s, a, s'), while the cost of doing so is defined as the reward function R(s, a, s').

b) When the search problem can be written as an MDP, what are the advantages and disadvantages of the value iteration algorithm over the A* algorithm?

When search problem written as MDP, the advantages of value iteration over the A* is value iteration is a simple and guaranteed-convergent algorithm that is best suited for MDPs with a small number of states and actions. The disadvantage of value iteration over the A* is value iteration is an algorithm for solving MDPs guaranteed to discover the best path, although the best path is not always the shortest path. A* is an algorithm for finding the shortest path in a graph and is generally more efficient than value iteration, but it may not find the absolute shortest path. For problem with more number of states and actions A* is more efficient than value iteration, as long as a good heuristic function is available.