Grassfire / Breadth First Search

SECTION 1.2

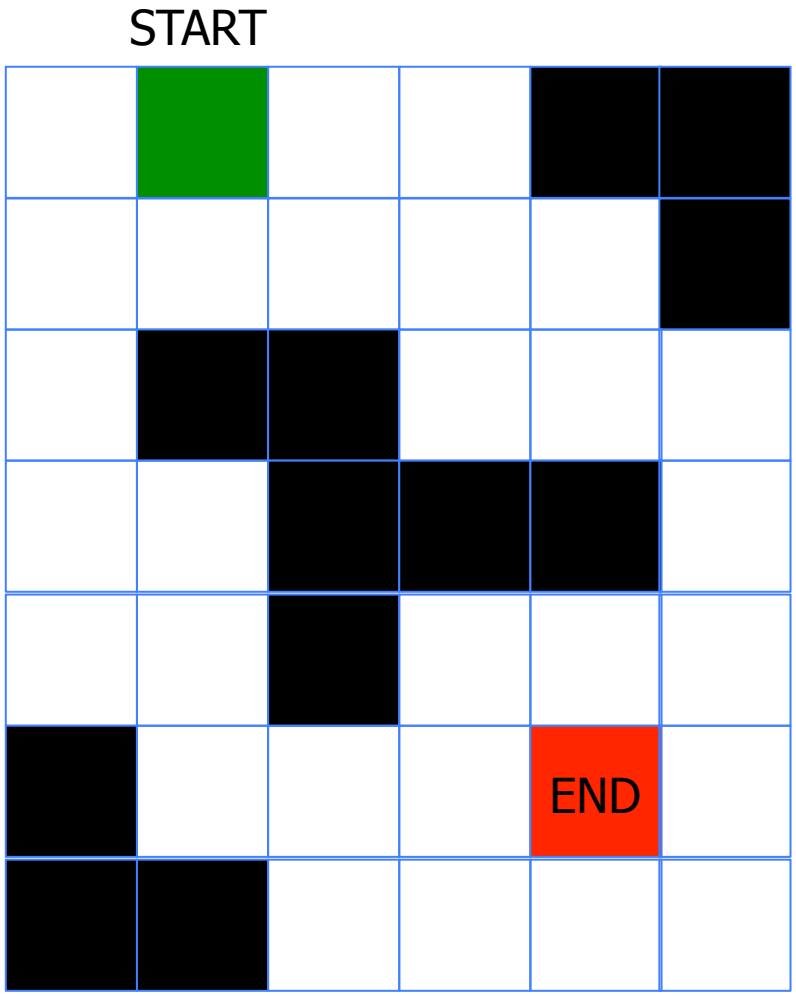
Penn

Section 1.2 - Grassfire

Prof. C.J. Taylor

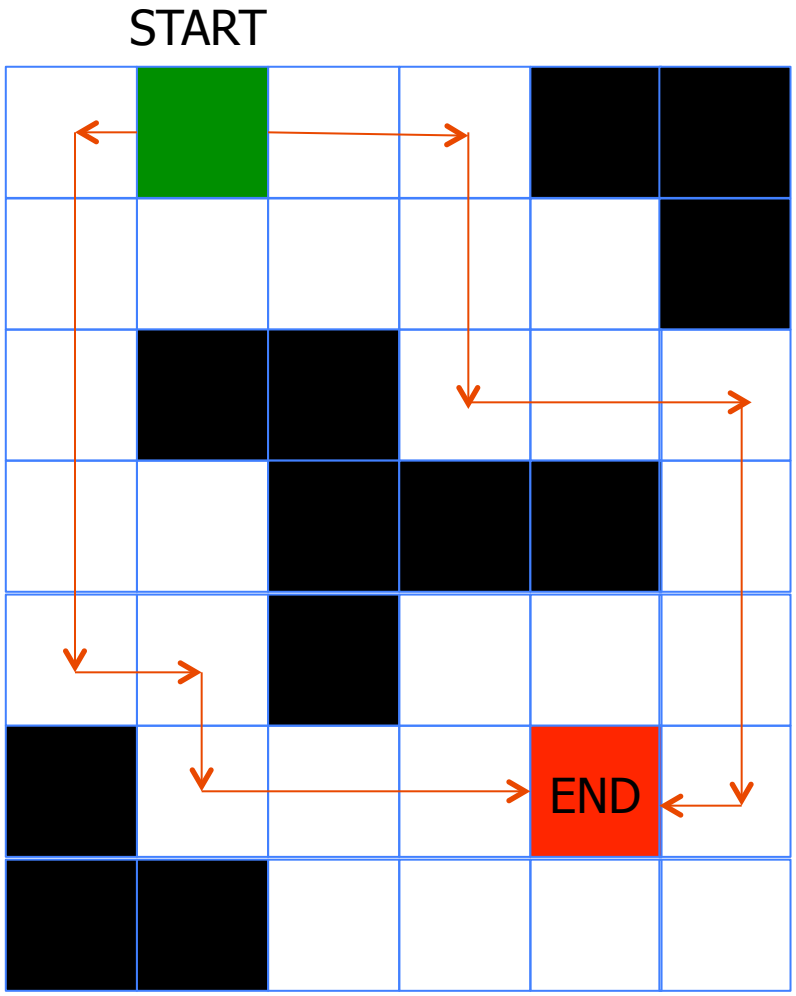
Planning on a grid

- The goal is to construct a path through the grid/ graph from the start to the goal



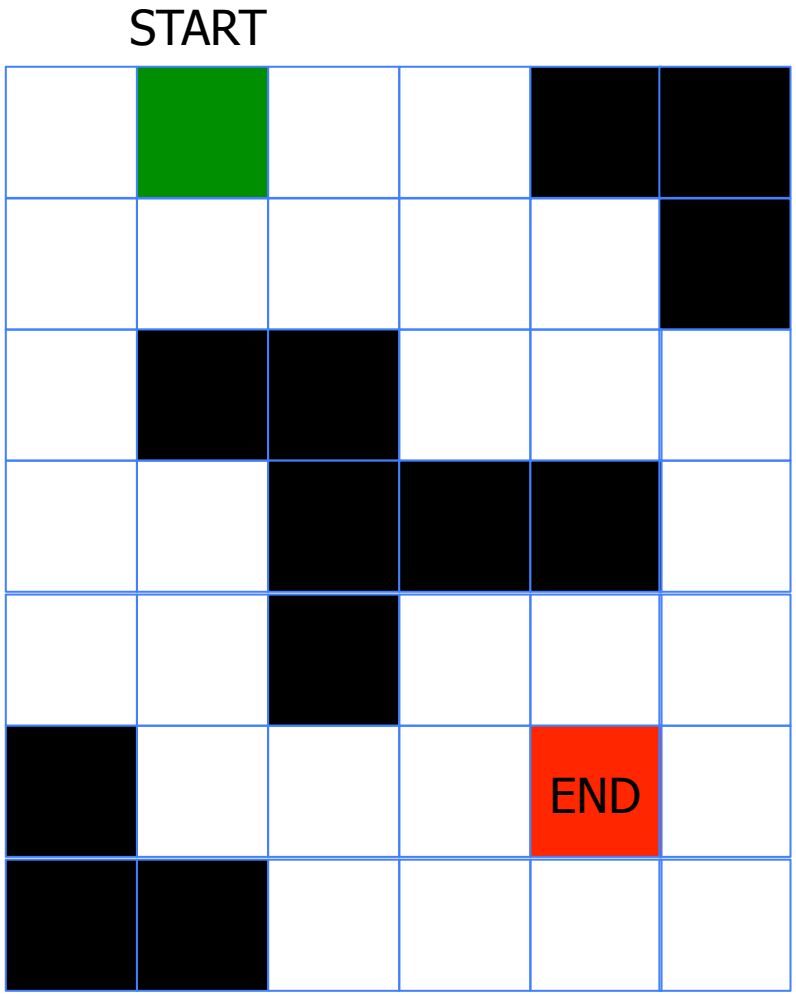
Planning on a grid

- Typically there are many possible paths between two nodes.
- We are usually interested in the shortest paths



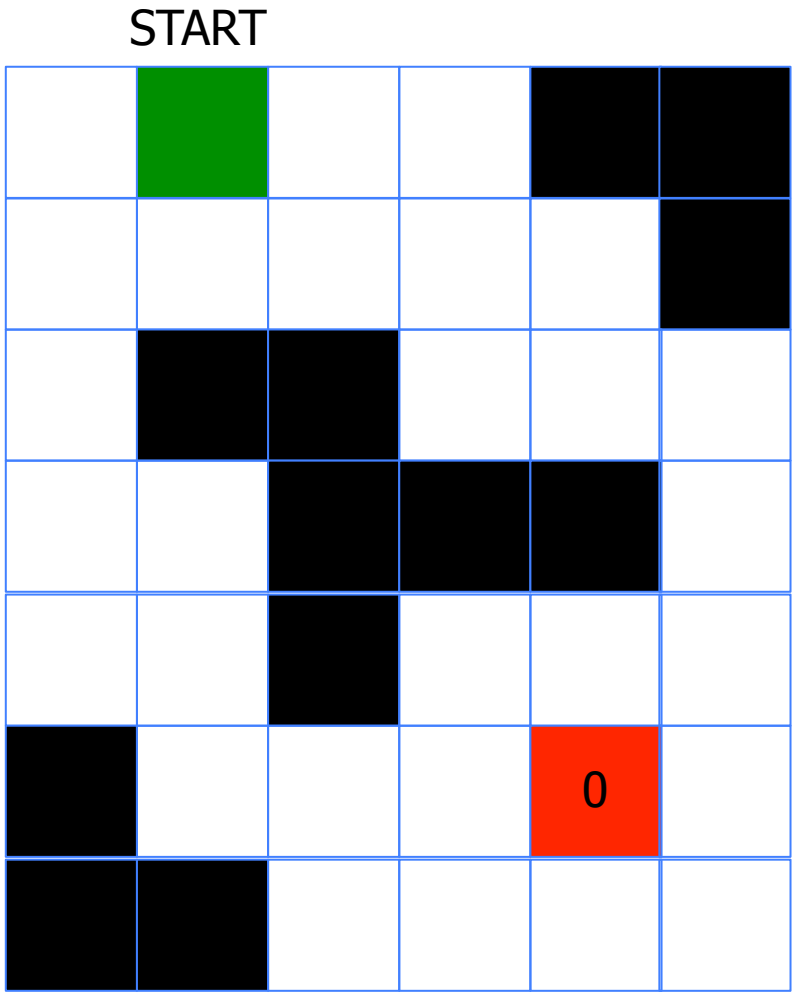
Planning on a grid

- Goal:
  - Construct the shortest path between the start and the goal location.



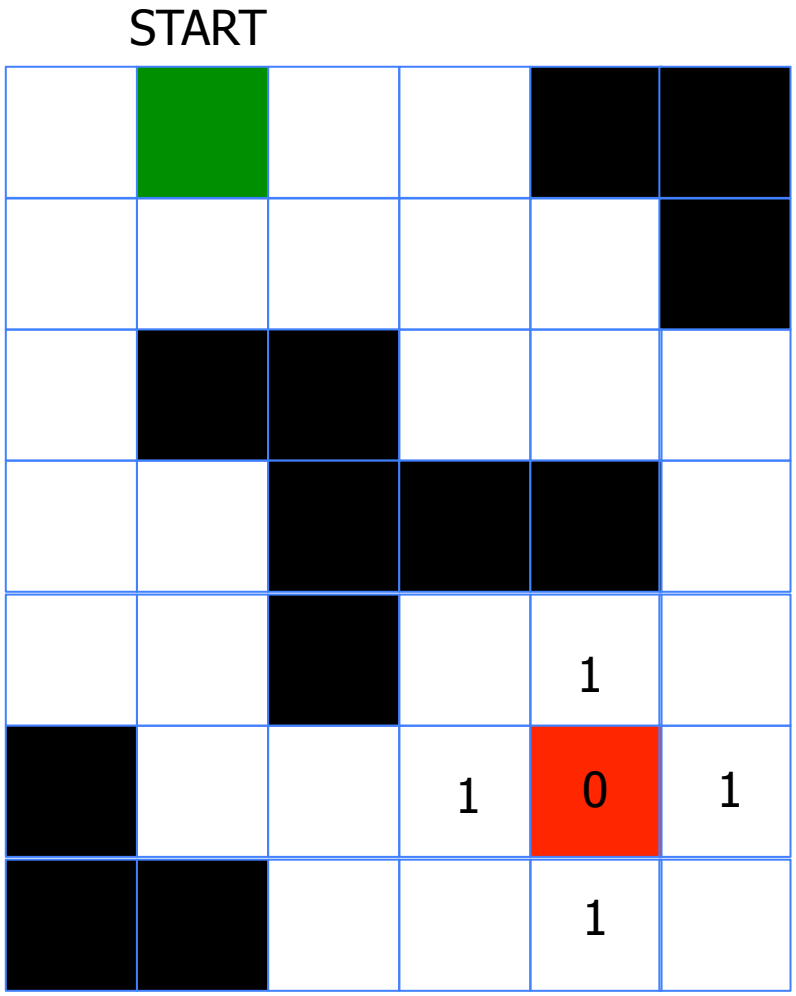
Planning Procedure – Grassfire Algorithm

- Begin by marking the destination node with a distance value of 0

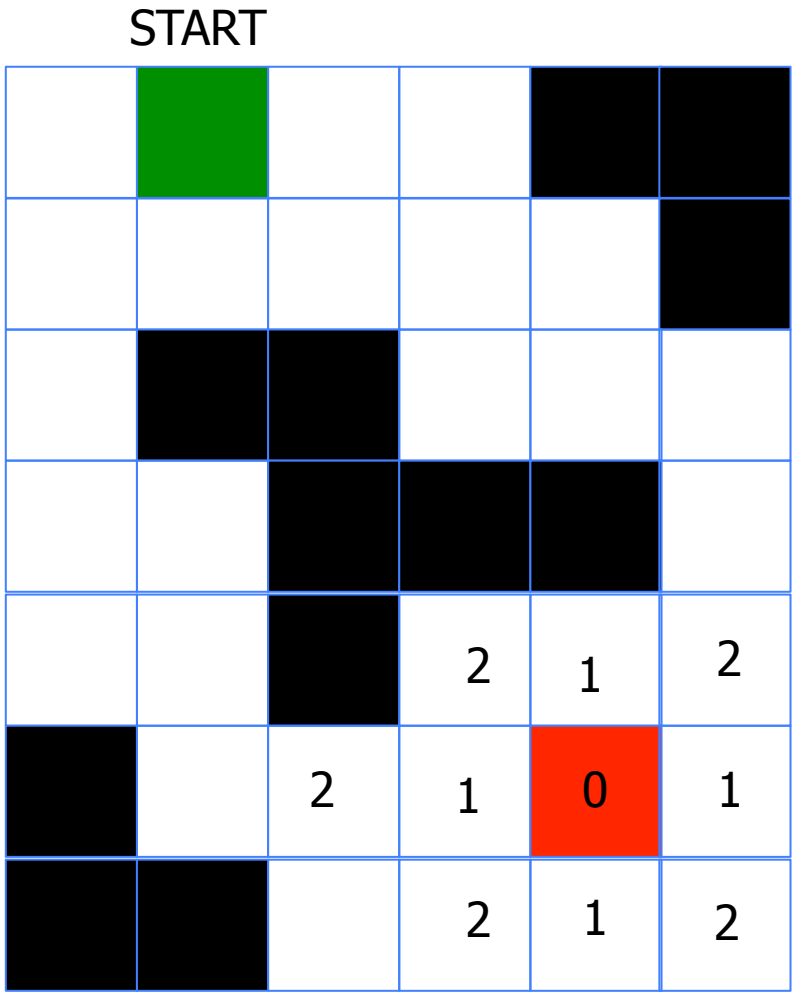


Planning Procedure – Grassfire Algorithm

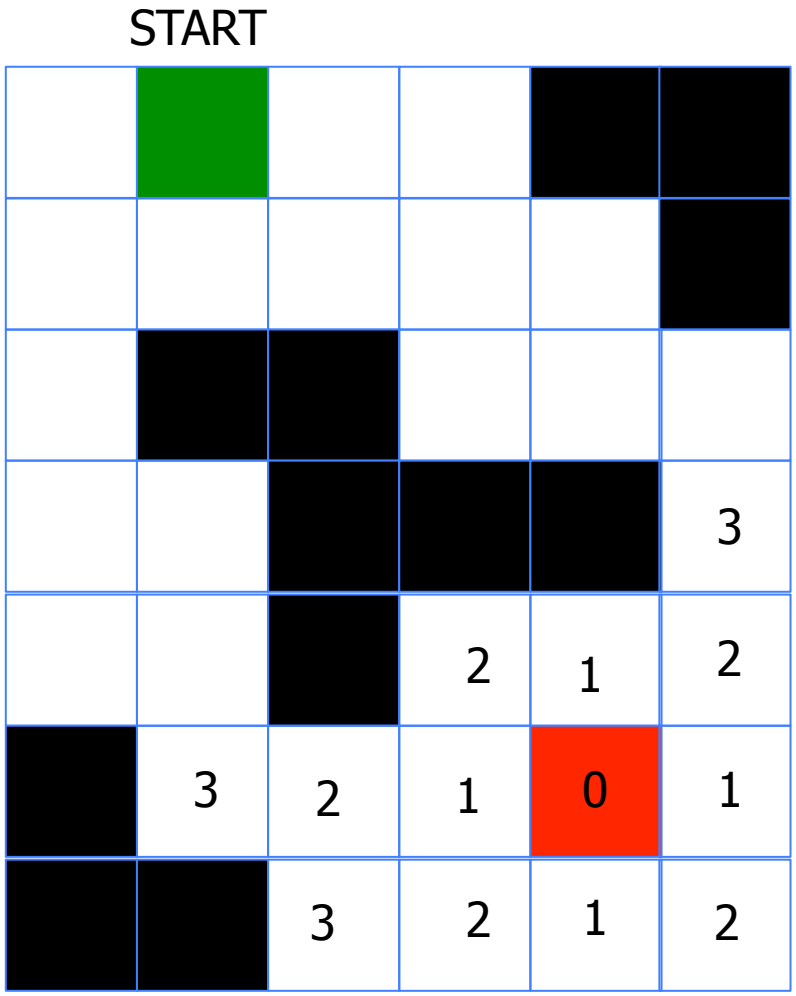
- On every iteration find all the unmarked nodes adjacent to marked nodes and mark them with that distance value + 1.



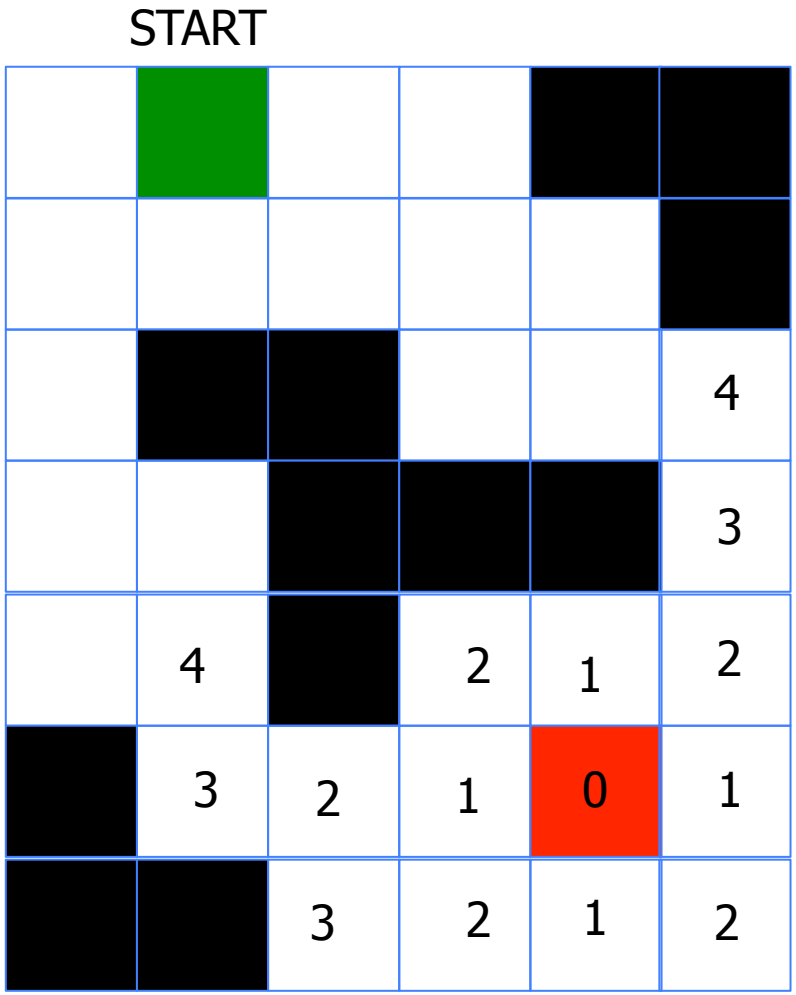
Planning Procedure – Grassfire Algorithm



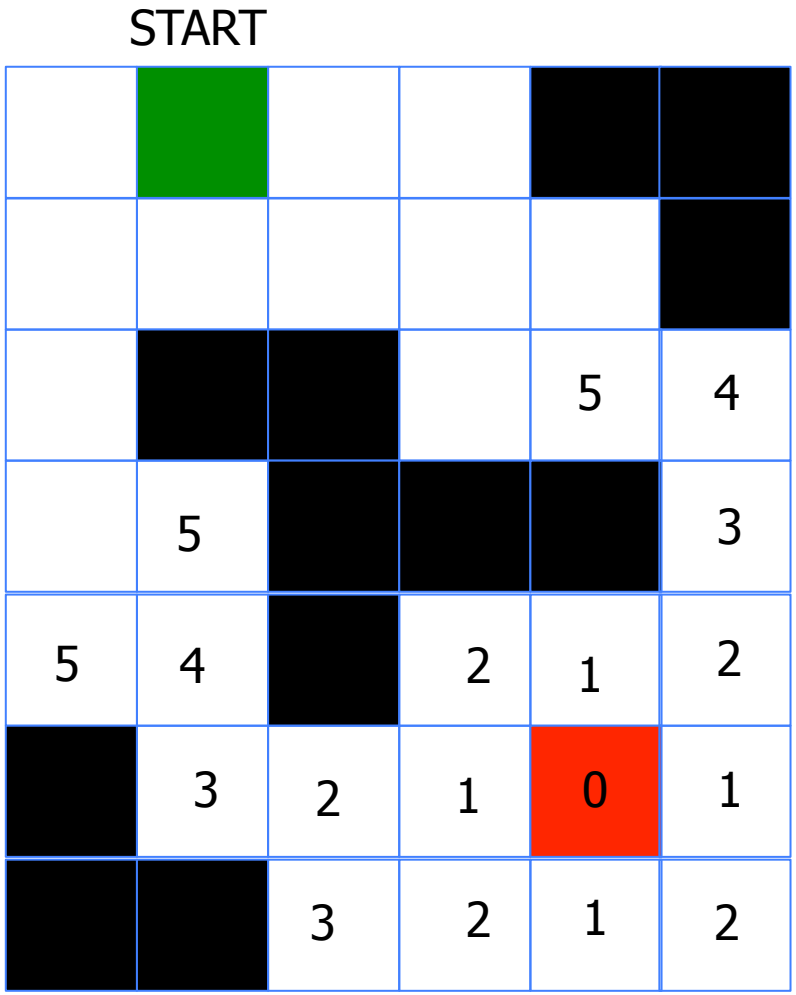
Planning Procedure – Grassfire Algorithm



Planning Procedure – Grassfire Algorithm

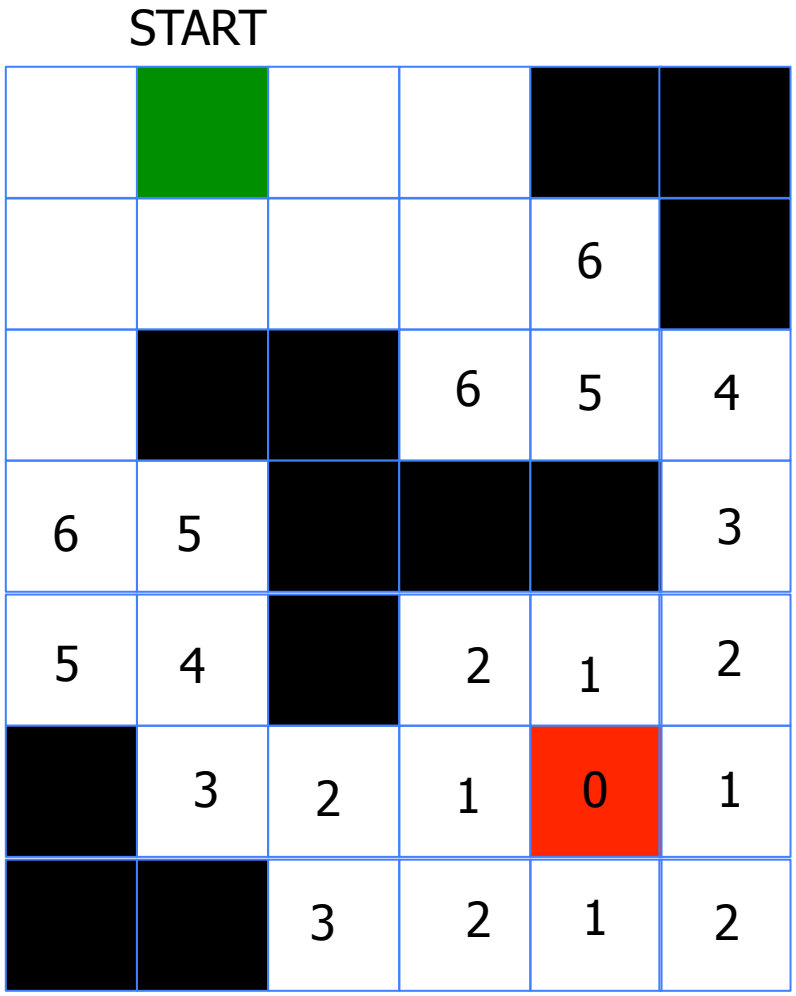


Planning Procedure – Grassfire Algorithm

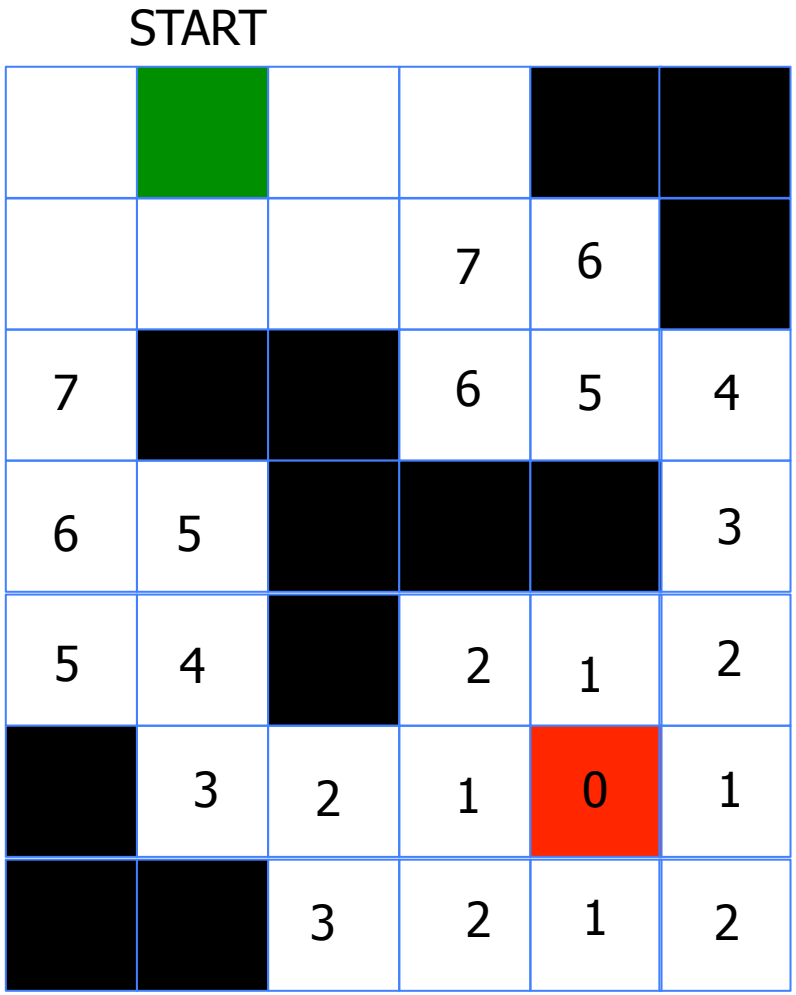


Planning Procedure – Grassfire Algorithm

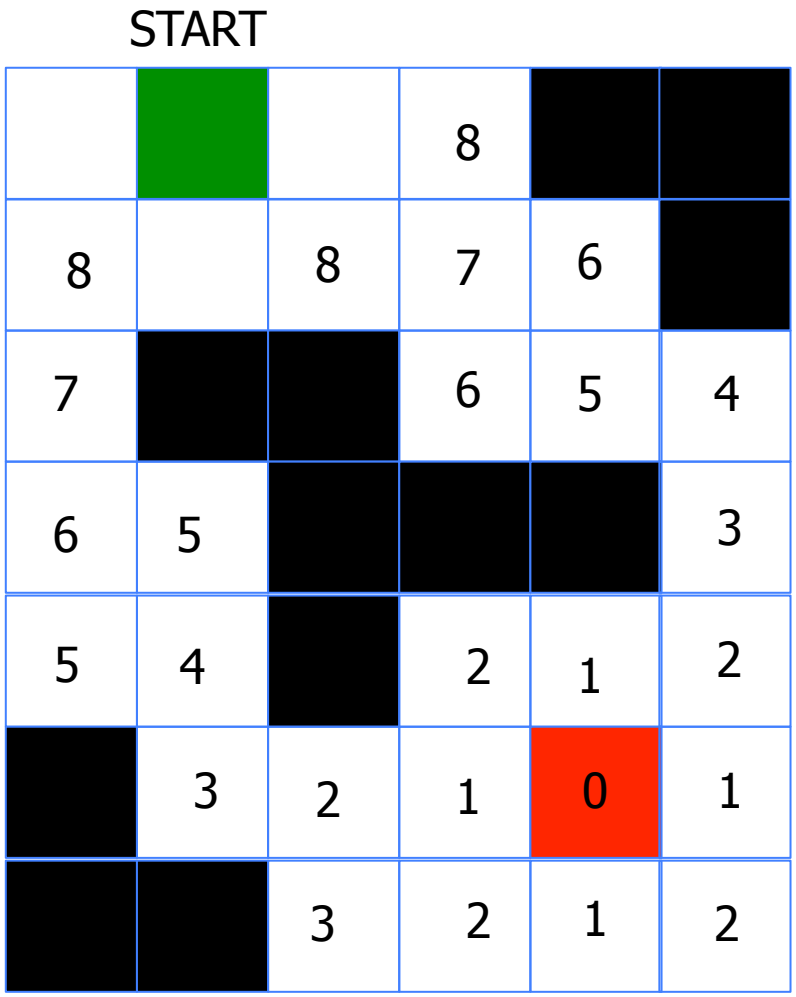
- On every iteration the marking radiates outward from the destination like a fire spreading – hence the name



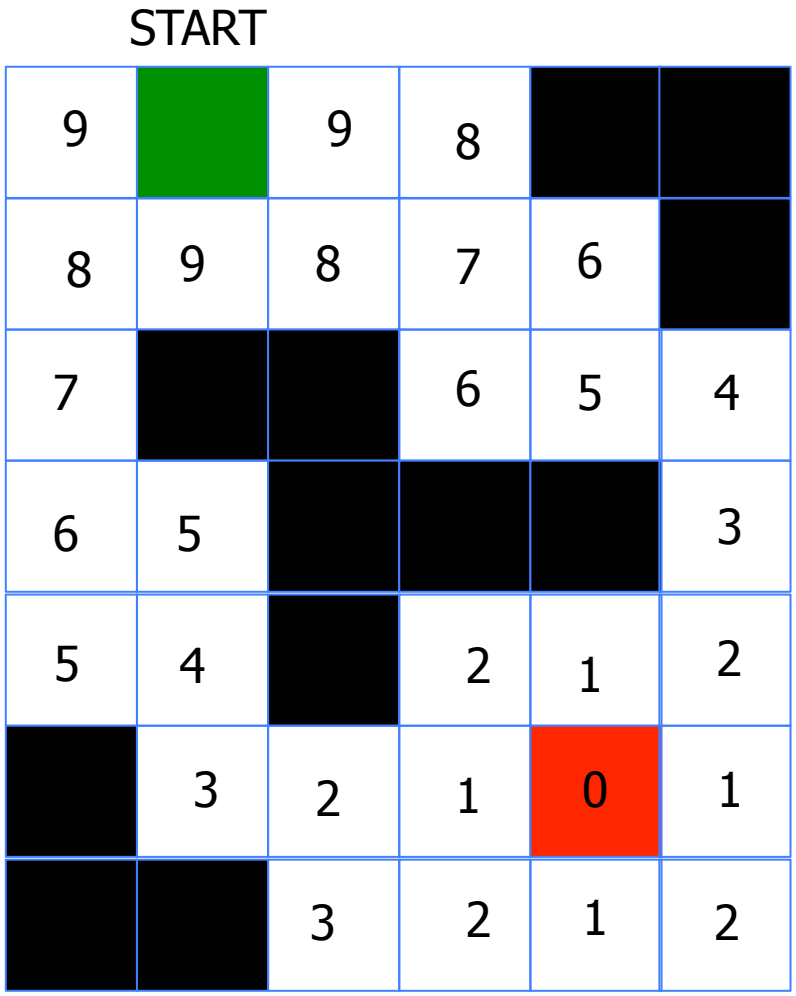
Planning Procedure – Grassfire Algorithm



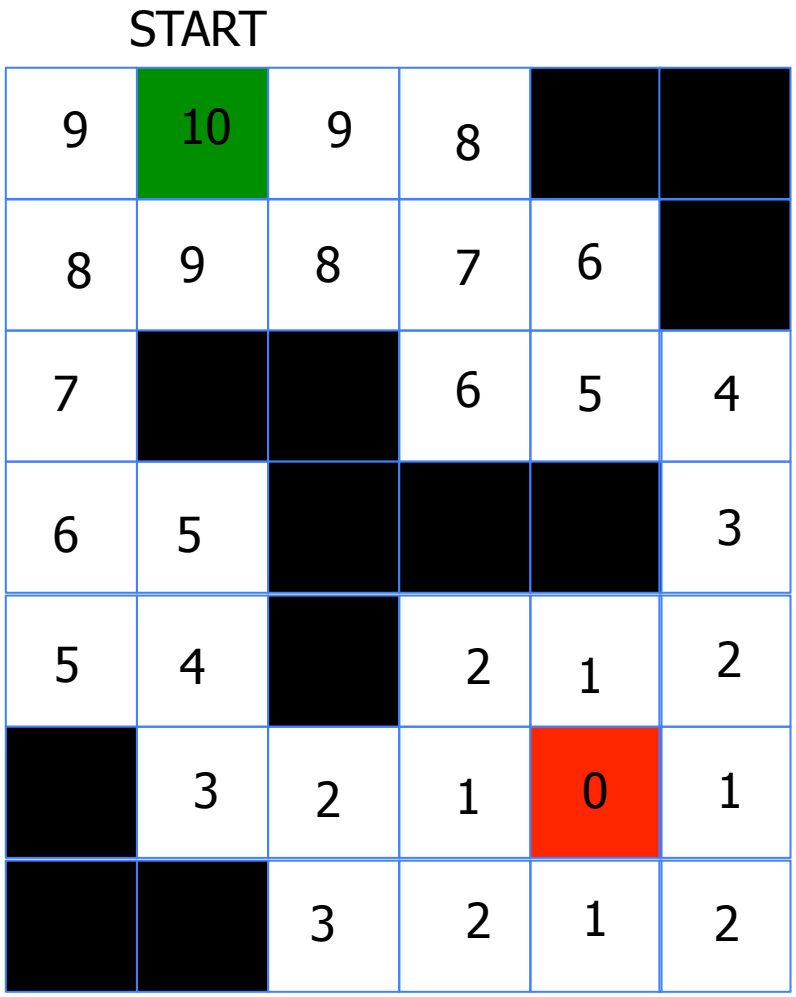
Planning Procedure – Grassfire Algorithm



Planning Procedure – Grassfire Algorithm

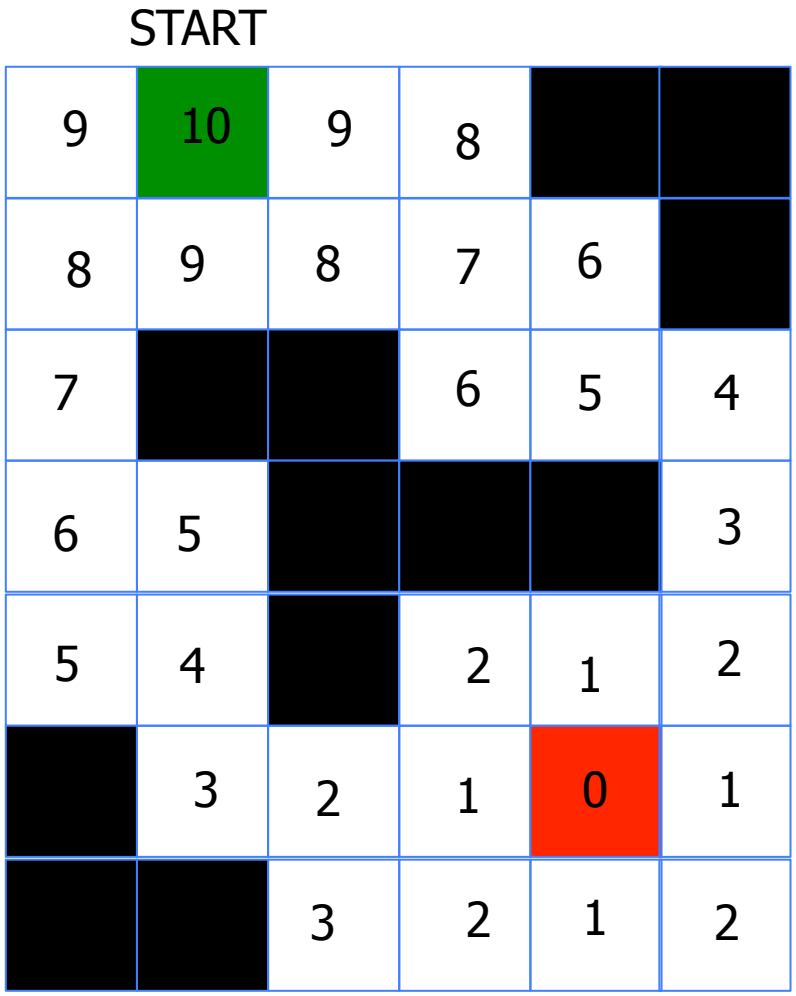


Planning Procedure – Grassfire Algorithm



Planning Procedure – Grassfire Algorithm

- The distance values produced by the grassfire algorithm indicate the smallest number of steps needed to move from each node to the goal



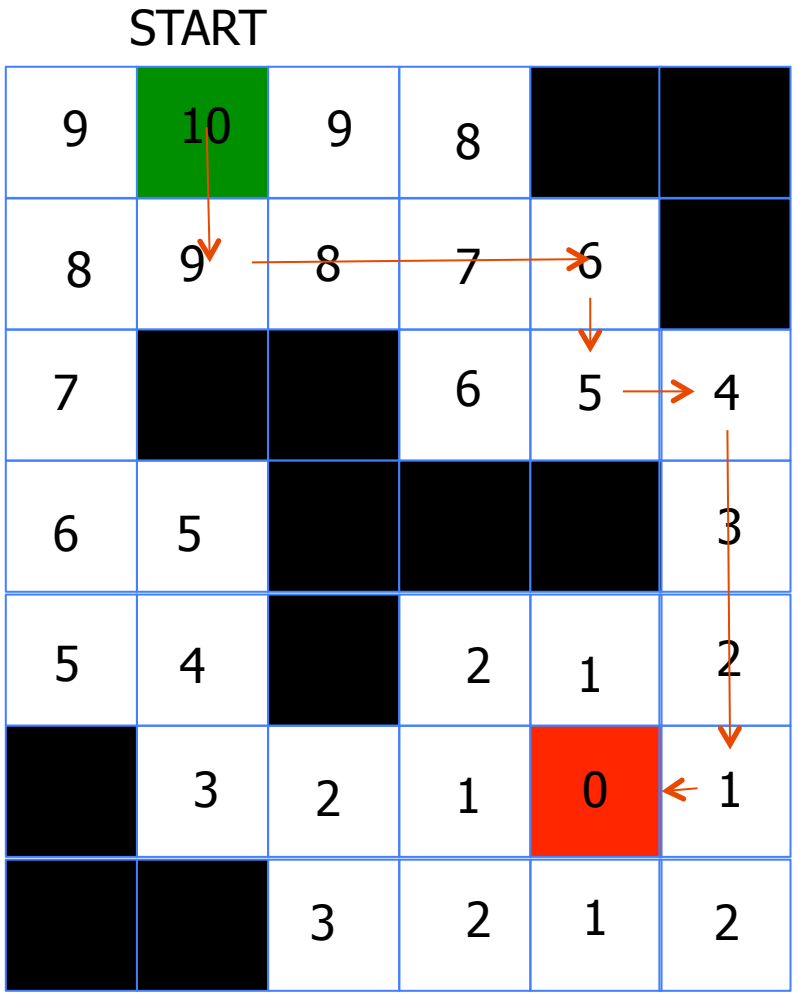
Grassfire algorithm – pseudo code

- For each node n in the graph
  - n.distance = Infinity
- Create an empty list.
- goal.distance = 0, add goal to list.
- While list not empty
  - Let current = first node in list, remove current from list
  - For each node, n that is adjacent to current

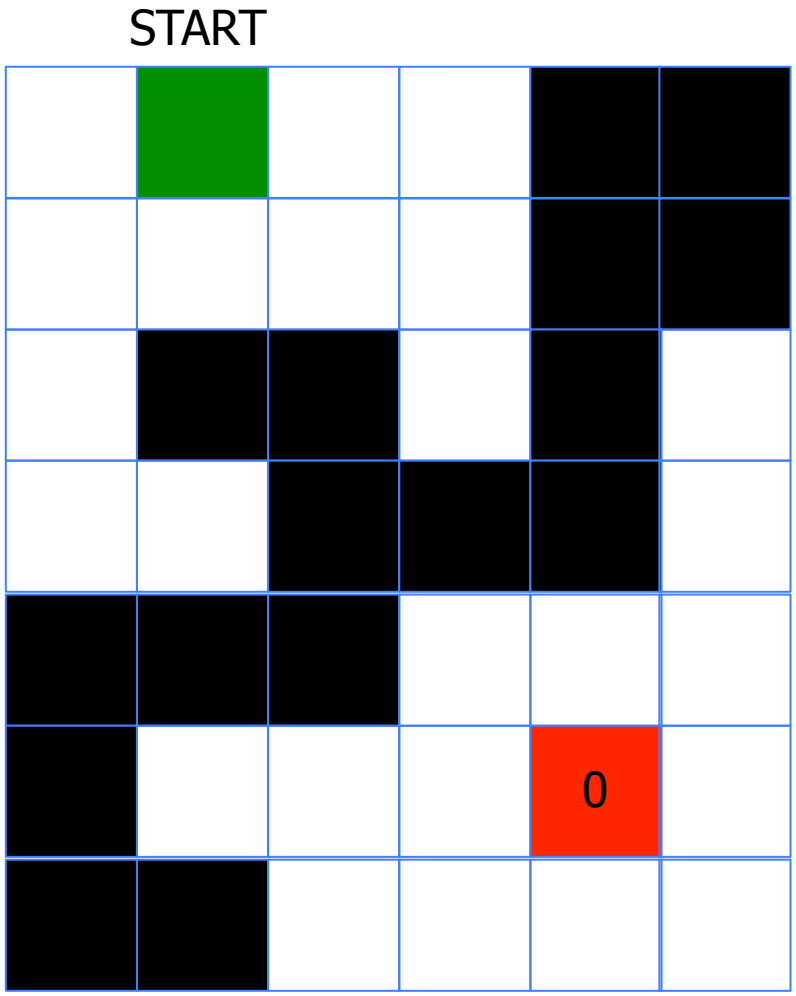
```
If n.distance = Infinity
    n.distance = current.distance + 1
    add n to the back of the list
```

Tracing a path to the destination

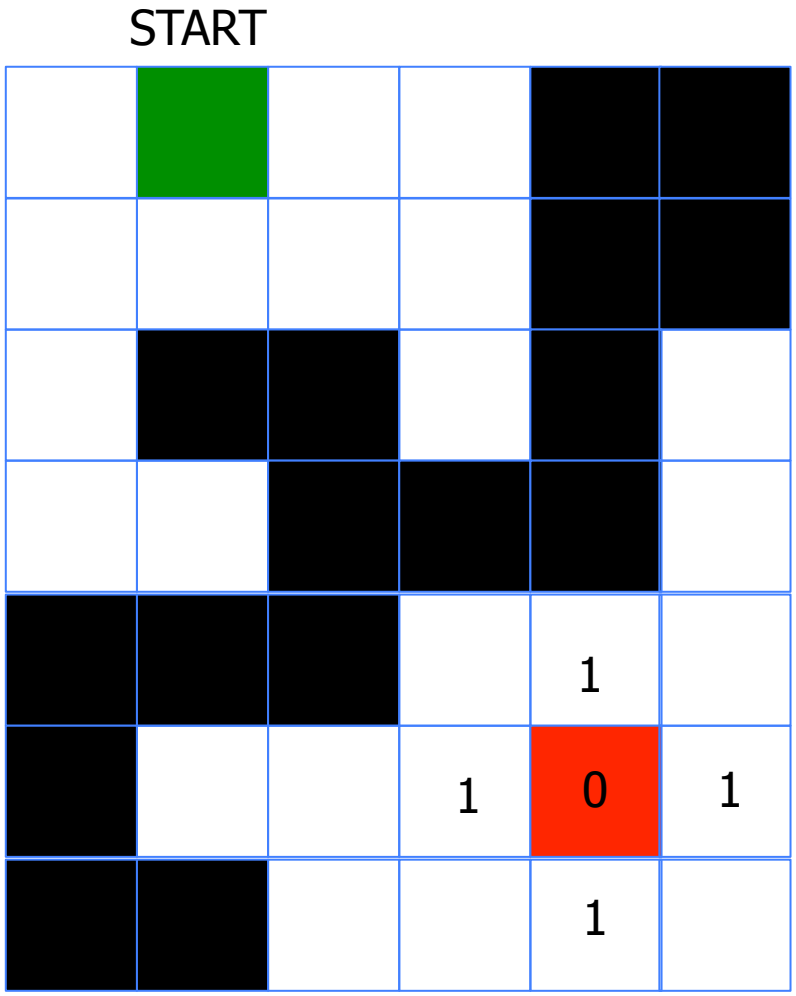
- To move towards the destination from any node simply move towards the neighbor with the smallest distance value, breaking ties arbitrarily.



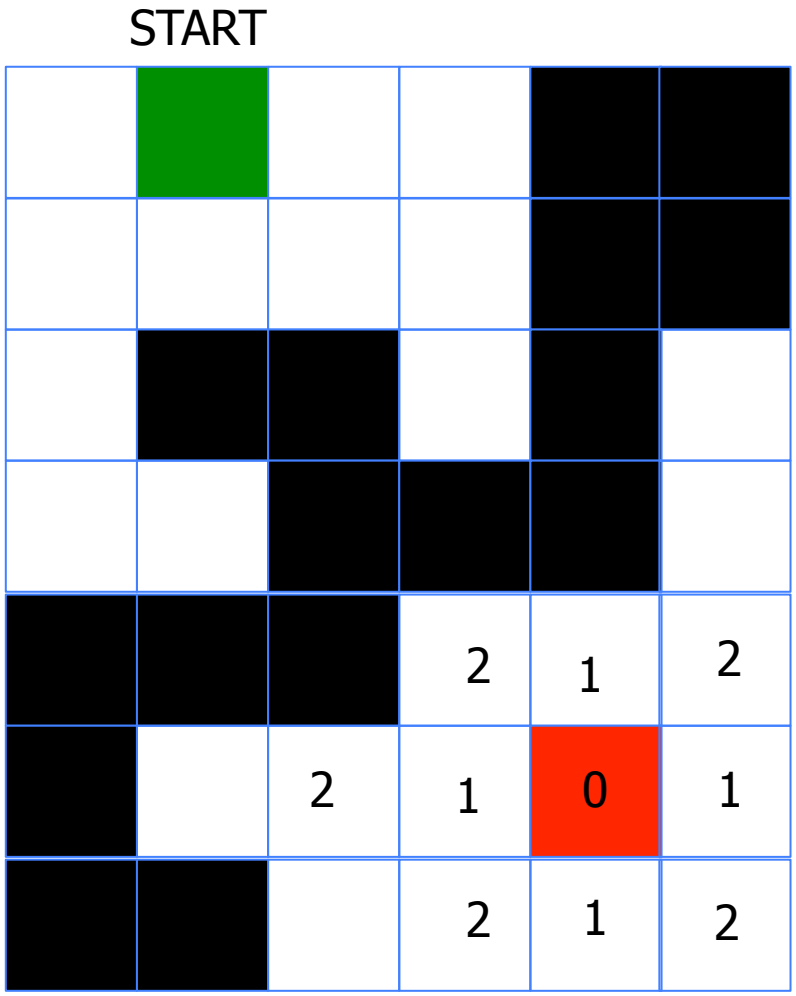
Another Example – Grassfire Algorithm



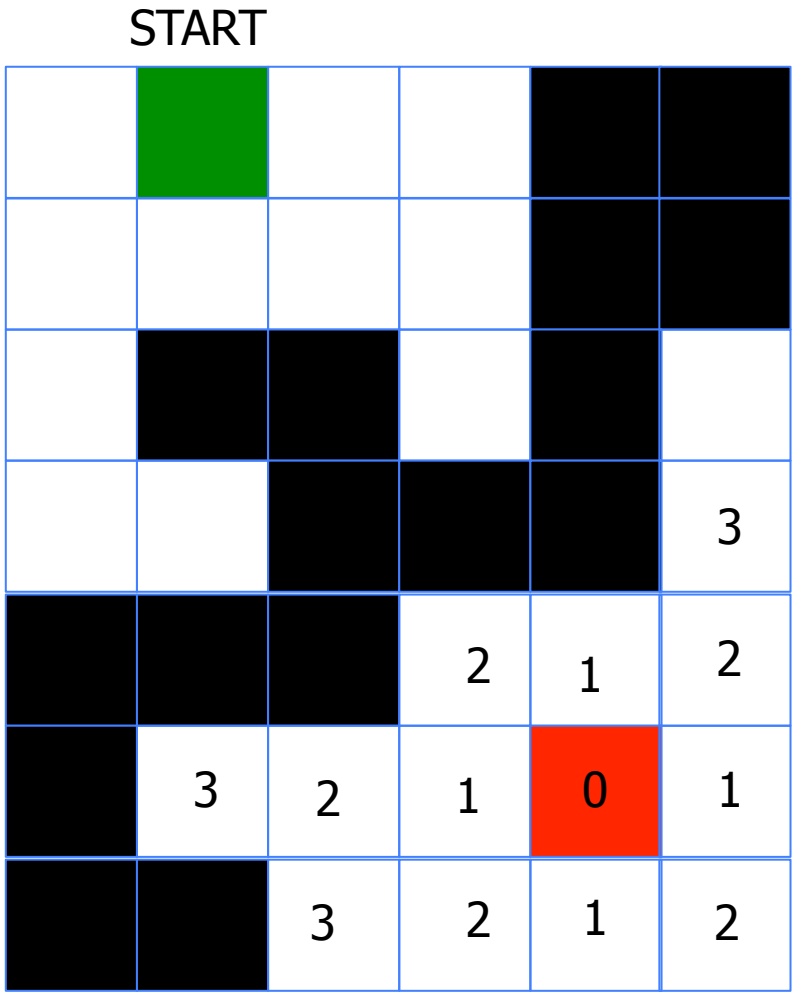
Another Example – Grassfire Algorithm



Planning Procedure – Grassfire Algorithm

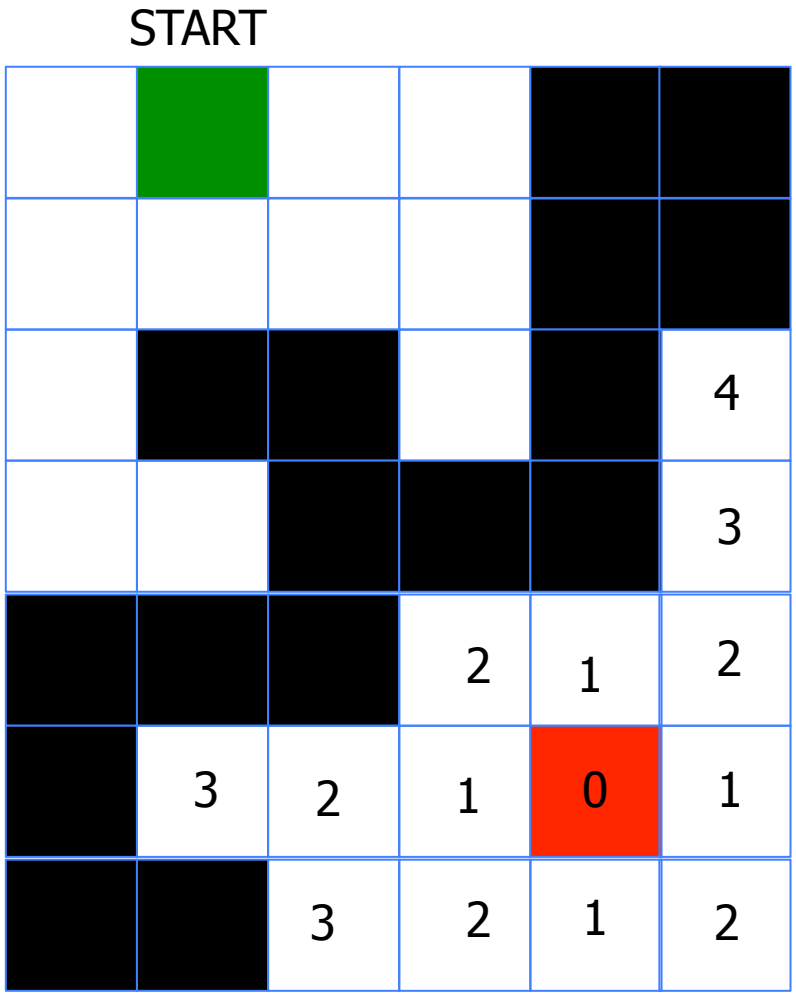


Planning Procedure – Grassfire Algorithm



Planning Procedure – Grassfire Algorithm

- In this case the procedure terminates before the start node is marked indicating that no path exists



Grassfire Algorithm

- It will find the shortest path between the start and the goal if one exists.
- If no path exists that fact will be discovered.

Computational Complexity - Grassfire

- The computational effort required to run the grassfire algorithm on a grid increases linearly with the number of edges.
- This can be expressed more formally as follows.

$$\mathcal{O}(|\mathbf{V}|) \tag{1}$$

Where $|\mathbf{V}|$ denotes the number of nodes in the graph

Computational Complexity - Grassfire

- Number of nodes in a 2D grid 100x100 = 10^4
- Number of nodes in a 3D grid 100x100x100 = 10^6
- Number of nodes in a 6D grid 100 cells on side = 10^{12}