1.

a)

Initial State: All pieces separated.

States: Every version of combinations for every piece of track that is left

Goal Test: Has the railway no overlapping tracks and no loose ends?

Path cost: one for setting up one piece of track.

b)

DFS would be the most suitable search algorithm because we will get a solution much faster than with BFS and with way less time-complexity and less storage-complexity. Every solution is at the last "depth point", with BFS we would get every solution, but every solution has the same path cost because we must place every piece of track. So, our goal is not to find the cheapest solution but the fastest way to a solution. Also, with BFS the big branching factor of this task would be a huge problem in terms of storage-complexity and time-complexity.

c)

Our goal is to have no overlapping tracks and no loose ends, that means we must make a circle. When one of the fork pieces gets removed the goal test could never be achieved and the task would be unsolvable.

2.

(Up, Right, Down, Left) (Cycle-Check)

(try which move has the lowest F)

L = 9 + 1

D = 9 + 2

R = 8 + 3

R = 7 + 4

D = 6 + 5

…

G: 0

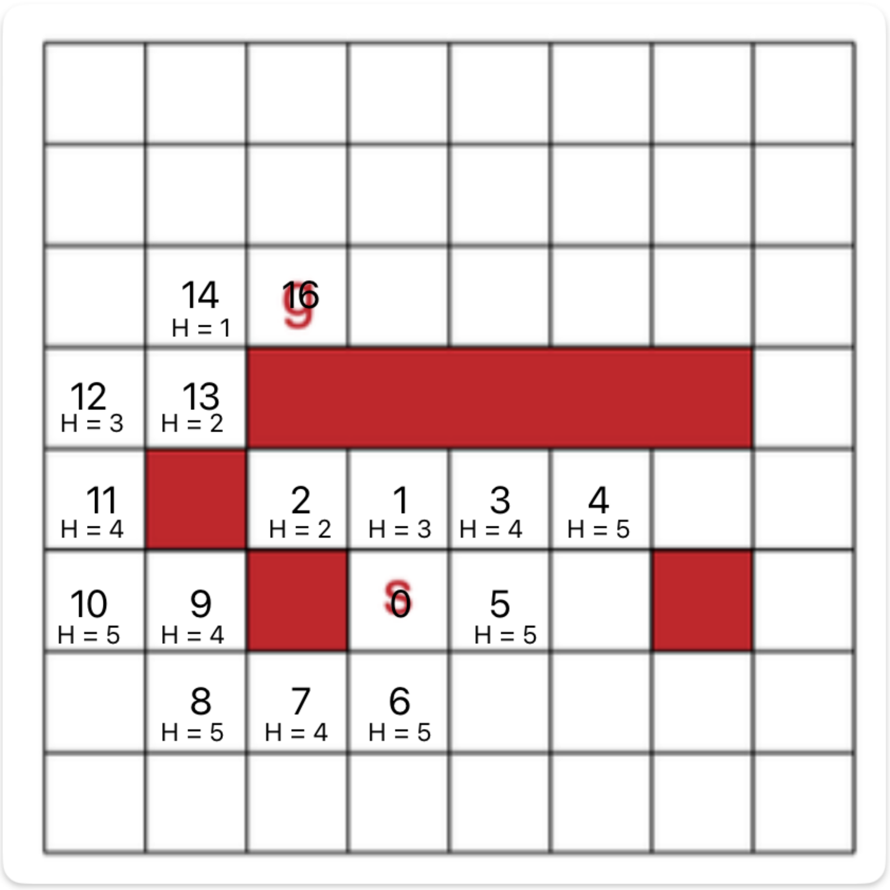
H: 10

F: G + H

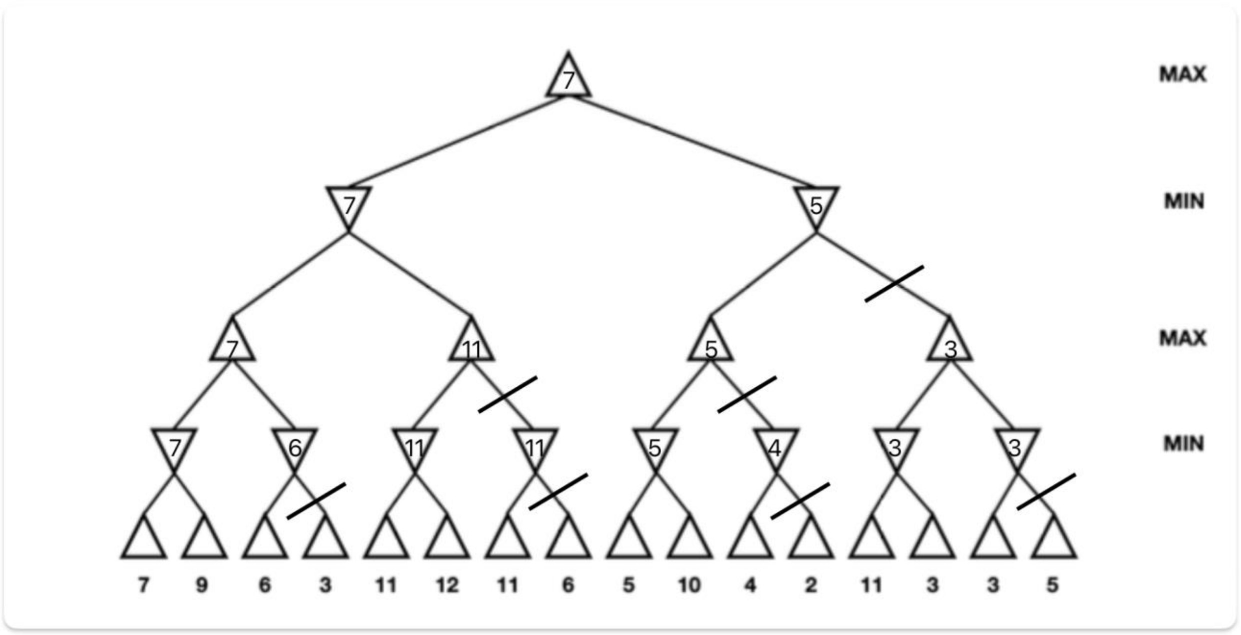
3.

Uniform-cost search is a special case of A*-search, but the difference is that Uniform-cost search uses the maximum priority in its priority queue (minimal path-cost) to find the best move to make and not a heuristic like the A*-search algorithm.

4.



5.



6.

Because the MAX player always plays like he is playing against an optimal MIN. So, if he is playing against a suboptimal MIN, he will still be playing like he is playing against an optimal MIN. That means the utility obtained by the MAX player will always be the same regardless of whether playing against an optimal MIN or a suboptimal MIN. Therefore, the utility obtained by MAX playing against a suboptimal MIN will always higher or equal then the utility playing against an optimal MIN.