1. a. Formulation of the task:

Initial state: Any piece which is chosen randomly.

Successor function: For a straight piece, we can choose any type of pieces from the rest of them. For a curved piece, we can choose any pieces with different orientations. For a fork piece, we can choose any pieces from the rest pieces. In the function, we need to check if there is any overlapping. If overlapping appears, break it.

Goal test: All of the pieces used in the track, no open slots and no overlapping.

Step cost: 1 / piece

b. My choice is Depth-First Search.

Breadth-First search won't work because the search space is so large.
Uniform-cost won't work because the same reason (search space is too large).
They are time comsuming.
Iterative Deepening can be a choice. However, some unneccessary works.
Depth-Limited search can be a choice.

c. The solution should has no open pegs or holes. In this problem, if we remove one 'folk' piece, then the total number of pegs would never be equal to the total number of holes. So the solution won't be exist any more.

d. The key point here is that only folk pieces have different number of pegs and holes. The maximum possible number of open pegs is 3. Any pieces can be connected to a peg, so there is at most $2^3+2^2+2\times16+12=56$ choice for each peg. The search depth is 32, because there are 32 pieces. So, the upper bound should be:

$$\frac{168^{32}}{(12!\,16!\,2!\,2!)}$$

2. a. If there is negative cost path, then after we add it to current path, the total cost will be small (no matter how large it is before we adding it). This situation shows that we need to find all the possible paths to make sure an optimal solution is found.

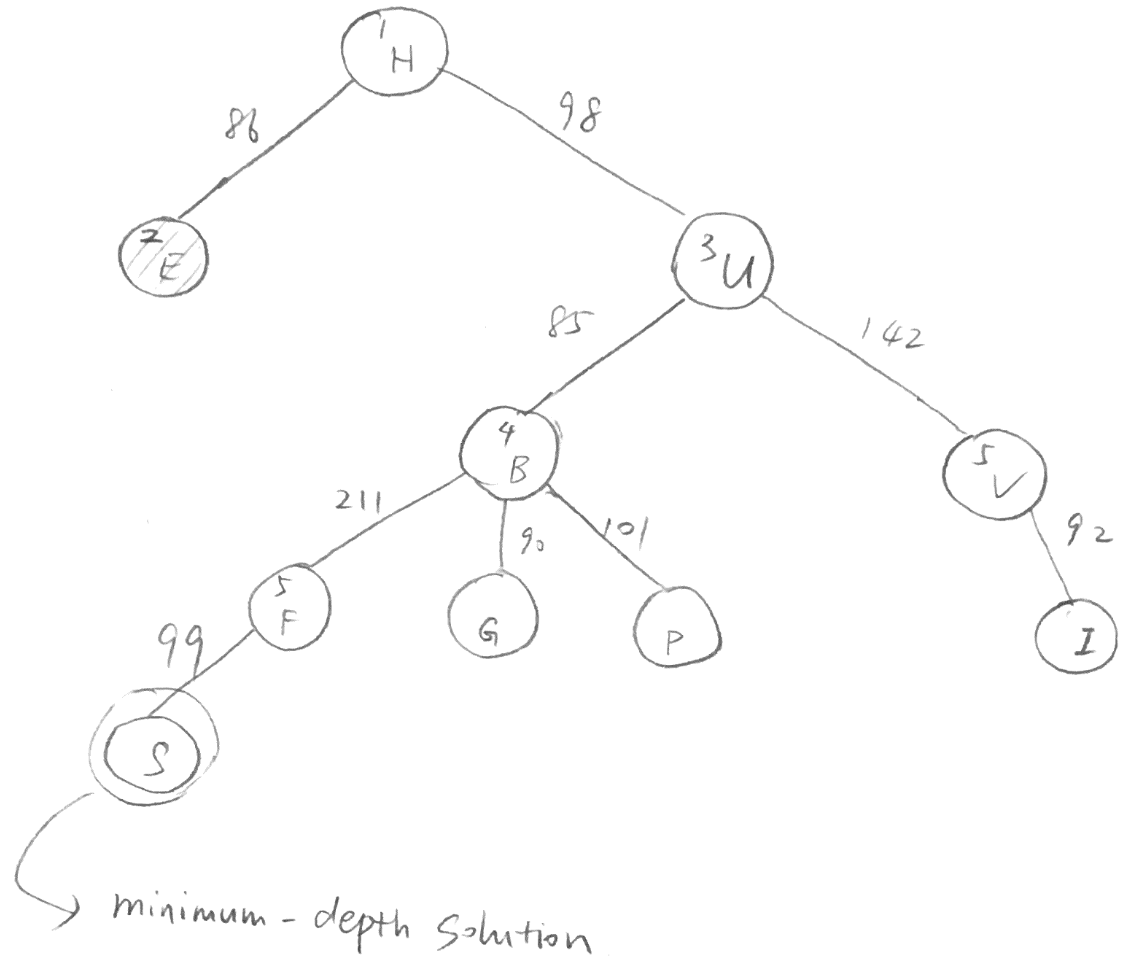
b. Suppose we know the maximum depth of the state space.
In trees: Any path which remains l levels can be improved by at most cl, so any path worse than cl and less than the best path can be pruned.
In graphs: Once there are loops in the state space, this guarantee won't work, because it is possible to go into the loop any numbers of time with the negative path c.

c. The agent should go around in the loop forever.

d. A novel scenic sight can be a large negative path. However, the value of this loop is increasing each time you visit it. Finally, it'll be a positive number, because you don't want to see it again and again even it is beautiful.
To solve this problem: we can put the times of visited places into the state. So a new location can have large reward, the reward decreases after we visit this place over and over again.

e. Eating food. If you are hungry, it's a big reward.

3 a) Breadth-first Search

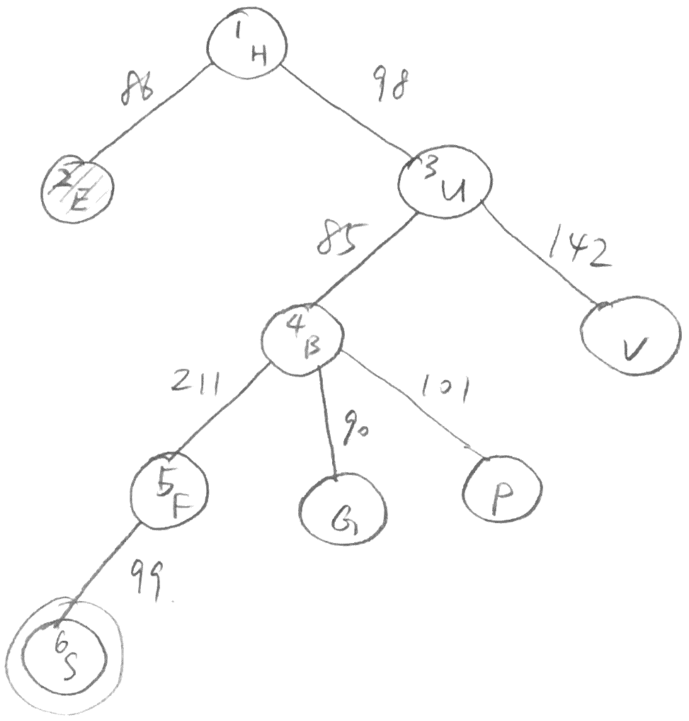


BFS solution:

H → U → B → F → S

Total path cost: 493

b) Depth-First Search

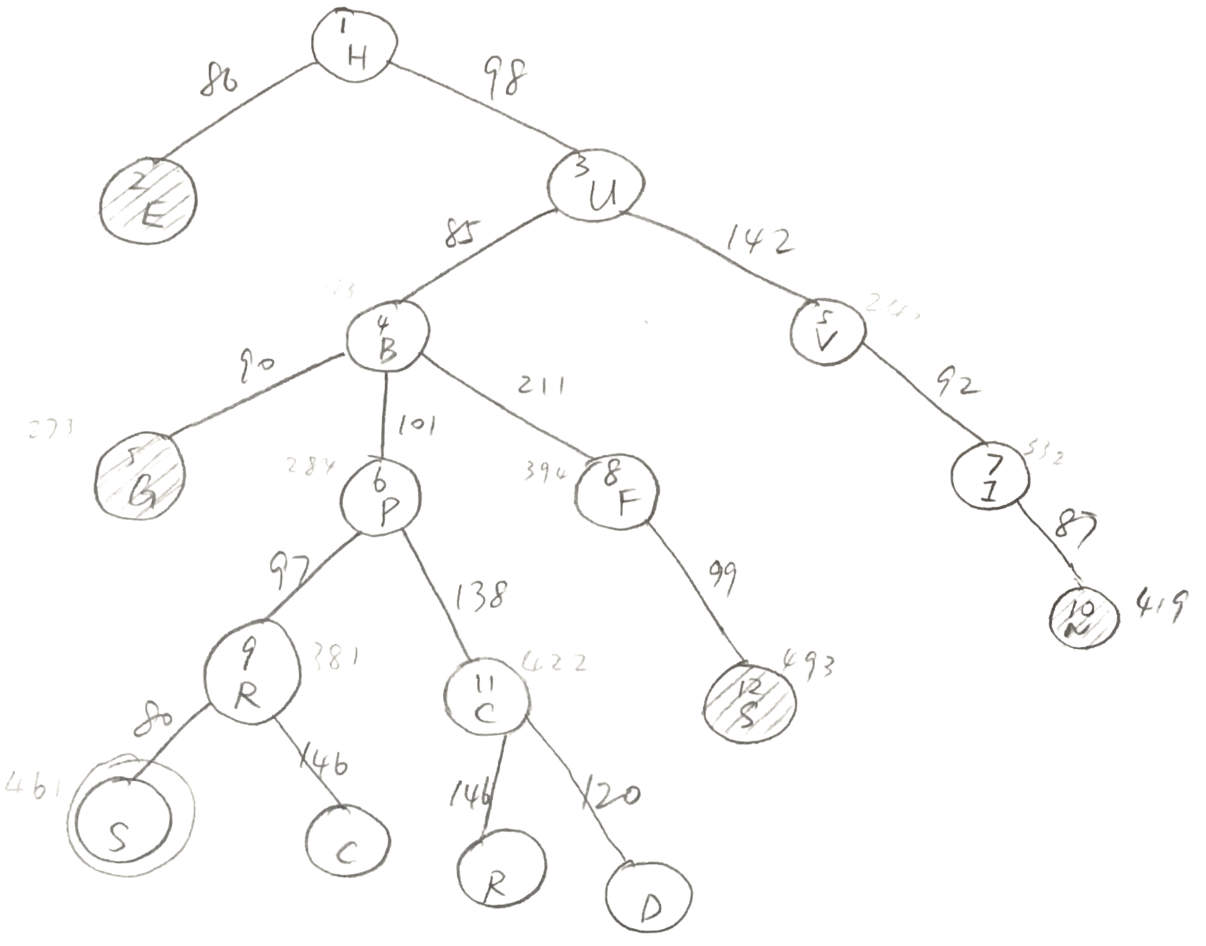


DFS solution:

H → U → B → F → S

Total path cost: 493

c) Uniform cost search

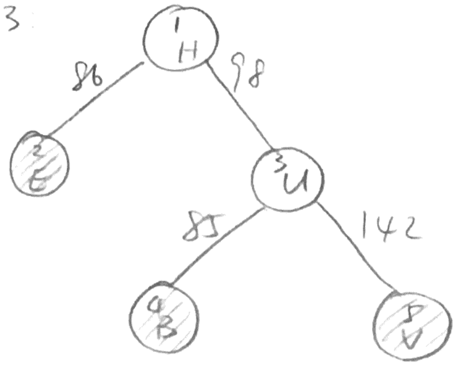


Solution path: H → U → B → P → R → S

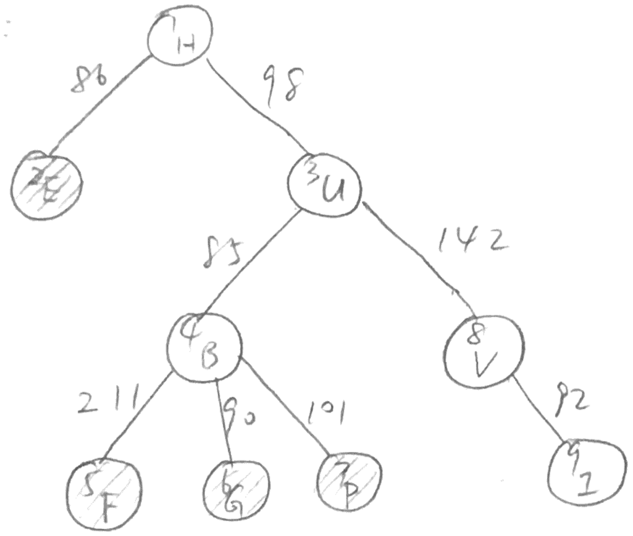
Total path costs: 461

d) Iterative Deepening

Depth = 3:



Depth = 4:

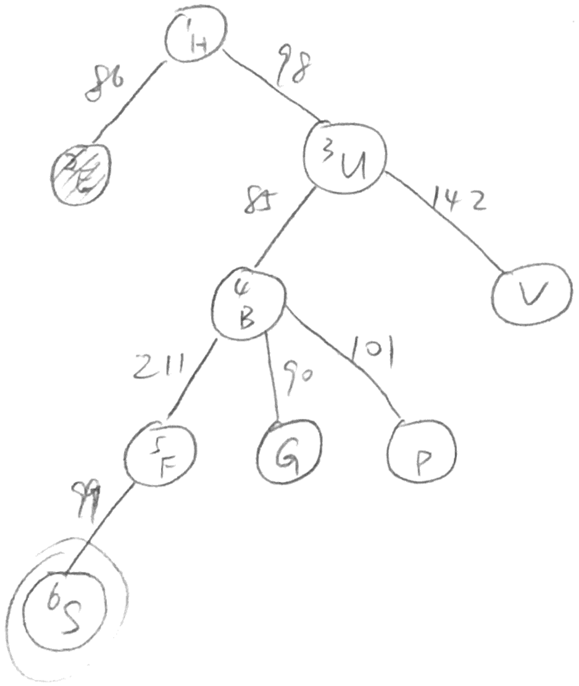


Solution path:

H → U → B → F → S

Total path costs: 493

Depth = 5:


1 H
86
98
2 E
3 U
85
142
4 B
V
211
90
101
5 F
G
P
99
6 S