**Introduction To Artificial Intelligence - HW1**

**Question 1:**

**1.1:** The search problem for the taxi environment is :

**S**: The state space consists of the 500 states in the environment, while a state is defined by the location of the taxi, the location of the passenger, and the passenger’s destination.

A state is a 4-tuple :

In each state the taxi is at one of the 25 locations on the map , the passenger can be at one of the R,G,B,Y locations or in the taxi, and the passenger’s desired destination can be one of the R,G,B,Y locations , therefore there are states in the state space.

**O**: The set of operators is .

Moves the taxi from the current position to the correct direction if there is no wall (pipe) between them with a penalty of -1 points. Moves the taxi to the south, moves the taxi to the north, moves the taxi to the east and moves the taxi to the west.

Is the pickup operation .It picks up the passenger with a penalty of -1 points if there is a passenger at the current location, or with a penalty of -10 points if there is no passenger at the current location.

Is the drop-off operation. If it is performed with no passenger aboard, the environment penalizes -10 points. Otherwise if the passenger is dropped at their destination the environment rewards 20 points, if the passenger is not dropped at their destination it penalizes -1 points.

: At the start state the taxi can be at any of the 25 locations on the map, the passenger and his/her destination can be at any of the R, G, B, Y locations.

The goal state is when the passenger is dropped at the destination .

**1.2 :**  , are all of the states that the agent is not at the top row of the map in them, and that don’t have a wall (pipe) between the position of the agent and the position north of it (directly on top of it in the map).

Therefore

In the given environment.

**1.3:**

**1.4:** infinite – solution may not be reached in the worst case.

**1.5:** the agent can reach the solution in 10 actions in the best case , since the passenger is at Y and the destination is R , the shortest way to reach the solution is this list of actions: .

**1.6:** since the agent has to perform this list of actions: , the list of operators is : , and the total reward is .

**1.7:** yes it is possible, for instance .

**Question 2:**

**2.2:** running BFS-G from state 328 leads to closing 31 vertices.

**2.3:** before opening a vertex (so that it may be closed later) we check whether it has been already opened or closed , if it has been then we do not open it again . That means a vertex can be closed only once, therefore all of the 31 vertices that were closed after running BFS-G from state 328 are different. So the number of the different vertices that were closed after running BFS-G from state 328 is 31.

**2.4:** BFS advantage over DFS in this environment: BFS is optimal, it is guaranteed that it returns the least cost path (explanation in 2.5), meanwhile DFS is not optimal.

DFS advantage over BFS in this environment: DFS’s memory requirements are much less than the memory requirements of BFS.

**2.5:** yes, BFS is optimal in the taxi problem. Proof:

All of the actions cost -1 points except for an illegal pickup or drop off which cost -10 points, and for the final drop-off at the destination which reward 20 points. Since the drop-off at the destination is the last action to be performed, the goal state is discovered before the action is performed (when the neighbors of the previous states are discovered) the reward is not relevant. Moreover, an illegal pickup or drop-off does not introduce a new state, meaning no vertex is opened, therefore these actions are also not relevant.

All of the relevant actions cost the same (-1 points), therefore we can run BFS as we learned, and the shortest path that is returned is guaranteed to be the least cost path.

**Question 3:**

**3.2:** running DFS from state 328 leads to closing 30 vertices.

**3.3:** before opening a vertex (so that it may be closed later) we check whether it has been already opened or closed , if it has been then we do not open it again . That means a vertex can be closed only once, therefore all of the 30 vertices that were closed after running DFS from state 328 are different. So the number of the different vertices that were closed after running DFS from state 328 is 30.

**Question 4:**

**4.2:** running DFS from state 328 leads to closing 308 vertices.

**4.3:** running DFS from state 328 leads to closing 31 vertices.

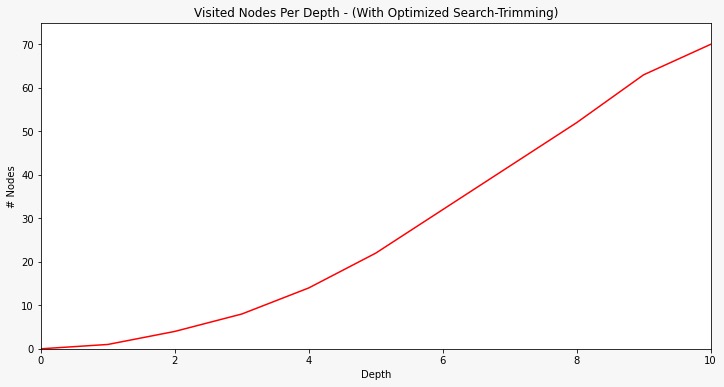
**4.4:** ID-DFS’s advantage over DFS: if a solutionwas found using ID-DFS then it is optimal, meanwhile if it was found using DFS it is not guaranteed.

ID-DFS’s disadvantage over DFS : the number of closed vertices in ID-DFS is much larger than the number of closed nodes in DFS (more running time).

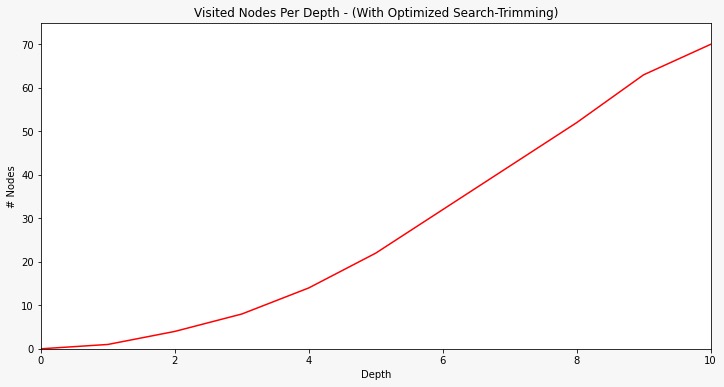
**4.5:**  ID-DFS’s advantage over BFS: a remarkable improvement in memory complexity.

ID-DFS’s disadvantage over BFS : the number of closed vertices in ID-DFS is larger than the number of closed nodes in BFS (more running time).

**4.6:** the deeper the search in ID-DFS the more vertices are closed.



**Question 5:**

**5.3:** yes it is possible, if a better path to the state was found then it is opened again.

We can see this at this line in the algorithm shown in lecture:

“OPEN.insert(n\_curr) “ // n\_curr <- node in CLOSED with state s

**5.5:** 1) greedyHeurisitc: Admissible.

Proof: for every .for.

2) ManhatanSumHeurisitc: Admissible.

Proof: first we acknowledge that in this specific environment Manhattan distance is

always less or equal than true cost. Dividing to two cases:

1. If the passenger has been picked up :

(a: the location of the taxi is the same location of the passenger since the passenger

has already been picked up)

(b: Manhattan distance is less or equal to true cost)

1. If the passenger has not been picked up yet:

(Manhattan distance is less or equal to true cost, moreover the taxi has to –at least-

reach pick up location then reach the destination from the pickup location)

We proved that the heuristic is admissible overall.

3) PickupSumHeuristic(s): Inadmissible.

Counter example: if

4) PickupMultHeuristic: Inadmissible.

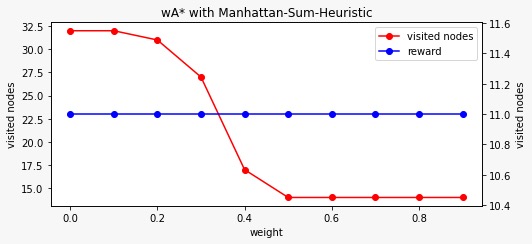
Counter example: : if

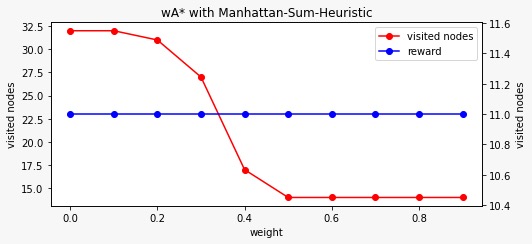
The most informative heuristic between the 4 given heuristics is the second one- ManhatanSumHeurisitc.

**5.7:** 14 vertices were closed when was performed, meanwhile the number of vertices that were closed when BFS was performed is 31, when DFS was performed is 30 and when ID-DFS was performed is 308 (31 different vertices).

Therefore, the number of closed vertices when is performed is relatively half of the number of closed vertices when a blind search is performed, or 4/100 of the number of the –not necessarily different-closed vertices when ID-DFS is performed.

**5.9:**



**5.10:** the graph describes that changing the weight does not affect the reward, it is fixed at 11, but increasing the weight decreases the number of closed nodes when running.

Therefore, it is better to run with a weight larger the 0.5, since decreasing the number of closed nodes decreases the running time, and there is no downside to increasing the weight since the reward is not affected by it.

We can see in this example that the principle learned in class is not necessarily always true, the reward with weight=0.6 is not better than the reward with weight=0.8 (there are equal) , and the number of closed nodes with weight=0.8 is not less the number of closed nodes with weight=0.6 (they are equal).

**Question 6:**

**6.3:**