Question 1 (part -2)

a) What is the time and space complexity of IDA *?

Solution:

Introduction:

The IDA* is actually a hybrid of both Iterative Deepening and A* Algorithm.

Iterative deepening A* (IDA*) is an optimal search algorithm with the performance properties of **A*—it is complete and optimal**—and the space requirements of **DFS—(essentially) linear in depth**. The main idea of iterative deepening A*is to repeatedly search in depth-first fashion, over sub graphs with f-cost less than α, less than 2α, less than 3α, and so on, until a goal is found, **where α is a lower bound on the cost between nodes and their successors throughout the search space**: i.e., $\alpha \leq c(n, m)$, for all n, m 2 ε δ(n).

Recall that the space complexity of ID is O(bd), where d is the depth of the goal node. Similarly, the space complexity of IDA* is $O(bC^*/\alpha)$, where C* is the optimal cost. Whereas the Space complexity of A* algorithm is $O(b^d)$, which is much expensive. This is due to that A* keep the record of each and every visited node.

<u>Time Complexity:</u>

<u>Case I :When Step cost is constant:</u>

For calculating the time complexity of IDA* we can consider it as equivalent to ID algorithm as it work in the same fashion if the f- cost is equal. So, the nodes on the bottom level are expanded once, those on the next to bottom level are expanded twice, and so on, up to the root of the search tree, which is expanded d+1 times .So the total number of expansions in an iterative deepening search is **$O(b^d)$.**

<u>Comparison with A*</u>

Where as the Time complexity of A* would surely be dependent on the type of heuristic we are applying. For problems with constant step cost the growth in run time as a function of the optimal solution depth d is analyzed in terms of absolute or relative error of the heuristic. The absolute error is defined as $\Delta = h^* - h$, where h* is the actual cost of getting from root to goal and relative error is defined as
$\varepsilon = (h^* - h)/h^*$.
For constant step cost the time complexity would be $O(b^{\varepsilon d})$,where d is the solution depth.

Now, for almost all the heuristics the absolute error is proportional to h*. Which means that ε is either constant or growing. As suppose h*-h is proportional to h* by a factor of 3. Which result in the value of **ε = (h*-h)/h* = (3h*)/h* = 3.**

So, the time complexity of A* would be O(b^d) as we can ignore ε.

Case II:

The time complexity of IDA*, however, can exceed that of A*.
In particular, in search spaces where the step - cost is different at every state, only one additional state is expanded during each iteration. In such a search space, if A* expands N nodes, IDA_*expands 1+. . .+N = **O(N^2)** nodes. The typical solution to this problem is to fix an increment β> α such that several nodes n have cost $f_i < f(n) \le f_i + \beta$, where fi is the i^{th} incremental value of the step-cost. This strategy reduces search time, since the total number of iterations is proportional to 1/β< 1/α, and returns solutions that are at worst β-optimal: i.e., if the algorithm returns m*, then g(m*) < C*+ β.

b) Is IDA* complete?

Solution:
Completeness: Completeness of heuristic/algorithm may be defined as if the algorithm guaranteed to find a solution.

Now, If suppose C* is the cost of the optimal solution path, then we can say that:

- IDA* expands all the nodes with f(n) < C*, that's is obvious because no heuristic can over estimate the cost of path ,it will always be certainly less than C*.
- IDA* might expand some nodes right on the GOAL path i.e. f(n) = C* before selecting a goal node.

So**, Completeness clearly requires that there be only finitely many nodes with cost less than or equal to C*,** a condition that will be true if all step costs exceed some finite γ and if b is finite.

Because there could me infinite node within the optimal path cost value and that will put the IDA* into infinite loop. Similarly, if the step comes out to be zero which is practically impossible than also IDA* won't give any results.

Conclusion: IDA* would give the result no matter what!!! But may take a little bit longer.

c) Is IDA* Optimal?

Solution:

OPTIMALITY: Optimality of any heuristic may be defined as answer to the question that does the strategy finds the optimal solution. The path cost function measures solution quality, and an optimal solution has the lowest path cost among all solutions.

So, the next step is to prove that whenever IDA* selects a node n for expansion, the optimal path to that node has been found.

We all know that these heuristics works in the condition that f is non decreasing along any path .

Since Iterative Deepening A* performs a series of **depth-first searches**, its memory requirement is linear with respect to the maximum search depth. In addition, if the heuristic function is admissible, IDA* finds an optimal solution. Finally, by an argument similar to that presented for DFID, IDA* expands the same number of nodes, asymptotically, as A* on a tree, provided that the number of nodes, asymptotically, as A* on a tree, provided that the number of nodes grows exponentially with solution cost. These costs, together with the optimality of A*, imply that IDA* is asymptotically optimal in time and space over all heuristic search algorithms that find optimal solutions on a tree. Additional benefits of IDA* are that it is much easier to implement, and often runs faster than A*, since it does not incur the overhead of managing the open and closed lists.

Conclusion: IDA* would give the optimal solution.

In Summary:

Criteria	IDA*
Time	$O(N^2)$,if step costs differ at all states ,when if A* expands N nodes.
Space	$O(bC^*/\alpha)$,if f is monotonically non-decreasing in depth and if C^*
Completeness	**YES,** if there do not exist ∞- many nodes n such that $f(n) < f^* + \alpha$ is the optimal cost.
β-Optimality	YES, if h is admissible and g is monotonically non decreasing at depth.

Question 1(1):

Solution:

Generalization: As given 8 puzzle problem, here we have branching factor, b = 3;

- When empty tile is in middle = 4 moves
- When empty tile in corner = 2 moves
- When empty tile is on edge = 3 move

So, average branching factor will be 3.

This will certainly lead to 3^{22} exhaustive tree search required to solve this problem. But we can cut short this kind of problem by applying some nice heuristic.

For the heuristic to be admissible it should never over estimate the path cost.

1) Misplaced Tile Heuristic:
   h_1 = the number of misplaced tiles. For the given puzzle two tiles i.e. "5 & 6" are misplaced, so the state would have $h_1 = 2$.

   As, $h_1 = 0 + 0 + 0 + 0 +1 +1 +0 +0 =$ **2.**

   This heuristic is obviously admissible heuristic because it is clear that any tile that is out of place must be moved at least once.

2) Manhattan Distance:
   H_2 = the sum of the distances of the tiles from the goal position. The distance will be sum of horizontal and vertical distance. Here, $h_2 = 2$.

   As, $h2 = 0 + 0 + 0 + 0 +1 +1 +0 +0 =$ **2.**

   H2 is also admissible because all any move can do is move one tile one step closer to the goal.

   Also, true solution cost here is 26.

From the above two observation, it looks like both the heuristic will perform equally with A*.But this not true in reality as h2 ≥ h1 no matter what. In other words h2 always dominate h1.

Suppose there is another 8-puzzle tile as follow:

7	6	
4	3	1
2	5	8

1	2	3
8		4
7	6	5

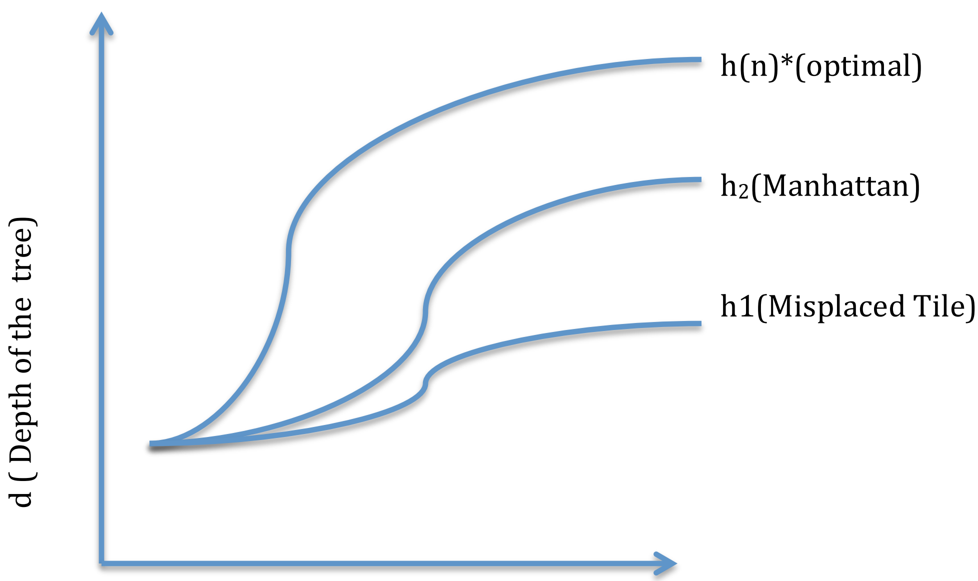
Initial State Goal State

Here,
H1 = 8 (as all of them are misplaced).

H2 = 3+ 3+ 2+ 2+ 1+ 2+ 2+ 3 = 18.

So, there it seems like H1 is better than h2 as it is giving us lesser moves or less cost than h2. Here, h2 is dominating h1. Domination translates directly into efficiency: A* using h2 will never expand more nodes than using h1(except for some nodes where f(n) = C*). This is same as saying that every node with h(n) < C* will surely be expanded. But because h2 is at least as big as h1 for all nodes, every node that is surely expanded by A* search with h2 will also surely be expanded with h1.

Hence, it is generally better to use a heuristic function with higher under estimate value. In other words, it is good to get the heuristic that gives the highest possible under estimate of cost.



b*(Branching Factor) {Graph not precise ,just to demonstrate}

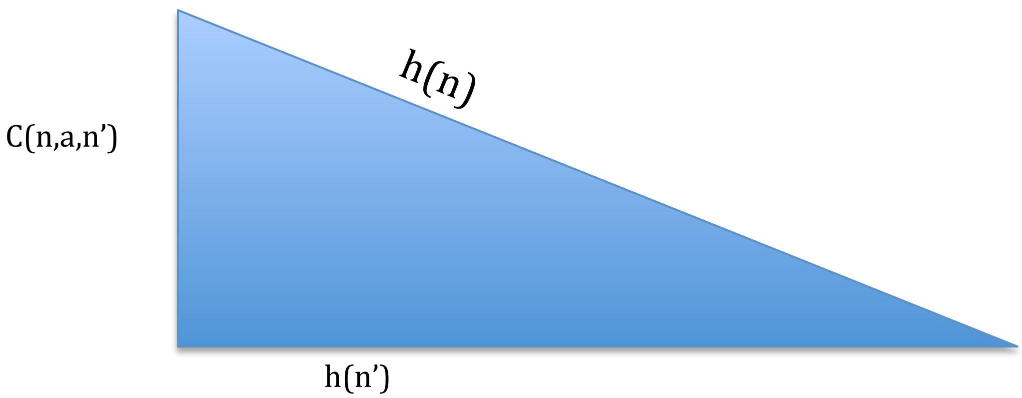
This is clear that we must have to choose that heuristic which is closer to the optimal heuristic that

Question 1 (1)

b)

Solution: Both h1 and h2 heuristics are admissible as they never over estimate the cost of reaching to the goal.

Consistent: For any heuristic to be consistent it should follow the triangle equality. Which is that sum of two sides will always be greater than or equal to the sum of the third side. This fact's analogy to the heuristic is as under:



This means that:

$h(n) \leq c(n,a,n') + h(n')$.

So, heuristic h(n) is consistent if, for every node n and every successor n' and n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n'.

<u>Proof of consistency of h1(Missing Tiles heuristic)</u> :

- $c(n,a,n') = 1$ for any action *a*
- Claim: $h_1(n) \leq h_1(n') + c(n,a,n') = h_1(n') + 1$
  - Now, no move (action) can get more than one misplaced tile into place.
  - Also, no move can create more than one new misplaced tile.
  - Hence, the above follows. I.e. **h_1 is consistent.**

<u>Proof of consistency of h2 (Manhattan Distance) :</u>

- $c(n,a,n') = 1$ for any action a.

- Claim: $h_1(n) \leq h_1(n') + c(n,a,n') = h_1(n') + 1$.
  - Now, if we view the above moving tiles as making right angled triangle than it will **prove the above heuristic as consistent**.
  - Because in right angled triangle $\mathbf{a + b \geq c}$

REFERENCES:

1) http://en.wikipedia.org/wiki/Heuristic.

2) http://www.cs.brown.edu/courses/cs141/amy_notes/astar.pdf

3) Russell and Norvig : A Modern Approach(3rd Edition)