**Q.1(c):**

For A\* to provide an optimal results, it is necessary that the heuristic function is admissible. For the heuristic function to be admissible, it means that it underestimates the cost to reach the goal state.

It is important and necessary for the heuristic function to be admissible because it is responsible for picking the next node to visit from the current node. The accuracy of the heuristic function determines the number of paths to explore in-order to find the most optimal path, if the heuristic is 0, the algorithm behaves like djikstra and explores quite a lot of nodes, as the heuristic gets close to the actual cost, the number of nodes the algorithm explores decreases. If the heuristic is 100% accurate, we do not have to explore any path other than the optimal path. Interestingly, if the heuristic overestimates there is a possibility that we might end up not exploring the optimal path.

This is because, an admissible heuristic shows a path as more promising than it actually is (because h(n) <= actual cost) and hence the algorithm explores it. On the other hand, if the heuristic is not admissible, its shows a path worse than it actually is, and hence it there is a possibility that the algorithm does not explore that path.

Suppose we are at node A, and have to reach node G. There are two paths, via C and D both at the distance of 5, suppose h(C) = 10 and h(D) = 5, while actual Cost from C to G is 5, and D to G is 7. At A, we f(D) = 5 + 7 = 12 and f(C) = 10 + 5 = 15, hence we end up going to D. From D, we have G at 7, so f(G) = 5 + 7 + 0 = 12, we end up exploring G, and arrive at the destination. This is because the heuristic, showed C as much more worse than it actually was, hence it went explored. If the heuristic was admissible, we would have explored C in any case before arriving at G.

Another requirement for the the A\* algorithm, is monotonicity of the heuristic function, i.e.

Where, c (n,n’) = cost to reach n’ from n.

This implies that the nodes along a path are in non-decreasing order of f(n) and hence any node n’ from node n would have a higher cost. This is precisely why when we reach the goal state from a path, we know for certainty that it is the most optimal path. If it was not the optimal path, other nodes in the frontier would have been explored earlier than the goal node.

**Q.2(c):**

The primary difference between search and optimization problem is the knowledge of the goal state. In optimization problem we are looking for an state which maximizes or minimizes the cost function, while in search problems we are aware of the goal state, we are just find the path.

We can say that in search problems we are more interested in the path, and in optimization we are more interesting in finding such an state.

Following are examples of search problem:

1. Finding a certain value in a list of numbers
2. Searching for a path in a map
3. Find a flight route such that the time of flight is minimum
4. Searching for a certain file within a directory
5. Maze traversal

Following are examples of optimization problem:

1. Finding a minimum value in a list of numbers.
2. Making a schedule such that the number of clashes are minimum
3. Finding the minimum and maximum values of function
4. Knapsack problem
5. Travelling Sales person problem