Neelam Aggarwal (6667974120)

Greedy Search and A Star Search

Greedy Search and A*(Star) Search, both are the examples of the best first search which expands most desirable node and the desirability is calculated through an evaluation function heuristic h(n). Heuristic is defined at each node that estimates the cost of reaching the goal.

In greedy search, the next node is chosen based on the smallest value of the heuristic function. This means it greedily chooses the node which appears promising (have smallest value to reach the goal). So, a greedy search may yield locally optimal solutions. But the local optimal solutions may approximate a global optimal solution in a reasonable time.

So, this strategy need not finds the best solution because it do not operate exhaustively on all the data but terminates in a fair number of steps which can be beneficial for some class of problems because for this class of problems finding a optimal solution requires unreasonably many steps.

If greedy search algorithm can be proven to yield global optimal solution for a given problem class, then it typically becomes method of choice because it is faster.

Summary of Greedy search:

1. It finds the local optimal solution. (+1)
2. It is faster and quicker compare to A*(star) search. (+1)
3. It can approximate the global optimal solution with less no of steps and reasonable time. (+1)
4. It is short sighted and non recoverable. (-1)
5. It does not operate exhaustively on data. (1)
6. It is not complete. (-1)
7. In worst case, it is no better than Breadth First Search. (-1)

In A*(star) search, it uses the cost of the path from the initial node to the current node g(n) in addition to the heuristic estimate h(n) (estimated cost from the current node to the goal node). So, the total function that determines the next node to be expanded is $f(n) = g(n) + h(n)$.

The secret of its success is that it combines the pieces of information that Dijkstra's algorithm uses (favoring vertices that are close to the starting point) and information that Greedy Best-First-Search uses (favoring vertices that are close to the goal). It's like Dijkstra's algorithm in that it can be used to find a shortest path. It's like Greedy Best-First-Search in that it can use a heuristic to guide itself.

Thus, the total function estimates the optimal cost of any solution path going through the current node and at each point a node with lowest function value is chosen for expansion. A* is both complete and optimal if we uses the admissible heuristic function means it never overestimates the actual cost.

Summary of A*(Star) search:

1. It is complete (finds the path if one exists).

2. It finds optimal solution (always finds the shortest path if heuristic function is admissible).
3. It can potentially search a huge area of the map.

Explanation of the output of our program:

We can analyze that greedy search greedily chooses the node which have lower value. We start from node m1 then looked for the heuristic value of its neighbors and found m3 to have lowest value and expanded the same and added its neighbors to queue then again looked for the least heuristic value. In this way we always choose the node with least heuristic value but resulted in dissimilarity cost of 37 which is not optimal. So, just based on the estimated cost to reach the goal may not result in the smallest path with minimal possible cost.

In A*, we uses total value of the dissimilarity cost (backward) and the heuristic function (forward) that gives more optimal solution. At each node we looked for the least total function cost and this end up having the total dissimilarity cost from start node to the goal node of 25 which is optimal and is less than 37 . So, greedy gave the approximate value but not optimal solution.

So, we can see the path taken by Greedy search (m1 ,m3 , m6 , m9 , m12) is not optimal in spite of taking least heuristic value while the A Star taken the different path of traversal from Greedy (m1 , m3 , m4 , m7 , m9 , m2 , m8 , m12) based on the total function value and resulting in the propagating path of (m1 , m3 , m7 , m9 , m12) using backtracking with the total cost of 25 which is optimal.