**Artificial Intelligence (MCA573D)**

**LAB-4**

**1. Algorithm –** Traversing or finding optimal path in a graph using A\* algo.

**2. Definition and Example -**

A\* algorithm works based on heuristic methods and this helps achieve optimality. A\* is a different form of the best-first algorithm. Optimality empowers an algorithm to find the best possible solution to a problem.

Such algorithms also offer completeness, if there is any solution possible to an existing problem, the algorithm will definitely find it.

When A\* enters into a problem, firstly it calculates the cost to travel to the neighbouring nodes and chooses the node with the lowest cost. If The f(n) denotes the cost, A\* chooses the node with the lowest f(n) value. Here ‘n’ denotes the neighbouring nodes.

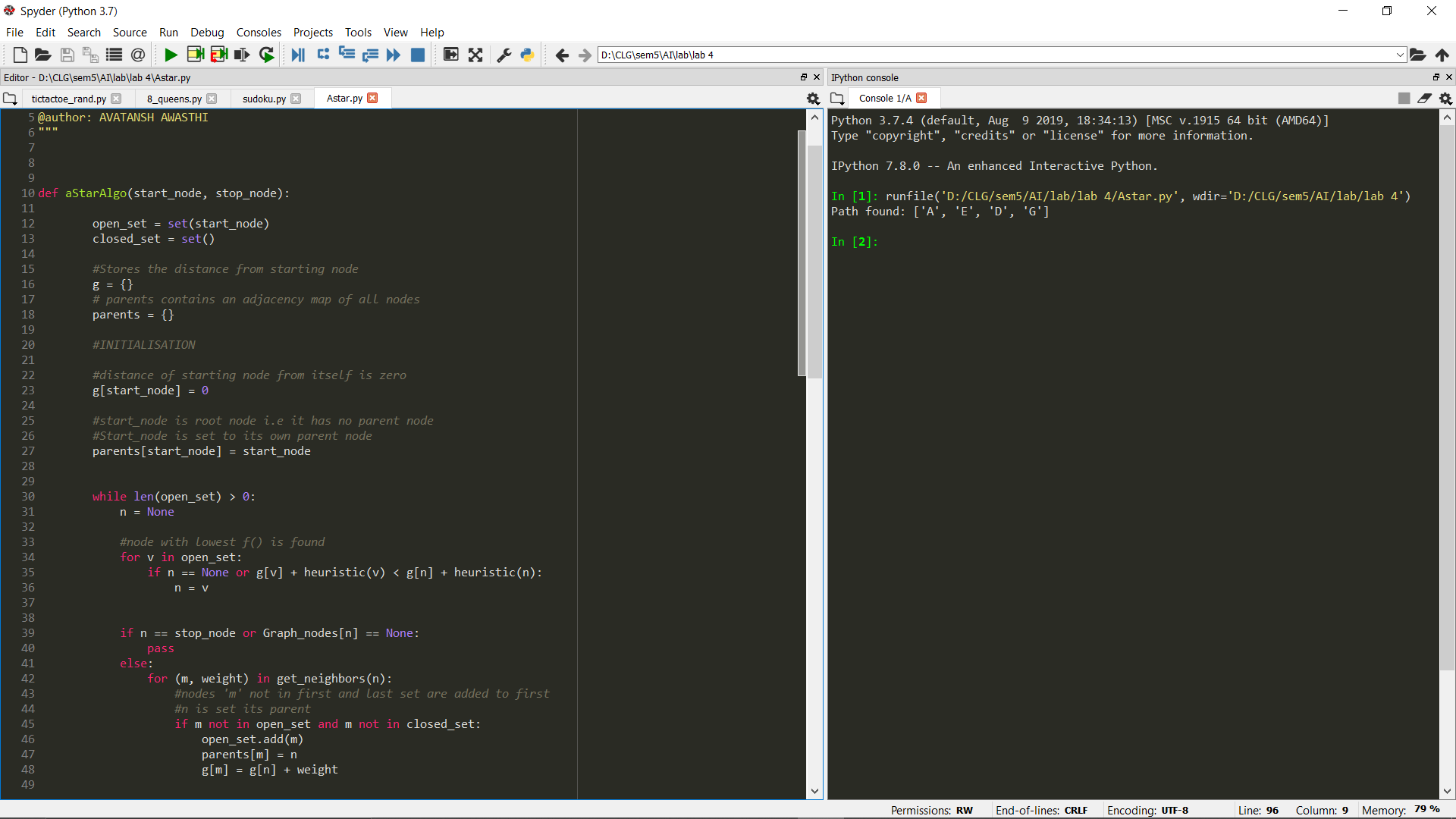
The calculation of the value can be done as shown below:  
f(n)=g(n)+h(n)f(n)=g(n)+h(n)

Here,  
g(n) = shows the shortest path’s value from the starting node to node n  
h(n) = The heuristic approximation of the value of the node.

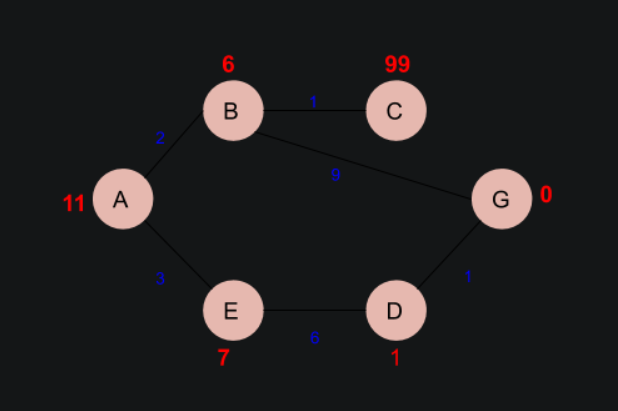
The heuristic value has an important role in the efficiency of the A\* algorithm. To find the best solution, we have to use different heuristic function according to the type of the problem. However, the creation of these functions is a difficult task, and this is the basic problem we face in AI.

**3. Implementation -**

Following is the code implementation of optimal path finding graph in Python :



**4. Explanation –**

****

Following is the explanation of the python code used in implementation:

The numbers written on edges represent the distance between the nodes while the numbers written on nodes represent the heuristic values. Let us find the most cost-effective path to reach from start state A to final state G using A\* Algorithm.

Let’s start with node A.Since A is a starting node, therefore, the value of g(x) for A is zero and from the graph, we get the heuristic value of A is 11, therefore

g(x) + h(x) = f(x)

0+ 11 =11

Thus for A, we can write

A=11

Now from A, we can go to point B or point E, so we compute f(x) for each of them

A → B = 2 + 6 = 8

A → E = 3 + 6 = 9

Since the cost for  A → B is less, we move forward with this path and compute the f(x) for the children nodes of B

Since there is no path between C and G, the heuristic cost is set infinity or a very high value

A → B → C = (2 + 1) + 99= 102

A → B → G = (2 + 9 ) + 0 = 11

Here the path A → B → G has the least cost but it is still more than the cost of A → E, thus we explore this path further

A → E → D = (3 + 6) + 1 = 10

Comparing the cost of A → E → D with all the paths we got so far and as this cost is least of all we move forward with this path. And compute the f(x) for the children of D

A → E → D → G = (3 + 6 + 1) +0 =10

Now comparing all the paths that lead us to the goal, we conclude that A → E → D → G is the most cost-effective path to get from A to G.

**5. Time Complexity -** The time complexity of path search in graph with V(Vertices) and E(Edges) is O(V+E).

For E.g. In our case the V = 6 and E = 6 so, the time complexity of the problem will be O(6+6) = O(12).