

Ahsanullah University of Science and Technology

Department of Computer Science and Engineering



CSE4108

Artificial Intelligence

TERM PROJECT 02

Topic no: 01

Topic Name: A* Search

Submitted By:

Mayisha Farzana

16.02.04.028

Date of Submission: **12 September, 2020**

A* Search:

A* Search algorithm is one of the best and popular technique used in path-finding and graph traversals. It is widely known form of best first search.

Evaluation function,

$f(n) = g(n) + h(n)$, where

$g(n)$ – an actual path cost from initial node to node n ,

$h(n)$ – estimated cost of the cheapest path from n to the goal,

$f(n)$ – estimated cost of the cheapest solution through node n .

Generates all neighbors (may be repeatedly), and puts in PQ.

1. A Priority Queue (PQ), which contains nodes in ascending order of $f(n) = g(n) + h(n)$ values, is also maintained. The PQ offers the node for expansion.

2. A Possible Path (PP) is maintained that contains nodes currently supposed to be in the solution.

3. A tree of visited nodes along with their children is also maintained which helps to update PQ and PP.

4. The process begins by placing the source node (initial state) in the empty PQ, and initiating a tree by placing that node as its root. The process terminates when the destination node (goal state) is placed in the PQ, and selected, consequently.

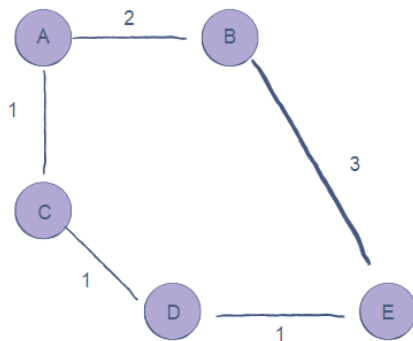
5. The 1st node from the PQ is selected repeatedly, and each time the tree, the PQ and the PP are updated.

Suboptimal solutions are avoided. If the goal is not chosen for expansion, that is, not the 1st element in the PQ, then not considered achieved. A* in tree search is optimal if $h(n)$ is admissible.

A* in general graph search is optimal if $h(n)$ is consistent. If $h(n)$ is consistent, then the values of $f(n)$ along any path are non decreasing. A* expands all nodes with $f(n) < C^*$, where C^* is the cost of the optimal solution. A* is complete, assuming that there are only finite number of nodes with cost less than or equal to C^* . A* is said to be optimally efficient, that is, no other optimal algorithm is guaranteed to expand fewer nodes than A* does. A* is a very powerful algorithm with almost unlimited potential. However, it is only as good as its heuristic function, which can be highly variable considering the nature of a problem.

Example:

Suppose you have the following graph and you apply A* algorithm on it. The initial node is **A** and the goal node is **E**.

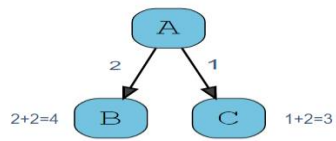


At every step, the f -value is being re-calculated by adding together the g and h values. The minimum f -value node is selected to reach the goal state. We can see that node **B** is *never* visited.



Root node A.

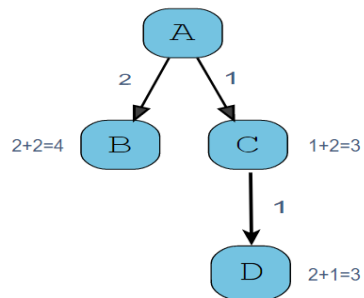
Figure 1



Node C is chosen.

In here, B node $h(n)$ is 2 and for C node $h(n)$ is 2.

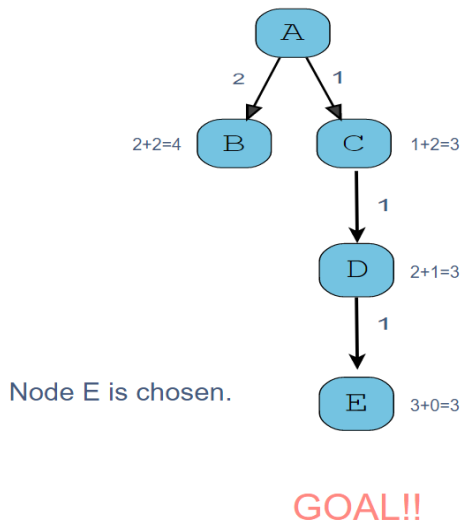
Figure 2



Node D is chosen.

In here, D node $h(n)$ is 1

Figure 3



In here, E node $h(n)$ is 0.

Figure 4

Python Code:

```

inf = 10000000
h = [55, 42, 34, 25, 20, 17, 0, inf, 80]

graph = [ [(2, 22), (3, 32), (8, 35)],
  [(8, 45), (4, 36), (5, 27), (3, 28)],
  [(0, 22), (3, 31), (6, 47)],
  [(0, 32), (2, 31), (1, 28), (6, 30)],
  [(1, 36), (6, 26)],
  [(1, 27)],
  [(2, 47), (3, 30), (4, 26)],
  [],
  [(0, 35), (1, 45)] ]
from queue import PriorityQueue
Q = PriorityQueue()
def empty():
    return Q.empty()
def push(val):
    Q.put(val)
def pop():
    if empty():
        return None
    return Q.get()
def a_star():

```

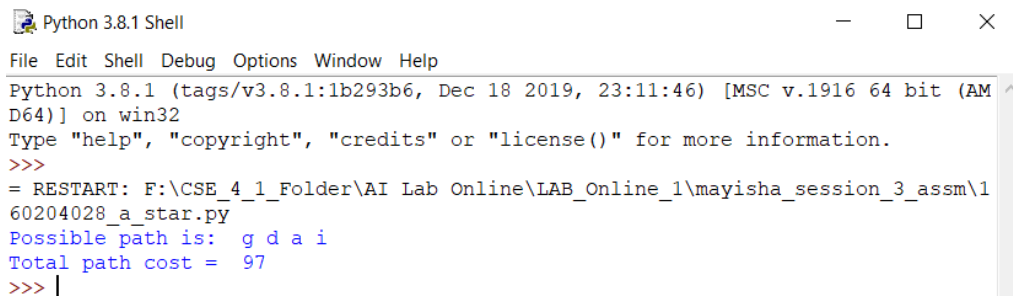
```

source = 8 # i
goal = 6 # g
parent = [-1] * len(graph)
g = [inf] * len(graph)
g[source] = 0
push((g[source] + h[source], source))
while (empty() == False):
    tmp = pop()
    u = tmp[1]
    for adj in graph[u]:
        v = adj[0]
        if (g[u] + adj[1] < g[v]):
            g[v] = g[u] + adj[1]
            push((g[u] + adj[1] + h[u], v))
            parent[v] = u
# finding path
tmp = goal
sum = 0
print("Possible path is: ", end = ' ')
while (parent[tmp] != -1):
    u = parent[tmp]
    v = tmp
    print(chr(ord('a') + tmp), end = ' ')
    for adj in graph[u]:
        if (adj[0] == v):
            sum += adj[1]
            break
    tmp = parent[tmp]
print(chr(ord('a') + tmp))
print("Total path cost = ", sum)

a_star()

```

OUTPUT:



```

Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\CSE_4_1_Folder\AI Lab Online\LAB_Online_1\mayisha_session_3_assm\160204028_a_star.py
Possible path is:  g d a i
Total path cost =  97
>>> |

```