Name: Shaikh Mohd Raza

Roll No: 20CO53

**Experiment No : 4**

**AIM:**

Program on Informed Search: A\* search.

**THEORY:-**

A \* algorithm is a searching algorithm that searches for the shortest path between the initial and the final state. It is used in various applications, such as maps.

A\* algorithm has 3 parameters:

* **g :** the cost of moving from the initial cell to the current cell. Basically, it is the sum of all the cells that have been visited since leaving the first cell.
* **h :** also known as the heuristic value, it is the estimated cost of moving from the current cell to the final cell. The actual cost cannot be calculated until the final cell is reached. Hence, h is the estimated cost. We must make sure that there is never an over estimation of the cost.
* **f :** it is the sum of g and h. So, f = g + h

The way that the algorithm makes its decisions is by taking the f-value into account. The algorithm selects the smallest f-valued cell and moves to that cell. This process continues until the algorithm reaches its goal cell.

**CODE :-**

def aStarAlgo(start\_node, stop\_node):

open\_set = set(start\_node)

closed\_set = set()

g = {}

parents = {}

g[start\_node] = 0

parents[start\_node] = start\_node

while len(open\_set) > 0:

n = None

for v in open\_set:

if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):

n = v

if n == stop\_node or Graph\_nodes[n] == None:

pass

else:

for (m, weight) in get\_neighbors(n):

#nodes 'm' not in first and last set are added to first

#n is set its parent

if m not in open\_set and m not in closed\_set:

open\_set.add(m)

parents[m] = n

g[m] = g[n] + weight

else:

if g[m] > g[n] + weight:

g[m] = g[n] + weight

parents[m] = n

if m in closed\_set:

closed\_set.remove(m)

open\_set.add(m)

if n == None:

print('Path does not exist!')

return None

if n == stop\_node:

path = []

while parents[n] != n:

path.append(n)

n = parents[n]

path.append(start\_node)

path.reverse()

print('Path found: {}'.format(path))

return path

open\_set.remove(n)

closed\_set.add(n)

print('Path does not exist!')

return None

def get\_neighbors(v):

if v in Graph\_nodes:

return Graph\_nodes[v]

else:

return None

def heuristic(n):

H\_dist = {

'A': 11,

'B': 6,

'C': 99,

'D': 1,

'E': 7,

'G': 0,

}

return H\_dist[n]

Graph\_nodes = {

'A': [('B', 2), ('E', 3)],

'B': [('C', 1),('G', 9)],

'C': None,

'E': [('D', 6)],

'D': [('G', 1)],

}

aStarAlgo('A', 'G')

**CONCLUSION :-**

In this experiment we learned about the informed search: A\* Search.

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.

It has found applications in many software systems, from Machine Learning and Search Optimization to game development where NPC characters navigate through complex terrain and obstacles to reach the player.