EXP NO: 4

DATE:

A * SEARCH ALGORITHM

AIM:

To implement an A* search algorithm using python.

PROGRAM:

```
from collections import deque
class Graph:
    def __init__(self, adjac_lis):
        self.adjac_lis = adjac_lis
    def get_neighbors (self, v):
        self.adjac_lis[v]
    def h(self, n):
        H = {
            'A':1,
            'B':1,
            'C':1,
            'D':1
        }
        return H[n]
    def a_star_algorithm(self, start, stop):
        open_lst = set([start])
```

```
closed_lst = set([])
poo = {}
poo[start] = 0
par = {}
par[start] = start
while len(open_lst) > 0 :
    n = None
    for v in open_lst :
        if n == None or poo[v] + self.h(v) < poo[n]
                                      + self.h(n):
            n = v;
    if n == None :
        print('Path does not exist!')
        return None
    if n == stop :
        reconst_path = []
        while par[n] != n:
            reconst_path.append(n)
            n = par[n]
        reconst_path.append(n)
        n = par[n]
    reconst_path.append(start)
    reconst_path.reverse()
    print('Path found : {}'.format(reconst_path))
    return reconst_path
```

```
for (m, weight) in self.get_neighbors(n):
    if m not in open_lst and m not in closed_lst:
        open_lst.add(m)
        par[m]=n
        poo[m] = poo[n] + weight
    else:
        if poo[m] > poo[n] + weight:
            poo[m] = poo[n] + weight
            par[m] = n
            if m in closed_lst:
                closed_lst.remove(m)
                open_lst.add(m)
    open_lst.remove(n)
    closed_lst.add(n)
print('Path does not exist')
return None
adjac_lis = {
    'A': [('B', 1), ('C', 3), ('D', 7)],
    'B': [('D', 5)],
    'C': [('D', 12)]
}
graph1 = Graph(adjac_lis)
graph1.a_star_algorithm('A', 'D')
```

Output :

```
Path found : ['A', 'B', 'D']
```

RESULT :

Thus the python program has been implemented to solve A* search algorithm.

[illegible]