ASSIGNMENT – 7

Aim : A* Algorithm

Code :

```
from collections import deque

class Node:
    def __init__(self, adjac_lis):
        self.adjac_lis = adjac_lis


    def get_neighbors(self, v):
        return self.adjac_lis[v]


    def h(self, n):
        H = {
            'P': 1,
            'Q': 1,
            'R': 1,
            'S': 1
        }


        return H[n]


    def 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(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 = {
    'P': [('Q', 1), ('R', 3), ('S', 7)],
    'Q': [('S', 5)],
    'R': [('S', 12)]
}
node1 = Node(adjac_lis)
node1.algorithm('P', 'S')
```

Output :

Shell

```
Path found: ['P', 'Q', 'S']
>
```