PROGRAM 15:

from collections import deque

class Graph:

def \_\_init\_\_(self, adjac\_lis):

self.adjac\_lis = adjac\_lis

def get\_neighbors(self, v):

return self.adjac\_lis[v]

# This is heuristic function which is having equal values for all nodes

def h(self, n):

H = {

'A': 1,

'B': 1,

'C': 1,

'D': 1

}

return H[n]

def a\_star\_algorithm(self, start, stop):

# In this open\_lst is a lisy of nodes which have been visited, but who's

# neighbours haven't all been always inspected, It starts off with the start

#node

# And closed\_lst is a list of nodes which have been visited

# and who's neighbors have been always inspected

open\_lst = set([start])

closed\_lst = set([])

# poo has present distances from start to all other nodes

# the default value is +infinity

poo = {}

poo[start] = 0

# par contains an adjac mapping of all nodes

par = {}

par[start] = start

while len(open\_lst) > 0:

n = None

# it will find a node with the lowest value of f() -

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

OUTPUT:

