PROGRAM 1

#!/usr/bin/env python # coding: utf-8

# In[1]:

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):

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': | 10, |
| 'B': | 8, |
| 'C': | 5, |
| 'D': | 7, |
| 'E': | 3, |
| 'F': | 6, |
| 'G': | 5, |
| 'H': | 3, |
| 'I': | 1, |
| 'J': | 0 |

}

return H\_dist[n]

Graph\_nodes = {

'A': [('B', 6), ('F', 3)],

'B': [('C', 3), ('D', 2)],

'C': [('D', 1), ('E', 5)],

'D': [('C', 1), ('E', 8)],

'E': [('I', 5), ('J', 5)],

'F': [('G', 1),('H', 7)] , 'G': [('I', 3)],

'H': [('I', 2)],

'I': [('E', 5), ('J', 3)],

}

aStarAlgo('A', 'J')