**Artificial Intelligence Lab**

**LAB 5 – Developing Best first search and A\* Algorithm for real world**

**problems**

**Aditya Kumar**

**RA1911030010100**

**O2 Section**

**Date – 18/2/22**

**Problem Statement :**

Implementation of A star algorithm for an application.

**Algorithm:**

1. Initialize two lists, open and closed, closed list with starting node
2. While open list is not empty:
3. Take initial node with least f and pop in from open list
4. Generate its succressors
5. For each successor:
6. If successor is goal stop search
7. Else compute g and h for successor using heuristic function
8. If a node with same position as successor has lower f, skip it,else add it to open list
9. End loop
10. Push the initial node to closed loop
11. End loop

**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):

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': [('A', 2), ('C', 1), ('G', 9)],

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

'D': [('E', 6), ('G', 1)],

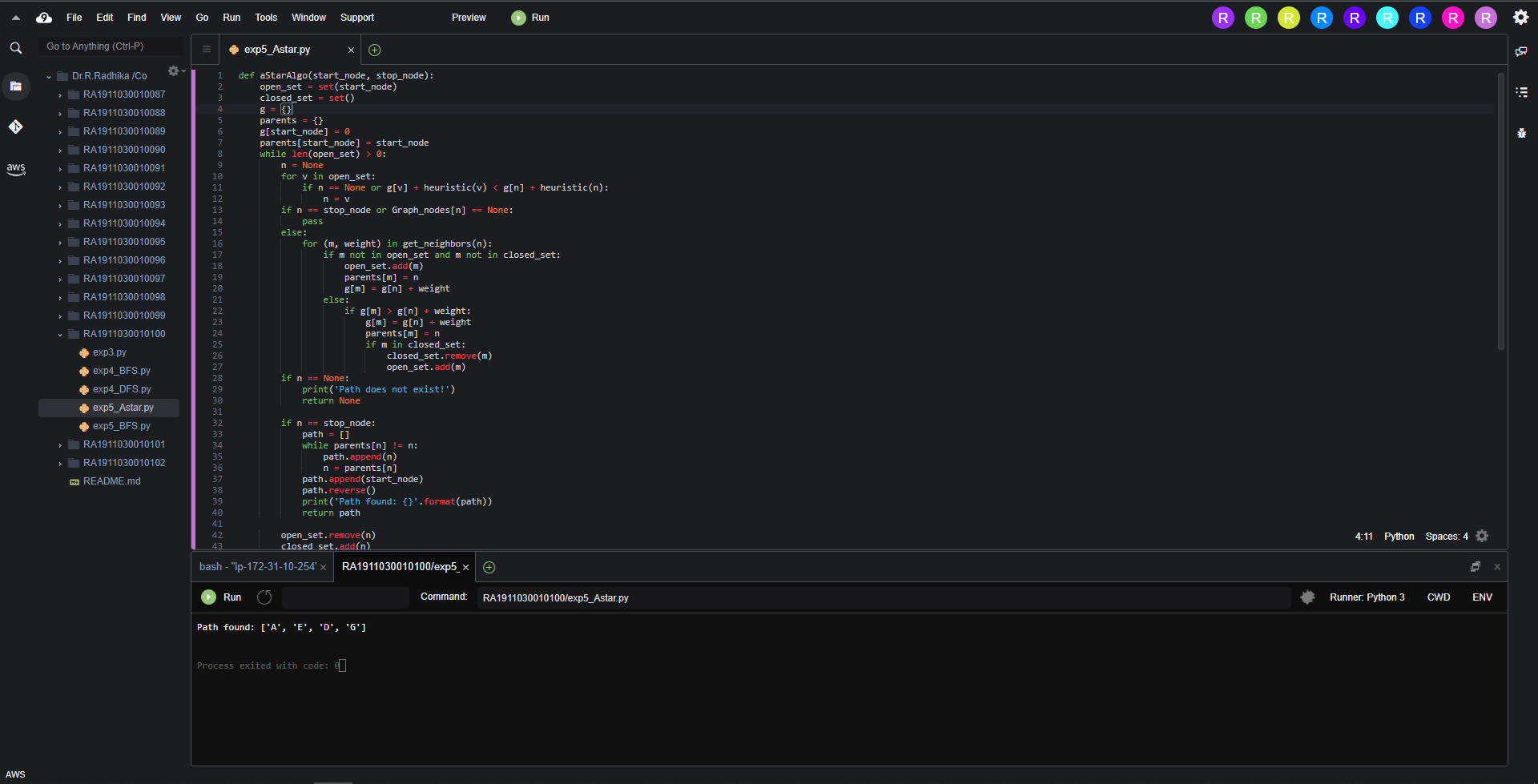
'E': [('A', 3), ('D', 6)],

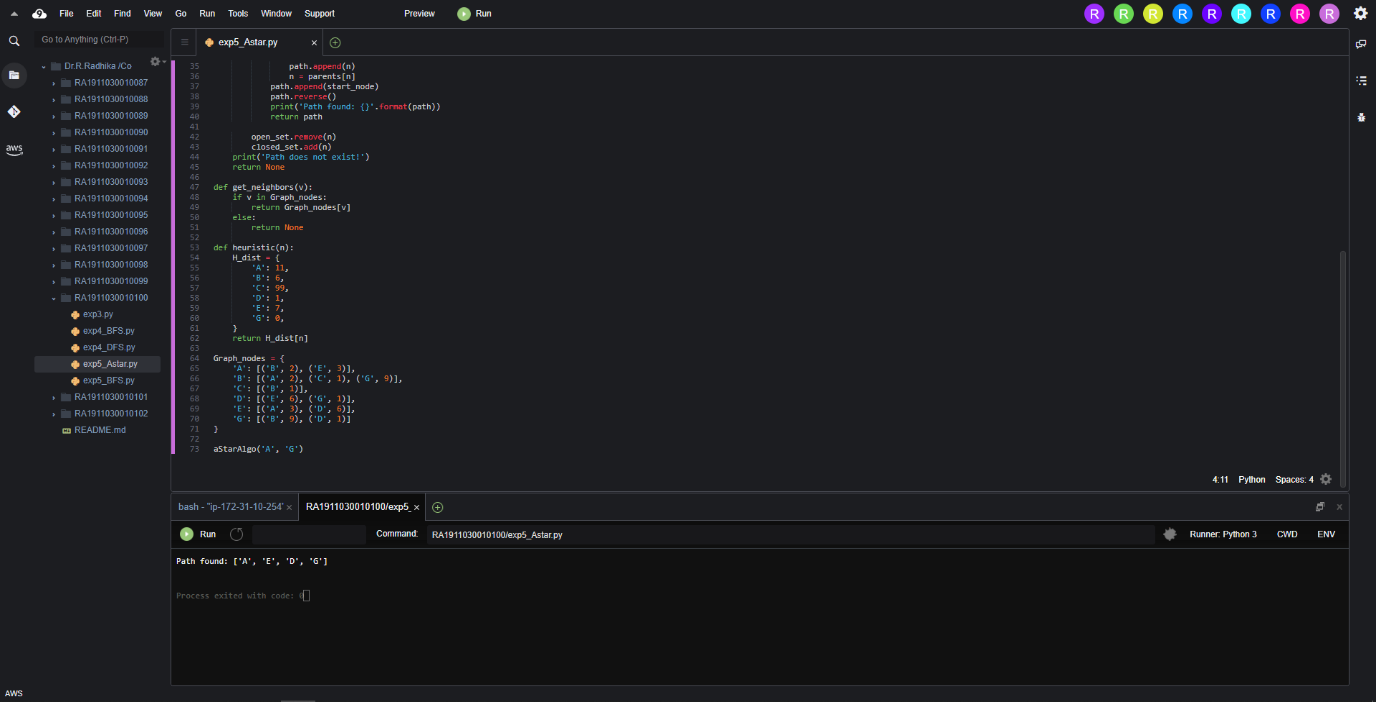
'G': [('B', 9), ('D', 1)]

}

aStarAlgo('A', 'G')

**Output:**





**Result:**

Hence A star algorithm was implemented and executed.