**AI Assignment 2**

**Code:**

def a\_star(start\_node, stop\_node):

open\_set = set([start\_node])

closed\_set = set()

g = {start\_node: 0}

parents = {start\_node: start\_node}

while open\_set:

current\_node = min(open\_set, key=lambda x: g[x] + heuristic(x))

if current\_node == stop\_node:

path = []

while current\_node != start\_node:

path.append(current\_node)

current\_node = parents[current\_node]

path.append(start\_node)

path.reverse()

print('Path found:', path)

return path

open\_set.remove(current\_node)

closed\_set.add(current\_node)

for neighbor, weight in get\_neighbors(current\_node):

if neighbor in closed\_set:

continue

tentative\_g = g[current\_node] + weight

if neighbor not in open\_set or tentative\_g < g[neighbor]:

g[neighbor] = tentative\_g

parents[neighbor] = current\_node

open\_set.add(neighbor)

print('Path does not exist!')

return None

def get\_neighbors(node):

return Graph\_nodes.get(node, [])

def heuristic(node):

H\_dist = {'A': 11, 'B': 6, 'C': 99, 'D': 1, 'E': 7, 'G': 0}

return H\_dist.get(node, float('inf'))

Graph\_nodes = {

'A': [('B', 2), ('E', 3)],

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

'C': [],

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

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

}

a\_star('A', 'G')

**Output:**

Path found: ['A', 'E', 'D', 'G']