**Experiment 3**

**AIM - Implement Breadth-First Search/UCS algorithm in Python.**

**Code –**

Tree\_BFS = {

"A": ['C', 'B'],

"B": ['E', 'D'],

"C": ['G', 'F'],

"D": ['H'],

"E": ['J', 'I'],

"F": ['L', 'K'],

"G": ['M'],

"H": [],

"I": [],

"J": [],

"K": [],

"L": [],

"M": []

}

def BFS(node,goal):

queue = []

queue.append(node)

print(queue)

while len(queue)!=0:

node = queue[0]

if queue[0]==goal:

return("Goal node found")

else:

queue.pop(0)

children = Tree\_BFS[node]

queue.extend(children)

print(queue)

return("Not exist After exploring all nodes")

def uniform\_cost\_search(goal, start):

global graph,cost

answer = []

queue = []

for i in range(len(goal)):

answer.append(10\*\*8)

queue.append([0, start])

visited = {}

count = 0

while (len(queue) > 0):

queue = sorted(queue)

p = queue[-1]

del queue[-1]

p[0] \*= -1

if (p[1] in goal):

index = goal.index(p[1])

if (answer[index] == 10\*\*8):

count += 1

if (answer[index] > p[0]):

answer[index] = p[0]

del queue[-1]

queue = sorted(queue)

if (count == len(goal)):

return answer

if (p[1] not in visited):

for i in range(len(graph[p[1]])):

queue.append( [(p[0] + cost[(p[1], graph[p[1]][i])])\* -1, graph[p[1]][i]])

visited[p[1]] = 1

return answer

graph,cost = [[] for i in range(8)],{}

graph[0].append(1)

graph[0].append(3)

graph[3].append(1)

graph[3].append(6)

graph[3].append(4)

graph[1].append(6)

graph[4].append(2)

graph[4].append(5)

graph[2].append(1)

graph[5].append(2)

graph[5].append(6)

graph[6].append(4)

cost[(0, 1)] = 2

cost[(0, 3)] = 5

cost[(1, 6)] = 1

cost[(3, 1)] = 5

cost[(3, 6)] = 6

cost[(3, 4)] = 2

cost[(2, 1)] = 4

cost[(4, 2)] = 4

cost[(4, 5)] = 3

cost[(5, 2)] = 6

cost[(5, 6)] = 3

cost[(6, 4)] = 7

goal = []

print("Niyati's Code for BFS & UCS")

print("The Tree structure is:{Parent:children}")

print(Tree\_BFS)

want\_to\_continue = 1

while want\_to\_continue == 1:

root\_node = input("Enter Root Node: ")

goal\_node = input("Enter Goal Node: ")

user\_inp = input("What algorithm to use? Press 1 for BFS, 2 for UCS: ")

stack = ['A']

if user\_inp == '1':

print(stack)

BFS(root\_node, goal\_node)

stack = ['A']

elif user\_inp == '2':

goal.append(int(goal\_node))

answer = uniform\_cost\_search(goal, int(root\_node))

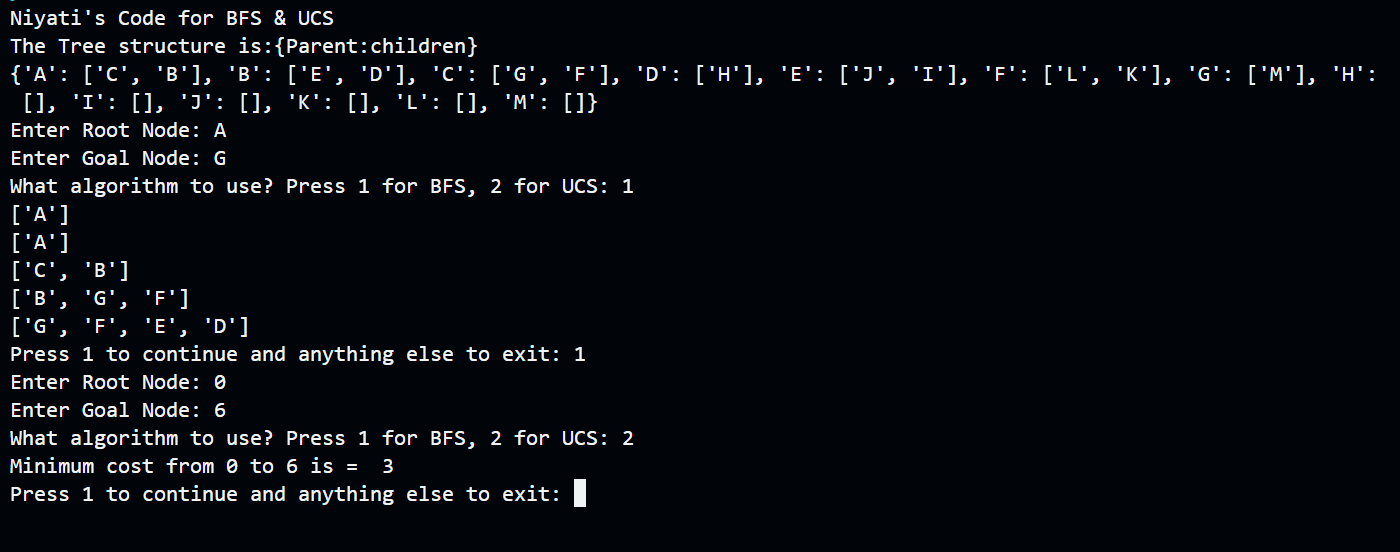
print("Minimum cost from 0 to 6 is = ",answer[0])

else:

print("Enter a valid number")

want\_to\_continue = int(input("Press 1 to continue and anything else to exit: "))

**Output –**

****