**BFS\_List Code :**

# -\*- coding: utf-8 -\*-

"""

Created on Mon Sep 19 12:26:17 2016

@author: jaysh

"""

# -\*- coding: utf-8 -\*-

"""

"""

# This is a Breadth First Search

# The problem of the program is stored in a list

# The program is always complete if the goal is present

# Variable path represents the path of the Breadth First Search

# Variable s represent the current state of the Breadth First Search

# This Program is implemented by a queue where the first node from the

#frontier is taken out and the new node is appended in the last of frontier

#intialise\_frontier = This function initialises the frontier node to the start node i.e in this case Arad

#choose\_node = This function chooses the first state from the frontier its like implementing the half of the queue the first node3 choosen becomes the current state

#update\_frontier = This fucntion updates frontier by inserting those nodes in the frontier which are present in g but they are not explored Basically froniter is the union of the list g and explored

#test\_goal = his function compares the current node with the goal node if the current node is the goal node than it sends True else it sends False

#expand\_node = #This fucntion Expands the current state It shows all the possibility that can be explored from the current state and it also shows the cost of that possibility

def graph\_search(start,goal,problem):

explored = [] #Represents the states which has been already explored

g = [] #Represents the nodes and one of the nodes will be explored

path = [] #Represents the path of the Breadth First Search

current\_state ='' #Represents the current state of the Breadth First Search

path\_cost=0 #calculates the cost of the path from start state to current state

flag=True

frontier=intialise\_frontier(start)

parent = []

print ('\n Initialisation value of the frontier = '+", ".join(frontier))

i=0

while frontier:

path.append(choose\_node(frontier))

frontier = frontier[1:]

length\_of\_path = len(path)

if(length\_of\_path > 1):

print ("\nThe parent node of current state "+path[-1]+" is "+path[-2])

print ("\n path value= "+", ".join(path))

current\_state= path[-1]

print ('\n The value of current state = '+current\_state)

print ('\n frontier bfore updation = '+", ".join(frontier))

explored.append(current\_state)

print ('\n explored node = '+", ".join(explored))

if(test\_goal(current\_state,goal) == False):

print ("\n The curent state is NOT the goal state ")

if(flag==True):

g,path\_cost,flag=expand\_node(g,problem,current\_state,path\_cost,goal,flag,parent)

update\_frontier(g,explored,frontier)

print ('\n frontier after updation = '+", ".join(frontier))

else:

print ("\n The curent state is NOT the goal state ")

break

g=[]

i = i+1

actual\_path,cost = backtracking(start,goal,parent)

actual\_path = actual\_path[::-1]

final\_explored = ', '.join(explored)

print("\n Final Output --------------------------------------------------------------")

print("\n The Final Explored states are :\n "+final\_explored)

print ("\n The Path from Arad to Bucharest using Breadth First Search : \n"+", ".join(actual\_path))

print("\n The cost from Arad to Bucharest using BFS : "+str(cost))

print("\n Final Output --------------------------------------------------------------")

def backtracking(start,goal,parent):

reverse\_path = [goal]

child = goal

cost = 0

while start not in reverse\_path:

for elements in parent:

if elements[1]==child:

reverse\_path.append(elements[0])

cost = cost + elements[2]

child=elements[0]

break

return reverse\_path,cost

#This function initialises the frontier node to the start node i.e in this case Arad

def intialise\_frontier(start):

frontier = [start]

return frontier

#This function chooses the first state from the frontier its like

#implementing the half of the queue

#The first node choosen becomes the current state

def choose\_node(frontier):

path = frontier[0]

return path

#This fucntion updates frontier by

#inserting those nodes in the frontier

#which are present in g but they are not explored

#Basically froniter is the union of the list g and explored

def update\_frontier(g, explored, frontier):

for element in g:

if element not in explored:

frontier.append(element)

return frontier

#This function compares the current node with the goal node

#if the current node is the goal node than it sends True else it sends False

def test\_goal(s,goal):

if (s==goal):

return True

return False

#This function Expands the current state

# It shows all the possibility that can be explored from the current state

# and it also shows the cost of that possibility

def expand\_node(g,problem,s,path\_cost,goal,flag,parent):

for place\_distance\_list in problem :

if place\_distance\_list[0] == s :

print ("\n The distance to be explored is :"+place\_distance\_list[0]+" to "+place\_distance\_list[1])

print ("The cost of the above distance to be explored is :"+str(place\_distance\_list[2]))

print (place\_distance\_list)

parent.append(place\_distance\_list)

print(parent)

g.append(place\_distance\_list[1])

if place\_distance\_list[0] == goal:

flag = False

path\_cost=path\_cost+place\_distance\_list[2]

return g,path\_cost,flag

problem = [ ['Arad','Zerind',75], ['Arad','Timisoara',118], ['Arad','Sibiu',140], ['Zerind','Oradea',71],

['Oradea','Sibiu',151], ['Timisoara','Lugoj',111], ['Lugoj','Mehadia',70], ['Mehadia','Drobeta',75],

['Drobeta','Craiova',120], ['Craiova','Rimnicu-Vilcea',146], ['Craiova','Pitesti',138],

['Sibiu','Fagaras',99], ['Sibiu','Rimnicu-Vilcea',80], ['Rimnicu-Vilcea','Pitesti',97],

['Fagaras','Bucharest',211], ['Pitesti','Bucharest',101],['Bucharest','Urziceni',85],

['Bucharest Giurgiu',90], ['Urziceni Vaslui',142], ['Vaslui Iasi',92], ['Iasi Neamt',87],

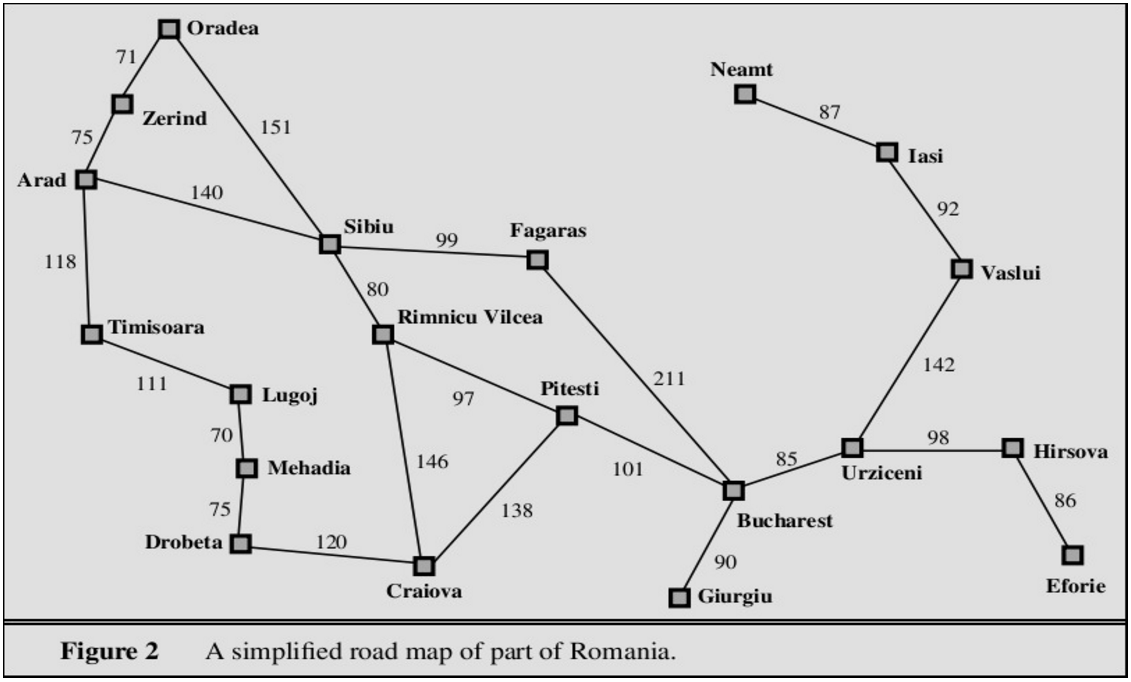
['Urziceni Hirsova',98], ['Hirsova Eforie',86] ]

print (problem)

start = "Arad"

goal = "Bucharest"

graph\_search(start,goal,problem)



**OUTPUT :**

Initialisation value of the frontier = Arad

path value= Arad

The value of current state = Arad

frontier bfore updation =

explored node = Arad

The curent state is NOT the goal state

The distance to be explored is :Arad to Zerind

The cost of the above distance to be explored is :75

['Arad', 'Zerind', 75]

[['Arad', 'Zerind', 75]]

The distance to be explored is :Arad to Timisoara

The cost of the above distance to be explored is :118

['Arad', 'Timisoara', 118]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118]]

The distance to be explored is :Arad to Sibiu

The cost of the above distance to be explored is :140

['Arad', 'Sibiu', 140]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118], ['Arad', 'Sibiu', 140]]

frontier after updation = Zerind, Timisoara, Sibiu

The parent node of current state Zerind is Arad

path value= Arad, Zerind

The value of current state = Zerind

frontier bfore updation = Timisoara, Sibiu

explored node = Arad, Zerind

The curent state is NOT the goal state

The distance to be explored is :Zerind to Oradea

The cost of the above distance to be explored is :71

['Zerind', 'Oradea', 71]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118], ['Arad', 'Sibiu', 140], ['Zerind', 'Oradea', 71]]

frontier after updation = Timisoara, Sibiu, Oradea

The parent node of current state Timisoara is Zerind

path value= Arad, Zerind, Timisoara

The value of current state = Timisoara

frontier bfore updation = Sibiu, Oradea

explored node = Arad, Zerind, Timisoara

The curent state is NOT the goal state

The distance to be explored is :Timisoara to Lugoj

The cost of the above distance to be explored is :111

['Timisoara', 'Lugoj', 111]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118], ['Arad', 'Sibiu', 140], ['Zerind', 'Oradea', 71], ['Timisoara', 'Lugoj', 111]]

frontier after updation = Sibiu, Oradea, Lugoj

The parent node of current state Sibiu is Timisoara

path value= Arad, Zerind, Timisoara, Sibiu

The value of current state = Sibiu

frontier bfore updation = Oradea, Lugoj

explored node = Arad, Zerind, Timisoara, Sibiu

The curent state is NOT the goal state

The distance to be explored is :Sibiu to Fagaras

The cost of the above distance to be explored is :99

['Sibiu', 'Fagaras', 99]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118], ['Arad', 'Sibiu', 140], ['Zerind', 'Oradea', 71], ['Timisoara', 'Lugoj', 111], ['Sibiu', 'Fagaras', 99]]

The distance to be explored is :Sibiu to Rimnicu-Vilcea

The cost of the above distance to be explored is :80

['Sibiu', 'Rimnicu-Vilcea', 80]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118], ['Arad', 'Sibiu', 140], ['Zerind', 'Oradea', 71], ['Timisoara', 'Lugoj', 111], ['Sibiu', 'Fagaras', 99], ['Sibiu', 'Rimnicu-Vilcea', 80]]

frontier after updation = Oradea, Lugoj, Fagaras, Rimnicu-Vilcea

The parent node of current state Oradea is Sibiu

path value= Arad, Zerind, Timisoara, Sibiu, Oradea

The value of current state = Oradea

frontier bfore updation = Lugoj, Fagaras, Rimnicu-Vilcea

explored node = Arad, Zerind, Timisoara, Sibiu, Oradea

The curent state is NOT the goal state

The distance to be explored is :Oradea to Sibiu

The cost of the above distance to be explored is :151

['Oradea', 'Sibiu', 151]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118], ['Arad', 'Sibiu', 140], ['Zerind', 'Oradea', 71], ['Timisoara', 'Lugoj', 111], ['Sibiu', 'Fagaras', 99], ['Sibiu', 'Rimnicu-Vilcea', 80], ['Oradea', 'Sibiu', 151]]

frontier after updation = Lugoj, Fagaras, Rimnicu-Vilcea

The parent node of current state Lugoj is Oradea

path value= Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj

The value of current state = Lugoj

frontier bfore updation = Fagaras, Rimnicu-Vilcea

explored node = Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj

The curent state is NOT the goal state

The distance to be explored is :Lugoj to Mehadia

The cost of the above distance to be explored is :70

['Lugoj', 'Mehadia', 70]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118], ['Arad', 'Sibiu', 140], ['Zerind', 'Oradea', 71], ['Timisoara', 'Lugoj', 111], ['Sibiu', 'Fagaras', 99], ['Sibiu', 'Rimnicu-Vilcea', 80], ['Oradea', 'Sibiu', 151], ['Lugoj', 'Mehadia', 70]]

frontier after updation = Fagaras, Rimnicu-Vilcea, Mehadia

The parent node of current state Fagaras is Lugoj

path value= Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj, Fagaras

The value of current state = Fagaras

frontier bfore updation = Rimnicu-Vilcea, Mehadia

explored node = Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj, Fagaras

The curent state is NOT the goal state

The distance to be explored is :Fagaras to Bucharest

The cost of the above distance to be explored is :211

['Fagaras', 'Bucharest', 211]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118], ['Arad', 'Sibiu', 140], ['Zerind', 'Oradea', 71], ['Timisoara', 'Lugoj', 111], ['Sibiu', 'Fagaras', 99], ['Sibiu', 'Rimnicu-Vilcea', 80], ['Oradea', 'Sibiu', 151], ['Lugoj', 'Mehadia', 70], ['Fagaras', 'Bucharest', 211]]

frontier after updation = Rimnicu-Vilcea, Mehadia, Bucharest

The parent node of current state Rimnicu-Vilcea is Fagaras

path value= Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj, Fagaras, Rimnicu-Vilcea

The value of current state = Rimnicu-Vilcea

frontier bfore updation = Mehadia, Bucharest

explored node = Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj, Fagaras, Rimnicu-Vilcea

The curent state is NOT the goal state

The distance to be explored is :Rimnicu-Vilcea to Pitesti

The cost of the above distance to be explored is :97

['Rimnicu-Vilcea', 'Pitesti', 97]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118], ['Arad', 'Sibiu', 140], ['Zerind', 'Oradea', 71], ['Timisoara', 'Lugoj', 111], ['Sibiu', 'Fagaras', 99], ['Sibiu', 'Rimnicu-Vilcea', 80], ['Oradea', 'Sibiu', 151], ['Lugoj', 'Mehadia', 70], ['Fagaras', 'Bucharest', 211], ['Rimnicu-Vilcea', 'Pitesti', 97]]

frontier after updation = Mehadia, Bucharest, Pitesti

The parent node of current state Mehadia is Rimnicu-Vilcea

path value= Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj, Fagaras, Rimnicu-Vilcea, Mehadia

The value of current state = Mehadia

frontier bfore updation = Bucharest, Pitesti

explored node = Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj, Fagaras, Rimnicu-Vilcea, Mehadia

The curent state is NOT the goal state

The distance to be explored is :Mehadia to Drobeta

The cost of the above distance to be explored is :75

['Mehadia', 'Drobeta', 75]

[['Arad', 'Zerind', 75], ['Arad', 'Timisoara', 118], ['Arad', 'Sibiu', 140], ['Zerind', 'Oradea', 71], ['Timisoara', 'Lugoj', 111], ['Sibiu', 'Fagaras', 99], ['Sibiu', 'Rimnicu-Vilcea', 80], ['Oradea', 'Sibiu', 151], ['Lugoj', 'Mehadia', 70], ['Fagaras', 'Bucharest', 211], ['Rimnicu-Vilcea', 'Pitesti', 97], ['Mehadia', 'Drobeta', 75]]

frontier after updation = Bucharest, Pitesti, Drobeta

The parent node of current state Bucharest is Mehadia

path value= Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj, Fagaras, Rimnicu-Vilcea, Mehadia, Bucharest

The value of current state = Bucharest

frontier bfore updation = Pitesti, Drobeta

explored node = Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj, Fagaras, Rimnicu-Vilcea, Mehadia, Bucharest

The curent state is NOT the goal state

Final Output --------------------------------------------------------------

The Final Explored states are :

Arad, Zerind, Timisoara, Sibiu, Oradea, Lugoj, Fagaras, Rimnicu-Vilcea, Mehadia, Bucharest

The Path from Arad to Bucharest using Breadth First Search :

Arad, Sibiu, Fagaras, Bucharest

The cost from Arad to Bucharest using BFS : 450

Final Output --------------------------------------------------------------