Practical 4

Aim: To implement RBFS Algorithm.

Code:

*import* queue *as* Q

dict\_hn = {"Arad": 336, "Bucharest": 0, "Craiova": 160, "Drobeta": 242, "Eforie": 161, "Fagaras": 176, "Giurgiu": 77, "Hirsova": 151, "lasi": 226, "Lugoj": 244, "Mehadia": 241,

           "Neamt": 234, "Oradea": 380, "Pitesti": 100, "Rimnicu": 193, "Sibiu": 253, "Timisoara": 329, "Urzicem": 80, "Vastu": 199, "Zerind": 374}

dict\_gn = dict(

    Arad=dict(Zerind=75, Timisoara=118, Sibiu=140),

    Bucharest=dict(Urziceni=85, Fagaras=211, Giurgiu=90, Pitesti=101),

    Craiova=dict(Drobeta=120, Rimnicu=146, Pitesti=138),

    Drobeta=dict(Mehadia=75, Craiova=120),

    Eforie=dict(Hirsova=86),

    Fagaras=dict(Sibiu=99, Bucharest=211),

    Giurgi=dict(Bucharest=90),

    Hirsova=dict(Urziceni=98, Eforie=86),

    lasi=dict(Vaslui=92, Neamt=87),

    Lugoj=dict(Timisoara=111, Mehadia=70),

    Mehadia=dict(Lugoj=70, Drobeta=75),

    Neamt=dict(lasi=87),

    Oradea=dict(Zerind=71, Sibiu=151),

    Pitesti=dict(Rimnicu=97, Craiova=138, Bucharest=101),

    Rimnicu=dict(Sibiu=80, Pitesti=97, Craiova=146),

    Sibiu=dict(Fagaras=99, Rimnicu=80, Oradea=151, Arad=140),

    Timisoara=dict(Arad=118, Lugoj=111),

    Urziceni=dict(Hirsova=98, Bucharest=85, Vaslui=142),

    Vaslui=dict(lasi=92, Urziceni=142),

    Zerind=dict(Arad=75, Oradea=71),

)

def get\_fn(citystr):

    cities = citystr.split(",")

    hn = 0

    gn = 0

    ctr = 0

*while* ctr != len(cities) - 1:

        gn = gn + dict\_gn[cities[ctr]][cities[ctr + 1]]

        ctr = ctr + 1

    hn = dict\_hn[cities[len(cities) - 1]]

*return* hn + gn

def expand(mycities, cityq, goal):

    tot, citystr = mycities

    cities = citystr.split(",")

    city2expand = cities[len(cities) - 1]

*if* city2expand == goal:

        ans = (

            "The Recursive Best First Search Path is "

            + citystr

            + " with the value as "

            + str(tot)

        )

*while* not cityq.empty():

            cityq.get()

*return* ans

    print("Expanded city--", city2expand)

    tempq = Q.PriorityQueue()

*for* cty *in* dict\_gn[city2expand]:

        tempq.put((get\_fn(citystr + "," + cty), citystr + "," + cty))

    print("First Best and Second Best Inserted in Priority Queue:")

    ctr = 1

*if* cityq.empty():

*while* not tempq.empty():

*if* ctr == 1 or ctr == 2:

                tempgn, tempcitystr = tempq.get()

                print("Inserting into CityQueue: ", tempgn, ",", tempcitystr)

                cityq.put((tempgn, tempcitystr))

                ctr = ctr + 1

*else*:

                tempq.get()

*else*:

        fn = 0

        citystring = ""

        fn = getSecondBest(cityq, fn, citystring)

*while* not tempq.empty():

*if* ctr == 1 or ctr == 2:

                tempgn, tempcitystr = tempq.get()

*if* tempgn > fn:

*if* ctr == 1:

                    print("Inserting into CityQueue:", tempgn, ",", citystr)

                    cityq.put((tempgn, citystr))

                ctr = 3

*continue*

*else*:

                tempq.get()

*while* not tempq.empty():

            tempq.get()

def getSecondBest(cityq, fn, citystring):

    fn, citystring = cityq.get()

    cityq.put((fn, citystring))

*return* fn

def main():

    start = "Arad"

    goal = "Bucharest"

    cityq = Q.PriorityQueue()

    cityq.put((get\_fn(start), start))

*while* not cityq.empty():

        mycities = cityq.get()

        ans = expand(mycities, cityq, goal)

    print("########", ans)

main()

Output:

