**Practical No :1**

**Aim:** Breadth First Search & Iterative Depth First Search

• Implement the Breadth First Search algorithm to solve a given problem

**Roll No: Date: Sign:**

**Source Code :**

#BFS Algorithm

import queue as Q

from RMP import dict\_gn

start='Arad'

goal='Vaslui'

result=''

def BFS(city, cityq, visitedq):

    global result

    if city==start:

        result=result+' '+city

    for eachcity in dict\_gn[city].keys():

        if eachcity==goal:

            result=result+' '+eachcity

            return

        if eachcity not in cityq.queue and eachcity not in visitedq.queue:

            cityq.put(eachcity)

            result=result+' '+eachcity

    visitedq.put(city)

    BFS(cityq.get(),cityq,visitedq)

def main():

    cityq=Q.Queue()

    visitedq=Q.Queue()

    BFS(start, cityq, visitedq)

    print("BFS Traversal from ",start," to ",goal," is: ")

    print(result)

main()

#RMP.py Romania Map

dict\_hn={'Arad':336,'Bucharest':0,'Craiova':160,'Drobeta':242,'Eforie':161,

         'Fagaras':176,'Giurgiu':77,'Hirsova':151,'Iasi':226,'Lugoj':244,

         'Mehadia':241,'Neamt':234,'Oradea':380,'Pitesti':100,'Rimnicu':193,

         'Sibiu':253,'Timisoara':329,'Urziceni':80,'Vaslui':199,'Zerind':374}

dict\_gn=dict(

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

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

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

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

Eforie=dict(Hirsova=86),

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

Giurgiu=dict(Bucharest=90),

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

Iasi=dict(Neamt=87,Vaslui=92),

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

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

Neamt=dict(Iasi=87),

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

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

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

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

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

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

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

Zerind=dict(Oradea=71,Arad=75)

)

**Output:**

PS D:\TYCS\AI> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe d:/TYCS/AI/BFSAlgoP1.py

BFS Traversal from Arad to Vaslui is:

Arad Zerind Timisoara Sibiu Oradea Lugoj Rimnicu Fagaras Mehadia Pitesti Craiova Bucharest Drobeta Urziceni Giurgiu Hirsova Vaslui

**Practical No :2**

**Aim:** Breadth First Search & Iterative Depth First Search

• Implement the Iterative Depth First Search algorithm to solve the same problem

**Roll No: Date: Sign:**

**Source Code :**

#AIM: Implement IDDFS(Iterative Deepening Depth-First Search).

import queue as Q

from RMP import dict\_gn

start='Arad'

goal='Vaslui'

result=''

def DLS(city, visitedstack, startlimit, endlimit):

    global result

    found=0

    result=result+city+' '

    visitedstack.append(city)

    if city==goal:

        return 1

    if startlimit==endlimit:

        return 0

    for eachcity in dict\_gn[city].keys():

        if eachcity not in visitedstack:

            found=DLS(eachcity, visitedstack, startlimit+1, endlimit)

            if found:

                return found

def IDDFS(city, visitedstack, endlimit):

    global result

    for i in range(0, endlimit):

        print("Searching at Limit: ",i)

        found=DLS(city, visitedstack, 0, i)

        if found:

            print("Found")

            break

        else:

            print("Not Found! ")

            print(result)

            print("-----")

            result=' '

            visitedstack=[]

def main():

    visitedstack=[]

    IDDFS(start, visitedstack, 9)

    print("IDDFS Traversal from ",start," to ", goal," is: ")

    print(result)

main()

**Output:**

PS D:\TYCS\AI> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe d:/TYCS/AI/iddfsAlgorithmP2.py

Searching at Limit: 0

Not Found!

Arad

-----

Searching at Limit: 1

Not Found!

Arad Zerind Timisoara Sibiu

-----

Searching at Limit: 2

Not Found!

Arad Zerind Oradea Timisoara Lugoj Sibiu Rimnicu Fagaras

-----

Searching at Limit: 3

Not Found!

Arad Zerind Oradea Sibiu Timisoara Lugoj Mehadia

-----

Searching at Limit: 4

Not Found!

Arad Zerind Oradea Sibiu Rimnicu Fagaras Timisoara Lugoj Mehadia Drobeta

-----

Searching at Limit: 5

Not Found!

Arad Zerind Oradea Sibiu Rimnicu Pitesti Craiova Fagaras Bucharest Timisoara Lugoj Mehadia Drobeta

-----

Searching at Limit: 6

Not Found!

Arad Zerind Oradea Sibiu Rimnicu Pitesti Bucharest Craiova Fagaras Timisoara Lugoj Mehadia Drobeta

-----

Searching at Limit: 7

Not Found!

Arad Zerind Oradea Sibiu Rimnicu Pitesti Bucharest Urziceni Giurgiu Fagaras Craiova Drobeta Timisoara Lugoj Mehadia

-----

Searching at Limit: 8

Found

IDDFS Traversal from Arad to Vaslui is:

Arad Zerind Oradea Sibiu Rimnicu Pitesti Bucharest Urziceni Hirsova Vaslui

**Practical No :3**

**Aim:** A\* Search and Recursive Best-First Search

• Implement the A\* Search algorithm for solving a pathfinding problem

**Roll No: Date: Sign:**

**Source Code :**

#AIM: Implement A\* search.

import queue as Q

from RMP import dict\_gn

from RMP import dict\_hn

start='Arad'

goal='Pitesti'

result=''

def get\_fn(citystr):

    cities=citystr.split(" , ")

    hn=gn=0

    for ctr in range(0, len(cities)-1):

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

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

    return(hn+gn)

def expand(cityq):

    global result

    tot, citystr, thiscity=cityq.get()

    if thiscity==goal:

        result=citystr+" : : "+str(tot)

        return

    for cty in dict\_gn[thiscity]:

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

    expand(cityq)

def main():

    cityq=Q.PriorityQueue()

    thiscity=start

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

    expand(cityq)

    print("The A\* path with the total is: ")

    print(result)

main()

**Output:**

PS D:\TYCS\AI> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe d:/TYCS/AI/AStarP3.py

The A\* path with the total is:

Arad , Sibiu , Rimnicu , Pitesti : : 417

**Practical No :4**

**Aim:** A\* Search and Recursive Best-First Search

• Implement the Recursive Best-First Search algorithm for the same problem.

**Roll No: Date: Sign:**

**Source Code :**

#AIM: RBFS(Recursive Breadth First Search)

import queue as Q

from RMP import dict\_gn

from RMP import dict\_hn

start='Arad'

goal='Bucharest'

result=''

def get\_fn(citystr):

    cities=citystr.split(',')

    hn=gn=0

    for ctr in range(0,len(cities)-1):

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

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

    return(hn+gn)

def printout(cityq):

    for i in range(0,cityq.qsize()):

        print(cityq.queue[i])

def expand(cityq):

    global result

    tot,citystr,thiscity=cityq.get()

    nexttot=999

    if not cityq.empty():

        nexttot,nextcitystr,nextthiscity=cityq.queue[0]

    if thiscity==goal and tot<nexttot:

        result=citystr+'::'+str(tot)

        return

    print("Expanded city------------------------------",thiscity)

    print("Second best f(n)------------------------------",nexttot)

    tempq=Q.PriorityQueue()

    for cty in dict\_gn[thiscity]:

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

    for ctr in range(1,3):

        ctrtot,ctrcitystr,ctrthiscity=tempq.get()

        if ctrtot<nexttot:

            cityq.put((ctrtot,ctrcitystr,ctrthiscity))

        else:

            cityq.put((ctrtot,citystr,thiscity))

            break

    printout(cityq)

    expand(cityq)

def main():

    cityq=Q.PriorityQueue()

    thiscity=start

    cityq.put((999,"NA","NA"))

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

    expand(cityq)

    print(result)

main()

**Output:**

PS D:\TYCS\AI> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe d:/TYCS/AI/RBFSAlgorithmP4.py

Expanded city------------------------------ Arad

Second best f(n)------------------------------ 999

(393, 'Arad,Sibiu', 'Sibiu')

(999, 'NA', 'NA')

(447, 'Arad,Timisoara', 'Timisoara')

Expanded city------------------------------ Sibiu

Second best f(n)------------------------------ 447

(413, 'Arad,Sibiu,Rimnicu', 'Rimnicu')

(415, 'Arad,Sibiu,Fagaras', 'Fagaras')

(447, 'Arad,Timisoara', 'Timisoara')

(999, 'NA', 'NA')

Expanded city------------------------------ Rimnicu

Second best f(n)------------------------------ 415

(415, 'Arad,Sibiu,Fagaras', 'Fagaras')

(417, 'Arad,Sibiu,Rimnicu', 'Rimnicu')

(447, 'Arad,Timisoara', 'Timisoara')

(999, 'NA', 'NA')

Expanded city------------------------------ Fagaras

Second best f(n)------------------------------ 417

(417, 'Arad,Sibiu,Rimnicu', 'Rimnicu')

(450, 'Arad,Sibiu,Fagaras', 'Fagaras')

(447, 'Arad,Timisoara', 'Timisoara')

(999, 'NA', 'NA')

Expanded city------------------------------ Rimnicu

Second best f(n)------------------------------ 447

(417, 'Arad,Sibiu,Rimnicu,Pitesti', 'Pitesti')

(447, 'Arad,Timisoara', 'Timisoara')

(999, 'NA', 'NA')

(450, 'Arad,Sibiu,Fagaras', 'Fagaras')

(526, 'Arad,Sibiu,Rimnicu', 'Rimnicu')

Expanded city------------------------------ Pitesti

Second best f(n)------------------------------ 447

(418, 'Arad,Sibiu,Rimnicu,Pitesti,Bucharest', 'Bucharest')

(447, 'Arad,Timisoara', 'Timisoara')

(607, 'Arad,Sibiu,Rimnicu,Pitesti', 'Pitesti')

(526, 'Arad,Sibiu,Rimnicu', 'Rimnicu')

(450, 'Arad,Sibiu,Fagaras', 'Fagaras')

(999, 'NA', 'NA')

Arad,Sibiu,Rimnicu,Pitesti,Bucharest::418