AI Practical 1

Aim: Implementing Breadth First Search using Romanian map problem.

DATA:

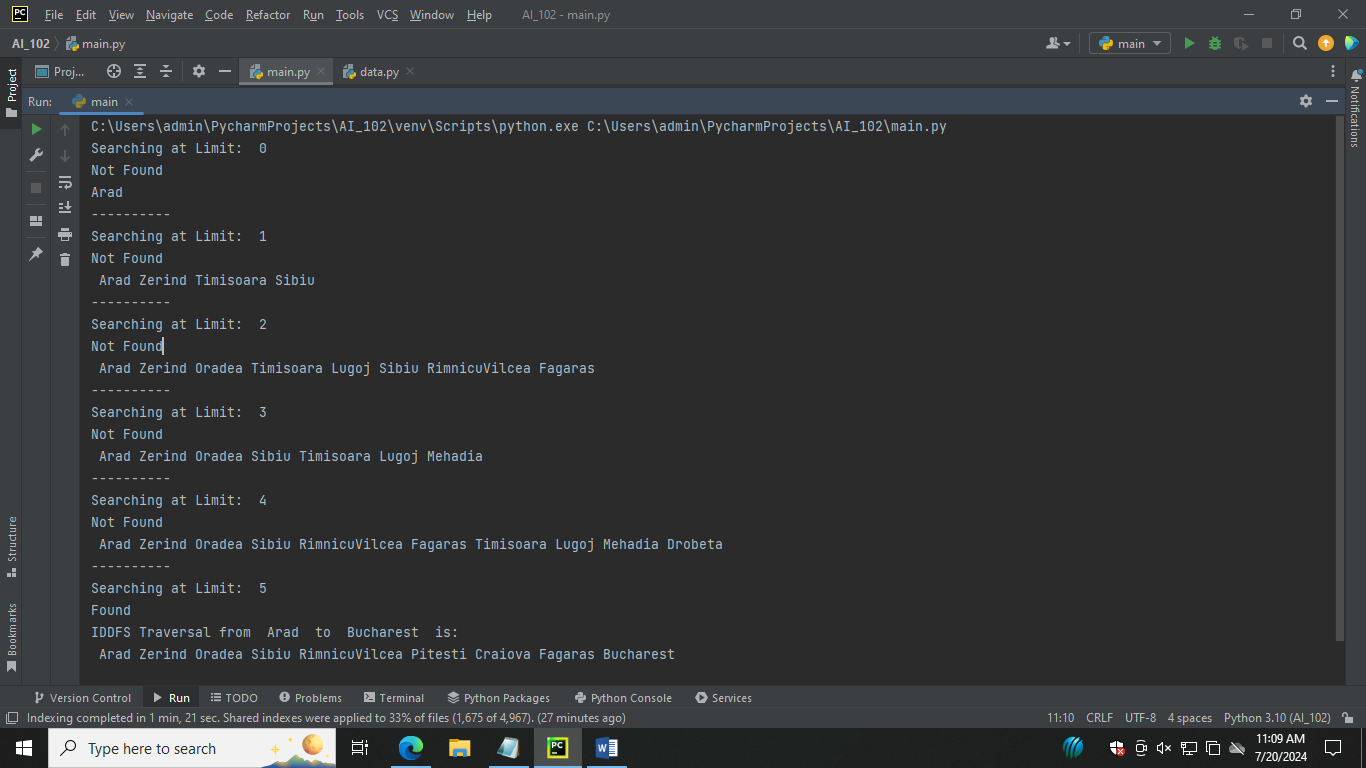
Save this in data.py file

dict\_hn={'Arad':336, 'Bucharest':0,'Craiova':160,'Drobeta':242, 'Eforie':161,'Fagaras':178 , 'Giurgiu':77,'Hirsova':151,  
 'Lasi':226,'Lugoj':244, 'Mehadia':241, 'Neamt':234, 'Oradea':380, 'Pitesti':98,'RimnicuVilcea':193, 'Sibiu':253,  
 'Timisoara':329,'Urziceni':80,'Valsui':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,RimnicuVilcea=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),  
 Lasi=dict(Neamt=87,Valsui=92),  
 Lugoj=dict(Mehadia=70,Timisoara=111),  
 Mehadia=dict(Lugoj=70,Drobeta=75),  
 Neamt=dict(Lasi=87),  
 Oradea=dict(Zerind=71,Sibiu=151),  
 Pitesti=dict(RimnicuVilcea=97,Bucharest=101,Craiova=138),  
 RimnicuVilcea=dict(Sibiu=80,Pitesti=97,Craiova=146),  
 Sibiu=dict(RimnicuVilcea=80,Fagaras=99,Arad=140,Oradea=151),  
 Timisoara=dict(Lugoj=111,Arad=118),  
 Urziceni=dict(Bucharest=85,Hirsova=98,Vaslui=142),  
 Valsui=dict(Lasi=92,Urziceni=142),  
 Zerind=dict(Oradea=71,Arad=75))

Make a main.py file and paste this code:

from data import dict\_gn  
  
start = 'Arad'  
goal = 'Bucharest'  
result = ''  
  
  
def DLS(city, visited\_stack, start\_limit, end\_limit):  
 global result  
 found = 0  
 result = result + city + ' '  
 visited\_stack.append(city)  
 if city == goal:  
 return 1  
 if start\_limit == end\_limit:  
 return 0  
 for each\_city in dict\_gn[city].keys():  
 if each\_city not in visited\_stack:  
 found = DLS(each\_city, visited\_stack, start\_limit + 1, end\_limit)  
 if found:  
 return found  
  
  
def IDDFS(city, visited\_stack, end\_limit):  
 global result  
 for i in range(0, end\_limit):  
 print("Searching at Limit: ", i)  
 found = DLS(city, visited\_stack, 0, i)  
 if found:  
 print("Found")  
 break  
 else:  
 print("Not Found")  
 print(result)  
 print("----------")  
 result = " "  
 visited\_stack = []  
  
def main():  
 visited\_stack = []  
 IDDFS(start, visited\_stack, 9)  
 print("IDDFS Traversal from ", start, " to ", goal, " is: ")  
 print(result)  
  
main()

Output:



Make another python file and paste this code:

Aim: A star algorithm

import queue as Q  
from data import dict\_gn, 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 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()