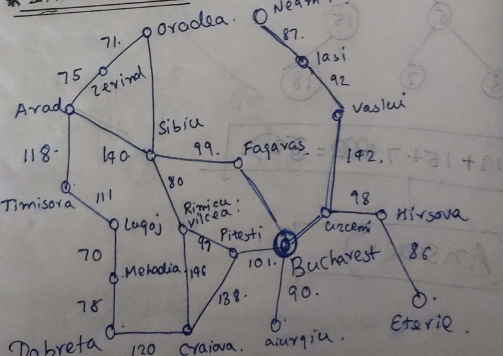
Expt 5:- Best First Search & A*Alg for Real World Problems

* **Problem formulation:-** A Romanian map is given we have to find the shortest path with least path cost using A* and Best First Search Problem.

* **Path cost:-** The sum of the cost from the initial to the final node.

* **Operators:-** The node ~~costs~~ and the paths (costs) are the operators in this problem

* **Initial State:-**



Romanian Map

Nodes.

Straight line Distance to Bucharest

Arad	366
Bucharest [Goal]	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

* Algorithm:-

(:) Best First Search:-

* Create a priority queue. pqueue.
* Insert 'start' in pqueue: pqueue. insert (search)
* Delete all elements of pqueue one by one
* If the element is goal. Exit.
* Else, traverse neighbours. and mark the node examined.

(:) A* Star:-

* Create two lists - Open and Closed.
* Initialize the open list
* Initialize the closed list.
* Put the starting node on the open list. (you can leave its f at zero).
* When the open list is not empty:-
(:) Find the node with the least f on the open list; call it 'q'.
(:) pop q off the open list.
(:) Generate q's 8 successors. and set their parents to q.

(c) for each successor.

(i). if successor is the goal, stop search.

(ii) else, compute both g & h for successor.

successor.g = q.g + distance between successor and q.

successor.h = distance from goal to successor.

successor.f = successor.g + successor.h.

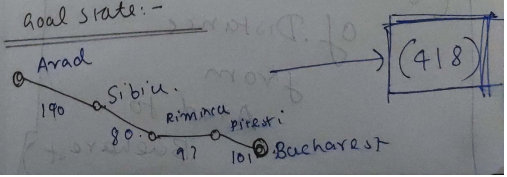
(iii). if a node with the same position as successor is in the open list which has a lower f than successor, skip this successor, skip this successor

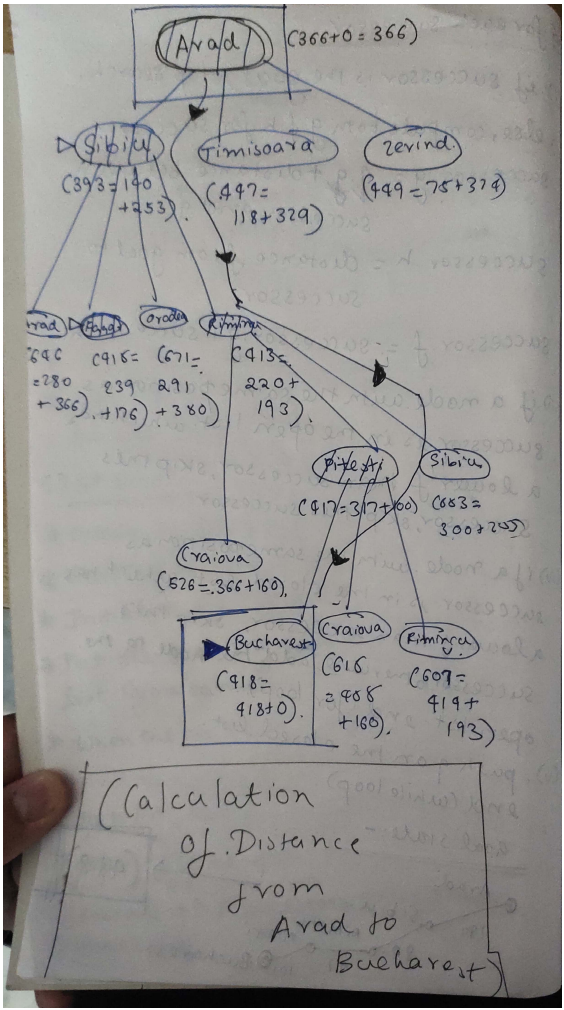
(iv) if a node with the same position as successor is in the closed list which has a lower f than successor, skip this successor otherwise, add the node to the open list. end (for loop).

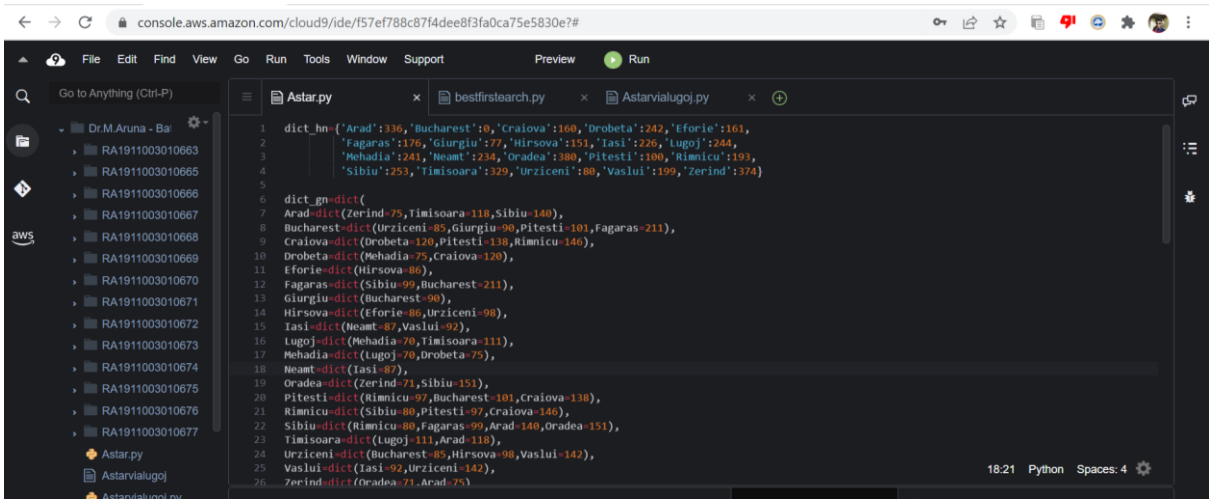
(v). push q on the closed list.

end (while loop).

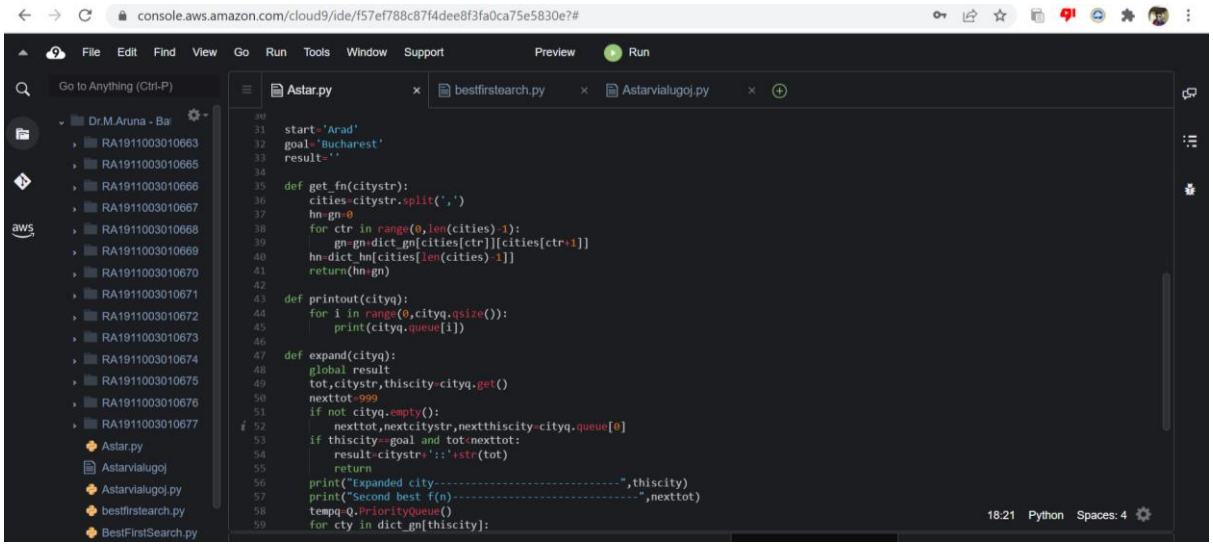
goal state:-



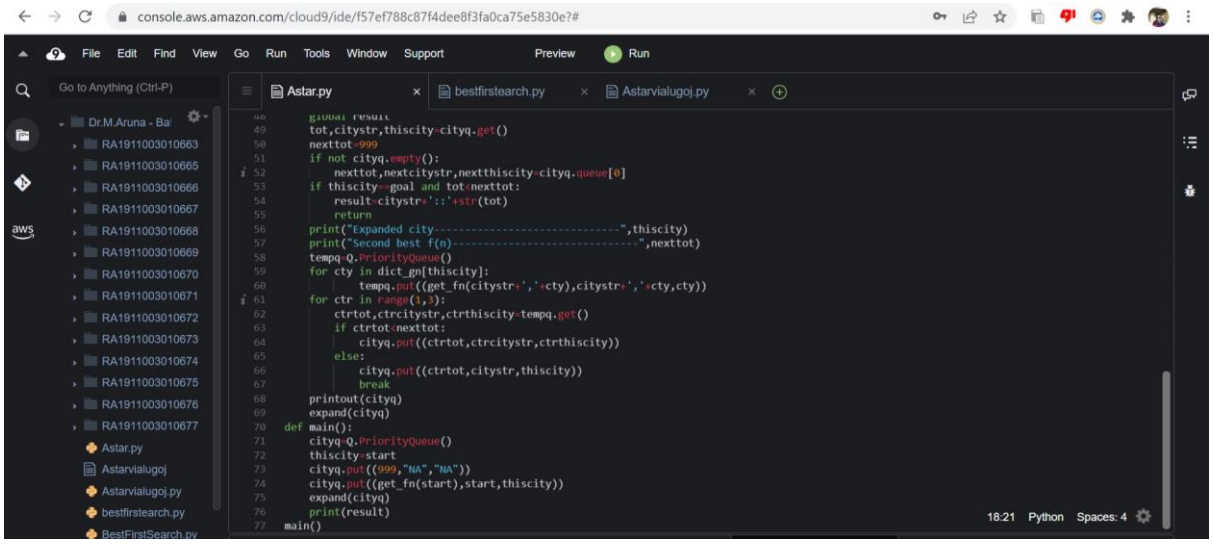

Arad
(366+0=366)
Sibiu
Timisoara
Zerind
(393=140+253)
(447=118+329)
(449=75+374)
Arad
Fagaras
Oradea
Rimnicu
(646=280+366)
(415=239+176)
(671=291+380)
(413=220+193)
Pitesti
Sibiu
(417=317+100)
(553=300+253)
Craiova
(526=366+160)
Bucharest
Craiova
Rimnicu
(418=418+0)
(615=455+160)
(607=414+193)
(Calculation of Distance from Arad to Bucharest)



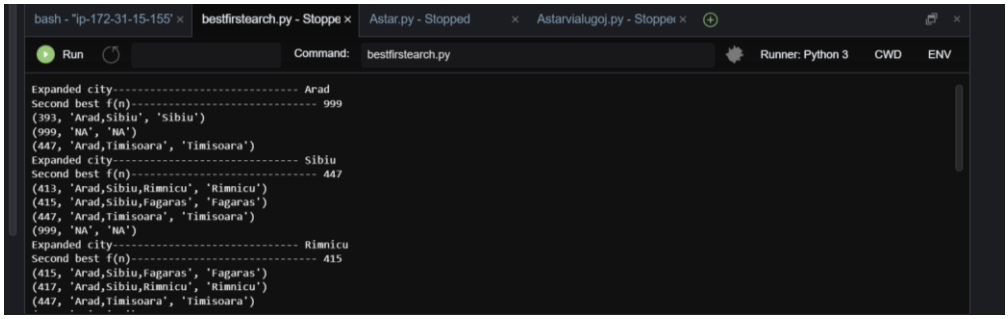

```
dict_hn={'Arad':336,'Bucharest':0,'Craiova':160,'Drobeta':242,'Eforie':161,
         'Fagaras':176,'Giurgiu':77,'Hirsova':151,'Iasi':226,'Lugoj':244,
         'Mehadia':241,'Neamt':234,'Oradea':380,'Pitesti':100,'Rimnicu':193,
         'Sibiu':253,'Timisoara':329,'Urziceni':80,'Vaslui':199,'Zerind':374}

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)
```

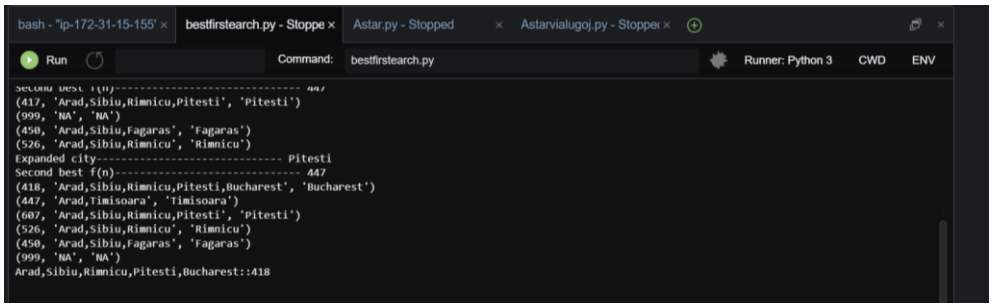


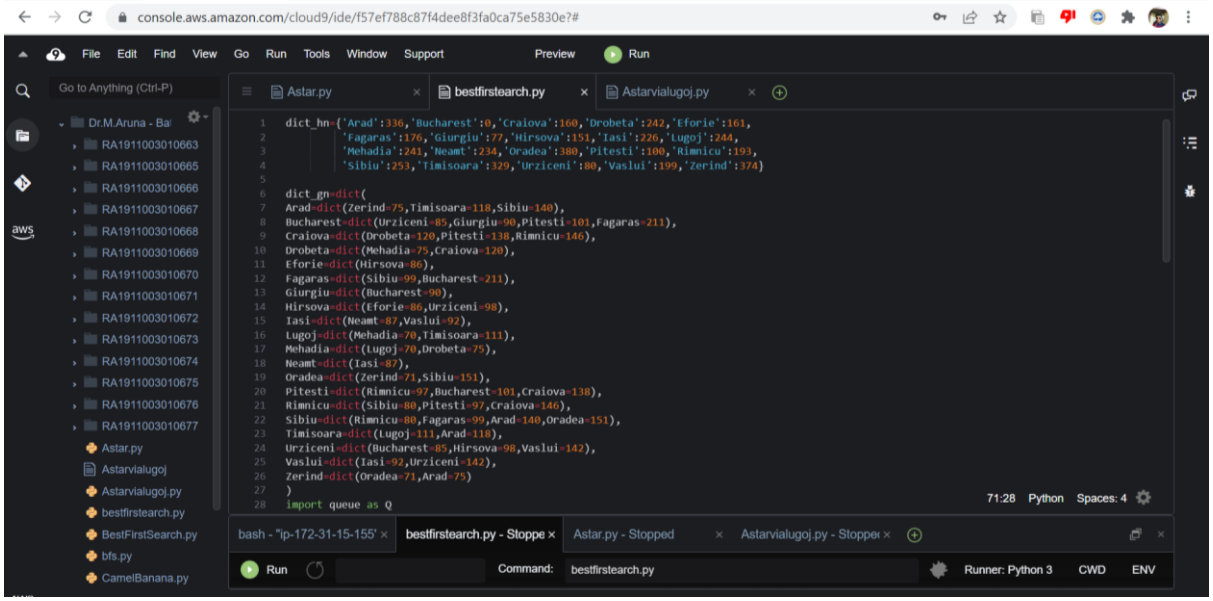

```
start='Arad'
goal='Bucharest'
result=''

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]:
```

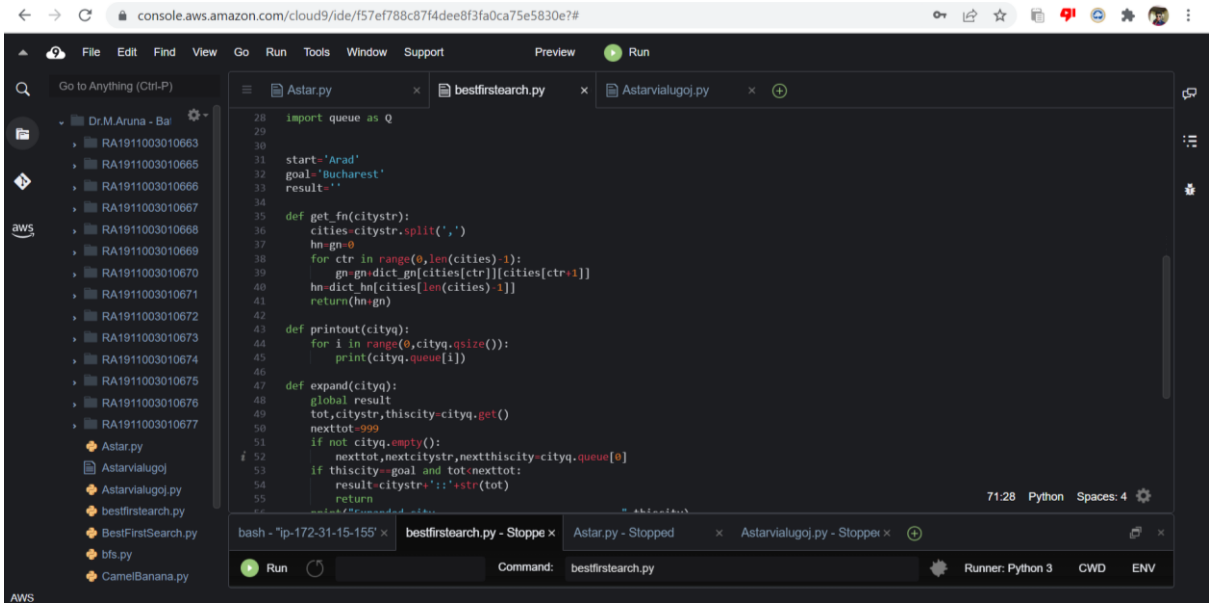


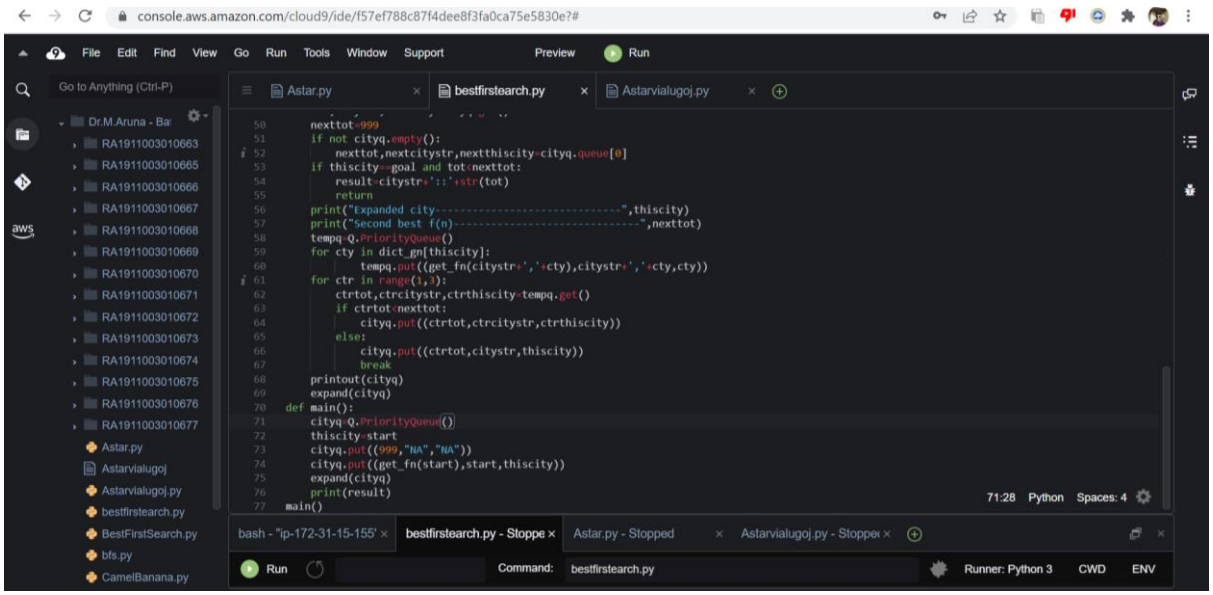

```
    tot,citystr,thiscity=cityq.get()
    nexttot=999
    if not cityq.empty():
        nexttot,nextcitystr,nextthiscity=cityq.queue[0]
    if thiscity==goal and tot<nexttot:
        result=citystr+'::'+str(tot)
        return
    print("Expanded city------------------------------",thiscity)
    print("Second 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()
```

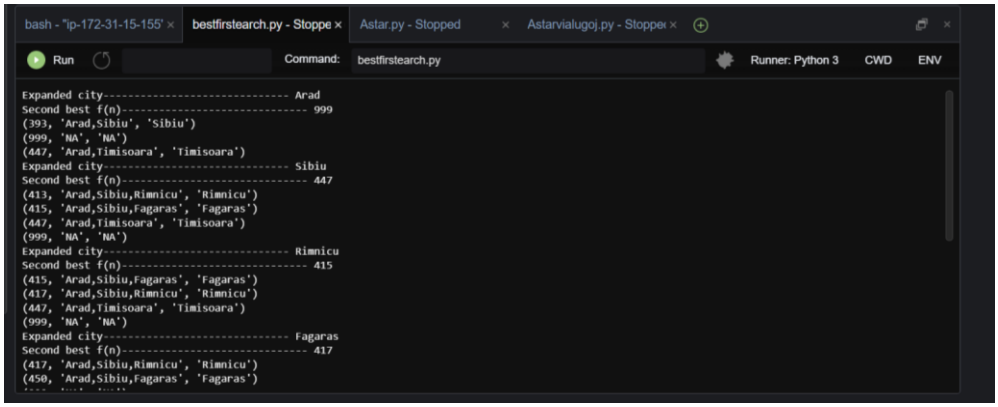

bash - "ip-172-31-15-155"
bestfirstearch.py - Stoppe
Astar.py - Stopped
Astarvialugoj.py - Stoppe
Run
Command: bestfirstearch.py
Runner: Python 3
CWD
ENV
Expanded city---------- Arad
Second best f(n)---------- 999
(393, 'Arad,Sibiu', 'Sibiu')
(999, 'NA', 'NA')
(447, 'Arad,Timisoara', 'Timisoara')
Expanded 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')

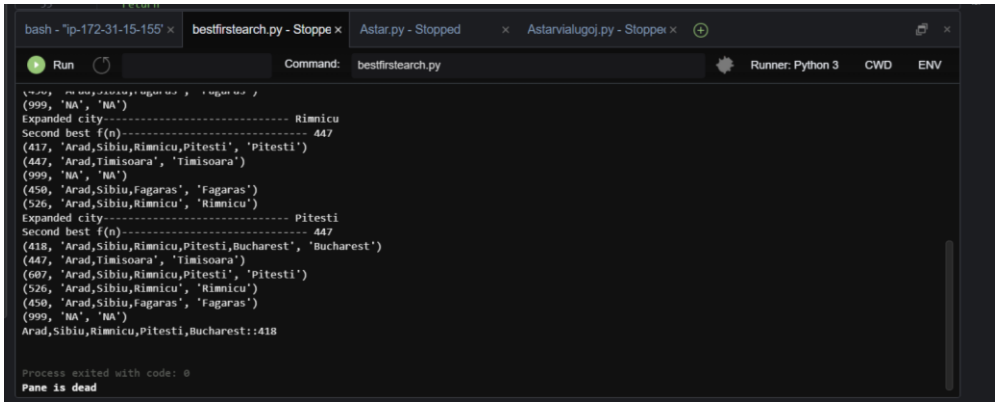

bash - "ip-172-31-15-155"
bestfirstearch.py - Stoppe
Astar.py - Stopped
Astarvialugoj.py - Stoppe
Run
Command: bestfirstearch.py
Runner: Python 3
CWD
ENV
(417, 'Arad,Sibiu,Rimnicu,Pitesti', 'Pitesti')
(999, 'NA', 'NA')
(450, 'Arad,Sibiu,Fagaras', 'Fagaras')
(526, 'Arad,Sibiu,Rimnicu', 'Rimnicu')
Expanded 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


console.aws.amazon.com/cloud9/ide/f57ef788c87f4dee8f3fa0ca75e5830e?#
File Edit Find View Go Run Tools Window Support
Preview
Run
Go to Anything (Ctrl-P)
Astar.py
bestfirstearch.py
Astarvialugoj.py
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)
)
import queue as Q
71:28 Python Spaces: 4
Dr.M.Aruna - Ba
RA1911003010663
RA1911003010665
RA1911003010666
RA1911003010667
RA1911003010668
RA1911003010669
RA1911003010670
RA1911003010671
RA1911003010672
RA1911003010673
RA1911003010674
RA1911003010675
RA1911003010676
RA1911003010677
Astar.py
Astarvialugoj
Astarvialugoj.py
bestfirstearch.py
BestFirstSearch.py
bfs.py
CamelBanana.py
bash - "ip-172-31-15-155"
bestfirstearch.py - Stoppe
Astar.py - Stopped
Astarvialugoj.py - Stoppe
Run
Command: bestfirstearch.py
Runner: Python 3
CWD
ENV


console.aws.amazon.com/cloud9/ide/f57ef788c87f4dee8f3fa0ca75e5830e?#
File Edit Find View Go Run Tools Window Support Preview Run
Go to Anything (Ctrl-P)
Astar.py
bestfirstearch.py
Astarvialugoj.py
Dr.M.Aruna - Bai
RA1911003010663
RA1911003010665
RA1911003010666
RA1911003010667
RA1911003010668
RA1911003010669
RA1911003010670
RA1911003010671
RA1911003010672
RA1911003010673
RA1911003010674
RA1911003010675
RA1911003010676
RA1911003010677
Astar.py
Astarvialugoj
Astarvialugoj.py
bestfirstearch.py
BestFirstSearch.py
bfs.py
CamelBanana.py
import queue as Q
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
71:28 Python Spaces: 4
bash - "ip-172-31-15-155" bestfirstearch.py - Stoppe Astar.py - Stopped Astarvialugoj.py - Stoppe
Run Command: bestfirstearch.py Runner: Python 3 CWD ENV
AWS


console.aws.amazon.com/cloud9/ide/f57ef788c87f4dee8f3fa0ca75e5830e?#
File Edit Find View Go Run Tools Window Support Preview Run
Go to Anything (Ctrl-P)
Astar.py
bestfirstearch.py
Astarvialugoj.py
Dr.M.Aruna - Bai
RA1911003010663
RA1911003010665
RA1911003010666
RA1911003010667
RA1911003010668
RA1911003010669
RA1911003010670
RA1911003010671
RA1911003010672
RA1911003010673
RA1911003010674
RA1911003010675
RA1911003010676
RA1911003010677
Astar.py
Astarvialugoj
Astarvialugoj.py
bestfirstearch.py
BestFirstSearch.py
bfs.py
CamelBanana.py
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()
71:28 Python Spaces: 4
bash - "ip-172-31-15-155" bestfirstearch.py - Stoppe Astar.py - Stopped Astarvialugoj.py - Stoppe
Run Command: bestfirstearch.py Runner: Python 3 CWD ENV


bash - "ip-172-31-15-155" bestfirstearch.py - Stoppe Astar.py - Stopped Astarvialugoj.py - Stoppe
Run Command: bestfirstearch.py Runner: Python 3 CWD ENV
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')



Result:-

A star and Best First Search was successfully implemented on the Romanian Map Problem