Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ19ITL504 **DATE:22/10/22**
COURSE NAME: Artificial Intelligence Laboratory **CLASS: TY-IT**

EXPERIMENT NO.04

CO/LO: Formulate the problem as a state space and select appropriate technique from blind, heuristic or adversarial search to generate the solution.

AIM / OBJECTIVE: To implement informed search techniques on a given problem. (Greedy BFS & A*)

DESCRIPTION OF EXPERIMENT:
We need to implement Best First Search and A* Search and study them based on Time Complexity, Space Complexity, Optimality and Completeness.

Algorithm astar(graph,h):

1. Append a touple of start state and its heuristic(h) to queue
2. While front of queue doesn't contain goal state do:
    1) Dequeue front
    2) Calculate the fx value for all next node as fx = gx + h
    3) Sort queue based on fx value
    4) Remove cycles
    5) Remove overlapping paths

Algorithm bestFirst(graph,h):

1. Append start state to queue
2. If path contains goal node:
    Success
   Else if leaf node reached with no goal node encounter:
    Terminate search goal not found
   Else:
    Find the neighbor with minimum heuristic
    Add this neighbor to traversal path
    Traverse bestFirst(neighbours of the last added node)

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Explanation/Solutions(Design):

Code:

A*:

```
#graph={"s":[("b",3),("c",4)] , "b":[("c",1),("d",2),("g",3),("s",3)] ,
"c":[("e",1),("f",3),("g",4),("s",4),("b",1)] , "d":[("g",1),("b",2)] , "e":[("c",1)],
"f":[("c",3)], "g":[("b",3),("c",4),("d",1)]}
#h={"s":11,"b":2,"c":4,"d":6,"e":4,"f":1,"g":0}

graph={}
h={}
n=int(input("enter the number of states : "))
for i in range(0,n):
    state=input("enter the state : ")
    huristic=int(input("enter the heurixtic of current state : "))
    h[state]=huristic
    n_line=input("enter the neighbours,distance of neighbour : ")
    neighbours=n_line.split()
    ni=[]
    for i in neighbours:
        ele=i.split(",")
        ni.append((ele[0],int(ele[1])))
    graph[state]=ni


q=[]
def dq(q):
    top=q[0]
    del q[0]
    return top[0]

def calculatefx(top,h,graph):
    f=0
    for i in range(1,len(top),1):
        n1=graph[top[i-1]]
        for j in n1:
            if(top[i]==j[0]):
                f=f+j[1]

    return f+h[top[-1]]

def removecycle(q):
    cycli=[]
    for i in range(0,len(q)):
        if(len(q[i][0])!=len(set(q[i][0]))):
            print("removing cycle " ,q[i])
            cycli.append(q[i])
    for i in cycli:
        del q[q.index(i)]
    print("--> ",q)

def removeoverlap(q):
    olapi=[]
    for i in range(0,len(q)-1):
        for j in range(i+1,len(q)):
            if(q[i][0][0]==q[j][0][0] and q[i][0][-1]==q[j][0][-1]):
                print("removing overlap " ,q[j])
                olapi.append(q[j])
    for i in olapi:
```

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

```
        del q[q.index(i)]
    print("--> ",q)

def traverse(graph,h):
    q.append(("s",h["s"]))
    #print(q)
    #print(q[0][0])

    while(True):
        top=dq(q)
        #print("top ",top)
        n=graph[top[-1]]
        for i in n:
            f=calculatefx(top+i[0],h,graph)
            q.append((top+i[0],f))
        q.sort(key = lambda x: x[1])
        print("--> ",q)
        removecycle(q)
        removeoverlap(q)
        if(q[0][0][-1]=="g"):
            return q[0]


print(traverse(graph,h))
```

best first search:

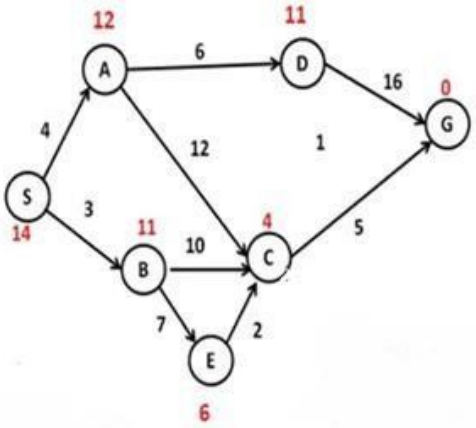
```
'''
graph={"s":["b","c"] , "b":["c","d","g","s"] , "c":["e","f","g","s","b"] , "d":["g","b"] ,
"e":["c"], "f":["c"], "g":["b","c","d"]}
h={"s":11,"b":2,"c":4,"d":6,"e":4,"f":1,"g":0}
'''
graph={}
h={}
n=int(input("enter the number of states : "))
for i in range(0,n):
    state=input("enter the state : ")
    huristic=int(input("enter the heurixtic of current state : "))
    h[state]=huristic
    n_line=input("enter the neighbours: ")
    neighbours=n_line.split()
    graph[state]=neighbours


def traversal(path,graph,h):
    if(path[-1]=="g"):
        print("search complete goal reached : ",path)
        return
    elif len(graph[path[-1]])==0:
        print("search terminated at :",path)
        return
    else:
        min=(h[graph[path[-1]][0]],graph[path[-1]][0])
        for i in graph[path[-1]]:
            if(h[i]<min[0]):
                min=(h[i],i)
        return traversal(path+min[1],graph,h)
```

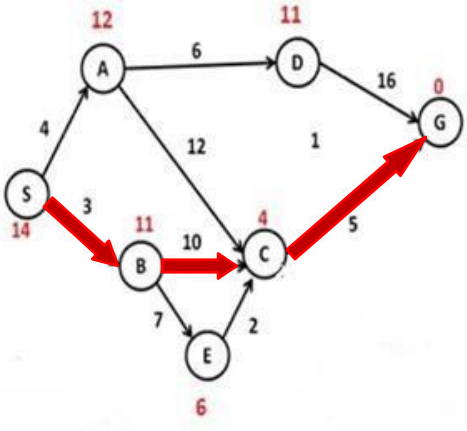
Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

```
traversal("s",graph,h)
```

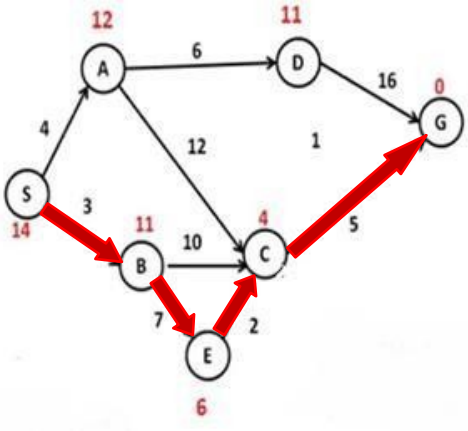
traversal paths:

example





Best first search



A*

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Output:

Best first search:

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\bestfirsrt.py"
enter the number of states : 7
enter the state : s
enter the heurixtic of current state : 14
enter the neighbours: a b
enter the state : a
enter the heurixtic of current state : 12
enter the neighbours: s d c
enter the state : b
enter the heurixtic of current state : 11
enter the neighbours: s c e
enter the state : c
enter the heurixtic of current state : 4
enter the neighbours: a b e g
enter the state : d
enter the heurixtic of current state : 11
enter the neighbours: a g
enter the state : e
enter the heurixtic of current state : 6
enter the neighbours: b c
enter the state : g
enter the heurixtic of current state : 0
enter the neighbours: d c
search complete goal reached :  sbcg
PS C:\Users\SHREE RAM\Desktop\ai>
```

A*:

```
PS C:\Users\SHREE RAM\Desktop\ai> python -u "c:\Users\SHREE RAM\Desktop\ai\astar.py"
enter the number of states : 7
enter the state : s
enter the heurixtic of current state : 14
enter the neighbours,distance of neighbour : a,4 b,3
enter the state : a
enter the heurixtic of current state : 12
enter the neighbours,distance of neighbour : s,4 d,6 c,12
enter the state : b
enter the heurixtic of current state : 11
enter the neighbours,distance of neighbour : s,3 c,10 e,7
enter the state : c
enter the heurixtic of current state : 4
enter the neighbours,distance of neighbour : b,10 a,12 e,2 g,5
enter the state : d
enter the heurixtic of current state : 11
enter the neighbours,distance of neighbour : a,6 g,16
enter the state : e
enter the heurixtic of current state : 6
enter the neighbours,distance of neighbour : b,7 c,2
enter the state : g
enter the heurixtic of current state : 0
enter the neighbours,distance of neighbour : d,16 c,5
-->  [('sb', 14), ('sa', 16)]
-->  [('sb', 14), ('sa', 16)]
-->  [('sb', 14), ('sa', 16)]
-->  [('sa', 16), ('sbe', 16), ('sbc', 17), ('sbs', 20)]
removing cycle  ('sbs', 20)
-->  [('sa', 16), ('sbe', 16), ('sbc', 17)]
-->  [('sa', 16), ('sbe', 16), ('sbc', 17)]
-->  [('sbe', 16), ('sbc', 17), ('sac', 20), ('sad', 21), ('sas', 22)]
removing cycle  ('sas', 22)
-->  [('sbe', 16), ('sbc', 17), ('sac', 20), ('sad', 21)]
removing overlap  ('sac', 20)
-->  [('sbe', 16), ('sbc', 17), ('sad', 21)]
-->  [('sbec', 16), ('sbc', 17), ('sad', 21), ('sbeb', 28)]
removing cycle  ('sbeb', 28)
```

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

```
removing cycle  ('sbeb', 28)
-->  [('sbec', 16), ('sbc', 17), ('sad', 21)]
removing overlap  ('sbc', 17)
-->  [('sbec', 16), ('sad', 21)]
-->  [('sbecg', 17), ('sbece', 20), ('sad', 21), ('sbecb', 33), ('sbeca', 36)]
removing cycle  ('sbece', 20)
removing cycle  ('sbecb', 33)
-->  [('sbecg', 17), ('sad', 21), ('sbeca', 36)]
-->  [('sbecg', 17), ('sad', 21), ('sbeca', 36)]
('sbecg', 17)
PS C:\Users\SHREE RAM\Desktop\ai>
```

comparison for Greedy best first search & A*:

Parameters	**Best first search**	**A***
completeness	No	Yes
Time complexity	O(b^d)	O(b^d)
Space complexity	O(b^d)	O(b^d)
optimality	No	Yes

CONCLUSION:

Hence, we have successfully implemented A* and Best First Search