Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

CSE4108: Artificial Intelligence Lab

Name: Rakib Hossain Rifat

ID: 16.02.04.099

Group: B2

Term Assignment # **02**

*A * Search*

Date of Submission: 13/09/2020

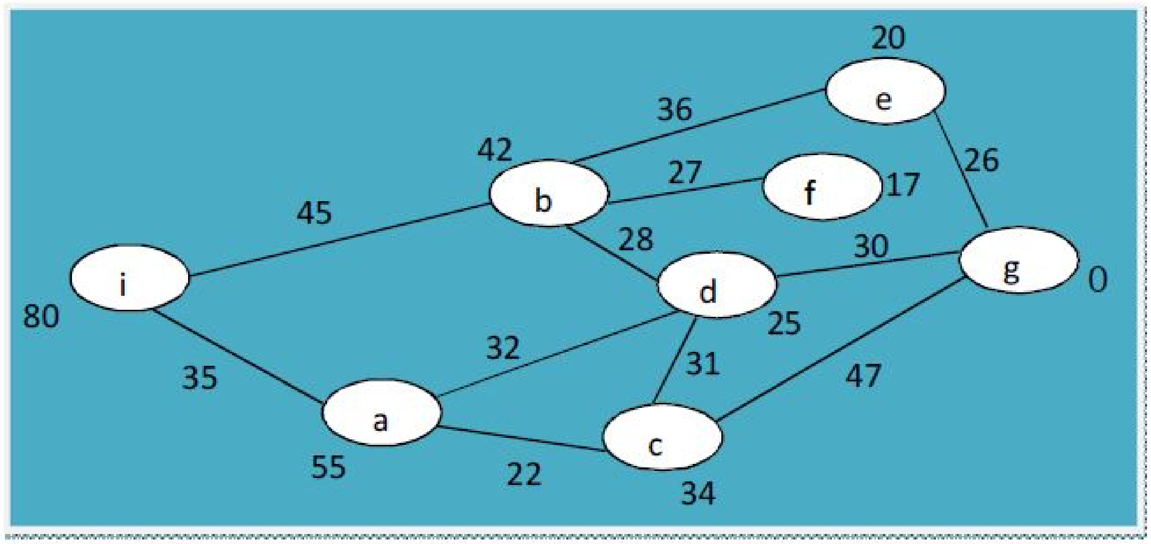
QUESTION-1:

Implement A* search algorithms in Python.

ANSWER:

- It is similar to GBFS. However, it has some other features.
- The index for all nodes is maintained. All Node index and parent index are also added to PQ.
- The evaluation function is f(n) = g(n) + h(n), where g(n) = an actual path cost from initial node to node n, and h(n) = estimated cost of the cheapest path from n to the goal.
- If a path is there then the process generates all the neighbors repeatedly and adds in PQ.
- The optimal solution is considered and sub-optimal ones are avoided.

INPUT GRAPH:



CODE:

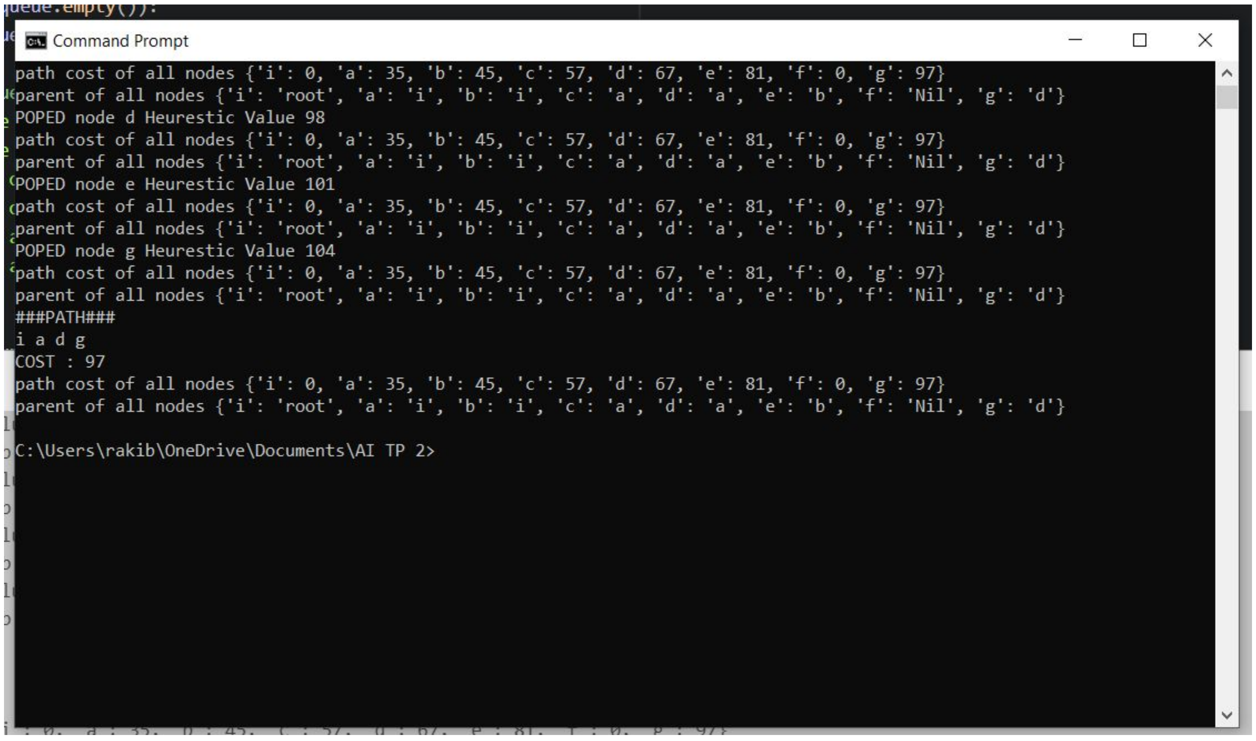
```
class NodePriority(object):
   def __init__(self, name,h_value):
      self.name = name
      self.h_value = h_value
      return

   def __lt__(self, other):
      return self.h_value < other.h_value
```

```
import queue as q
import sModule as pri_queue

neighbour = [('i', 'a', 35), ('i', 'b', 45), ('a', 'c', 22),
            ('a', 'd', 32), ('b', 'd', 28),
            ('b', 'e', 36),  ('b', 'f', 27),
            ('c', 'd', 31), ('c', 'g', 47),
            ('d', 'g', 30),  ('e', 'g', 26)]

heu_fn = [('i', 80), ('a', 55), ('b', 42), ('c', 34), ('d', 25), ('e', 20), ('f', 17), ('g', 0)]

visited={}
parent={}
a_cost={}
next_node={}
path=[]
for i in range(0,len(heu_fn)):
   next_node[heu_fn[i][0]]=False
   visited[heu_fn[i][0]]=False
   parent[heu_fn[i][0]]='Nil'
   a_cost[heu_fn[i][0]]=0
next_node[heu_fn[0][0]]=True

for i in neighbour:

   (m,n,l) = i
   for k in neighbour :
      if n ==k[0] and k!=i:
         next_node[n]=True
```

```
priority_queue = q.PriorityQueue()

s = str(input('Enter start node:'))
g = str(input('Enter goal node:'))

for i in range(0,len(heu_fn)):
   if heu_fn[i][0]==s:
      priority_queue.put(pri_queue.NodePriority(heu_fn[i][0],heu_fn[i][1]))

parent[s]="root"
a_cost[s]=0
f1=open("output.txt", "w")
while not (priority_queue.empty()):
   v = priority_queue.get()
   node = v.name
   visited[node]=True
   print("POPED node {} Heurestic Value {}".format(v.name,v.h_value),file=f1)
   print("POPED node {} Heurestic Value {}".format(v.name,v.h_value))
   print("path cost of all nodes {}".format(a_cost),file=f1)
   print("path cost of all nodes {}".format(a_cost))
   print("parent of all nodes {}".format(parent),file=f1)
   print("parent of all nodes {}".format(parent))


   for i in neighbour:
      if i[0]==node:

         next_v=i[1]
         if next_node[next_v]==True or next_v==g:

            if a_cost[next_v]==0:
               a_cost[next_v]=a_cost[i[0]]+i[2]
               parent[next_v]=i[0]
               for k in range(0,len(heu_fn)):
                  if heu_fn[k][0]==next_v:

priority_queue.put(pri_queue.NodePriority(next_v,a_cost[next_v]+heu_fn[k][1]))

            elif (a_cost[next_v]!=0) and (a_cost[next_v]>a_cost[i[0]]+i[2]):
               a_cost[next_v]=a_cost[i[0]]+i[2]
               parent[next_v]=i[0]
               for k in range(0,len(heu_fn)):
                  if heu_fn[k][0]==next_v:

priority_queue.put(pri_queue.NodePriority(next_v,a_cost[next_v]+heu_fn[k][1]))
```

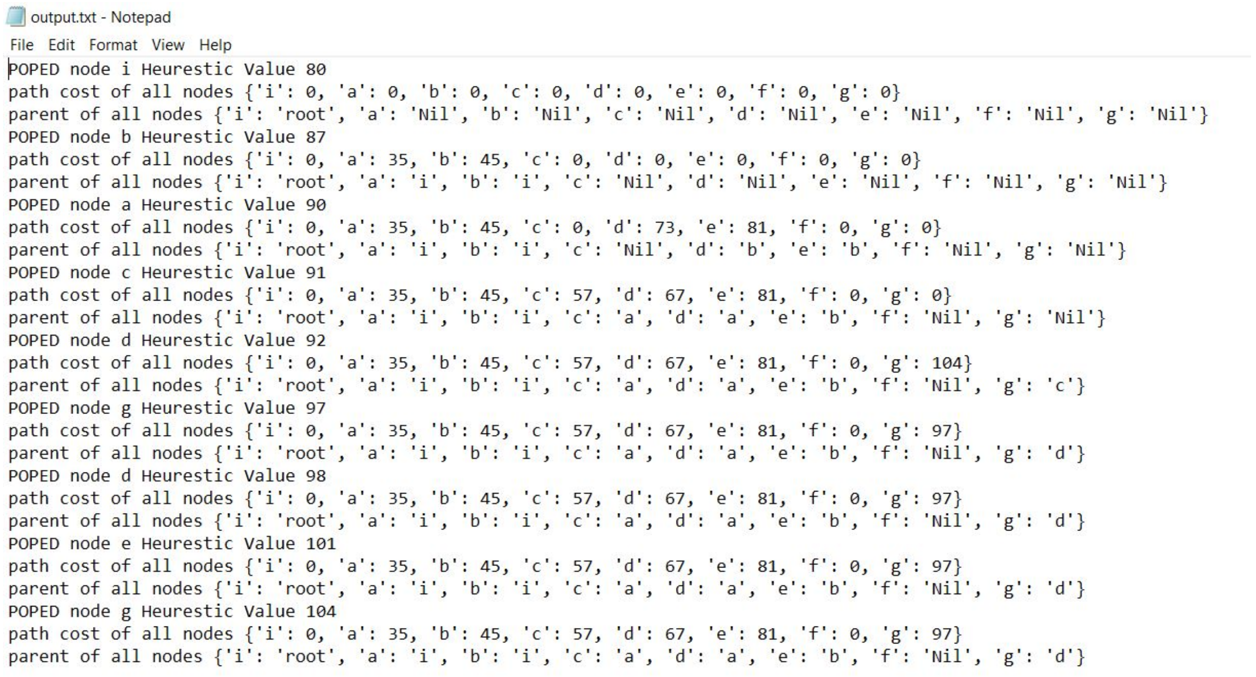
```
f1.close
def value_return(key):
   return parent[key]

r=g
print("###PATH###")
path.append(r)
while(r!='root'):
   r=value_return(r)
   if(r!='root'):
      path.append(r)

path.reverse()
for i in path:
   print(i,end=' ')
print("\nCOST : {}".format(a_cost[g]))

print("path cost of all nodes {}".format(a_cost))

print("parent of all nodes {}".format(parent))
```

OUTPUT:



```
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 97}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'd'}
POPED node d Heurestic Value 98
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 97}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'd'}
POPED node e Heurestic Value 101
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 97}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'd'}
POPED node g Heurestic Value 104
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 97}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'd'}
###PATH###
i a d g
COST : 97
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 97}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'd'}

C:\Users\rakib\OneDrive\Documents\AI TP 2>
```

Saved in File:



```
POPED node i Heurestic Value 80
path cost of all nodes {'i': 0, 'a': 0, 'b': 0, 'c': 0, 'd': 0, 'e': 0, 'f': 0, 'g': 0}
parent of all nodes {'i': 'root', 'a': 'Nil', 'b': 'Nil', 'c': 'Nil', 'd': 'Nil', 'e': 'Nil', 'f': 'Nil', 'g': 'Nil'}
POPED node b Heurestic Value 87
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 0, 'd': 0, 'e': 0, 'f': 0, 'g': 0}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'Nil', 'd': 'Nil', 'e': 'Nil', 'f': 'Nil', 'g': 'Nil'}
POPED node a Heurestic Value 90
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 0, 'd': 73, 'e': 81, 'f': 0, 'g': 0}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'Nil', 'd': 'b', 'e': 'b', 'f': 'Nil', 'g': 'Nil'}
POPED node c Heurestic Value 91
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 0}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'Nil'}
POPED node d Heurestic Value 92
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 104}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'c'}
POPED node g Heurestic Value 97
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 97}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'd'}
POPED node d Heurestic Value 98
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 97}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'd'}
POPED node e Heurestic Value 101
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 97}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'd'}
POPED node g Heurestic Value 104
path cost of all nodes {'i': 0, 'a': 35, 'b': 45, 'c': 57, 'd': 67, 'e': 81, 'f': 0, 'g': 97}
parent of all nodes {'i': 'root', 'a': 'i', 'b': 'i', 'c': 'a', 'd': 'a', 'e': 'b', 'f': 'Nil', 'g': 'd'}
```