

# Ahsanullah University of Science and Technology

Department of Computer Science and Engineering



CSE4108

Artificial Intelligence

Submitted By:

Mayisha Farzana

16.02.04.028

Date of Submission: **19 August, 2020**

**Ques: 1) Implement in generic ways (as multi-modular and interactive systems) the Greedy Best-First algorithms in Python.**

**Description:** In greedy best first search, we have set the heuristic values. After that we made the graph where we use list in where the first value is of the node and the second value is the cost of the node for traverse. Then we import Priority Queue Function. In empty function it will return true if the Queue is empty. In push we will add the value. In pop, we will pop some value if there any value otherwise we will return null if there is no value.

In greedy best first we will fix the source and destination node. At first there is no path. First we will push the source value. We will visit the adjacent node and will mark the node if visited. After that, we calculate possible path.

**Python Code:**

```
inf = 10000000
h = [55, 42, 34, 25, 20, 17, 0, inf, 80]
```

```
graph = [ [(2, 22), (3, 32), (8, 35)],
  [(8, 45), (4, 36), (5, 27), (3, 28)],
  [(0, 22), (3, 31), (6, 47)],
  [(0, 32), (2, 31), (1, 28), (6, 30)],
  [(1, 36), (6, 26)],
  [(1, 27)],
  [(2, 47), (3, 30), (4, 26)],
  [],
  [(0, 35), (1, 45)] ]
```

```
from queue import PriorityQueue
Q = PriorityQueue()
```

```
def empty():
    return Q.empty()
```

```
def push(val):
    Q.put(val)
```

```
def pop():
    if empty():
        return None
    return Q.get()
```

```
def greedy_best_first():
    source = 8 # i
    goal = 6 # g
```

```

parent = [-1] * len(graph)
vis = [False] * len(graph)

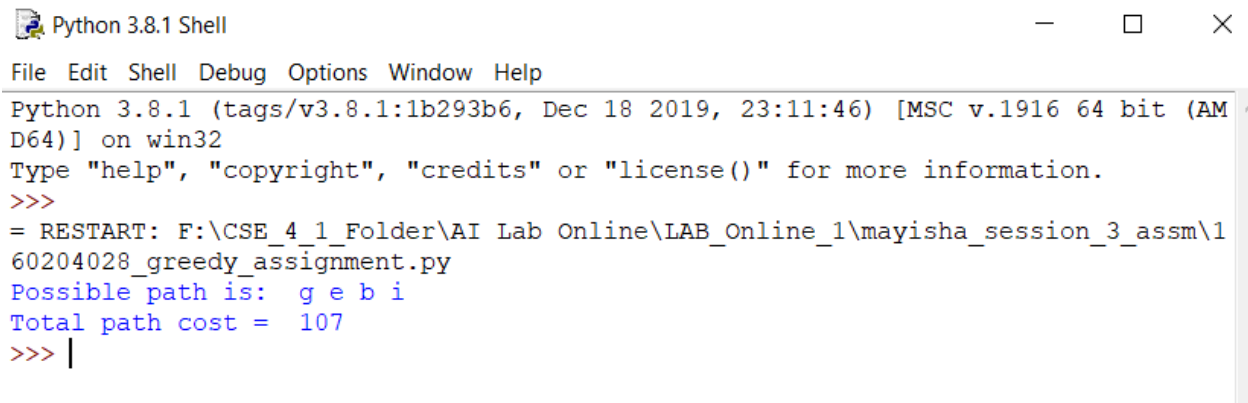
push((h[source], source))
vis[source] = True
while (empty() == False):
    u = pop()
    if (u[1] == goal):
        break
    #cost = u[0]
    #node_name = u[1]
    for adj in graph[u[1]]:
        v = adj[0]
        if (vis[v] == False):
            push((h[v], v))
            vis[v] = True
            parent[v] = u[1]

# finding path
tmp = goal
sum = 0
print("Possible path is: ", end = ' ')
while (parent[tmp] != -1):
    u = parent[tmp]
    v = tmp
    print(chr(ord('a') + tmp), end = ' ')
    for adj in graph[u]:
        if (adj[0] == v):
            sum += adj[1]
            break
    tmp = parent[tmp]
print(chr(ord('a') + tmp))
print("Total path cost = ", sum)

greedy_best_first()

```

## Output :



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\CSE_4_1_Folder\AI Lab Online\LAB_Online_1\mayisha_session_3_assm\160204028_greedy_assignment.py
Possible path is: g e b i
Total path cost = 107
>>> |
```

## Ques:

**2) Implement in generic ways (as multi-modular and interactive systems) the A\* search algorithms in Python.**

**Description:** In A\* search, we have set the heuristic values. After that we made the graph where we use list in where the first value is of the node and the second value is the cost of the node for traverse. Then we import Priority Queue Function. In empty function it will return true if the Queue is empty. In push we will add the value. In pop, we will pop some value if there any value otherwise we will return null if there is no value.

In A\* search, first we push the heuristic value and the cost. Then we check if there is any new cost is less than the old cost. We will update the value. We will add the cost of each node and the weight of the path. By doing this, we will get the final optimal solution for destination. In here, we can traverse the same node for many times to get better solution.

## Python Code:

```
inf = 10000000
h = [55, 42, 34, 25, 20, 17, 0, inf, 80]

graph = [ [(2, 22), (3, 32), (8, 35)],
  [(8, 45), (4, 36), (5, 27), (3, 28)],
  [(0, 22), (3, 31), (6, 47)],
  [(0, 32), (2, 31), (1, 28), (6, 30)],
  [(1, 36), (6, 26)],
```

```

[(1, 27)],
[(2, 47), (3, 30), (4, 26)],
[],
[(0, 35), (1, 45)] ]

from queue import PriorityQueue
Q = PriorityQueue()

def empty():
    return Q.empty()
def push(val):
    Q.put(val)
def pop():
    if (empty()):
        return None
    return Q.get()
def a_star():
    source = 8 # i
    goal = 6 # g
    parent = [-1] * len(graph)
    g = [inf] * len(graph)

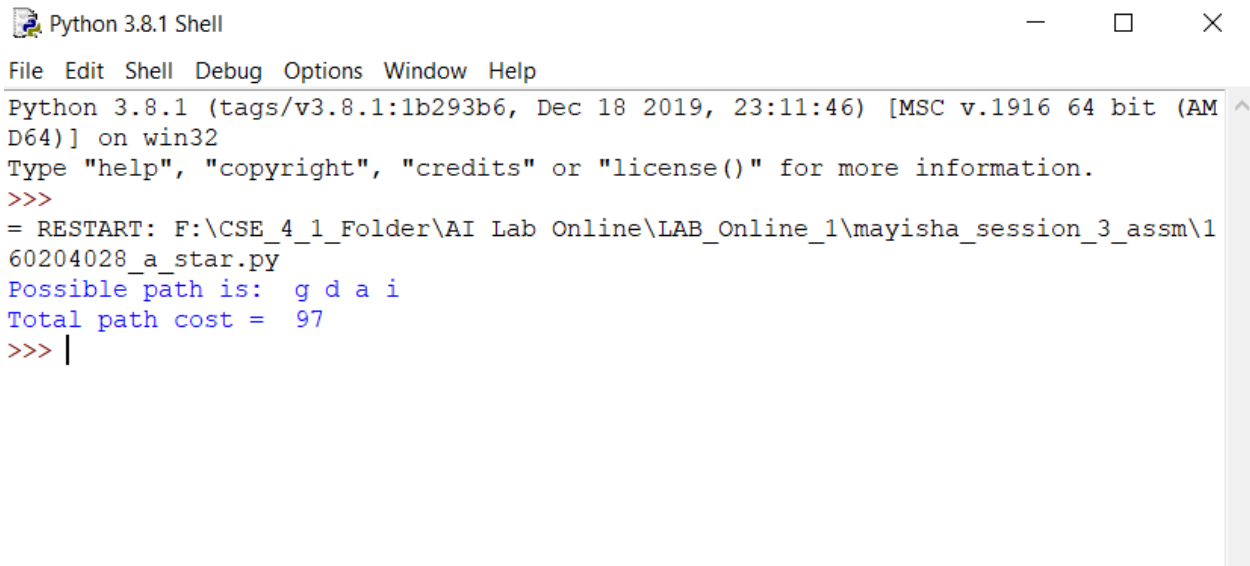
    g[source] = 0
    push((g[source] + h[source], source))
    while (empty() == False):
        tmp = pop()
        u = tmp[1]
        for adj in graph[u]:
            v = adj[0]
            if (g[u] + adj[1] < g[v]):
                g[v] = g[u] + adj[1]
                push((g[u] + adj[1] + h[u], v))
                parent[v] = u

    # finding path
    tmp = goal
    sum = 0
    print("Possible path is: ", end = ' ')
    while (parent[tmp] != -1):
        u = parent[tmp]
        v = tmp
        print(chr(ord('a') + tmp), end = ' ')
        for adj in graph[u]:
            if (adj[0] == v):
                sum += adj[1]
                break
        tmp = parent[tmp]
    print(chr(ord('a') + tmp))
    print("Total path cost = ", sum)

a_star()

```

## Output:



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\CSE_4_1_Folder\AI Lab Online\LAB_Online_1\mayisha_session_3_assm\1
60204028_a_star.py
Possible path is:  g d a i
Total path cost =  97
>>> |
```

**Ques: 3) Write a Python program that reads the file created as demonstrated into a dictionary taking 'name' as the key and a list consisting of 'dept' and 'cgpa' as the value for each line. Make changes in some 'cgpa' and then write back the whole file.**

### Description:

First we take an input in text file where we saved student name, department and cgpa. After storing the information, we write the python code. First we have opened the file in read mode. In the file we read the whole lines and split the lines with ' '. After that, we update some values cgpa in student table. After modifying the result, we store the value in the list. After that, we close the file. After that we opened the file in write mode then we rewrite the value with the updating values and then we closed the file. If we check the file later, we can see the modify cgpa in the text file.

### Python Code:

```
file_name = "input.txt"

def read():
    fp = open(file_name, "r")
    lst = []
```

```
for line in fp:
    ar = line.split(' ')
    #modify values
    if (ar[0] == 'mayisha'):
        ar[2] = '2.5\n'


    lst.append(ar)
fp.close()

fp = open(file_name, "w")
for val in lst:
    tmp = str(val[0]) + ' ' + str(val[1]) + ' ' + str(val[2])
    fp.write(tmp)

fp.close()
print('done')
read()
```

### Output:

---

 input.txt - Notepad

File Edit Format View Help

```
mayisha cse 2.5
akash cse 3.3
badhan eee 3.4
```