

AI LAB EXP – 5

BFS AND A* ALGORITHM FOR REAL WORLD PROBLEMS

Tejas Ashok

RA1911030010090

AIM

To implement Best First Algorithm and A* Algorithm using python.

BEST FIRST SEARCH

ALGORITHM

- Define a list, OPEN, consisting solely of a single node, the start node, s.
- IF the list is empty, return failure.
- Remove from the list the node n with the best score (the node where f is the minimum), and move it to a list, CLOSED.
- Expand node n.
- IF any successor to n is the goal node, return success and the solution (by tracing the path from the goal node to s).
- FOR each successor node:
 1. Apply the evaluation function, f, to the node.
 2. IF the node has not been in either list, add it to OPEN.
- Looping structure by sending the algorithm back to the second step.

CODE

```
from queue import PriorityQueue

v = 14

graph = [[] for i in range(v)]


def best_first_search(source, target, n):
    visited = [0] * n
    visited[0] = True
    pq = PriorityQueue()
```

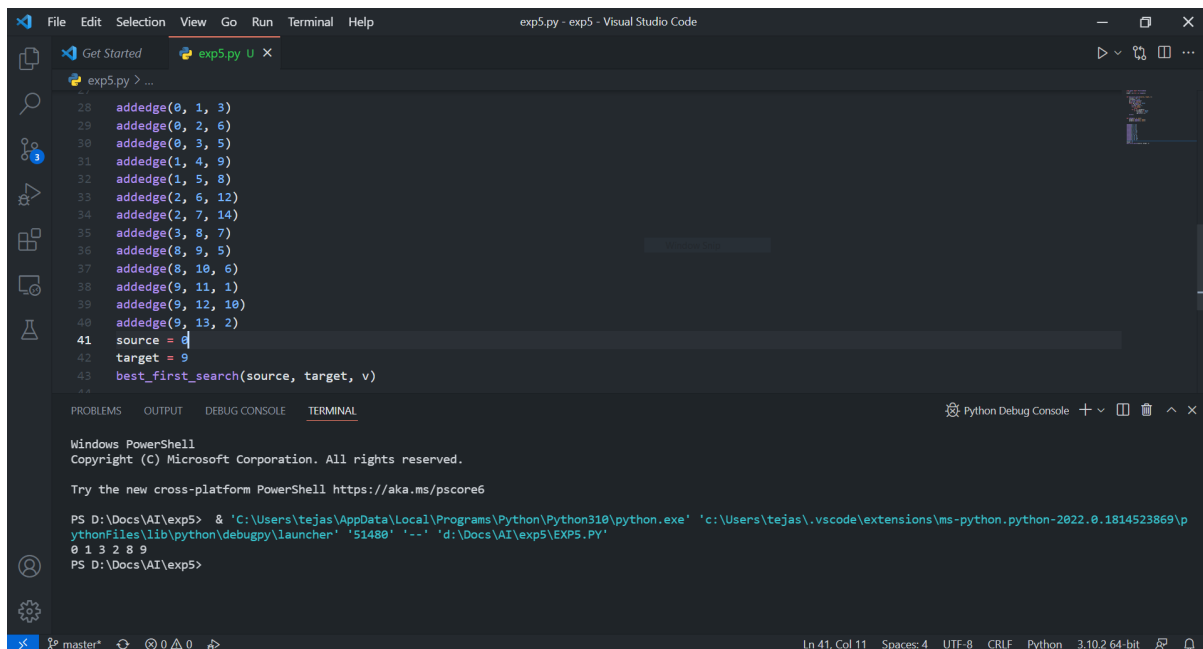
```
    pq.put((0, source))
    while pq.empty() == False:
        u = pq.get()[1]
        print(u, end=" ")
        if u == target:
            break
        for v, c in graph[u]:
            if visited[v] == False:
                visited[v] = True
                pq.put((c, v))
    print()
```

```
def addedge(x, y, cost):
    graph[x].append((y, cost))
    graph[y].append((x, cost))
```

```
adddedge(0, 1, 3)
adddedge(0, 2, 6)
adddedge(0, 3, 5)
adddedge(1, 4, 9)
adddedge(1, 5, 8)
adddedge(2, 6, 12)
adddedge(2, 7, 14)
adddedge(3, 8, 7)
adddedge(8, 9, 5)
adddedge(8, 10, 6)
adddedge(9, 11, 1)
adddedge(9, 12, 10)
adddedge(9, 13, 2)
source = 0
```

```
target = 9
best_first_search(source, target, v)
```

OUTPUT



```
File Edit Selection View Go Run Terminal Help
exp5.py - exp5 - Visual Studio Code

exp5.py > ...
28 addedge(0, 1, 3)
29 addedge(0, 2, 6)
30 addedge(0, 3, 5)
31 addedge(1, 4, 9)
32 addedge(1, 5, 8)
33 addedge(2, 6, 12)
34 addedge(2, 7, 14)
35 addedge(3, 8, 7)
36 addedge(8, 9, 5)
37 addedge(8, 10, 6)
38 addedge(9, 11, 1)
39 addedge(9, 12, 10)
40 addedge(9, 13, 2)
41 source = 0
42 target = 9
43 best_first_search(source, target, v)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python Debug Console

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Docs\AI\exp5> & 'C:\Users\tejas\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\tejas\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher' '51480' '--' 'd:\Docs\AI\exp5\EXP5.PY'
0 1 3 2 8 9
PS D:\Docs\AI\exp5>
```

A* SEARCH ALGORITHM

ALGORITHM

- We create two lists – Open List and Closed List (just like Dijkstra Algorithm)
- Initialize the open list
- Initialize the closed list put the starting node on the open list (you can leave its f at zero)
- While the open list is not empty
 1. Find the node with the least f on the open list, call it "q"
 2. Pop q off the open list
 3. Generate q's 8 successors and set their parents to q
 4. For each successor
 - i. If successor is the goal, stop search
 - ii. Else, compute both g and h for successor
 - $\text{successor.g} = \text{q.g} + \text{distance between successor and q}$
 - $\text{successor.h} = \text{distance from goal to successor}$ (This can be done using many ways, we will discuss three heuristics- Manhattan, Diagonal and Euclidean Heuristics)
 - $\text{successor.f} = \text{successor.g} + \text{successor.h}$
 - 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

- 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)

CODE

```
def aStarAlgo(start_node, stop_node):  
    open_set = set(start_node)  
    closed_set = set()  
    g = {}  
    parents = {}  
    g[start_node] = 0  
    parents[start_node] = start_node  
    while len(open_set) > 0:  
        n = None  
        for v in open_set:  
            if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):  
                n = v  
        if n == stop_node or Graph_nodes[n] == None:  
            pass  
        else:  
            for (m, weight) in get_neighbors(n):  
                if m not in open_set and m not in closed_set:  
                    open_set.add(m)  
                    parents[m] = n  
                    g[m] = g[n] + weight  
                else:  
                    if g[m] > g[n] + weight:  
                        g[m] = g[n] + weight  
                        parents[m] = n  
                        if m in closed_set:
```

```

        closed_set.remove(m)
        open_set.add(m)

    if n == None:
        print('Path does not exist!')
        return None

    if n == stop_node:
        path = []
        while parents[n] != n:
            path.append(n)
            n = parents[n]
        path.append(start_node)
        path.reverse()
        print('Path found: {}'.format(path))
        return path

    open_set.remove(n)
    closed_set.add(n)

print('Path does not exist!')
return None


def get_neighbors(v):
    if v in Graph_nodes:
        return Graph_nodes[v]
    else:
        return None


def heuristic(n):
    H_dist = {
        'A': 11,
        'B': 6,
        'C': 99,
    }

```

```

        'D': 1,
        'E': 7,
        'G': 0,
    }

    return H_dist[n]

```

```

Graph_nodes = {
    'A': [('B', 2), ('E', 3)],
    'B': [('C', 1), ('G', 9)],
    'C': None,
    'E': [('D', 6)],
    'D': [('G', 1)],
}

aStarAlgo('A', 'G')

```

OUTPUT

The screenshot shows the Visual Studio Code interface with the file `exp5-astar.py` open. The code defines the `aStarAlgo` function, which implements the A* search algorithm. The terminal window at the bottom shows the command prompt output, indicating that the path found is `['A', 'E', 'D', 'G']`.

```

exp5-astar.py > aStarAlgo
1 def aStarAlgo(start_node, stop_node):
2     open_set = set(start_node)
3     closed_set = set()
4     g = {}
5     parents = {}
6     g[start_node] = 0
7     parents[start_node] = start_node
8     while len(open_set) > 0:
9         n = None
10        for v in open_set:
11            if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
12                n = v
13        if n == stop_node or Graph_nodes[n] == None:
14            pass
15        else:
16            for (m, weight) in get_neighbors(n):
17                if m not in open_set and m not in closed_set:

```

Windows PowerShell
 Copyright (C) Microsoft Corporation. All rights reserved.
 Try the new cross-platform PowerShell <https://aka.ms/pscore6>

```

PS D:\Docs\AI\exp5> & 'C:\Users\tejas\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\tejas\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher' '61577' '--' 'd:\Docs\AI\exp5\exp5-astar.py'
Path found: ['A', 'E', 'D', 'G']
PS D:\Docs\AI\exp5>

```

RESULT

Best first search and A* search algorithm were successfully executed in python.