

Roll. No. A016	Name: Varun Khadayate
Class B.Tech CsBs	Batch: 1
Date of Experiment: 10-1-2021	Date of Submission: 31-1-2022

## To implement Greedy Best First Search

### Theory

Greedy best-first search algorithm always selects the path which appears best at that moment. It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search. Best-first search allows us to take the advantages of both algorithms. With the help of best-first search, at each step, we can choose the most promising node. In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function, i.e.

$$f(n) = g(n) + h(n)$$

Where,  $h(n)$  = estimated cost from node  $n$  to the goal.

The greedy best first algorithm is implemented by the priority queue.

### Steps to follow

- **Step 1:** Place the starting node into the OPEN list.
- **Step 2:** If the OPEN list is empty, Stop and return failure.
- **Step 3:** Remove the node  $n$ , from the OPEN list which has the lowest value of  $h(n)$ , and places it in the CLOSED list.
- **Step 4:** Expand the node  $n$ , and generate the successors of node  $n$ .
- **Step 5:** Check each successor of node  $n$ , and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.
- **Step 6:** For each successor node, algorithm checks for evaluation function  $f(n)$ , and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.
- **Step 7:** Return to Step 2.

### Code

```
from queue import PriorityQueue
v = 5
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))
addege(0, 1, 5)
addege(0, 2, 1)
addege(2, 3, 2)
addege(1, 4, 1)
addege(3, 4, 2)
source = 0
target = 4
best_first_search(source, target, v)
```

## Output

```
PS E:\TY\SEM VI\AI> & C:/Users/varun/AppData/Local/Programs/Python/Python310/python.exe "e:/TY/SEM VI/AI/PRAC_3.py"
0 2 3 4
```