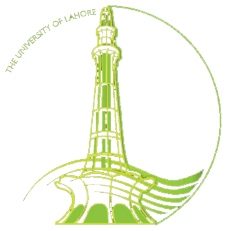

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Artificial Intelligence
(CS13217)

Lab Report

Name:	Wardha kanwal
Registration #:	CSU-S15-103
Lab Report #:	04
Dated:	06-04-2018
Submitted To:	Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 4
Breath First Search Problem

Objective
To understand Breath First Searchi Problem.

Software Tool
1. Python
2. Sublime, version 3.0
3. Operating System, window 8.1

1 Theory

There are many ways to traverse graphs. BFS is the most commonly used approach.

BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layerwise thus exploring the neighbour nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbour nodes.

As the name BFS suggests, you are required to traverse the graph breadthwise as follows:

1. First move horizontally and visit all the nodes of the current level
2. Move to the next level

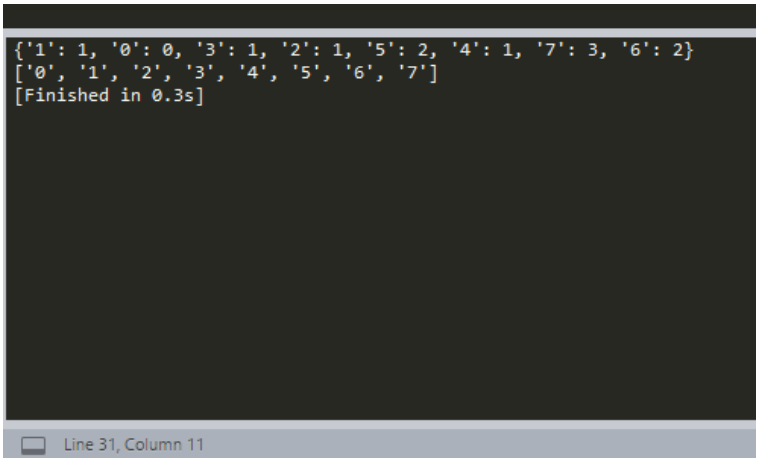

Figure 1: Time Independent Feature Set

2 Task

2.1 Procedure: Task 1

1. Enqueue the root node.
2. Dequeue a node and examine it.
3. If the element sought is found in this node, quit the search and return a result
4. Otherwise enqueue any successors (the direct child nodes) that have not yet been discovered.
5. If the queue is empty, every node on the graph has been examined, quit the search and return "not found".
6. If the queue is not empty, repeat from Step 2.

2.2 Procedure: Task 2

```
graph = {
    '0' : ['1','2','3','4'],
    '1' : ['0','5'],
    '2' : ['0','5'],
    '3' : ['0','6'],
```

```
    '4' : ['0','6'],
    '5' : ['1','2','7'],
    '6' : ['3','4','7'],
    '7' : ['5','6']}
def bfs_connected_component(graph, start):
    explored = []
    queue = [start]
    levels = {}
    levels[start]= 0
    visited= [start]

    while queue:
        node = queue.pop(0)
        explored.append(node)
        neighbours = graph[node]
        for neighbour in neighbours:
            if neighbour not in visited:
                queue.append(neighbour)
                visited.append(neighbour)
                levels[neighbour]= levels[node]+1

    print(levels)
    return explored
ans = bfs_connected_component(graph,'0')
print(ans)
```

3 Conclusion

1.Solution will definitely found out by BFS If there are some solution.
2.BFS will never get trapped in blind alley , means unwanted nodes.
3.If there are more than one solution then it will find solution with minimal steps.