Artificial Intelligence (CS13217)

Lab Report

Name:	Iqra Arshad
Registration #:	CSU-S15-120
Lab Report #:	04
Dated:	16-04-2018
Submitted To:	Mr. Usman Ahmed

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

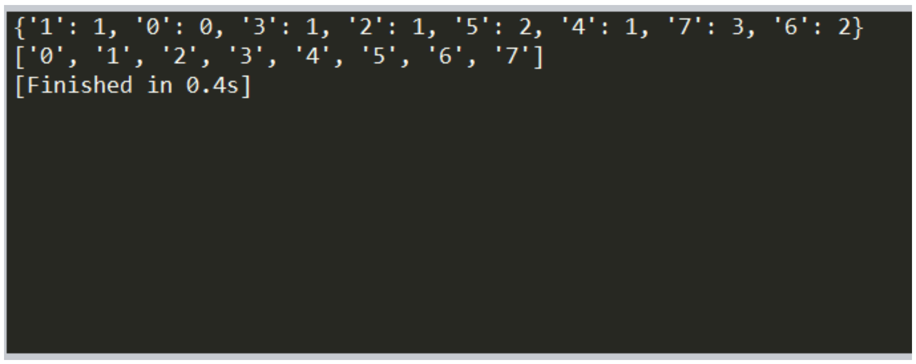
Experiment # 4
Implementing Breath First Search

Objective
To understand and implement the Breath First Search .

Software Tool
1. Operating System , Window 10
2. Sublime text , version 3.0
3. python

1 Theory

Breadth-first search (BFS) is an important graph search algorithm that is used to solve many problems including finding the shortest path in a graph and solving puzzle games (such as Rubik's Cubes). Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root and explores the neighbor nodes first, before moving to the next level neighbors. Breadth First Search algorithm(BFS) traverses a graph in a breadth wards motion and uses a queue to remember to get the next vertex to start a search when a dead end occurs in any iteration.
Consisting of vertices (nodes) and the edges (optionally directed/weighted) that connect them, the data-structure is effectively able to represent and solve many problem domains. One of the most popular areas of algorithm design within this space is the problem of checking for the existence or (shortest) path between two or more vertices in the graph. Properties such as edge weighting and direction are two such factors that the algorithm designer can take into consideration. .



```
{'1': 1, '0': 0, '3': 1, '2': 1, '5': 2, '4': 1, '7': 3, '6': 2}
['0', '1', '2', '3', '4', '5', '6', '7']
[Finished in 0.4s]
```

Figure 1: Time Independent Feature Set

2 Task

2.1 Procedure: Task 1

Depth-First and Breath-First search to achieve the goals highlighted below:

1.Find all vertices in a subject vertices connected component.
2.Return all available paths between two vertices.
3.And in the case of BFS, return the shortest path (length measured by number of path edges).

2.2 Procedure: Task 2

```
    '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'],
}
# visits all the nodes of a graph (connected component) using BFS
def bfs_connected_component(graph, start):
    explored = [] # keep track of all visited nodes
    queue = [start] # keep track of nodes to be checked
    levels = {}         # this dict keeps track of levels
    levels[start]= 0   # depth of start node is 0
    visited= [start]     # to avoid inserting the same node twice into the
    # keep looping until there are nodes still to be checked
    while queue:
   node = queue.pop(0) # pop shallowest node (first node) from queue
   explored.append(node)
   neighbours = graph[node]
   for neighbour in neighbours: # add neighbours of node to queue
   if neighbour not in visited:
   queue.append(neighbour)
   visited.append(neighbour)
    levels[neighbour]= levels[node]+1 # print(neighbour, ">>", levels[neigh

    print(levels)
    return explored
ans = bfs_connected_component(graph,'0') # returns ['A', 'B', 'C', 'E', 'D
print(ans)
```

3 Conclusion

Thus sucessfullly implemented Breath First Search program using python.