Premier University Chittagong.

Artificial Intelligence Laboratory

CSE 318

Implementation of BFS Algorithm

Author:
Mohiuddin Tamim
1903610201763
Sec: C1
Performance Date: 25/07/22

Submitted to:
Faisal Ahmed
Lecturer
Premier University Chittagong

Report No: 01.

Name of the report: Implementation of BFS Algorithm.

Objective:

- To understand the basic concept of BFS Algorithm
- To implement BFS Algorithm
- Using queue data structure for traversal.

Theory: Breadth-first search is a graph traversal algorithm that starts traversing the graph from the root node and explores all the neighboring nodes. Then, it selects the nearest node and explores all the unexplored nodes. While using BFS for traversal, any node in the graph can be considered as the root node. There are many ways to traverse the graph, but among them, BFS is the most commonly used approach. It is a recursive algorithm to search all the vertices of a tree or graph data structure. BFS puts every vertex of the graph into two categories - visited and non-visited. It selects a single node in a graph and, after that, visits all the nodes adjacent to the selected node.

Applications:

- BFS can be used to locate the nearest locations to a given source location.
- The BFS algorithm can be used as a traversal method in a peer-to-peer network to find all neighboring nodes. Most torrent clients, including BitTorrent and uTorrent, use this method to locate "seeds" and "peers" in the network.
- BFS is used to find the shortest path and the minimum spanning tree.
- Cheney's technique also implements in BFS to duplicate garbage collection.
- It can be used to compute the maximum flow in a flow network using the Ford-Fulkerson method.

Algorithm:

- Declare a queue and insert the starting vertex.
- Initialize a visited array and mark the starting vertex as visited
- Follow the below process till the queue becomes empty:
    - Remove the first vertex of the queue
    - Mark that vertex as visited
    - Insert all the unvisited neighbours of the vertex into the queue.

Complexity of BFS Search:

- **Time Complexity**: O(V+E), where V is the number of nodes and E is the number of edges.
- **Space Complexity**: O(V)

Code:

```
from queue import Queue
adjacencyList = {
        "A" : [ "B" , "D" ] ,
        "B" : [ "A" , "C" ] ,
        "C" : [ "B" ] ,
        "D" : [ "A" , "E" , "F" ] ,
        "E" : [ "D" , "F" , "G" ] ,
        "F" : [ "D" , "E" , "H" ] ,
        "G" : [ "E" , "H" ] ,
        "H" : [ "G" , "F" ]
}
visitedNodes = {}
l e v e l = {}
pa ren t = {}
bfs_traversal_output = [ ]
queue = Queue ( )

for node in  adjacencyList.key s ( ) :
        visited Nodes [ node ] = False
        pa ren t [ node ] = None
        l e v e l [ node ] = −1

s ta r tNode="A"
vi si t e d N o d e s [ s ta r tNode ] = True
l e v e l [ s ta r tNode ] = 0
queue . put ( s ta r tNode )

while not queue . empty ( ) :
        queueVal = queue . g e t ( )
        b f s_ t ra v e r sal_o u t p u t . append ( queueVal )
        for val in a d j a c e n c y Li s t [ queueVal ] :
```

```
        i f  not visitedNodes [ val ] :
                visitedNodes [ val ] = True
                pa ren t [ val ] = queueVal
                l e v e l [ val ] = l e v e l [ queueVal ]+1
                queue . put ( val )

d="G"
path =[]
while d is not None :
        path . append ( d )
        d = pa ren t [ d ]
        path . r e v e r s e ( )

print ( " Traversal_output : {}" . format ( bfs_traversal_output ) )
print ( " Levels_list : {}" . format ( level ) )
print ( "Path : {}" . format ( path ) )
```

Sample Input: There is no sample input because the code is initialized with an adjacency list containing values of the graph.

Sample Output:

Traversal output : ['A' , 'B' , 'D' , 'C' , 'E' , 'F' , 'G' , 'H']

Levelslist : { 'A' : 0 , 'B ' : 1 , 'C' : 2 , 'D' : 1 , 'E ' : 2 , 'F ' : 2 , 'G' : 3 ,'H' : 3}

Path : ['A' , 'D' , 'E' , 'G']

Discussion: We used the BFS algorithm in this experiment. We discovered that it chooses the closest node and explores all unexplored nodes. Later, we could take the input and output it without error and get the correct result.

Code Link: Click Here