Assignment No: 02
Breadth First Search (BFS)

CSE-0408 Summer 2021

SABER AHMED
UG02-44-17-006

Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
dev.saberahmed@gmail.com

***Abstract*—Breadth First Search (BFS) Algorithm.**

***Index Terms*—C++**

I. Introduction

BFS is a traversing algorithm where we 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). We must then move towards the next-level neighbour nodes.

II. Literature Review

BFS and its application in finding connected components of graphs were invented in 1945 by Konrad Zuse, in his (rejected) Ph.D. thesis on the Plankalk¨ul programming language, but this was not published until 1972. It was reinvented in 1959 by Edward F. Moore, who used it to find the shortest path out of a maze, and later developed by C. Y. Lee into a wire routing algorithm (published 1961). In 2012 Farhad S. et. al. proposed new resolution for solving N-queens by using combination of DFS (Depth First Search) and BFS (Breadth First Search) techniques.

III. Proposed Methodology

1. for each u in V s
2. do color[u] $\leftarrow WHITE$
3. $d[u] \leftarrow infinity$
4. $[u] \leftarrow NIL$
5. $color[s] \leftarrow GRAY$
6. $d[s] \leftarrow 0$
7. $[s] \leftarrow NIL$
8. $Q \leftarrow$
9. $ENQUEUE(Q, s)$
10. $whileQisnon - empty$
11. $dou \leftarrow DEQUEUE(Q)$
12. $foreachvadjacenttou$
13. $doifcolor[v] \leftarrow WHITE$
14. $thencolor[v] \leftarrow GRAY$
15. $d[v] \leftarrow d[u] + 1$
16. $[v] \leftarrow u$
17. $ENQUEUE(Q, v)$
18. $DEQUEUE(Q)$
19. $color[u] \leftarrow BLACK$

Acknowledgment

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this assignment.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, “Title of paper if known,” unpublished.
[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
[6] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

```
8 7
1 5
1 2
2 6
3 6
3 4
6 7
7 8
2
From node 2
Distance of 1 is : 1
Distance of 2 is : 0
Distance of 3 is : 2
Distance of 4 is : 3
Distance of 5 is : 2
Distance of 6 is : 1
Distance of 7 is : 2
Distance of 8 is : 3


...Program finished with exit code 0
Press ENTER to exit console.
```

Fig. 1. Output

```
#include
using namespace std;

#define MX 110

vector < int > graph[MX];
bool vis[MX];
int dist[MX];
int parent[MX];

void bfs(int source){
    queue < int > Q;
    // initialization
    vis[source] = 1;
    dist[source] = 0;
    Q.push(source);

    while(!Q.empty()){
        int node = Q.front();
        Q.pop();

        for (int i = 0; i < graph[node].size(); i++){
            int next = graph[node][i];
            if (vis[next] == 0){
                vis[next] = 1; // visit
                dist[next] = dist[node] + 1; // update
                Q.push(next); // push to queue

                // set parent
                parent[next] = node;
            }
        }
    }
}

/*
    input:
    8 7
    1 5
    1 2
    2 6
    3 6
    3 4
    6 7
    7 8
    2
*/

// recursive function
void printPathRecursive(int source, int node){
    if (node == source){
        cout << node << " "; // print from source
        return;
    }
    printPathRecursive(source, parent[node]);
    cout << node << " ";
}

// iterative function
void printPathIterative(int source, int node){
    vector path_vector;

    while(node != source){
        path_vector.push_back(node);
        node = parent[node];
    }
    path_vector.push_back(source); // inserting source

    for (int i = path_vector.size() - 1; i >= 0; i--){
        cout << path_vector[i] << " ";
    }
}
```

```
int main()
{
    int nodes, edges;
    cin >> nodes >> edges;

    for (int i = 1; i <= edges; i++){
        int u, v;
        cin >> u >> v;
        graph[u].push_back(v);
        graph[v].push_back(u);
    }

    int source;
    cin >> source;

    bfs(source);

    cout << "From node " << source << endl;
    for (int i = 1; i <= nodes; i++){
        cout << "Distance of " << i << " is : " << dist[i] << endl;
    }

    return 0;

}
```