Green University of Bangladesh

Department of Computer Science and Engineering(CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2021), B.Sc. in CSE (Day)

LAB REPORT NO 04

Course Code:206 Section:DB

Lab Experiment Name: Traverse all the vertices of graph using BFS method and find the shortest path between a source node and a destination node.

Student Details

	Name	**ID**
1.	Shamim Ahmed	201902067

Course Teacher's Name : Monoshi Kumar Roy

[For Teachers use only: Don't Write Anything inside this box]

L ab Report Status Marks: ……………………………… **Signature:………………….**

TITLE OF THE LAB EXPERIMENT

Traverse all the vertices of graph using BFS method and find the shortest path between a source node and a destination node.

OBJECTIVES/AIM

From this lab we will learn about BFS algorithm. We will will learn how to find the shortest path between a source node and a destination node in a graph.

PROCEDURE / ANALYSIS / DESIGN

BFS: BFS stands for Breadth First Search is a vertex based technique for finding a shortest path in graph.

BFS Algorithm:

```
1 let S be a stack
2 S.push(v)
3 while S is not empty
4 v = S.pop()
5 if v is not labeled as discovered:
6 label v as discovered
7 for all edges from v to w in G.adjacentEdges(v) do
8 S.push(w).
```

BFS Pseudocode:

```
Procedure BFS (G, s)
    G is the graph and s is the source node
begin
          let q be queue to store nodes
          q.enqueue(s) //insert source node in the queue

          mark s as visited.
          while (q is not empty)
          //remove the element from the queue whose adjacent nodes are to be processed
          n = q.dequeue( )

           //processing all the adjacent nodes of n
           for all neighbors m of n in Graph G if w is not visited
           q.enqueue (m)        //Stores m in Q to in turn visit its adjacent nodes
           mark m as visited.
end
```

IMPLEMENTATION

BFS Traverse and shortest path:

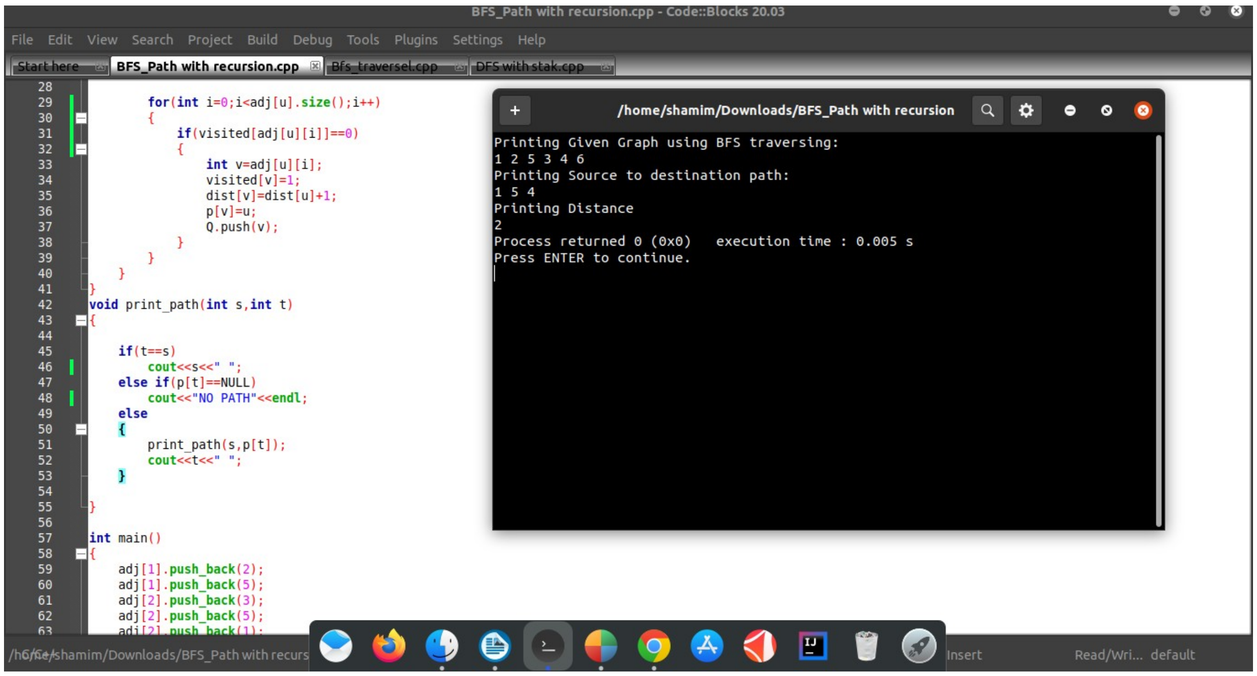
```
#include <bits/stdc++.h>
using namespace std;
```

```
vector <int> adj[6];
int visited[6];
int dist[6];
int p[6];

void bfs(int s, int n)
{
    for(int i=1;i<=n;i++){
        visited[i]=0;
    }
    queue <int> Q;
    Q.push(s);
    visited[s]=1;

    cout<<"Printing Given Graph using BFS traversing:"<<endl;

    while(!Q.empty())
    {
        int u=Q.front();
        cout<<u<<" ";
        Q.pop();




        for(int i=0;i<adj[u].size();i++)
        {
            if(visited[adj[u][i]]==0)
            {
                int v=adj[u][i];
                visited[v]=1;
                dist[v]=dist[u]+1;
                p[v]=u;
                Q.push(v);
            }
        }
    }
}
```

```
void print_path(int s,int t)
{

    if(t==s)
        cout<<s<<" ";
    else if(p[t]==NULL)
        cout<<"NO PATH"<<endl;
    else
    {
        print_path(s,p[t]);
        cout<<t<<" ";
    }

}

int main()
{
    adj[1].push_back(2);
    adj[1].push_back(5);
    adj[2].push_back(3);
    adj[2].push_back(5);
    adj[2].push_back(1);
    adj[5].push_back(1);
    adj[5].push_back(2);
    adj[5].push_back(4);
    adj[3].push_back(2);
    adj[3].push_back(4);
    adj[4].push_back(5);
    adj[4].push_back(6);
    bfs(1,6);
    cout<<endl;
    cout<<"Printing Source to destination path:"<<endl;
    print_path(1,4);

    cout<<endl<<"Printing Distance"<<endl;
    cout<<dist[4];
}
```

5.TEST RESULT / OUTPUT

BFS Traverse and shortest path:


BFS_Path with recursion.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Tools Plugins Settings Help
Start here
BFS_Path with recursion.cpp
Bfs_traversel.cpp
DFS with stak.cpp
/home/shamim/Downloads/BFS_Path with recursion
/h6/Ge/shamim/Downloads/BFS_Path with recurs
Insert
Read/Wri... default

```
        for(int i=0;i<adj[u].size();i++)
        {
            if(visited[adj[u][i]]==0)
            {
                int v=adj[u][i];
                visited[v]=1;
                dist[v]=dist[u]+1;
                p[v]=u;
                Q.push(v);
            }
        }
    }
}
void print_path(int s,int t)
{

    if(t==s)
        cout<<s<<" ";
    else if(p[t]==NULL)
        cout<<"NO PATH"<<endl;
    else
    {
        print_path(s,p[t]);
        cout<<t<<" ";
    }

}

int main()
{
    adj[1].push_back(2);
    adj[1].push_back(5);
    adj[2].push_back(3);
    adj[2].push_back(5);
    adj[2].push_back(1);
```

```
Printing Given Graph using BFS traversing:
1 2 5 3 4 6
Printing Source to destination path:
1 5 4
Printing Distance
2
Process returned 0 (0x0)   execution time : 0.005 s
Press ENTER to continue.
```