BFS Implementation in C++

Presented by: Adarsh Singh

File Edit Selection Find View Goto Tools Project Preferences Help

adja.cpp

```cpp
#include<bits/stdc++.h>

using namespace std;

vector <int> v[10] ;   // vector for maintaining adjacency list explained above.

int level[10]; // to determine the level of each node
bool vis[10]; //mark the node if visited
void bfs(int s)
{
    queue <int> q;
    q.push(s);
    level[ s ] = 0 ;  //setting the level of sources node as 0.
    vis[ s ] = true;
    cout<<s<<" level "<<level[s]<<endl;
    while(!q.empty())
    {
        int p = q.front();
        q.pop();
        for(int i = 0;i < v[ p ].size() ; i++)
        {
            if(vis[ v[ p ][ i ] ] == false)
            {
        //setting the level of each node with an increment in the level of parent node
                level[ v[ p ][ i ] ] = level[ p ]+1;
                q.push(v[ p ][ i ]);
                cout<<v[p][i]<<" level "<<level[ v[ p ][ i ] ]<<endl;
                vis[ v[ p ][ i ] ] = true;
            }
        }
```

Line 7, Column 1 Spaces: 4 C++

File Edit Selection Find View Goto Tools Project Preferences Help

adja.cpp

```cpp
        //setting the level of each node with an increment in the level of parent node
                level[ v[ p ][ i ] ] = level[ p ]+1;
                 q.push(v[ p ][ i ]);
                 cout<<v[p][i]<<" level "<<level[ v[ p ][ i ] ]<<endl;
                 vis[ v[ p ][ i ] ] = true;
            }
        }
    }
}

int main()
{
    int edges_count;
    cin>>edges_count;
    while(edges_count--)
    {
        int m,n;
        cin>>m>>n;
        v[m].push_back(n);
    }
    bfs(1);
    return 0;
}
```

Line 40, Column 17 Spaces: 4 C++

Input/Output

Input	Output
5	1 level 0
1 2	2 level 1
2 4	4 level 1
1 4	3 level 2
2 3	5 level 3
3 5	

Time Complexity

Time complexity of BFS is **O(V + E)** , where V is the number of nodes and E is the number of Edges.

Thanks