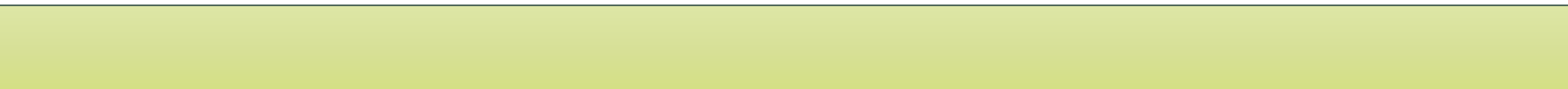




BFS Implementation in C++

Presented by: Adarsh Singh



adja.cpp

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  vector <int> v[10] ;    // vector for maintaining adjacency list explained above.
6
7  int level[10]; // to determine the level of each node
8  bool vis[10]; //mark the node if visited
9  void bfs(int s)
10 {
11     queue <int> q;
12     q.push(s);
13     level[ s ] = 0 ;    //setting the level of sources node as 0.
14     vis[ s ] = true;
15     cout<<s<<" level "<<level[s]<<endl;
16     while(!q.empty())
17     {
18         int p = q.front();
19         q.pop();
20         for(int i = 0;i < v[ p ].size() ; i++)
21         {
22             if(vis[ v[ p ][ i ] ] == false)
23             {
24                 //setting the level of each node with an increment in the level of parent node
25                 level[ v[ p ][ i ] ] = level[ p ]+1;
26                 q.push(v[ p ][ i ]);
27                 cout<<v[p][i]<<" level "<<level[ v[ p ][ i ] ]<<endl;
28                 vis[ v[ p ][ i ] ] = true;
29             }
30         }
31     }
```

adja.cpp

```
24 //setting the level of each node with an increment in the level of parent node
25     level[ v[ p ][ i ] ] = level[ p ]+1;
26     q.push(v[ p ][ i ]);
27     cout<<v[p][i]<<" level "<<level[ v[ p ][ i ] ]<<endl;
28     vis[ v[ p ][ i ] ] = true;
29 }
30 }
31 }
32 }
33
34 int main()
35 {
36     int edges_count;
37     cin>>edges_count;
38     while(edges_count-->0)
39     {
40         int m,n;
41         cin>>m>>n;
42         v[m].push_back(n);
43     }
44     bfs(1);
45     return 0;
46 }
47
```

Input/Output

Input

5

1 2

2 4

1 4

2 3

3 5

Output

1 level 0

2 level 1

4 level 1

3 level 2

5 level 3

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