BFS BY BUBU

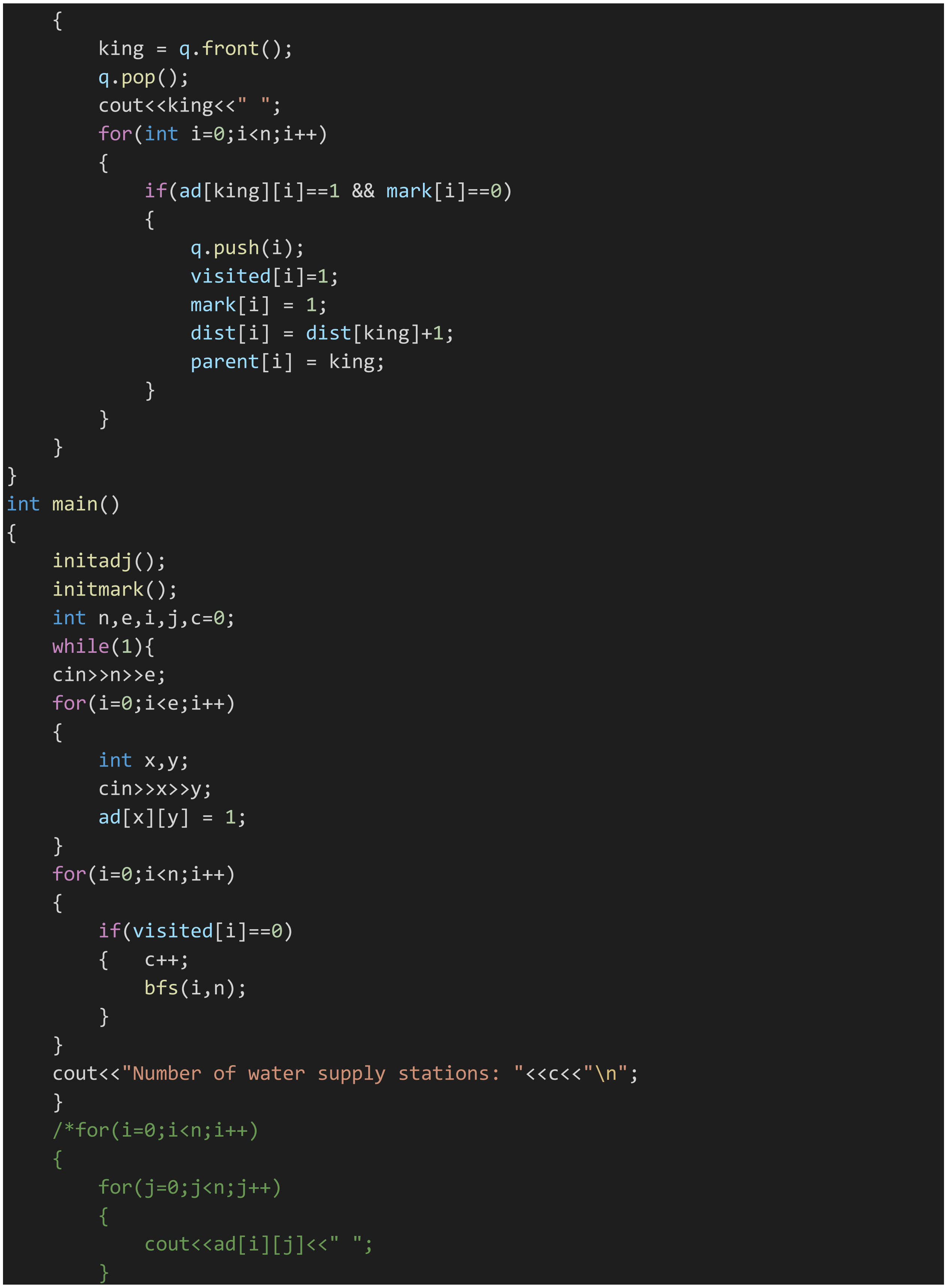
```cpp
#include<iostream>
#include<queue>
//#include<bits/stdc++.h>
using namespace std;
int n = 9;
int main(){
    int v[n];
    int g[n][n];
    int color[n];
    int parent[n];
    int dist[n];
    int i;
    for(i=0; i<n; i++){
        v[i] = 0;
        color[i] = 0;
        parent[i] = 0;
        dist[i] = 0;
    }
    for(i=0; i<n; i++){
        for(int j = 0; j<n; j++){
            g[i][j] = 0;
        }
    }
    g[0][1] = 1;
    g[0][2] = 1;
    g[0][3] = 1;
    g[1][4] = 1;
    g[1][5] = 1;
    g[2][3] = 1;
    g[2][6] = 1;
    g[3][7] = 1;
    g[3][8] = 1;
    g[7][8] = 1;
    int c = 0;
    queue<int>q;
    q.push(0);
    color[0] = 1;
    parent[0] = -1;
    v[0] = 1;
    while(!q.empty()){
        int k = q.front();
        cout<<k<<" "<<dist[k]<<"\n";
        q.pop();
        for(i=0; i<n; i++){
            if(g[k][i]==1 && color[i] == 0){
                q.push(i);
                color[i] = 1;
                parent[i] = k;
                dist[i] = dist[k] + 1;
                v[i] = 1;
            }
            else if(g[k][i]==1 && color[i] == 1 && v[i]==1 && parent[k]!=i){
                c++;
                g[k][i] = 0;
            }
        }
        color[k] = 2;
    }
    cout<<"Number of cycles: "<<c<<"\n";
    for(i=0;i<n;i++){
        if(parent[i]!=-1) cout<<"Child: "<<i<<" -> "<<"Parent: 
"<<parent[i]<<"\n";
        if(parent[i]==-1) cout<<"Root Child: "<<i<<" -> "<<"NULL(since its the 
root)\n";
    }
    int x;
    cout<<"the node to find: ";
    cin>>x;
    vector<int>s;
    if(x>-1){
            while(x!=-1){
        s.push_back(x);
        x = parent[x];
    }
    for(i=0;i<s.size();i++) cout<<s[i]<<" ";

    }
    else cout<<"No files were found"<<endl<<-1<<endl;

}
```

MOCK ONLINE

```
#include<bits/stdc++.h>
using namespace std;
int ad[100][100];
int mark[100];
int dist[100];
int parent[100];
int visited[100];
void initadj()
{
    int i,j;
    for(i=0;i<100;i++)
    {
        for(j=0;j<100;j++)
        {
            ad[i][j] = 0;
        }
    }
}
void initmark()
{
    int i;
    for(i=0;i<100;i++)
    {
        mark[i] = 0;
        visited[i] = 0;
    }
}
void initdist()
{
    int i;
    for(i=0;i<100;i++)
    {
        dist[i] = -1;
    }
}
void initparent()
{
    int i;
    for(i=0;i<100;i++)
    {
        parent[i] = -1;
    }
}
void bfs(int start,int n)
{
    queue<int> q;
    q.push(start);
    int king;
    while(q.size()!=0)
```



```
    {
        king = q.front();
        q.pop();
        cout<<king<<" ";
        for(int i=0;i<n;i++)
        {
            if(ad[king][i]==1 && mark[i]==0)
            {
                q.push(i);
                visited[i]=1;
                mark[i] = 1;
                dist[i] = dist[king]+1;
                parent[i] = king;
            }
        }
    }
}
int main()
{
    initadj();
    initmark();
    int n,e,i,j,c=0;
    while(1){
    cin>>n>>e;
    for(i=0;i<e;i++)
    {
        int x,y;
        cin>>x>>y;
        ad[x][y] = 1;
    }
    for(i=0;i<n;i++)
    {
        if(visited[i]==0)
        {   c++;
            bfs(i,n);
        }
    }
    cout<<"Number of water supply stations: "<<c<<"\n";
    }
    /*for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            cout<<ad[i][j]<<" ";
        }
        cout<<"\n";
    }*/
}
```

DFS PARENT CYCLE DETECT

```
#include<bits/stdc++.h>
using namespace std;
int n = 9;

int main()
{
    int v[n];
    int g[n][n];
    int color[n];
    int parent[n];
    int dist[n];
    int i;
    for(i=0; i<n; i++)
    {
        v[i] = 0;
        color[i] = 0;
        parent[i] = 0;
        dist[i] = 0;
    }
    for(i=0; i<n; i++)
    {
        for(int j = 0; j<n; j++)
        {
            g[i][j] = 0;
        }
    }
    g[0][1] = 1;
    g[0][2] = 1;
    g[0][3] = 1;
    g[1][4] = 1;
    g[1][5] = 1;
    g[2][3] = 1;
    g[2][6] = 1;
    g[3][7] = 1;
    g[3][8] = 1;
    g[7][8] = 1;
    int c = 0;
    stack<int>s;
    s.push(0);
    color[0] = 1;
    parent[0] = 0;
    v[0] = 1;
    while(!s.empty())
    {
        int k = s.top();
        cout<<k<<"\n";
        s.pop();
        for(i=0; i<n; i++)
        {
            if(g[k][i]==1 && color[i] == 0 && v[i]==0)
            {
                s.push(i);
                color[i] = 1;
                parent[i] = k;
                dist[i] = dist[k] + 1;
                v[i] = 1;
            }
            else if(g[k][i]==1 && v[i]==1 && parent[k]!=i)
            {
                c++;
                g[k][i] = 0;
            }
        }
        color[k] = 2;

    }
    cout<<"Number of cycles: "<<c<<"\n";
    for(i=0;i<n;i++)
    {
        if(parent[i]!=-1)
        {
            cout<<i<<" -> "<<parent[i]<<"\n";
        }
        if(parent[i]==-1)
        {
            cout<<i<<" -> "<<"NULL(since its the root)\n";
        }
    }

}
```

DFS ANIKA

```
#include<bits/stdc++.h>
using namespace std;
int count_=0;
 int g[5][5]={0};
    int v[5];
    int color[5]={0};
    int dist[5]={0};
void dfs_visit(int k)
{
    cout<<k<< " "<<dist[k]<<endl;
    color[k]=1;
    for(int i=0;i<5;i++)
    {
        if(g[k][i]==1 && color[i]==0)
        {
            dist[i]=dist[k]+1;
            dfs_visit(i);
        }
    }
    color[k]=2;
}
void parent()
{
    for(int i=0; i<5; i++)
        for(int j=0; j<5; j++)
    {
        if(g[i][j]==1)
            cout<<j<<"->"<<i<<endl;
    }
}
int main()
{

    g[0][1]=1;
    g[0][2]=1;
    g[1][4]=1;
    g[1][3]=1;
    dist[0]=0;
    dfs_visit(0);
    parent();
}
```