```
/*** Biconnected Component:
A graph is Biconnected if it has no vertex such that its removal increases the number of
connected components in the graph. And if there exists such a vertex then it is not Biconnected.
A vertex whose removal increases the number of connected components is called an Articulation Point.
So simply check if the given graph has any articulation point or not. If it has no articulation
point then it is Biconnected otherwise not. Now let's move on to Biconnected Components.
For a given graph, a Biconnected Component, is one of its subgraphs which is Biconnected.
***/

#include<bits/stdc++.h>
using namespace std;
#define pii pair<int,int>
int gr,tim,root,dis[100005],low[100005],par[100005];
vector<int>ed[100005];
bool vis[100005];
vector<pii>bcc[100005];
stack<pii>st;

void dfs(int u){
    vis[u]=true;
    dis[u]=++tim;
    low[u]=tim;
    int child=0;

    for(int i=0; i<ed[u].size(); i++){
        int v = ed[u][i];

        if(vis[v]==false){
            par[v]=u;
            child++;
            pii uv = make_pair(u,v);
            st.push(uv);

            dfs(v);

            low[u]=min(low[u],low[v]);

            if(u==root && child>1){
               //found articulation bridge
                gr++;
                while(st.top()!=uv){
                    pii w = st.top();
                    st.pop();
                    bcc[gr].push_back(w);
                }

                bcc[gr].push_back(st.top());
                st.pop();
            }

            if(u!=root && dis[u]<=low[v]){
                //found articulation bridge
                gr++;
                while(st.top()!=uv){
                    pii w = st.top();
                    st.pop();
                    bcc[gr].push_back(w);
                }

                bcc[gr].push_back(st.top());
                st.pop();
            }

        }
        else if(v!=par[u] && dis[v]<low[u]) {
            // found back edge & ignore cross edge
            low[u]=dis[v];
            st.push(make_pair(u,v));
        }
    }
}

```

```cpp
int main(){
    int tt; scanf("%d",&tt);
    for(int ks=1; ks<=tt; ks++){
        int n,m; scanf("%d%d",&n,&m);

        for(int i=1; i<=m; i++){
            int u,v; scanf("%d%d",&u,&v);
            ed[u].push_back(v);
            ed[v].push_back(u);
        }

        memset(vis,false,sizeof(vis));
        tim=gr=0;

        for(int i=1; i<=n; i++){
            if(vis[i]==false) {
                root = i;
                dfs(i);

                if(!st.empty())gr++;

                while(!st.empty()){
                    pii w = st.top();
                    st.pop();
                    bcc[gr].push_back(w);
                }
            }
        }

        for(int i=1; i<=gr; i++){
            printf("Group %d:\n",i);
            for(int j=0; j<bcc[i].size(); j++){
                pii w = bcc[i][j];
                int u = w.first; int v = w.second;
                printf("%d %d\n",u,v);
            }
            printf("\n");
        }

        for(int i=1; i<=n; i++)ed[i].clear();
        for(int i=1; i<=gr; i++)bcc[i].clear();
    }

    return 0;
}
```