

```

1  /** Biconnected Component:
2  A graph is Biconnected if it has no vertex such that its removal increases the number of
3  connected components in the graph. And if there exists such a vertex then it is not Biconnected.
4  A vertex whose removal increases the number of connected components is called an Articulation Point.
5  So simply check if the given graph has any articulation point or not. If it has no articulation
6  point then it is Biconnected otherwise not. Now Let's move on to Biconnected Components.
7  For a given graph, a Biconnected Component, is one of its subgraphs which is Biconnected.
8  */
9
10 #include<bits/stdc++.h>
11 using namespace std;
12 #define pii pair<int,int>
13 int gr,tim,root,dis[100005],low[100005],par[100005];
14 vector<int>ed[100005];
15 bool vis[100005];
16 vector<pii>bcc[100005];
17 stack<pii>st;
18
19 void dfs(int u){
20     vis[u]=true;
21     dis[u]=++tim;
22     low[u]=tim;
23     int child=0;
24
25     for(int i=0; i<ed[u].size(); i++){
26         int v = ed[u][i];
27
28         if(vis[v]==false){
29             par[v]=u;
30             child++;
31             pii uv = make_pair(u,v);
32             st.push(uv);
33
34             dfs(v);
35
36             low[u]=min(low[u],low[v]);
37
38             if(u==root && child>1){
39                 //found articulation bridge
40                 gr++;
41                 while(st.top()!=uv){
42                     pii w = st.top();
43                     st.pop();
44                     bcc[gr].push_back(w);
45                 }
46
47                 bcc[gr].push_back(st.top());
48                 st.pop();
49             }
50
51             if(u!=root && dis[u]<=low[v]){
52                 //found articulation bridge
53                 gr++;
54                 while(st.top()!=uv){
55                     pii w = st.top();
56                     st.pop();
57                     bcc[gr].push_back(w);
58                 }
59
60                 bcc[gr].push_back(st.top());
61                 st.pop();
62             }
63
64         }
65         else if(v!=par[u] && dis[v]<low[u]) {
66             // found back edge & ignore cross edge
67             low[u]=dis[v];
68             st.push(make_pair(u,v));
69         }
70     }
71 }
72

```



```
73 int main(){
74     int tt; scanf("%d",&tt);
75     for(int ks=1; ks<=tt; ks++){
76         int n,m; scanf("%d%d",&n,&m);
77
78         for(int i=1; i<=m; i++){
79             int u,v; scanf("%d%d",&u,&v);
80             ed[u].push_back(v);
81             ed[v].push_back(u);
82         }
83
84         memset(vis,false,sizeof(vis));
85         tim=gr=0;
86
87         for(int i=1; i<=n; i++){
88             if(vis[i]==false) {
89                 root = i;
90                 dfs(i);
91
92                 if(!st.empty())gr++;
93
94                 while(!st.empty()){
95                     pii w = st.top();
96                     st.pop();
97                     bcc[gr].push_back(w);
98                 }
99             }
100         }
101
102         for(int i=1; i<=gr; i++){
103             printf("Group %d:\n",i);
104             for(int j=0; j<bcc[i].size(); j++){
105                 pii w = bcc[i][j];
106                 int u = w.first; int v = w.second;
107                 printf("%d %d\n",u,v);
108             }
109             printf("\n");
110         }
111
112         for(int i=1; i<=n; i++)ed[i].clear();
113         for(int i=1; i<=gr; i++)bcc[i].clear();
114     }
115
116     return 0;
117 }
```