## 5. Biconnected components in a graph (using Tarjan’s algorithm)

**Description:** Finds articulation points and biconnected components using DFS, discovery time and low values, stack of edges.

```c
// biconnected.c
#include <stdio.h>
#include <stdlib.h>
#define MAXV 100

typedef struct Edge{ int u,v; struct Edge* next; } Edge;

int time_dfs;
int disc[MAXV], low[MAXV], st_u[1000], st_v[1000], top=-1;
int n; Edge* adj[MAXV];

void pushEdge(int u,int v){ st_u[++top]=u; st_v[top]=v; }
void popComponent(){ printf("Biconnected component: ");
    while(top!=-1){ printf("(%d,%d) ", st_u[top], st_v[top]); top--; }
    printf("\n"); }

void addEdge(int u,int v){ Edge* e = malloc(sizeof(Edge)); e->u=u; e->v=v;
e->next=adj[u]; adj[u]=e; }

void bccDFS(int u, int parent){ disc[u]=low[u]=++time_dfs; int children=0;
    for(Edge* e=adj[u]; e; e=e->next){ int v=e->v;
        if(!disc[v]){ children++; pushEdge(u,v); bccDFS(v,u); low[u]=
(low[u]<low[v])?low[u]:low[v];
            if((parent==-1 && children>1) || (parent!=-1 &&
low[v]>=disc[u])){ popComponent(); }
        } else if(v!=parent && disc[v] < disc[u]){ low[u] = (low[u] <
disc[v])? low[u] : disc[v]; pushEdge(u,v); }
    }
}

int main(){
    int m,u,v; scanf("%d %d", &n, &m);
    for(int i=0;i<n;i++){ adj[i]=NULL; disc[i]=0; low[i]=0; }
    for(int i=0;i<m;i++){ scanf("%d %d", &u, &v); addEdge(u,v); addEdge(v,u);
}
    time_dfs=0; top=-1;
    for(int i=0;i<n;i++) if(!disc[i]) bccDFS(i,-1);
    return 0;
}
```