B. BODY BUILDING

PROBLEM DESCRIPTION

題目給一張無向圖，問在這張圖中有幾個子圖滿足：可由一條邊被切開成兩個相同大小的完全子圖。

SOLUTION TECHINQUES

Tarjan’s Algorithm, BFS

SOLUTION SKETCHES

由一條邊斷開等同於是找圖上的 Bridge，所以我先使用 Tarjan’s Algorithm 找出圖上的所有 Bridge；

接著我們依次查看每座 Bridge，檢查 Bridge 兩端點連出去的圖點數是否相同、是否是完全圖，這部分我是用 BFS 統計連出去的點數 V 與邊數 E，檢查是否 E == V * (V – 1) + 1 (加上 Bridge 本身)。

TIME COMPLEXITY

每筆測資 O(NM)，N 為點的數量、M 為邊的數量。

SOLUTION PROGRAM FOR REFERENCE

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <vector>
#include <utility>
#include <queue>

using namespace std;
typedef pair<int, int> pii;

const int N = 102;

int v, e;
vector<pii> bri;
vector<int> g[N];
int deg[N];
int low[N], dfn[N]; int t;
char vst[N];

int bfs(int vi, int no)
{
     fill(vst+1, vst+1+v, 0);
     int ee = 0, vv = 0;
     queue<int> q;
     vst[vi] = 1;
     q.push(vi);
     while (!q.empty())
```

```
    {
        int vi = q.front(); q.pop();
        if (vi == no) continue;
        ee += deg[vi];
        ++vv;
        for (auto &vj: g[vi])
            if (!vst[vj])
            {
                vst[vj] = 1;
                q.push(vj);
            }
    }
    return ee == vv * (vv - 1) + 1 ? vv : 0;
}

int dfs(int p, int vi)
{
    dfn[vi] = low[vi] = ++t;
    for (auto &vj: g[vi])
    {
        if (vj == p) continue;
        if (dfn[vj] == -1)
        {
            low[vi] = min(low[vi], dfs(vi, vj));
            if (low[vj] > dfn[vi])
                bri.push_back(make_pair(vi, vj));
        }
        else
            low[vi] = min(low[vi], dfn[vj]);
    }
    return low[vi];
}

int main()
{
    int i, j, tt;
    int vi, vj;
    int ans;
    scanf("%d", &tt);
    for (int cc = 1; cc <= tt; cc++)
    {
        scanf("%d%d", &v, &e);
        ans = 0;
        bri.clear();
        for (i = 1; i <= v; i++)
        {
            dfn[i] = -1;
            g[i].clear();
            deg[i] = 0;
        }
        t = 0;
        for (i = 0; i < e; i++)
        {
            scanf("%d%d", &vi, &vj);
            g[vi].push_back(vj);
            g[vj].push_back(vi);
            deg[vi]++;
            deg[vj]++;
        }
        for (i = 1; i <= v; i++)
            if (dfn[i] == -1)
                dfs(-1, i);
        for (auto &e: bri)
        {
            vi = bfs(e.first, e.second);
            vj = bfs(e.second, e.first);
            if (vi && vj && vi == vj)
                ans++;
        }
        printf("Case #%d: %d\n", cc, ans);
    }
    return 0;
}
```