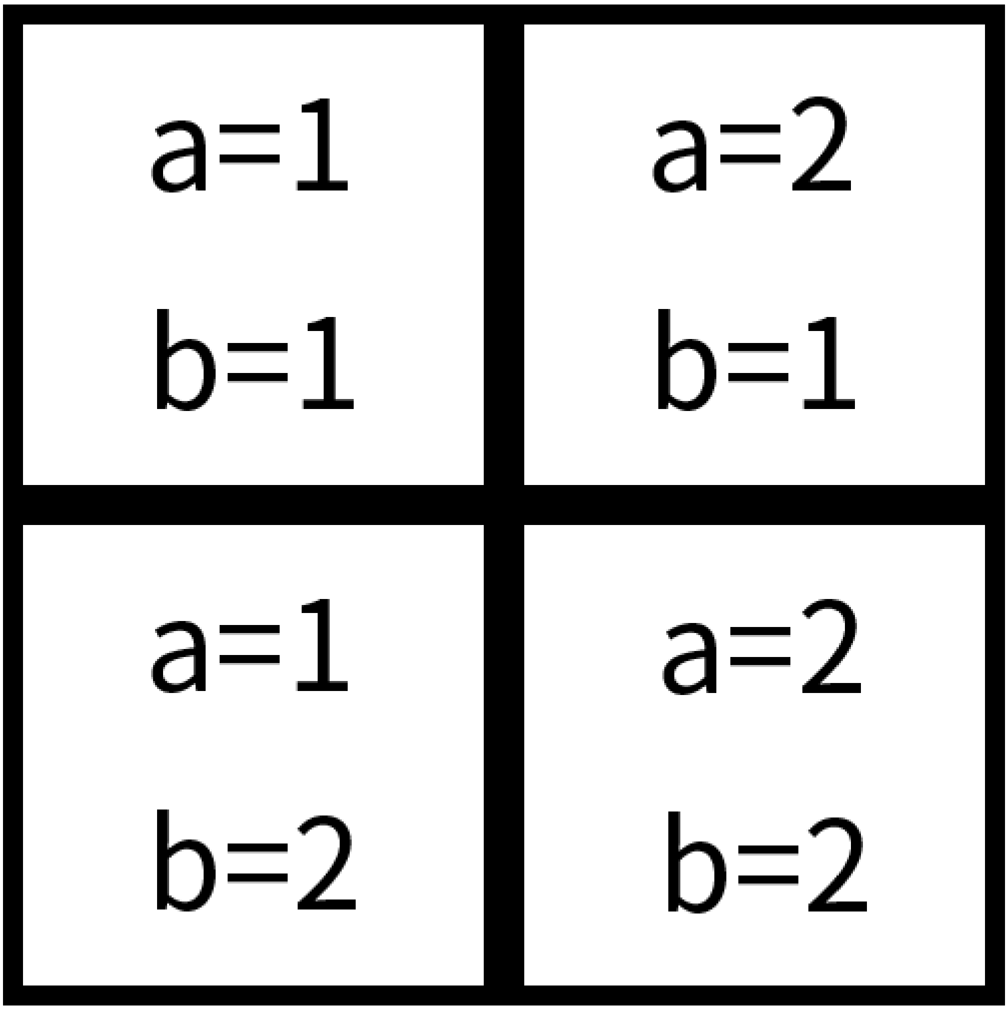
A 小红的图



a=1
b=1
	a=2
	b=1

a=1
b=2
	a=2
	b=2

Figure 1

```cpp
void solve() {
    int a, b;
    cin >> a >> b;
    if (a == 1 && b == 1) cout << "LU" << "\n";
    if (a == 1 && b == 2) cout << "LD" << "\n";
    if (a == 2 && b == 1) cout << "RU" << "\n";
    if (a == 2 && b == 2) cout << "RD" << "\n";
}
```

Fence 1

B 小红的菊花

因为和其他点都有边，所以度为 $n-1$ 的点就是中心。

```cpp
void solve() {
    int n;
    cin >> n;
    vector<int> d(n + 1);
    for (int i = 1; i < n; ++i) {
        int u, v;
        cin >> u >> v;
        d[u]++, d[v]++;
    }
    for (int i = 1; i <= n; ++i) {
        if (d[i] == n - 1) {
            cout << i << "\n";
            return;
        }
    }
}
```

Fence 2

C 小红的好矩形

面积为 2 的矩形要么是 1×2 的，要么是 2×1 的，统计一下即可。

```
void solve() {
    ll n, m;
    cin >> n >> m;
    cout << n*(m - 1) + m*(n - 1) << "\n";
}
```

Fence 3

D 小红嫁接

对于一个度大于 2 的点，把 $d-2$ 条边移走即可。

```cpp
void solve() {
    int n;
    cin >> n;
    vector<int> a(n + 1);
    for (int i = 1; i < n; ++i) {
        int u, v;
        cin >> u >> v;
        a[u]++, a[v]++;
    }
    int ans = 0;
    for (int i = 1; i <= n; ++i) {
        if (a[i] > 2) ans += a[i] - 2;
    }
    cout << ans << "\n";
}
```

Fence 4

E 小红玩马

先 BFS 求到 (x_2, y_2) 的距离，距离如果大于 k 或者到达不了就肯定不行。

然后找到这条路径，如果这条路径的长度 n 满足 $k-n$ 为偶数，我们就能在 n 步到达终点后通过跳过来跳回去的方法消耗偶数个步数。

关于只有偶数个步数才能被消耗的证明：

- 记当前坐标为 (x, y) ，由于是日字形移动，所以在移动一次后，横纵坐标之和的奇偶性必然改变。
- 故只有偶数步才能回到原点。

注意需要特判当起点与终点相同，且步数为偶数时，棋盘上是否存在可以移动的点。

```cpp
void solve() {
    int n, m, k;
    cin >> n >> m >> k;
    int x1, y1, x2, y2;
    cin >> x1 >> y1 >> x2 >> y2;

    auto check = [&](int x, int y) -> bool{
        return 1 <= x && x <= n && 1 <= y && y <= m;
    };
    auto encode = [&](int x, int y) {
        return (x - 1) * m + (y - 1);
    };
    auto decode = [&](int id)->PII{
        return {id / m + 1, id % m + 1};
    };
    int s = encode(x1, y1), t = encode(x2, y2);

    vector<int> dist(n * m, -1), par(n * m, -1);
    queue<int> q;

    dist[s] = 0;
    q.push(s);
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        auto [x, y] = decode(u);
        for (int i = 0; i < 8; i++) {
            int nx = x + dx[i];
            int ny = y + dy[i];
            if (!check(nx, ny)) continue;
            int v = encode(nx, ny);
            if (dist[v] == -1) {
                dist[v] = dist[u] + 1;
                par[v] = u;
                q.push(v);
            }
        }
    }
    if (dist[t] == -1 || k < dist[t] || ((k - dist[t]) & 1)) {
        cout << "No" << "\n";
        return;
    }

    if (dist[t] == 0) {
        if (k == 0) {
            cout << "Yes" << "\n";
            return;
        }
        int f = -1;
        auto [sx, sy] = decode(s);
        for (int i = 0; i < 8; i++) {
            int nx = sx + dx[i];
            int ny = sy + dy[i];
            if (check(nx, ny)) {
                f = encode(nx, ny);
                break;
            }
        }
        if (f == -1) {
            cout << "No" << "\n";
            return;
        }
        cout << "Yes" << "\n";
        int cycles = k / 2;
        for (int i = 0; i < cycles; i++) {
            auto p1 = decode(f);
            cout << p1.fi << " " << p1.se << "\n";
            auto p2 = decode(s);
            cout << p2.fi << " " << p2.se << "\n";
        }
        return;
    }

    cout << "Yes" << "\n";

    vector<int> ans;
    int cur = t;
    while (cur != -1) {
        ans.pb(cur);
        cur = par[cur];
    }
    reverse(all(ans));
    int d = ans.size() - 1;
    int c = (k - d) / 2;

    for (int i = 1; i < ans.size(); i++) {
        auto [u, v] = decode(ans[i]);
        cout << u << " " << v << "\n";
    }
    auto [u1, v1] = decode(ans[ans.size() - 2]);
    auto [u2, v2] = decode(ans[ans.size() - 1]);
    for (int i = 0; i < c; i++) {
        cout << u1 << " " << v1 << "\n";
        cout << u2 << " " << v2 << "\n";
    }
}
```

F 小红的⑨

- 下推 $dp1[u][d]$：节点 u 的子树中距离 u 恰好为 d 的节点数；
- 上推 $dp2[u][d]$：不在 u 子树（即“向上及其它分支”）中距离 u 恰好为 d 的节点数；
- 记 $K = 9$

对于 $child[v]$（父是 u），可以用 $total[u][d-1]$（即 u 的所有方向在距离 $d-1$ 的节点数）减去来自 v 子树的那些在 u 距离为 $d-1$ 的节点（对应 $dp1[v][d-2]$）得到 $dp2[v][d]$；

最终每个节点答案为 $dp1[u][K] + dp2[u][K]$（等于 $total[u][K]$）。

```
void solve() {
    int n;
    cin >> n;
    vector<vector<int>> g(n + 1);
    for (int i = 1; i < n; ++i) {
        int u, v;
        cin >> u >> v;
        g[u].pb(v);
        g[v].pb(u);
    }

    vector<int> fa(n + 1, -1);
    vector<int> order;
    order.pb(1);
    fa[1] = -1;
    for (int i = 0; i < order.size(); ++i) {
        int u = order[i];
        for (int v : g[u]) {
            if (v == fa[u]) continue;
            fa[v] = u;
            order.pb(v);
        }
    }

    vector<array<int, 10>> dp1(n + 1), dp2(n + 1);
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j <= 9; ++j) {
            dp1[i][j] = 0;
            dp2[i][j] = 0;
        }
    }

    for (int i = n - 1; i >= 0; --i) {
        int u = order[i];
        dp1[u][0] = 1;
        for (int v : g[u]) {
            if (v == fa[u]) continue;
            for (int j = 1; j <= 9; ++j) {
                dp1[u][j] += dp1[v][j - 1];
            }
        }
    }

    for (int i = 0; i < n; ++i) {
        int u = order[i];
        array<int, 10> tot;
        for (int j = 0; j <= 9; ++j) tot[j] = dp1[u][j] + dp2[u][j];

        for (int v : g[u]) {
            if (v == fa[u]) continue;
            dp2[v][0] = 0;
            for (int j = 1; j <= 9; ++j) {
                int t = tot[j - 1];
                if (j - 2 >= 0) t -= dp1[v][j - 2];
                dp2[v][j] = t;
            }
        }
    }

    for (int i = 1; i <= n; ++i) {
        cout << dp1[i][9] + dp2[i][9] << " ";
    }
    cout << "\n";
}
```

Fence 6