A. 二叉树的遍历

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 100010;

int lson[N], rson[N];

void dfs(int u,int op)
{
        if(op == 1)
                cout << u << ' ';
        if(lson[u] != -1)
                dfs(lson[u], op);
        if(op == 2)
                cout << u << ' ';
        if(rson[u] != -1)
                dfs(rson[u], op);
        if(op == 3)
                cout << u << ' ';
}

int main()
{
        int n;
        cin >> n;
        for(int i = 1; i <= n; i ++)
        {
                cin >> lson[i] >> rson[i];
        }
        dfs(1, 1);
        cout << '\n';
        dfs(1, 2);
        cout << '\n';
        dfs(1, 3);
        cout << '\n';

}
```

B. 树的重心

```
#include<bits/stdc++.h>
using namespace std;

const int N = 100010;

vector<int> edge[N];

int siz[N];
int n, ans = 1e9;

void dfs(int u,int p)
{
        siz[u] = 1;
        int mx = 0;
        for(int v : edge[u])
        {
                if(v == p) continue;
                dfs(v, u);
                siz[u] += siz[v];
                mx = max(mx, siz[v]);
        }
        mx = max(mx, n - siz[u]);
        ans = min(ans, mx);
}

int main()
{
        cin >> n;
        for(int i = 1; i <= n - 1; i ++)
        {
                int u, v;
                cin >> u >> v;
                edge[u].push_back(v);
                edge[v].push_back(u);
        }
    dfs(1, -1);
        cout << ans << '\n';
}
```

C. 间谍风云

实际上就是求树的重心，注意一下如果树有两个重心应该取编号较小的一个。

```cpp
#include <bits/stdc++.h>
using namespace std;
#define IOS ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
// #define int long long
#define endl '\n'
#define all(_x) _x.begin(), _x.end()
#define range(_x, _st, _ed) (_x.begin() + _st), (_x.begin() + _ed)
#define rep(_x, _y, _z) for (int _x = _y; _x <= _z; _x++)
#define matrix(_x, _y, _z) vector<vector<int>>(_x, vector<int>(_y, _z))
#define debug(_x) cout << #_x << '=' << _x << endl
using i64 = long long;
using i128 = __int128;
using pii = pair<int, int>;
using mat = vector<vector<int>>;
using u64 = unsigned long long;
constexpr int N = 1e5 + 10;
// dont use umap!!!

vector<int> edge[N];

int siz[N];

int ans = 1e9, ansp, n;

void dfs(int u,int p)
{
    int mx = 0;
    siz[u] = 1;
    for(int v : edge[u])
    {
        if(v == p) continue;
        dfs(v, u);
        mx = max(mx, siz[v]);
        siz[u] += siz[v];
    }
    mx = max(mx, n - siz[u]);
    if(mx < ans) ans = mx, ansp = u;
    else if(mx == ans) ansp = min(ansp, u);
}

signed main()
{
    IOS;
    cin >> n;
    for(int i = 1; i <= n - 1; i ++)
    {
        int u, v;
        cin >> u >> v;
        edge[u].push_back(v);
        edge[v].push_back(u);
    }
    dfs(1, -1);
    cout << ansp << '\n';
    return 0;
}
```

D. 树的直径

```
#include<bits/stdc++.h>
using namespace std;

const int N = 100010;

vector<int> edge[N];

int siz[N];
int n, ans = 1e9;

void bfs(int st,int &P,int &D)
{
        P = D = 0;
        queue<int> q;
        q.push(st);
        vector<int> dist(n + 1, 1e9);
        dist[st] = 0;
        while(q.size())
        {
                int u = q.front();
                q.pop();
                for(int v : edge[u])
                {
                        if(dist[v] > dist[u] + 1)
                        {
                                dist[v] = dist[u] + 1;
                                if(dist[v] > D)
                                {
                                        D = dist[v];
                                        P = v;
                                }
                                q.push(v);
                        }
                }
        }
}

int main()
{
        cin >> n;
        for(int i = 1; i <= n - 1; i ++)
        {
                int u, v;
                cin >> u >> v;
                edge[u].push_back(v);
                edge[v].push_back(u);
        }
        int p, d;
        bfs(1, p, d);
        bfs(p, p, d);
        cout << d << '\n';
}
```

E. 图上最短路I

简单bfs即可，跟走迷宫一个原理。

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 100010;

int n, m;
vector<int> edge[N];

int bfs(int st,int ed)
{
        queue<int> q;
        q.push(st);
        vector<int> dist(n + 1, -1);
        dist[st] = 0;
        while(q.size())
        {
                int u = q.front();
                q.pop();
                for(int v : edge[u])
                {
                        if(dist[v] == -1)
                        {
                                dist[v] = dist[u] + 1;
                                q.push(v);
                        }
                }
        }
        return dist[ed];
}

int main()
{
        int ST, ED;
        cin >> n >> m >> ST >> ED;
        for(int i = 1; i <= m; i ++)
        {
                int u, v;
                cin >> u >> v;
                edge[u].push_back(v);
        }
        int ans = bfs(ST, ED);
        if(ans == 1e9) cout << "-1\n";
        cout << ans << '\n';
}
```

F. 安全员Orange

注意到，在 u 和 v 之间连接一条新便刚好可以减少 $distance(u,v)$ 的距离。那么根据贪心的思想，一定是连接树上 $distance(u,v)$ 最大的两点，即树的直径两端，最后可以减少树的直径长度 -1 个单位的时间(因为新边需要 1 个单位的时间遍历)。

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 100010;

vector<int> edge[N];

int siz[N];
int n, ans = 1e9;

void bfs(int st,int &P,int &D)
{
        P = D = 0;
        queue<int> q;
        q.push(st);
        vector<int> dist(n + 1, 1e9);
        dist[st] = 0;
        while(q.size())
        {
                int u = q.front();
                q.pop();
                for(int v : edge[u])
                {
                        if(dist[v] > dist[u] + 1)
                        {
                                dist[v] = dist[u] + 1;
                                if(dist[v] > D)
                                {
                                        D = dist[v];
                                        P = v;
                                }
                                q.push(v);
                        }
                }
        }
}

int main()
{
        cin >> n;
        for(int i = 1; i <= n - 1; i ++)
        {
                int u, v;
                cin >> u >> v;
                edge[u].push_back(v);
                edge[v].push_back(u);
        }
        int p, d;
        bfs(1, p, d);
        bfs(p, p, d);
        cout << 2 * (n - 1) - d + 1 << '\n';
}
```

G. 拓扑排序

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 100010;

vector<int> edge[N];

int din[N];

int main()
{
        int n, m;
        cin >> n >> m;
        for(int i = 1; i <= m; i ++)
        {
                int u, v;
                cin >> u >> v;
                edge[u].push_back(v);
                din[v]++;
        }
        queue<int> q;
        vector<int> toporder;
        for(int i = 1; i <= n; i ++)
        {
                if(din[i] == 0) q.push(i);
        }
        while(q.size())
        {
                int u = q.front();
                q.pop();
                toporder.push_back(u);
                for(int v : edge[u])
                {
                        if(--din[v] == 0) q.push(v);
                }
        }
        for(int i = 0; i < toporder.size(); i ++)
                cout << toporder[i] << ' ';
}
```

H. Orange的旅行计划

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 100010;

vector<int> edge[N];

int din[N], f[N];

int main()
{
        int n, m;
        cin >> n >> m;
        for(int i = 1; i <= m; i ++)
        {
                int u, v;
                cin >> u >> v;
                edge[u].push_back(v);
                din[v]++;
        }
        queue<int> q;
        vector<int> toporder;
        for(int i = 1; i <= n; i ++)
        {
                if(din[i] == 0) q.push(i), f[i] = 1;
        }
        while(q.size())
        {
                int u = q.front();
                q.pop();
                toporder.push_back(u);
                for(int v : edge[u])
                {
                        if(--din[v] == 0) q.push(v);
                }
        }
        for(int u : toporder)
        {
                for(int v : edge[u])
                {
                        f[v] = max(f[v], f[u] + 1);
                }
        }
        for(int i = 1; i <= n; i ++) cout << f[i] << '\n';
}
```

I. 奖金

我们将题目中所有的语句成为 断言 ： 实际上，对于任意一条断言 $u \to v$ (表示 u 的工资应该大于 v)，实际上，u 应该满足的是:

$$u > \max v_i$$

也就是所有关于 u 的断言中，最大的一条，这样显然满足了所有 u 的断言，因此 $u = \max v_i + 1$。而确定 u 之前应该保证所有的 v_i 均已经被确认，符合拓扑序的思想，因此对所有的断言做一次拓扑排序，再依次更新即可。注意，拓扑排序的起点 $u = 100$(题目要求)。

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 100010;

vector<int> edge[N];

int din[N];

int main()
{
        int n, m;
        cin >> n >> m;
        for(int i = 1; i <= m; i ++)
        {
                int u, v;
                cin >> u >> v;
                edge[v].push_back(u);
                din[u]++;
        }
        queue<int> q;
        vector<int> toporder, f(n + 1, 100);
        for(int i = 1; i <= n; i ++)
        {
                if(din[i] == 0) q.push(i);
        }
        while(q.size())
        {
                int u = q.front();
                q.pop();
                toporder.push_back(u);
                for(int v : edge[u])
                {
                        if(--din[v] == 0) q.push(v);
                }
        }
    if(toporder.size() < n)
    {
        cout << "Poor Xed\n";
        return 0;
    }
        for(int u : toporder)
        {
                for(int v : edge[u])
                {
                        f[v] = max(f[v], f[u] + 1);
                }
        }
    int ans = 0;
        for(int i = 1; i <= n; i ++) ans += f[i];
    cout << ans << '\n';
}
```

J. 可达性统计

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 100010;

vector<int> edge[N];

int din[N];
bitset<30010> f[N];

int main()
{
        int n, m;
        cin >> n >> m;
        for(int i = 1; i <= m; i ++)
        {
                int u, v;
                cin >> u >> v;
                edge[u].push_back(v);
                din[v]++;
        }
        queue<int> q;
        vector<int> toporder;
        for(int i = 1; i <= n; i ++)
        {
                if(din[i] == 0) q.push(i);
        }
        while(q.size())
        {
                int u = q.front();
                q.pop();
                toporder.push_back(u);
                for(int v : edge[u])
                {
                        if(--din[v] == 0) q.push(v);
                }
        }
        for(int i = toporder.size() - 1; i >= 0; i --)
        {
                int u = toporder[i];
                f[u][u] = 1;
                for(int v : edge[u])
                {
                        f[u] |= f[v];
                }
        }
        for(int i = 1; i <= n; i ++) cout << f[i].count() << '\n';
}
```

K. 跳石头

发现本题跟上一题题意完全相同，因此只需要略微改动建图部分和计算答案部分即可，其他代码完全照搬上一题。

```cpp
#include <bits/stdc++.h>
using namespace std;
#define IOS ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
using i64 = long long;
using i128 = __int128;
using pii = pair<int, int>;
using mat = vector<vector<int>>;
using u64 = unsigned long long;
constexpr int N = 4e4 + 10;
// dont use umap!!!

vector<int> edge[N];

bitset<N> vis[N];

signed main()
{
    IOS;
    int n;
    cin >> n;
    vector<int> c(n + 1), din(n + 1);
    for(int i = 1; i <= n; i ++) cin >> c[i];
    for(int i = 1; i <= n; i ++)
    {
        if(2 * i <= n) edge[i].push_back(2 * i), din[2 * i]++;
        if(i + c[i] <= n) edge[i].push_back(i + c[i]), din[i + c[i]]++;
    }
    queue<int> q;
    vector<int> toporder;
    for(int i = 1; i <= n; i ++)
    {
        if(!din[i]) q.push(i);
    }
    while(q.size())
    {
        auto u = q.front();
        toporder.push_back(u);
        q.pop();
        for(auto v : edge[u])
            if(--din[v] == 0) q.push(v);
    }
    for(int i = toporder.size() - 1; i >= 0; i --)
    {
        int u = toporder[i];
        vis[u][u] = 1;
        for(auto v : edge[u])
            vis[u] |= vis[v];
    }
    int ans = 0;
    for(int i = 1; i <= n; i ++)
        ans = max(ans, (int)vis[i].count());
    cout << ans << '\n';
    return 0;
}
```