动态规划 树形

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <cstring>
#include <set>
#include <map>
#include <queue>
#include <ctime>
#include <random>
#include <sstream>
#include <numeric>
#include <stack>
#include <stdio.h>
#include <algorithm>
using namespace std;

#define Multiple_groups_of_examples
#define rep(i,x,n) for(int i = x; i <= n; i++)
#define vf first
#define vs second

typedef long long LL;
typedef pair<int,int> PII;

const int INF = 0x3f3f3f3f;
const int N = 1e2 + 21;
namespace golitter {
namespace treedp {

    // 树上连续片段最大
    // https://ac.nowcoder.com/acm/problem/202475
void NC202475() {
    int n; cin>>n;
    vector<vector<int>> g(n+1);
    vector<int> val(n+1);
    vector<int> f(n+1, -INF), vis(n+1);
    for(int i = 1; i <= n; ++i) cin>>val[i];
    for(int i = 1; i < n; ++i) { // 读入
        int u,v; cin>>u>>v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    int res = -INF;
    auto dfs = [&](auto&& dfs, int u, int fu) -> void { // 树形dp模板
        f[u] = val[u];
        for(auto y: g[u]) { // 遍历子结点
            if(y == fu) continue;
            dfs(dfs,y, u);
            res = max(res, f[u] + f[y]); // 以u为父亲的两个子结点组成的最大链长
```

```
            f[u] = max(f[u], f[y] + val[u]); // 一个子结点组成的最大链长
        }
        res = max(res, f[u]);
    };
    dfs(dfs,1, 0);
    cout<<res;
}
    // 最大独立集
    // https://ac.nowcoder.com/acm/problem/51178
void NC51178() {
    int n; cin>>n;
    vector<int> val(n+1),vis(val);
    vector<vector<int>> g(n+1);
    vector<vector<int>> f(n+1, vector<int>(2));
    for(int i = 1; i <= n; ++i) cin>>val[i];
    int u,v;
    while(cin>>u>>v, u != 0 || v != 0) {
        vis[u] = 1; // 找没有父节点的那个根
        g[v].push_back(u);
    }
    int root = 0;
    for(int i = 1; i <= n; ++i) if(!vis[i]) root = i; // 找根
    auto dfs = [&](auto &dfs, int u) -> void {
        f[u][0] = 0;
        f[u][1] = val[u];
        for(auto t: g[u]) {
            dfs(dfs, t);
            f[u][0] += max(f[t][0], f[t][1]);
            f[u][1] += f[t][0];
        }
    };
    dfs(dfs, root);
    cout<<max(f[root][0], f[root][1]);
}

void NC51222() {
    int n;
    while(cin>>n) {

    vector<vector<int>> g(n);
    vector<int> vis(n);
    vector<vector<int>> f(n, vector<int>(2));
    for(int i = 0; i < n; ++i) {
        int u,cnt; scanf("%d:(%d)", &u, &cnt);
        for(int j = 0; j < cnt; ++j) {
            int v; cin>>v;
            g[u].push_back(v);
            g[v].push_back(u);
        }
    }
    auto dfs = [&](auto&&dfs, int u, int fu) -> void {
        f[u][0] = 0;
        f[u][1] = 1;
        vis[u] = 1;
        for(auto y: g[u]) {
            if(y ==fu) continue;
            dfs(dfs, y, u);
            f[u][0] += f[y][1];
```

```
            f[u][1] += min(f[y][0], f[y][1]);
        }
    };
    int ans = 0;
    for(int i = 0; i < n; ++i) {
        if(vis[i]) continue;
        dfs(dfs, i, -3);
        ans += min(f[i][1], f[i][0]);
    }
    cout<<ans<<endl;

    }
}

}}
```