CF1379C

有 m 种花，每种花第一次选有 a_i 的快乐值，之后每次有 b_i 的快乐值。要选 n 朵花，并最大化快乐值

直接贪心似乎没什么思路，考虑比较暴力的做法：枚举第 i 种花，把所有 $a_j \ge b_i$ 的其他花先取一次，剩下全部取第 i 种花。

```
#include<bits/stdc++.h>
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define db double
#define ull unsigned long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 1e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

int a[maxn],b[maxn];

int main()
{
    fastio();
    int _,n,m;
    for(cin>>_;_;_--){
        cin>>n>>m;
        ll ans = 0;
        vector<ll> vec,suf;
        rep(i,1,m){
            cin>>a[i]>>b[i];
            vec.pb(a[i]);
        }
        sort(ALL(vec));
        per(i,m-1,0){
            if(i==m-1) suf.pb(vec[i]);
            else suf.pb(suf.back() + vec[i]);
            //show1(suf.back());
        }
        rep(i,1,m){
            int pos = lower_bound(ALL(vec),b[i]) - vec.begin();
```

```
            int r = m - pos;
            //show2(b[i],r);
            if(r>=n){
                //show1(suf[n-1]);
                ans = max(ans,suf[n-1]);
            }else{
                ll tmp = r?suf[r-1]:0;
                if(a[i] >= b[i]){
                    tmp += (ll) b[i] * (n-r);
                }else{
                    tmp += (ll)a[i] + (ll)b[i] * (n-r-1);
                }
                //show1(tmp);
                ans = max(ans,tmp);
            }
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

CF1385F

给一棵树，每次选 k 个叶子删掉，问最多能操作多少次。

每次找儿子叶子最多的一直删即可，关键是要维护每个点的儿子叶子的个数。可以用线段树或者 set 维护。

```
#include<bits/stdc++.h>
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define db double
#define ull unsigned long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e5+5;
```

```
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}
int cnt[maxn],n,k;
vector<int> g[maxn];
set<int> sg[maxn];
struct node
{
    int id,num;
    bool operator < (const node e) const
    {
        if(num != e.num) return num < e.num;
        return id < e.id;
    }
}tr[maxn<<2];
void dfs(int u,int f)
{
    cnt[u] = 0;
    for(int v:g[u])if(v!=f){
        dfs(v,u);
        if(g[v].size()==1) cnt[u]++;
    }
}
void push_up(int id)
{
    tr[id] = max(tr[id<<1],tr[id<<1|1]);
}
void build(int id,int l,int r)
{
    if(l==r){
        tr[id] = (node){l,cnt[l]};
        return ;
    }
    int mid = (l+r)>>1;
    build(id<<1,l,mid);build(id<<1|1,mid+1,r);
    push_up(id);
}
void update(int id,int stl,int str,int pos,int flag)
{
    if(stl==str){
        if(!flag) tr[id].num %= k;
        else tr[id].num++;
        return ;
    }
    int mid = (stl+str)>>1;
    if(pos<=mid) update(id<<1,stl,mid,pos,flag);
    else update(id<<1|1,mid+1,str,pos,flag);
    push_up(id);
}
int main()
{
    fastio();
    int _;
```

```
    for(cin>>_;_;_--){
        cin>>n>>k;
        rep(i,1,n) g[i].clear(),sg[i].clear();
        rep(i,1,n-1){ int u,v;
            cin>>u>>v;
            g[u].pb(v);
            g[v].pb(u);
        }
        if(k==1){
            cout<<n-1<<endl;continue;
        }
        rep(i,1,n){
            if(g[i].size()!=1) {dfs(i,0);break;}
        }
        rep(u,1,n){
            for(int v:g[u]) if(g[v].size() > 1) sg[u].insert(v);
        }
        build(1,1,n);
        int ans = 0;
        while(1){
            node tmp = tr[1];
            if(tmp.num < k) break;
            int num = tmp.num / k;
            ans += num;
            update(1,1,n,tmp.id,0);
            cnt[tmp.id] -= num*k;
            if(cnt[tmp.id]==0 && sg[tmp.id].size()==1){
                int f = *sg[tmp.id].begin();
                sg[f].erase(sg[f].find(tmp.id));
                cnt[f]++;
                update(1,1,n,f,1);
            }
        }
        cout<<ans<<endl;
    }
    return 0;
}
</hidden>
```