1. P8314 Parkovi

二分答案，之后考虑自下往上贪心的放点，原则是若在父亲放点仍然能覆盖子树中最深的未覆盖点，就不会在当前点放，因为我们让子树内合法的同时，对子树外的覆盖范围还扩大了。当然这里要处理一下根的情况。具体的，算处理完 x 所在子树后只需记下来最深的未覆盖点到 x 的距离，以及最浅的放的点到 x 的距离即可。

```
#include<bits/stdc++.h>
using namespace std;
int n,M;
#define ll long long
int U[301000],V[301000];ll W[300100];
vector<int>g[301000];
const ll I=1e18;
ll a[301000],b[301000],m;
int su;
bool vis[301000];
void dfs(int x,int f,ll pw){
    a[x]=I,b[x]=-1;
    for(int i:g[x]){
        int v=U[i]^V[i]^x,w=W[i];if(v==f)continue;
        dfs(v,x,w),a[x]=min(a[x],a[v]+w);
        if(b[v]!=-1)b[x]=max(b[x],b[v]+w);
    }
    b[x]=max(b[x],0ll);
    if(b[x]!=-1&&a[x]+b[x]<=m)b[x]=-1;
    if(b[x]!=-1&&(x==1||b[x]+pw>m))a[x]=0,b[x]=-1,su++,vis[x]=1;
}
bool chk(){
    memset(vis,0,sizeof(vis));
    su=0,dfs(1,0,0);return su<=M;
}
#define pb push_back
int main(){
    scanf("%d%d",&n,&M);
    for(int
i=1;i<n;i++)scanf("%d%d%lld",&U[i],&V[i],&W[i]),g[U[i]].pb(i),g[V[i]].pb(i);
    ll l=0,r=1e15,ans=-1;while(l<=r){
        ll mi=(l+r)>>1;m=mi;
        if(chk())ans=mi,r=mi-1;
        else l=mi+1;
    }
    m=ans,chk();
    printf("%lld\n",m);
    vector<int>X;
    for(int i=1;i<=n;i++)if(vis[i])X.pb(i),M--;
    for(int i=1;i<=n;i++)if(!vis[i]&&M)X.pb(i),M--;
    sort(X.begin(),X.end());for(int x:X)printf("%d ",x);
    return 0;
}
```

2. P4698 Hotel

先思考选的次数不受限制时的做法，考虑贪心，把房间按费用排序，每次用最小费用的房间接它可承受范围内的最值订单，不难通过反证说明这样取是最优的。

然后考虑选的次数受限制怎么办。由于这个题实际上可以建图费用流解决，因此是具有凸性的，wqs 二分即可。

```
#include<bits/stdc++.h>
using namespace std;
int n,m,o;
#define ll long long
struct qq{ll x,y;int t;}a[1000100];
ll su;int jz;
bool chk(ll K){
    priority_queue<ll>q;
    su=0,jz=0;
    for(int i=1;i<=n+m;i++){
        if(a[i].t){if(!q.empty()&&q.top()-a[i].x-K>=0)jz++,su+=q.top()-a[i].x-
K,q.pop();}
        else q.push(a[i].x);
    }
    return jz>=o;
}
int main(){
    scanf("%d%d%d",&n,&m,&o);
    for(int i=1;i<=n;i++)scanf("%lld%lld",&a[i].x,&a[i].y),a[i].t=1;
    for(int i=n+1;i<=n+m;i++)scanf("%lld%lld",&a[i].x,&a[i].y),a[i].t=0;
    sort(a+1,a+n+m+1,[&](qq a,qq b){if(a.y!=b.y)return a.y<b.y;if(a.t!=b.t)return
a.t<b.t;return a.x<b.x;});
    if(!chk(0)){printf("%lld",su);return 0;}
    ll l=0,r=1e18,at=-1;
    while(l<=r){
        ll mi=(l+r)>>1;
        if(chk(mi))at=mi,l=mi+1;
        else r=mi-1;
    }
    chk(at);
    printf("%lld\n",su+o*at);
    return 0;
}
```

3. P9128 Fertilizing Pastures G

先考虑 $T=0$ 时的做法，考虑遍历顺序对费用产生的影响，对每个子树记录二元组 (a,b) 为该子树的大小和权值和，那么我们要让这些二元组排序，使 $\sum\limits_{i<j} a_ib_j$ 最小。这是经典问题，按 $\frac{a}{b}$ 从小到大排序即可。

然后考虑 T=1，那这个时候要取根到某个最深的叶子的路径，这个路径上的点代表的子树都必须最后一个遍历。考虑 dp ，记 dp_i 为子树 i 内要满足这个限制的最小费用，f_i 为无限制的最小费

用, d_i 是 i 到最深叶子的距离，则我们选一个儿子满足 $d_i = d_v + 1$ ，不难通过记录一些前缀信息得到强制 i 最后一个遍历时的最小费用，然后再加上 dp_v 和其他儿子的 f_v 贡献给 dp_i 即可。

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
int n,d[301000],sz[301000],typ;
ll a[301000],sw[301000];
vector<int>g[300100];
#define pb push_back
ll dp[301000],dp2[300100],qs[300100];
void dfs(int x){
    sz[x]=1,sw[x]=a[x];
    dp[x]=0;
    for(int
v:g[x])dfs(v),sz[x]+=sz[v],sw[x]+=sw[v],dp[x]+=sw[v],dp[x]+=dp[v],d[x]=max(d[x],d[v]+1
);
    sort(g[x].begin(),g[x].end(),[&](int a,int b){return sw[b]*sz[a]<sw[a]*sz[b];});
    ll ks=0;
    for(int v:g[x])qs[v]=ks,dp[x]+=ks*sw[v]*2,ks+=sz[v];
    if(g[x].empty())return;dp2[x]=2e18;
    reverse(g[x].begin(),g[x].end());
    ks=0;
    for(int v:g[x]){
        if(d[v]+1==d[x])
            dp2[x]=min(dp2[x],dp[x]+dp2[v]-dp[v]-qs[v]*sw[v]*2+sw[v]*(sz[x]-
sz[v]-1)*2-ks*sz[v]*2);
        ks+=sw[v];
    }
}
int main(){
    scanf("%d%d",&n,&typ);
    for(int i=2,f;i<=n;i++)scanf("%d%lld",&f,&a[i]),g[f].pb(i);
    dfs(1);
    if(typ)printf("%d %lld",(n-1)*2-d[1],dp2[1]);
    else printf("%d %lld",(n-1)*2,dp[1]);
    return 0;
}
```

4. AGC034C Tests

你发现取的 a_i 是有良好性质的，我们可以分类讨论说明：若存在 $i \neq j$ 满足 $0 < a_i, a_j < X$ ，则一定不优，我们可以调整。

于是只有至多一个数满足 $0 < a_i < X$ 。先二分答案 mid，然后枚举这个数，显然此时 $a_i = X \mod mid$ ，容易计算其贡献。再在剩下的数中尽量选 $(X - b_i)r_i + b_i l_i$ 最大的，我们一开始就把三元组按这个值排序，再求前缀和即可。

```cpp
#include<bits/stdc++.h>
using namespace std;
#define ll long long
int n;ll X,O;
struct qq{ll b,l,r,v;}a[101000];
ll su[101000];
vector<pair<ll,ll> >e;
#define mp make_pair
#define pb push_back
#define fi first
#define se second
ll val(int x,ll z){
    if(z<=a[x].b)return z*a[x].l;
    return a[x].b*a[x].l+(z-a[x].b)*a[x].r;
}
bool chk(ll S){
    if(S==n*X)return 1;
    ll ans=-1e18;
    for(int i=1;i<=n;i++){
        if(S/X<i)ans=max(ans,su[S/X]+val(i,S%X));
        else ans=max(ans,su[S/X+1]-a[i].v+val(i,S%X));
    }
    return ans+O>=0;
}
int main(){
    scanf("%d%lld",&n,&X);
    for(int i=1;i<=n;i++)scanf("%lld%lld%lld",&a[i].b,&a[i].l,&a[i].r),
    a[i].v=a[i].r*(X-a[i].b)+a[i].l*a[i].b,O-=a[i].l*a[i].b;
    sort(a+1,a+n+1,[&](qq x,qq y){return x.v>y.v;});
    for(int i=1;i<=n;i++)su[i]=su[i-1]+a[i].v;
    ll l=0,r=X*n,ans=-1;
    while(l<=r){
        ll mid=(l+r)>>1;
        if(chk(mid))ans=mid,r=mid-1;
        else l=mid+1;
    }
    return printf("%lld",ans),0;
}
```

5. CF436E Cardboard Box

令 $b_i := a_i - b_i$ 。

考虑如果所有 i 都满足： $a_i \le b_i$ ，就可以把所有的 a_i, b_i 拉到一起排序，然后求前 w 小的数之和。

再来思考如果都满足 $a_i > b_i$ 怎么办。对于两个满足 $a_i > b_i$ 的数 i_1, i_2 ，我们可以发现一个很好的结论：不可能同时在关卡 i_1, i_2 取恰好一颗星，理由是令 $a_{i_1} \le a_{i_2}$ ，那由于 $b_{i_1} < a_{i_1}$ ，

把 a_{i_2} 换成 b_{i_1} 更优。

于是把它们按 $a_i + b_i$ 排序，发现 w 为偶数时肯定就会对于前 $\frac{w}{2}$ 个关卡，每个关卡取两颗星。w 为奇数时设 $w = 2k + 1$，发现要么会让前 k 个关卡都打两遍，在剩下的部分里取 a 最小的；要么先对前 $k + 1$ 个关卡打两遍，然后让这些关卡里 b 最大的只打一次。预处理一些东西就可以计算了。

最后，把这两部分拼起来就好了，枚举其中一部分打了多少星即可。复杂度 $O(n \log n)$

```
#include<bits/stdc++.h>
using namespace std;
int n,m;
#define ll long long
#define pb push_back
#define pi pair<ll,ll>
#define F first
#define S second
#define mp make_pair
vector<pi>e1;
vector<array<ll,3> >e2;
ll o1[301000],o2[300100],o3[300100];
int tt[301000];
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1,a,b;i<=n;i++){
        scanf("%d%d",&a,&b);b-=a;
        if(a>=b)e2.pb({a+b,a,i});
        else e1.pb(mp(a,i)),e1.pb(mp(b,i));
    }
    sort(e1.begin(),e1.end());sort(e2.begin(),e2.end());
    int s1=e1.size(),s2=e2.size();
    for(int i=1;i<s1;i++)e1[i].F+=e1[i-1].F;
    for(int i=0;i<s2;i++)o1[i+1]=o1[i]+e2[i][0];
    o2[s2+1]=1e18;
    for(int i=s2;i;i--)o2[i]=min(o2[i+1],e2[i-1][1]);
    o3[0]=-1e18;
    for(int i=0;i<s2;i++)o3[i+1]=max(o3[i],e2[i][0]-e2[i][1]);
    ll ans=1e18;
    int zi=-1;
    for(int i=0;i<=m&&i<=s2*2;i++)if(m-i<=s1){
        ll tz;
        if(!(i&1))tz=o1[i/2];
        else tz=min(o1[i/2]+o2[i/2+1],o1[i/2+1]-o3[i/2+1]);
        ll vp=tz+((i==m)?0:e1[m-1-i].F);
        if(ans>vp)ans=vp,zi=i;
    }
    for(int i=0;i<m-zi;i++)tt[e1[i].S]++;
    if(!(zi&1)){for(int i=0;i<zi/2;i++)tt[e2[i][2]]=2;}
    else{
        int h=zi/2;
        if(o1[h]+o2[h+1]<o1[h+1]-o3[h+1]){
            for(int i=0;i<h;i++)tt[e2[i][2]]+=2;
            ll X=1e18;int ix=-1;
            for(int i=h;i<s2;i++)if(X>e2[i][1])X=e2[i][1],ix=e2[i][2];
            tt[ix]++;
        }
        else{
            for(int i=0;i<=h;i++)tt[e2[i][2]]+=2;
            ll X=-1e18;int ix=-1;
```

```
            for(int i=0;i<=h;i++)if(X<e2[i][0]-e2[i][1])X=e2[i][0]-e2[i][1],ix=e2[i]
[2];
            tt[ix]--;
        }
    }
    printf("%lld\n",ans);for(int i=1;i<=n;i++)printf("%d",tt[i]);
    return 0;
}
```

]

6. P5912 [POI2004] JAS

题意是求一个深度最小的点分树，我们转化为，对点标号使得对于一对点 $u \neq v$ ，若 $a_u = a_v$ ，则路径上存在至少一点使 $a_k > a_u$ ，最后让 a_u 的最大值最小。

我们知道每次取重心就是 $O(\log n)$ 的了，虽然这不一定是最优的，但可以帮助我们后面的计算。

可以感受到填 a_u 就直接从下往上填，每次填和子树内的数不冲突的最小数即可，感性理解一下，选最小的 a_u 对上面的点的影响是最好的。

我们记下来 S_u 为 u 子树内那些不存在比它大的祖先的 a_v 的集合。那么填 a_u 的时候，我们需要 $a_u > \max\{S_{v_1} \cap S_{v_2}\}$ ，且 $a_u \notin S_v$ 。由于 a_u 的值域是 $O(\log n)$ 的，我们只需二进制压下来 S 就好了。

现在来证明我们每次取最小的合法的点是不劣的，有一个很巧妙的证明。

发现 S_u 具有很好的性质。令 $V(S) = \sum\limits_{i \in S} B^i$ ，其中 B 是一个很大的数。

令 $G = \sum V(S_v)$ 。发现无论 a 怎么填 S_u 一定满足 $V(S_u) > G$ ，且 a 取最小值时 S_u 恰好是满足 $V > G$ 的集合中 val 最小者！这是一个很巧的事情。

于是贪心的填法取到了 $V(S_1)$ 可能的最小值，而 $V(S_1)$ 最小就让答案最小了。

复杂度 $O(n)$ 。

```cpp
#include<bits/stdc++.h>
using namespace std;
int n;
vector<int>g[101000];
#define pb push_back
int st[101000],t[101000],ans;
void dfs(int x,int f){
    int al=0,fk=0;
    for(int v:g[x])if(v!=f)dfs(v,x),fk|=st[v],al|=(st[x]&st[v]),st[x]|=st[v];
    int e=-1;
    for(int i=19;i>=0;i--)if((al>>i)&1){e=i;break;}
    e++;while((fk>>e)&1)e++;
    t[x]=e,st[x]|=(1<<t[x]);
    for(int i=0;i<t[x];i++)if((st[x]>>i)&1)st[x]^=(1<<i);
    ans=max(ans,t[x]);
}
int main(){
    scanf("%d",&n);
    for(int i=1,u,v;i<n;i++)scanf("%d%d",&u,&v),g[u].pb(v),g[v].pb(u);
    dfs(1,0);
    return printf("%d",ans),0;
}
```

7. P3543 [POI2012] WYR-Leveling Ground

令 $g = gcd(a, b)$，先把 A_i, a, b 都除掉 g，如果 A_i 不是 g 倍数就无解了。

令 $s_i = A_i - A_{i-1}$，我们相当于每次可以选两个数，让一个 +a/b 一个 -a/b。

我们先用 exgcd 解出 $ax_0 + by_0 = 1$ 的解，然后令 x_i, y_i 分别为 s_i 在 +a/-a 过程做的次数，y_i 为 +b/-b 时做的次数，则 $ax_i + by_i = s_i$，$\sum x_i = \sum y_i = 0$，然后要让 $\sum |x_i| + \sum |y_i|$ 最小。但其实这里由于 x, y 之和皆为 0，我们改成算 $\sum \max(0, x_i) + \max(0, -y_i)$，后面可以发现这样是很优雅的：

把 x_i, y_i 写成 $s_i x_0 + k_i b, s_i y_0 - k_i a$ 的形式。由于 $\sum s_i = 0$，我们就需要 $\sum k_i = 0$。然后其贡献就是 $\max(0, s_i x_0 + k_i b) + \max(0, k_i a - s_i y_0)$。可以发现这关于 k_i 是个分段的下凸函数。又要让费用最小，于是只需要做个闵可夫斯基和就好了，复杂度 $O(n \log n)$。

```cpp
#include<bits/stdc++.h>
using namespace std;
int n;
#define ll long long
ll gcd(ll a,ll b){
    if(!b)return a;
    return gcd(b,a%b);
}
#define pi pair<ll,ll>
#define mp make_pair
#define pb push_back
ll a[1001000],A,B,g,x_,y_;
void exgcd(ll a,ll b,ll &x,ll &y){
    if(!b){x=1,y=0;return;}
    exgcd(b,a%b,y,x),y-=(a/b)*x;
}
ll lz(ll a,ll b){if(a<0)return (a+1)/b-1;return a/b;}
ll lb(ll a,ll b){if(a<=0)return a/b;return (a-1)/b+1;}
int main(){
    scanf("%d%lld%lld",&n,&A,&B);g=gcd(A,B),A/=g,B/=g;
    exgcd(A,B,x_,y_);
    for(int i=1;i<=n;i++){
        scanf("%lld",&a[i]);
        if(a[i]%g){puts("-1");return 0;}
        a[i]/=g;
    }
    ll db=0,X=0;
    vector<pi>E;
    for(int i=1;i<=n+1;i++){
        ll e=a[i]-a[i-1],p=x_*e,q=-y_*e,r=lb(p,B),l=lb(q,A);
        X+=min(l,r)-1;
        if(l==r)E.pb(mp((A+B)*l-(p+q),1));
        else if(l<r){
            E.pb(mp(A*l-q,1));
            E.pb(mp(A,r-l-1));
            E.pb(mp((A+B)*r-p-A*(r-1),1));
        }
        else{
            swap(l,r);
            E.pb(mp(B*l-p,1));
            E.pb(mp(B,r-l-1));
            E.pb(mp((A+B)*r-q-B*(r-1),1));
        }
        //max(Bk-p,0)+max(Ak-q,0)
        //<l:0
        //[l,r):Ak-q
        //>=r:(A+B)k-(p+q)
    }
    if(X>=0){puts("0");return 0;}
    X=-X,sort(E.begin(),E.end());
```

```
    for(auto e:E){
        ll x=e.first,y=e.second;
        if(X>y)X-=y,db+=x*y;
        else{db+=X*x,X=0;break;}
    }
    db+=(A+B)*X;
    return printf("%lld",db),0;
}
```

8. P3571 [POI2014] SUP-Supercomputer

我们设根是第 1 层，令 s_i 为层数 $> i$ 的节点个数，可以发现答案有一个显然的下界为 $\max\limits_{\text{第}i\text{层存在点}} i+\lceil\frac{s_i}{k}\rceil$ 。

我们尝试构造一种操作取到这个下界。我们把题目的操作倒着看，当成每次要删掉至多 k 个叶子，只要每次操作让这个式子的值减 1 就好了。

令 p 是最大的满足 $s_p>0$ 的数。分类讨论：

如果 $s_p>k$ ，就直接在 $p+1$ 层删 k 个点，发现对于 $i\le p$,s_i 都减去了 k ，于是 $\lceil\frac{s_i}{k}\rceil$ 都会减去 1 。

如果 $s_p\le k$ ，直接删掉层数前 k 大的叶子。

我们令第 k 大的叶子层数为 q ，可以发现：

对于 $i\ge q$ ，发现第 i 层的非叶子的点数 $<k$ ，理由是一个非叶子的子树内一定有叶子。

于是操作后 $s_i-s_{i+1}\le k$, 则有 $i+\lceil\frac{s_i}{k}\rceil\le i+1+\lceil\frac{s_{i+1}}{k}\rceil$ ，这样最大值一定取在 $p-1$ 这个位置。

不难说明操作前这部分 max 是 p 。

对于 $i<q$ ，$i+\lceil\frac{s_i}{k}\rceil$ 一定减 1 。

这就证完了。

处理出 f_i 为最大的 j 使得 $s_j\ge i$ ， 那么我们处理 k 的答案时就枚举 $t=\lceil\frac{s_i}{k}\rceil$ 的值，用 $f_{(t-1)k+1}+t$ 更新答案即可。复杂度 $O(n\log n)$ 。

```
#include<bits/stdc++.h>
using namespace std;
int n,q,K[1001000],d[1010000],t[1001000],ans[1010000],fi[1010000];
int main(){
    scanf("%d%d",&n,&q);for(int i=1;i<=q;i++)scanf("%d",&K[i]),K[i]=min(K[i],n);
    d[1]=1,t[1]=1;
    for(int i=2,f;i<=n;i++)scanf("%d",&f),d[i]=d[f]+1,t[d[i]]++;
    for(int i=n;i;i--){t[i]+=t[i+1];for(int j=t[i+1]+1;j<=t[i];j++)fi[j]=i;}
    for(int k=1;k<=n;k++)
        for(int j=1;(j-1)*k+1<=n;j++){
            int v=(j-1)*k+1;
            ans[k]=max(ans[k],fi[v]-1+j);
            //s_i>=v中最大的
        }
    for(int i=1;i<=q;i++)printf("%d ",ans[K[i]]);
    return 0;
}
```

9. AGC032E Modulo Pairing

先思考所有数都小于 $\frac{m}{2}$ 时怎么做，不难发现 a_i 和 a_{2n-i} 配对即可，理由是考虑四个数 $a \le b \le c \le d$ ，$\max(a+d,b+c) \le b+d,c+d$ 。于是考虑一般的情况，我们继续尝试通过 a,b,c,d 四个数找一些性质，你发现 $\min(a+d,b+c) \ge a+c,a+b$ ，结合上面的式子我们就可以得知，若有两组匹配，它们的和同时 $<m$ or $\ge m$ ，则其位置关系一定是 $a-d,b-c$ 的关系。

最后再来简化一组 $<m$ 的匹配和一组 $\ge m$ 的匹配的位置关系。

相似的用 a,b,c,d 来看，发现最优的一定是 $a-b,c-d$ 。

其他的情况 $a-d,b-c$ ； $b-c,a-d$ ； $a-c,b-d$ ，调整成 $a-b,c-d$ 后，因为我们容易发现 $<m$ 的匹配其和更小了，$\ge m$ 的匹配其和更大了，于是仍然满足 $<m/\ge m$ ；而考虑 $c+d-m$ 和 $a+b$ ，它们都是 $\le a+d,b+c,a+c$ 的，这样就直接说明了其结果也是更优的。

于是我们一定是取一个分界线 t ，使前 $2t$ 个数匹配，后 $2n-2t$ 个数匹配。t 越小结果越优，但不能让后 $2n-2t$ 个数中存在和 $<m$ 的情况，于是二分出这个 t 即可。

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,a[201000],m;
bool chk(int X){
    int l=X*2+1,r=n*2;
    for(int i=l;i<=r;i++)if(a[i]+a[l+r-i]<m)return 0;
    return 1;
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n*2;i++)scanf("%d",&a[i]);sort(a+1,a+n*2+1);
    int l=0,r=n,t=-1;
    while(l<=r){
        int mid=(l+r)>>1;
        if(chk(mid))t=mid,r=mid-1;
        else l=mid+1;
    }
    int ans=-1;
    for(int i=1;i<=t*2;i++)ans=max(ans,(a[i]+a[t*2+1-i])%m);
    for(int i=t*2+1;i<=n*2;i++)ans=max(ans,(a[i]+a[n*2+t*2+1-i])%m);
    return printf("%d",ans),0;
}
```