1372F - Omkar and Modes

存在一个长度 n 的不下降序列，一次询问可以得到指定 l,r 区间内的众数及其出现次数，有 k 未给出）种不同数字，要在 $4k$ 次询问内得到原序列。

题解里的第一个做法挺好懂的（Solution1）。记录 $length[x]$ 用一次，取出 $length[x]$ 用一次，在询问统计其他数字答案时连续 x 可能被分成两段（之后处在区间一端不可能再分），所以不超过 $4k$

```
#include<bits/stdc++.h>
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
using namespace std;
const int N=2e5+10;
int n,a[N];
map<int,int>length;
int solve(int l,int r)
{
    if(l>r)return r;
    printf("? %d %d\n",l,r),fflush(stdout);
    int x,f;
    scanf("%d%d",&x,&f);
    if(length[x])
    {
        int ll=r-f+1,rr=ll+length[x]-1;
        for(int i=ll;i<=rr;i++)a[i]=x;
        length[x]=0,solve(l,ll-1);
        return rr;
    }
    else
    {
        length[x]=f;
        int p=l;
        while(length[x])p=solve(p,p+f-1)+1;
        return solve(p,r);
    }
}
int main()
{
    scanf("%d",&n);
    solve(1,n);
    printf("! ");
    for(int i=1;i<=n;i++)printf("%d ",a[i]);
    puts("");
    return 0;
```

```
}
```

是在standing里看到下面这个做法的，但是很神秘，我很难想象是怎么想到的。

所有做法基本上都会搞一个递归函数 $solve(l,r)$ 我们先对 l,r 进行询问得到 x,f 此时我们可以保证如果区间 $[r-f+1,l+f-1]$ 合法，里面的数字均为 x 因为不下降，所有 x 必定在一个连续的块内，想象有一个长度 f 的滑动条，右端最左能到 $l+f-1$ 左端最右同理），之后递归处理已确认区间之外的部分。如果区间不合法（即遵循上述规则不能确认任何一个位置为 x ）我们均分两半递归处理。

证明不会超过 $4k$ 次：**证明失败**。呃尴尬本来想证每种数字作为答案的询问都不会超过 4 次但是发现是可以超过 4 次的比如 $1111222222|2|2222344444$ 以 2 为答案的询问会有 5 个，虽然均摊下来是满足条件的，啊啊啊啊不会证明也不会证伪。。。但如果是假做法不会能这么长时间占据这题榜一吧qaq...但是到底是为什么，不能确认就分两半这个操作也很灵性但就不明白有什么道理能保证不超过 $4k$ 想出这个做法的数学直觉也太好了吧。。

```
#include<bits/stdc++.h>
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
using namespace std;
const int N=2e5+10;
int n,a[N];
void solve(int l,int r)
{
    if(l>r)return;
    printf("? %d %d\n",l,r),fflush(stdout);
    int x,f;
    scanf("%d%d",&x,&f);
    int ll=r-f+1,rr=l+f-1;
    if(ll<=rr)
    {
        for(int i=ll;i<=rr;i++)a[i]=x;
        solve(l,ll-1),solve(rr+1,r);
    }
    else
    {
        int mid=(l+r)>>1;
        solve(l,mid),solve(mid+1,r);
    }
}
int main()
{
    scanf("%d",&n);
    solve(1,n);
    printf("! ");
    for(int i=1;i<=n;i++)printf("%d ",a[i]);
```

```
    puts("");
    return 0;
}
```

1296F - Berland Beauty

给出一棵树，现在需要给每条边赋权值，使得若干条件 $a_i~b_i~g_i$ 满足 a_i 到 b_i 的简单路径上最小值为 g_i 。

可以按 g_i 从大到小处理，将 a_i 到 b_i 路径上未赋值的边赋值为 g_i 这样保证不破坏先前更大的最小值，同时也保证这条路上最小值为 g_i （如果没有未赋值的边无解）。

```
#include<bits/stdc++.h>
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
using namespace std;
const int N=5005;
int head[N],n,m,cnt,p[N],d[N],f[N],res[N];
struct Node1{int nxt,to,id;}edges[N*2];
struct Node2{int a,b,g;}q[N];
void addedge(int u,int v,int
w){edges[++cnt]=Node1{head[u],v,w},head[u]=cnt;}
void dfs(int u)
{
    for(int i=head[u];~i;i=edges[i].nxt)
    {
        int v=edges[i].to;
        if(p[u]==v)continue;
        p[v]=u,d[v]=d[u]+1,f[v]=edges[i].id,dfs(v);
    }
}
int main()
{
    memset(head,-1,sizeof(head));
    scanf("%d",&n);
    for(int i=1;i<n;i++)
    {
        int u,v;
        scanf("%d%d",&u,&v);
        addedge(u,v,i),addedge(v,u,i);
    }
    dfs(1);
    scanf("%d",&m);
```

```
    for(int i=1;i<=m;i++)scanf("%d%d%d",&q[i].a,&q[i].b,&q[i].g);
    sort(q+1,q+1+m,[&](Node2 x,Node2 y){return x.g>y.g;});
    int g=1;
    for(int i=1;i<=m;i++)
    {
        int x=q[i].a,y=q[i].b,h=0;
        if(d[x]<d[y])swap(x,y);
        while(d[x]!=d[y])
        {
            if(!res[f[x]]||res[f[x]]==q[i].g)res[f[x]]=q[i].g,h=1;
            x=p[x];
        }
        while(x!=y)
        {
            if(!res[f[x]]||res[f[x]]==q[i].g)res[f[x]]=q[i].g,h=1;
            if(!res[f[y]]||res[f[y]]==q[i].g)res[f[y]]=q[i].g,h=1;
            x=p[x],y=p[y];
        }
        if(!h)g=0;
    }
    if(!g)puts("-1");
    else for(int i=1;i<n;i++)printf("%d ",res[i]?res[i]:1000000);
    return 0;
}
```