可持久化线段树练习

例1 poj2107 k-th number

一个长度为 n 的序列，q次询问，
每次询问区间 [l, r] 中的第 k 小的数。

n<=1e5, 0<q<5000.

In:	Out:
7 3	5
1 5 2 6 3 7 4	6
2 5 3	3
4 4 1	
1 7 3	

sol

一个想法就是二分答案 x,查询一个区间内有几个数比 x 小。

对值的范围建一棵线段树，把区间中的数全部插入线段树，查询 [1, x-1] 的和即可。

更优化的是不需要在外层二分，直接在树上二分即可。（参考线段树练习）

sol

发现区间会变，不可能每次把区间里的数插入线段树里去。

考虑可持久化：

T[0] 是一颗空的线段树。

T[1] 是在 T[0] 中插入 a[1]

T[2] 是在 T[1] 中插入 a[2]

...

T[i] 是在 T[i-1] 中插入 a[i]

求 [l,r] 的答案可以用类似前缀和的思想，[1,r] 的答案减去[1,l-1]的答案。

sol

具体实现，只要维护2个指针 x 和 y。

x 指向 T[l-1] 的根节点

y 指向 T[r] 的根节点

每次用 sum[y]-sum[x] 就等价于线段树 S (区间[l,r]建树）上对应节点的值。

采用树上二分的方法找第k小。

时空复杂度均为 O(nlogn)

sol

核心代码

```
void change(int p,int v,int l,int r,int &x){
    x=++tot;
    lc[x]=lc[p];
    rc[x]=rc[p];
    sum[x]=sum[p]+1;
    if(l==r) return;
    int mid=(l+r)>>1;
    if(v<=mid) change(lc[p],v,l,mid,lc[x]);
    else change(rc[p],v,mid+1,r,rc[x]);
}

int query(int k,int l,int r,int x,int y){
    if(l==r) return 1;
    int s=sum[lc[y]]-sum[lc[x]],mid=(l+r)>>1;
    if(s<k) return query(k-s,mid+1,r,rc[x],rc[y]);
    else return query(k,l,mid,lc[x],lc[y]);
}
```

sol

核心代码

```
void solve(){
    tot=0;
    cin>>n>>m;
    for(int i=1;i<=n;i++) cin>>a[i],bin[i]=a[i],t[i]=0;
    sort(bin+1,bin+n+1);
    int cnt=unique(bin+1,bin+b+1)-bin-1;
    t[0]=0;
    for(int i=1;i<=n;i++){
        a[i]=lower_bound(bin+1,bin+cnt+1,a[i])-bin;
        change(t[i-1],a[i],1,cnt,t[i]);
    }
    int x,y,k;
    while(m--){
        cin>>x>>y>>k;
        cout<<bin[query(k,1,cnt,t[x-1],t[y])]<<endl;
    }
}
```

例2 cf484e sign on fence

有 n 个篱笆排成一列，第 i 个 篱笆高度为 h[i].
每次询问给定 l,r,w, 问 [l, r]中的连续的 w 个篱笆最小高度的最大值。

n<=1e5, h[i]<=1e9

In:	Out:
5	2
1 2 2 3 3	3
3	1
2 5 3	
2 5 2	
1 5 5	

sol

很显然可以二分答案。

设二分的最小高度 h_min, 只要在区间 [l, r] 中检查是否有连续 w 个数都大于等于h_min。

我们把 h[i]>=h_min 的位置填1，否则填0.

问题就变成检查区间 [l, r] 中是否有连续 w 个1.

可以用线段树来实现。（参考线段树练习连续字段和）

sol

考虑上述做法的前提是在固定 h_min的情况下的,

很显然 h_min 最多有n种取值, 最多可能有 n 颗线段树。
因此可以使用可持久化线段树。

sol

对 h 从大到小排序。

依次往线段树中插入 1，得到一颗新的线段树。

二分 h_min 的时候，在h_min 对应的线段树上做查询即可。

时间复杂度 $O(n\log n+m\log^2 n)$

空间复杂度 $O(n\log n)$

例3 hdu5820 lights

在二维平面上有 n 个路灯。

问是否每一对路灯之间都存在一条道路，使得长度为它们之间的曼哈顿距离，且每个拐弯点都是路灯。

n<=5e5

In:	Out:
2	no
1 1	yes
3 3	
3	
1 1	
1 3	
3 3	

sol

去验证 n^2 对路灯是否都可行显然不现实。

所以我们应该简化题目的限制,

即找到与题目等价的另一种描述，去验证条件。

sol

考虑一对路灯（A, B）
假设存在一条拐弯点都是路灯的路径 A-C1-C2…Cn-B
那么（A, C1), (Cn, B）合法。
所有的相邻的 (Ci,Ci+1) 都合法。

如果只考虑这条路径上的相邻的两个路灯，他们肯定具有相同的 x 坐标或 y 坐标。

考虑相邻的三个路灯C1,C2,C3，他们要么形成一个线段，要么形成一个矩形，矩形内部的点到C1是不合法的。

sol

换句话说，对于一个路灯 (x,y)

找到它上面最近的灯 (x, y’)

找到它右边最近的等 (x’, y)

这三个点可以组成一个矩形，如果矩形内如果有点，那一定到不了（x, y）.

问题则变成 n个矩形，询问 n 个矩形中是否有点。

可以用可持久化线段树来实现。

sol

对 x 这一维建可持久化线段树，维护 y 轴信息。

T[x] 存的是横坐标小于等于 x 的 y 的信息。

T[x] 只需要在 T[x-1]中插入横坐标为 x 的 y 即可。

查询 (x1,x2)-(y1,y2) 矩形中点的个数就是：

query(T[x2],y1,y2) – query(T[x1-1],y1,y2)

sol

还需要考虑边界问题。
一个路灯如果上面和右边都没有点，则认为点在边界上。

还要再对左边和上边算一遍。
左下和右下就不用了。

最后怎样计算每个点上方和右边最近的点?
只要对 y坐标/x坐标排序，依次赋值即可。
时空复杂度都是 O(nlogn)

例4 cf813e army creation

给定一个长度为 n 的序列。

每次询问一个区间 [l,r] ,相同的数最多只能取 k 个，问总共能取几个数。

题目要求强制在线。

n,k <=1e5

In:	Out:
6 2	2
1 1 1 2 2 2	4
5	1
1 2	3
1 6	2
6 6	
2 4	
4 6	

sol

考虑这样一个问题，询问一个区间中有几个不同的数。
即 k=1 并且离线的情况 (参考线段树练习)。

本题要求强制在线怎么办?
考虑可持久化线段树。
T[l] 存的是 last[i]<l 的i，在 i 的位置上+1.
这样每次在 T[l] 这棵树上查询 [l,r] 的区间和即可。

sol

那要求每个数最多可以算 k 次怎么处理?

只需要处理出 lastk[i] 即可。（即 i 前面第k次出现 a[i] 的位置）

时间复杂度 O(nlogn)

例5 cf893f Subree Minimum Query

给定一棵树，树上每个点有一个点权，每条边的边权都是 1.

每次询问给出一对 (x, k) 问点 x 的子树中，距离 x 小于等于 k 的所有点的最小值是多少。

n<=1e5

sol

设每个节点深度为 d[i]

如果对 x 的子树的节点记录深度为 i 的点权最小值为 min[i]。

这样查询距离 x 不超过 k 的点就是在 min[d[x], ... d[x]+k] 中找最小值。

因此我们可以用线段树来维护 min 数组。

sol

继续分析，树上每个节点都应对应一棵线段树。
且节点 x 的线段树可以由 x 的孩子的线段树合并得到。
需要进行线段树的合并操作。
最开始从叶子开始，线段树只有一个值为其点权。
然后逐步往上合并。
最后记录每个节点对应线段树的根节点，这样对于询问 (x, k)
只要在 rt[x] 上查询 d[x]...min(n,d[x]+k) 的区间最小值即可。

核心代码

```cpp
#define mid (l+r)>>1
void pushup(int x){
    minn[x]=min(minn[lc[x]],minn[rc[x]]);
}
void change(int &x,int l,int r,int p,int v){
    if(!x) x=++tot;
    if(l==r){
        minn[x]=v;
        return;
    }
    if(p<=mid) change(lc[x],l,mid,p,v);
    else change(rc[x],mid+1,r,p,v);
    pushup(x);
}
```

```cpp
int merge(int x,int y){
    if(!x||!y){
        return x+y;
    }
    minn[x]=min(minn[x],minn[y]);
    lc[x]=merge(lc[x],lc[y]);
    rc[x]=merge(rc[x],rc[y]);
    return x;
}

void dfs(int u,int fa){
    change(rt[u],1,n,dep[u],a[u]);
    for(int i=head[u];i;i=e[i].next){
        int v=e[i].v;
        if(v==fa) continue;
        d[v]=d[u]+1;
        dfs(v,u);
        rt[u]=merge(rt[u],rt[v]);
    }
}
```

例6 cf600e Lomsat Gelral

给定一棵树，树上每个点有一个数。
查询以每个节点为根的子树中，众数之和是多少。

n<=1e5

In:
4
1 2 3 4
1 2
2 3
2
4

Out:
10 9 3 4

sol

首先考虑如果在序列中查询众数之和怎么解决?

对值域建线段树，第 i 个位置表示 i 出现的次数。

对每个区间维护:

max：区间出现最多的数出现的次数

sum：区间出现最多数之和

```
void pushup(int x){
    maxx[x]=max(maxx[lc[x]],maxx[rc[x]]);
    sum[x]=0;
    if(maxx[lc[x]]==maxx[x]) sum[x]+=sum[lc[x]];
    if(maxx[rc[x]]==maxx[x]) sum[x]+=sum[rc[x]];
}
```

sol

现在我们只需要对每个点 x，维护以 x 为根的子树所有点的信息 T[x]。

T[x] 可以通过 x 的所有孩子的线段树合并得到。

查询 只需要在 T[x] 上查询 [1,n] 的 sum 即可。

时空复杂度都是 O(nlogn)

```
int merge(int x,int y){
	if(!x || !y) return x+y;
	int t=++tot;
	lc[t]=merge(lc[x],lc[y]);
	rc[t]=merge(rc[x],rc[y]);
	pushup(t);
	return t;
}
```

核心代码

```
#define mid (l+r)>>1
void change(int &x,int l,int r,int p,int v){
    if(!x) x=++tot;
    if(l==r){
        maxx[x]=1;
        sum[x]=v;
        return;
    }
    if(p<=mid) change(lc[x],l,mid,p,v);
    else change(rc[x],mid+1,r,p,v);
    update(x);
}
```

```
void dfs(int u,int fa){
    change(rt[u],1,n,a[u],a[u]);
    for(int i=head[u];i;i=e[i].next){
        int v=e[i].v;
        if(v==fa)continue;
        d[v]=d[u]+1;
        dfs(v,u);
        rt[u]=merge(rt[u],rt[v]);
    }
}
```