线段树
- 算法流程
- 复杂度分析
- 实际应用
- 选讲

树状数组
- 算法流程
- 复杂度证明
- 实际应用
- 选讲

线段树合并
- 算法流程
- 复杂度分析
- 实际应用
- 练习题

线段树分裂
- 算法流程
- 复杂度分析
- 实际应用

可持久化线段树
- 算法流程
- 复杂度分析
- 实际应用
- 练习题

李超线段树
- 算法流程
- 复杂度分析
- 实际应用

线段树

算法流程

有一个序列，要区间修改、区间求和和求区间最值，怎么做?

显然有一种nq的做法，暴力查、改所选区间。

算法复杂度是不能接受的。

还可以怎么做?

显然可以将长度为n序列分为等长的$\sqrt{n}$块，每一块分别维护 整块的和 和 对整块的序列加上的相同的数是多少，还有区间最值。

区间加和区间求和相同，查询或修改区间由 中间连续的整块 和 两边长度小于$\sqrt{n}$的散块构成。

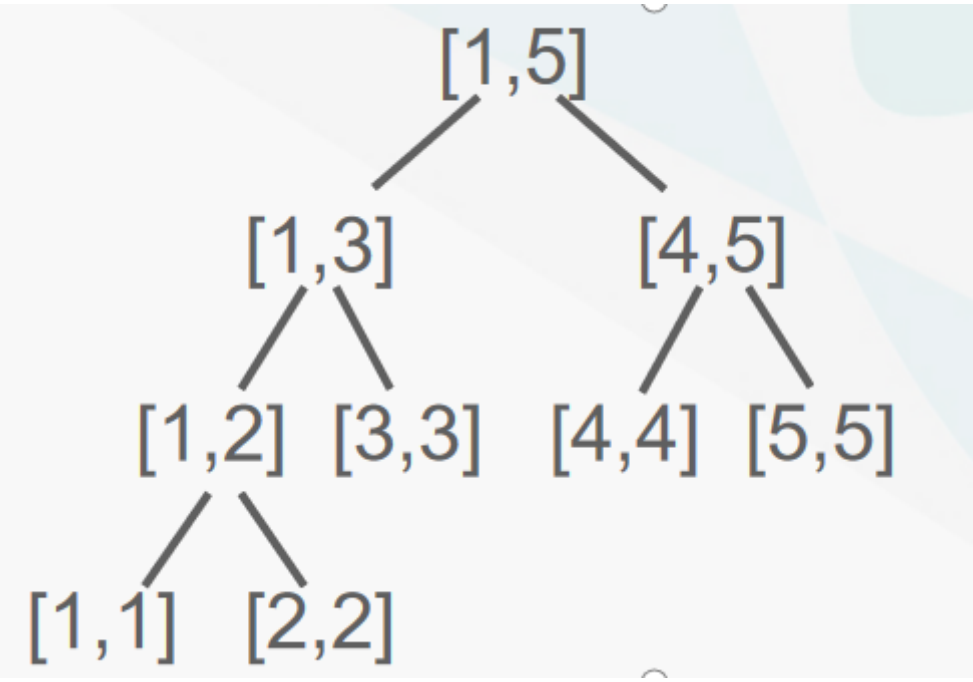
中间连续的整块最多有$\sqrt{n}$块，散块有$O(\sqrt{n})$个，总复杂度$n\sqrt{n}$。

这种方式被称为分块思想，即拆分序列。

而线段树，则是在此基础上的进一步优化，进行优雅的分块，可用$logn$的复杂度完成每次询问。

线段树的本质是一颗二叉树，不同于其它二叉树，线段树的每一个节点记录的是一段区间的信息。

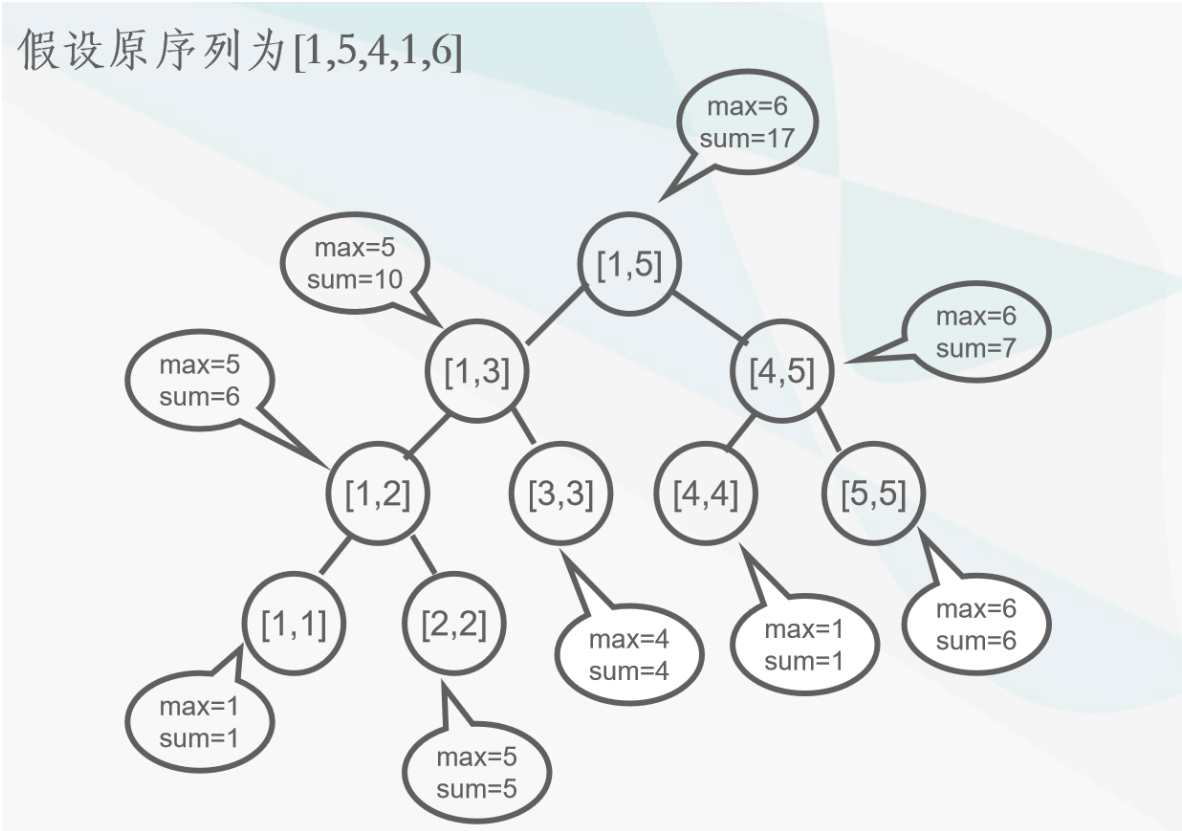
先来看线段树的结构（下面是对于长度5的序列建的树）：

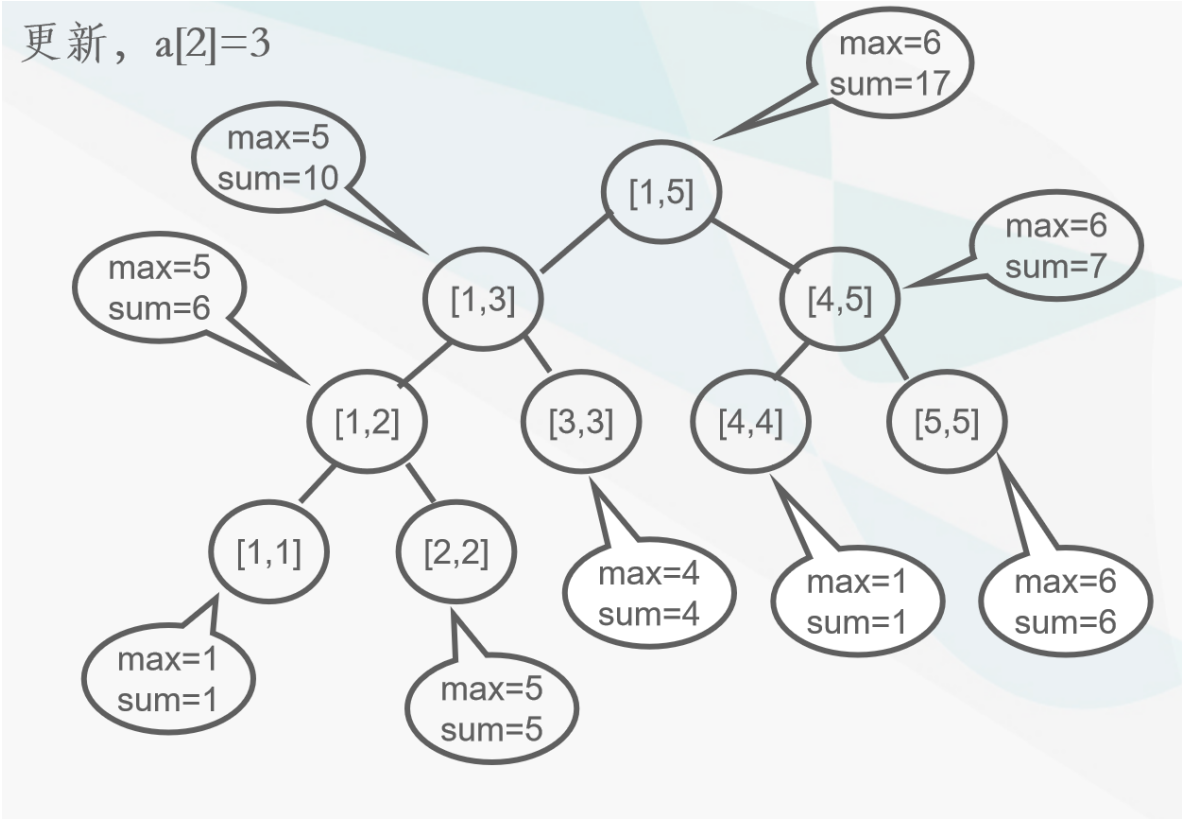


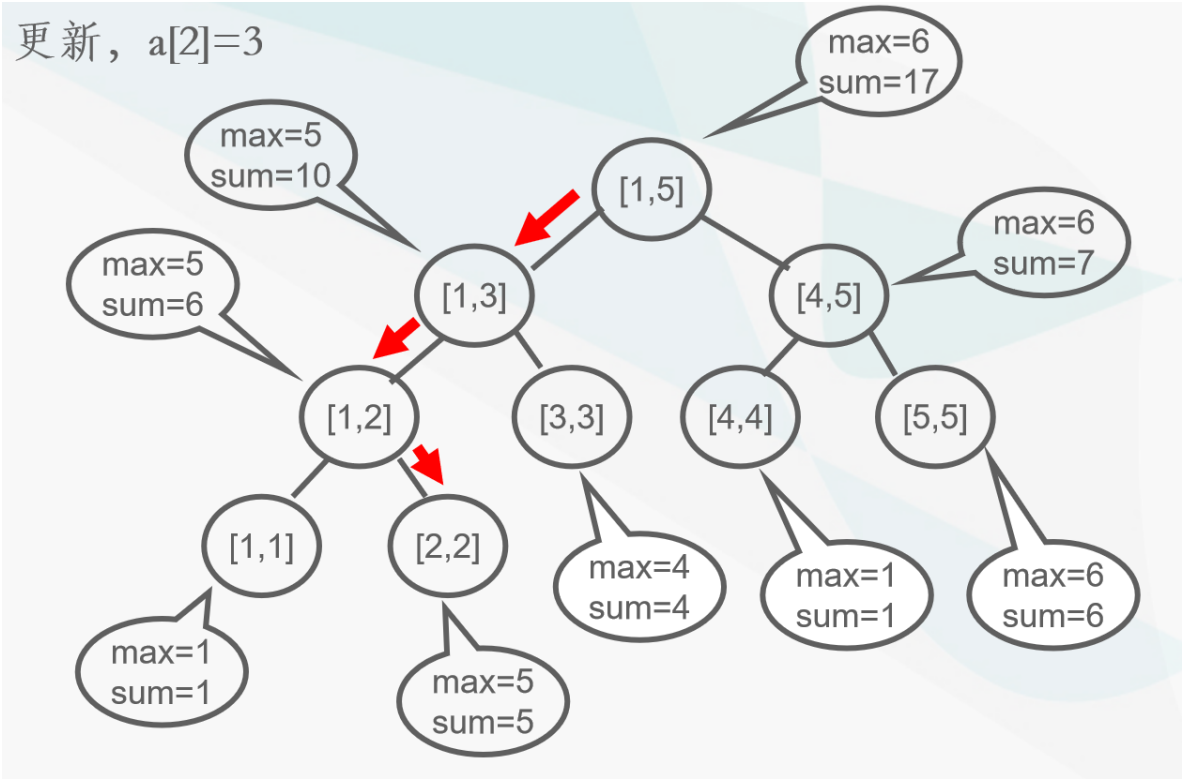
实际就是将上一层的节点每个都砍作均匀的两块。

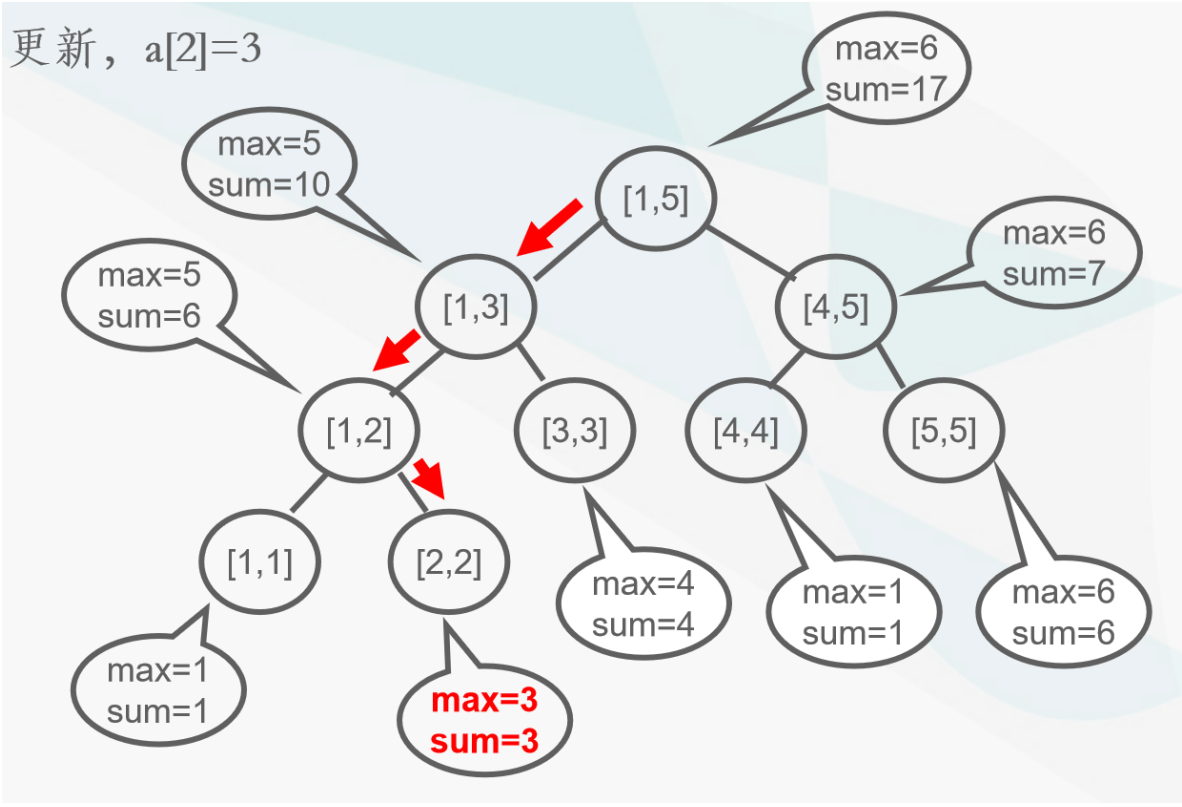
节点上的数字即代表某个点所管辖的范围。

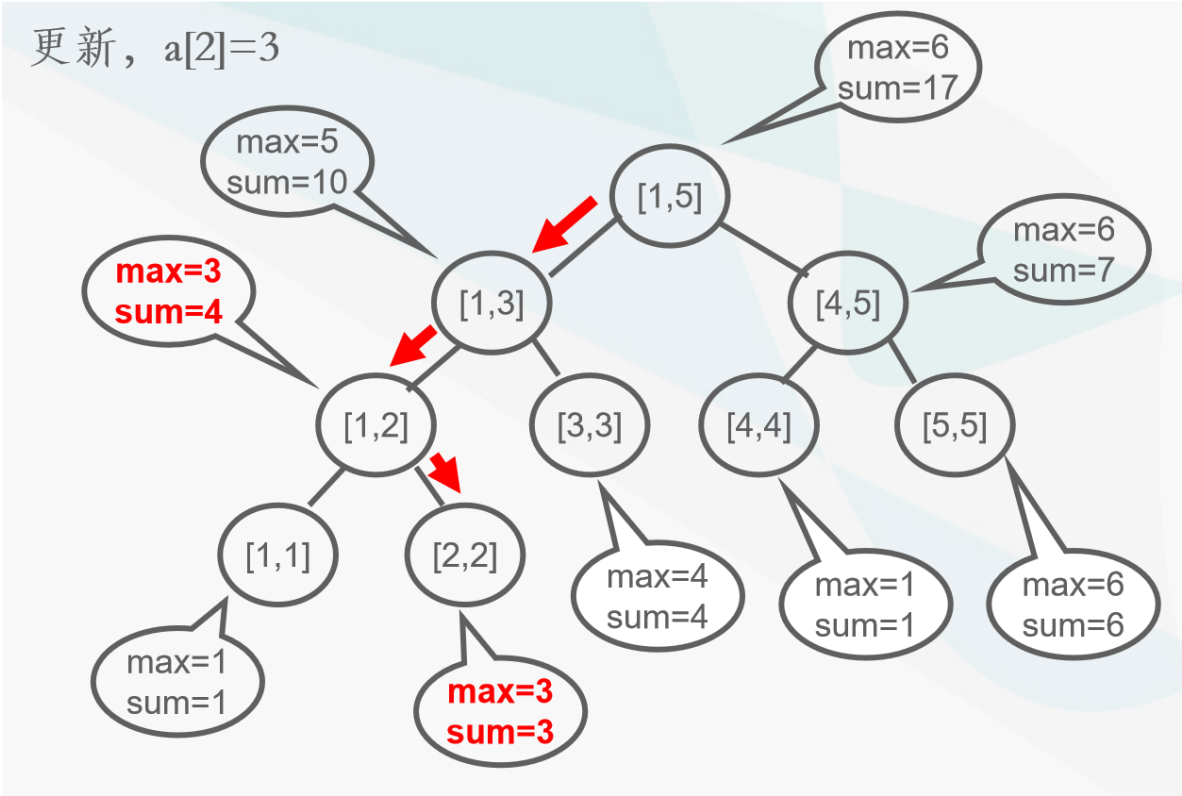
如下演示更改某一个点的过程。

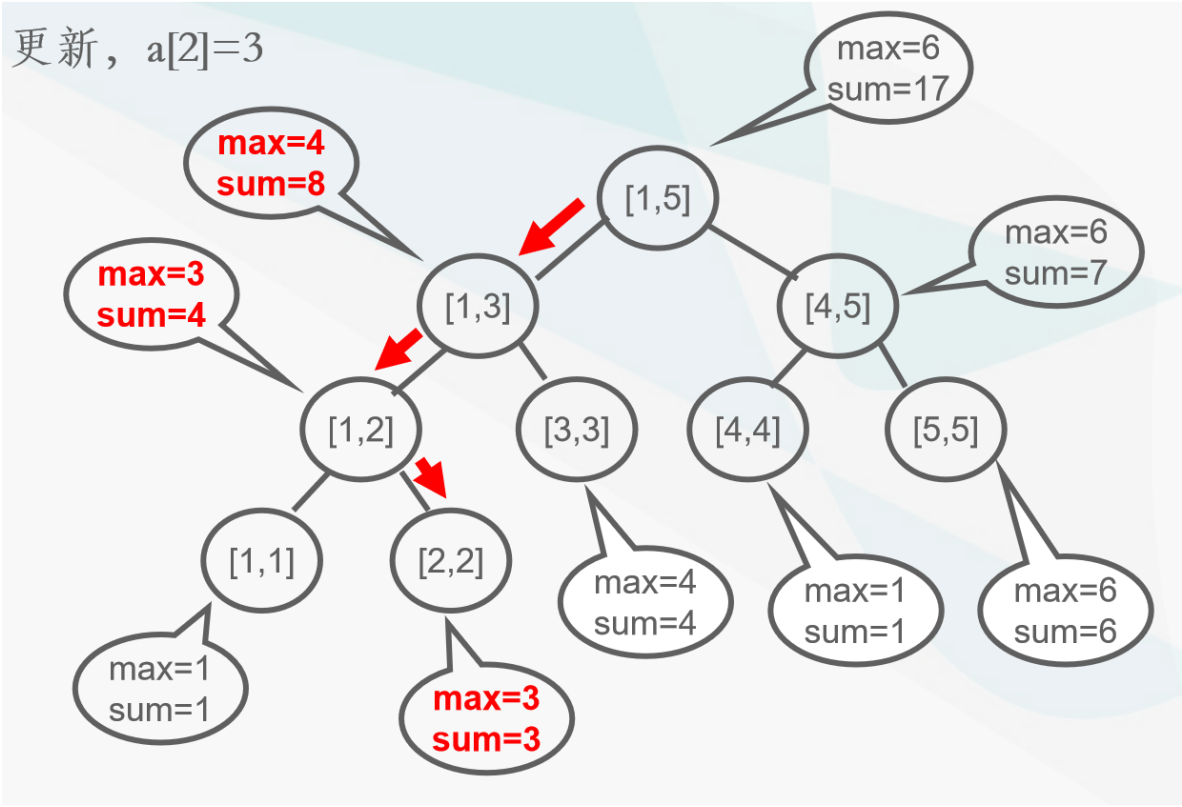


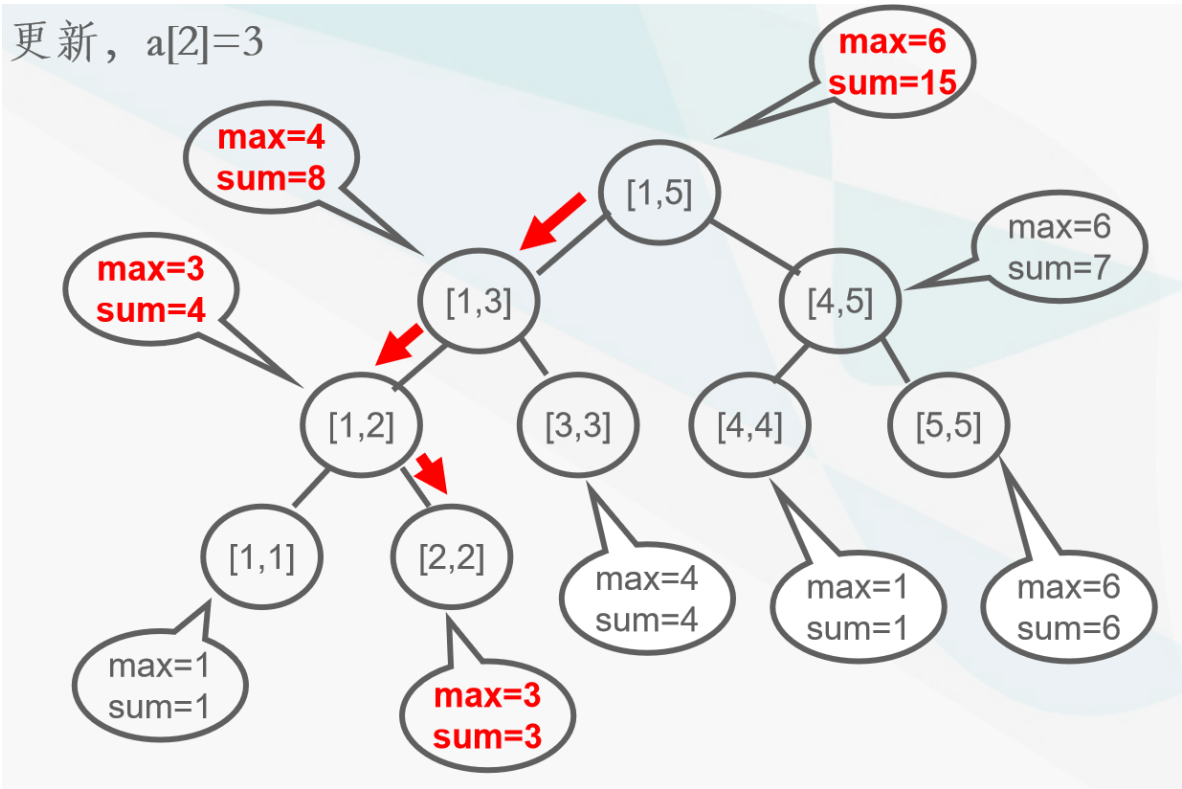

更新，a[2]=3
max=6
sum=17
max=5
sum=10
[1,5]
max=5
sum=6
[1,3]
[4,5]
max=6
sum=7
[1,2]
[3,3]
[4,4]
[5,5]
[1,1]
[2,2]
max=4
sum=4
max=1
sum=1
max=6
sum=6
max=1
sum=1
max=5
sum=5


更新，a[2]=3
max=6
sum=17
max=5
sum=10
[1,5]
max=5
sum=6
[1,3]
[4,5]
max=6
sum=7
[1,2]
[3,3]
[4,4]
[5,5]
[1,1]
[2,2]
max=4
sum=4
max=1
sum=1
max=6
sum=6
max=1
sum=1
max=5
sum=5


更新，a[2]=3
max=6
sum=17
max=5
sum=10
[1,5]
max=5
sum=6
[1,3]
[4,5]
max=6
sum=7
[1,2]
[3,3]
[4,4]
[5,5]
[1,1]
[2,2]
max=4
sum=4
max=1
sum=1
max=6
sum=6
max=1
sum=1
max=3
sum=3


更新，a[2]=3
max=6
sum=17
max=5
sum=10
[1,5]
max=3
sum=4
[1,3]
[4,5]
max=6
sum=7
[1,2]
[3,3]
[4,4]
[5,5]
[1,1]
[2,2]
max=4
sum=4
max=1
sum=1
max=6
sum=6
max=1
sum=1
max=3
sum=3





以下将用仅有区间修改和区间求和的线段树进行演示。

- 创建节点

通常会选用结构体，不过题目不复杂情况下可用数组模拟。

- 快速找子节点

设当前点编号为id，左儿子编号为$id \times 2$,右儿子编号为$id \times 2 + 1$。

当然你也可以在建树时将左右儿子编号存下来。

- 建树

```
void built(int u,int l,int r){//初始化线段树
    len[u]=r-l+1;//计算当前点对应的区间长度
    if(l==r)return tr[u]=sum[l],void(0);
    int mid=(l+r)/2;
    built(u*2,l,mid),built(u*2+1,mid+1,r);
    tr[u]=tr[u*2]+tr[u*2+1];
    return ;
}
```

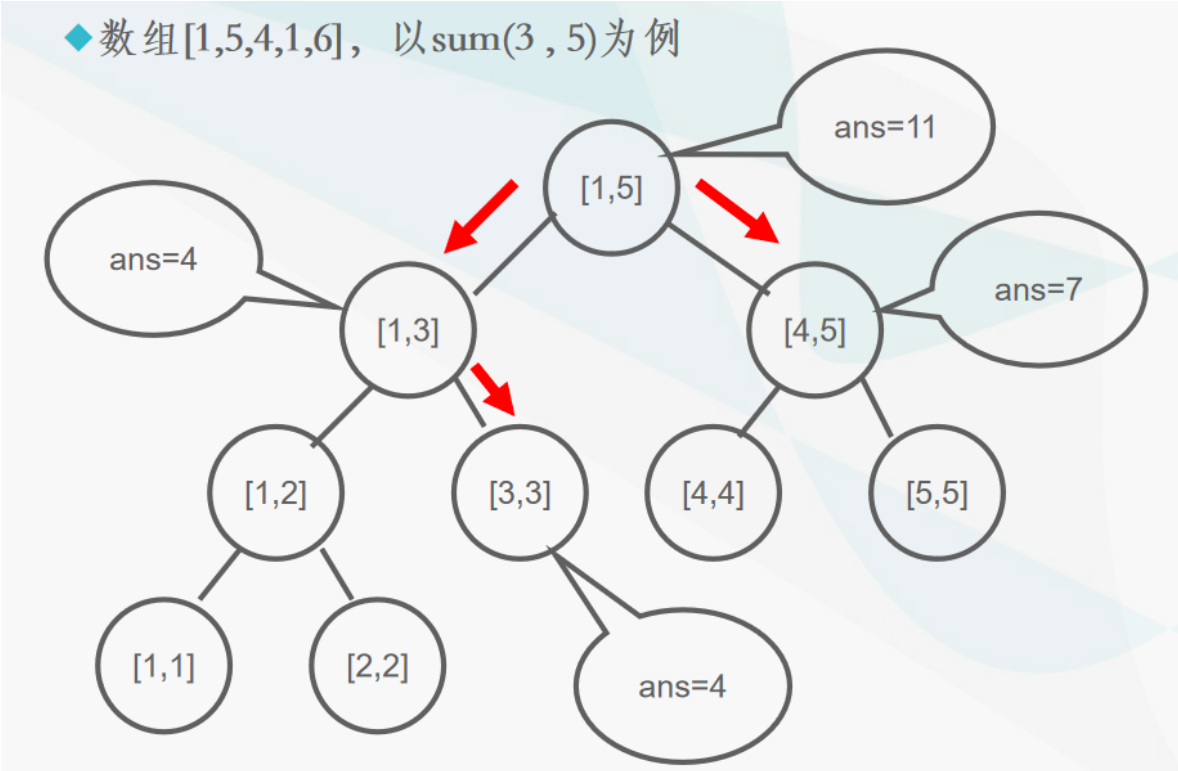
主函数内调用built(1,1,n)建树。

- 单点修改?

```
void modify(int u,int l,int r,int pla,int sumn){
    if(l==r)return tr[u]=sumn,void(0);
    int mid=(l+r)/2;
    if(pla<=mid)modify(u*2,l,mid,pla,sumn);
    else modify(u*2+1,mid+1,r,pla,sumn);
    tr[u]=tr[u*2]+tr[u*2+1];//需要往上继续更新节点信息
    return;
}
```

- 区间查询?

有三种情况:
1、区间在id节点的左半边，递归往下ls（左儿子）
2、区间在id节点的右半边，递归往下rs（右儿子）
3、区间包含id节点的中点，区间以中点分段，分别递归ls、rs



```
int query(int u,int l,int r,int L,int R){
    if(L<=l&&r<=R)return tr[u];
    int mid=(l+r)/2,ansn=0;
    if(L<=mid)ansn+=query(u*2,l,mid,L,R);
    if(R>mid)ansn+=query(u*2+1,mid+1,r,L,R);
    return ansn;
};
```

- 区间修改?

引入lazy标记，某一个点上的lazy标记即代表这个区间每个数被加上的共同值是多少。

将标记lazy打在区间上，表示该区间会有一个lazy的增加。

如果在之后的维护或查询过程中，需要对这个结点递归地进行处理，则当场将这个标记分解，传递给它的两个子结点，这样就不需要考虑自根结点开始的影响了。

方便起见，我们将lazy tag的下传和值的上传分别写成两个函数。

```
void pushdown(int u){//下传信息
    if(!tag[u])return ;
    //若当前点有lazy tag
    tr[u*2]+=tag[u]*len[u*2];
    tr[u*2+1]+=tag[u]*len[u*2+1];//左右儿子的值分别更新
    tag[u*2]+=tag[u];
    tag[u*2+1]+=tag[u];//lazy tag继续下传
    tag[u]=0;//当前点的lazy tag用过了，清零
    return ;
}
void update(int u){//更新当前节点的信息（即将被包含于它的区间信息回传给它）
    tr[u]=tr[u*2]+tr[u*2+1];
    return ;
}
```

如下为区间修改和查询：

```
void modify(int u,int l,int r,int L,int R,int k){//修改
    if(L<=l&&r<=R){
        tag[u]+=k;//给区间打上lazy tag标记
        tr[u]+=len[u]*k;
        return ;
    }
    pushdown(u);//下传标记
    int mid=(l+r)/2;
    if(L<=mid)modify(u*2,l,mid,L,R,k);
    if(R>mid)modify(u*2+1,mid+1,r,L,R,k);
    update(u);//回传值更新当前点
    return ;
}
int query(int u,int l,int r,int L,int R){//查询
    if(L<=l&&r<=R)return tr[u];
    pushdown(u);//下传标记
    int mid=(l+r)/2,ansn=0;
    if(L<=mid)ansn+=query(u*2,l,mid,L,R);
    if(R>mid)ansn+=query(u*2+1,mid+1,r,L,R);
    return ansn;
}
```

复杂度分析

线段树有$log(n)$层，单点修改和单点查询都是$log(n)$的。

区间修改查询时，注意到每层只会下传区间左右端点所属的区间，每层下传两个节点，复杂度$log(n)$。

总复杂度$nlog(n)$。

实际应用

线段树非常的万能，什么区间信息都可以试着用线段树搞一搞。

P3372 【模板】线段树 1

建议先写一写这道。

P1198 [JSOI2008] 最大数

P3373 【模板】线段树 2

上面三道基本上是板子。

P1083 [NOIP2012 提高组] 借教室

题解：线段树区间减，维护最小值，最小值小于零就完蛋。也有二分做法，要听就讲。

给定长度为n的数列a，以及m条指令(n,m<=50000)，每条指令是以下两种之一：
1、“0 x y”，把a[x]改成y
2、“1 x y”，查询区间[x,y]中的最大连续子段和
对于每一个询问，输出一个整数表示答案

题解：本题建立的线段树节点除了区间的端点，还需要维护4个信息：区间和sum，区间最大连续子段和temp，紧靠左端的最大连续字段和lmax，紧靠右端的最大连续子段和rmax。update里顺便处理一下就行。

给定一个长度为n的序列a，以及m条指令($n, m <= 2 * 10^5$)，每条指令可能是以下两种之一：
1、“C l r d”，表示把a[l],a[l+1],...,a[r]都加上d
2、“Q l r”，表示询问a[l],a[l+1],...,a[r]的最大公约数
对于每个询问，输出一个整数表示答案

（tips：和欧几里得算法有一定关系，关系不大）

（tips plus：gcd(x,y)=gcd(x-y,y)）

题解：~~相信大家已经掌握了线段树了，咕咕咕。~~

我们知道，gcd(x,y)=gcd(x,y-x)，对于三个数，也成立：gcd(x,y,z)=gcd(x,y-x,z-y)
实际上，对于任意多个整数，上述式子都成立
因此，构造原数组的差分序列b
这样一来，询问“Q l r”，就等价于求出gcd(a[l],query(1,l+1,r))
因此原序列的区间修改变成了单点修改。

求单点的值还需要线段树求前缀和，不过用接下来要讲的树状数组也可以。

P2672 [NOIP2015 普及组] 推销员

P1442 铁球落地（较难，没时间或者没学过dp就算了）

P8868 [NOIP2022] 比赛 深刻理解线段树打标记

PS：线段树要多写，写几遍就熟练了。

选讲

zkw线段树，不用递归！~~然而并没有什么用~~

猫树！超速处理线段树上的静态问题！~~有点用但是不多~~

有时间再讲吧，咕咕咕。

P6242 【模板】线段树 3（区间最值操作、区间历史最值） 吉司机线段树，尽在线段树3！

- 动态开点线段树

如果线段树开在值域上，值域很大直接把线段树每一个节点建出来空间起飞。

当我们访问到一个区间时，在对对应节点开点。

此时我们不再用 $u \times 2$ 和 $u \times 2 + 1$ 表示左右儿子，而用两个指针指向左右儿子。

对应到刚才单点修改代码：

```
void modify(int &u,int l,int r,int pla,int sumn){
    if(!u)u=++cnt;
    if(l==r)return tr[u]=sumn,void(0);
    int mid=(l+r)/2;
    if(pla<=mid)modify(ls[u],l,mid,pla,sumn);
    else modify(rs[u],mid+1,r,pla,sumn);
    tr[u]=tr[ls[u]]+tr[rs[u]];
    return;
}
```

- 线段树优化建图

有一类问题，从一个点向 $[l,r]$ 区间内每一个点连边，或者将 $[l_1,r_1]$ 中每个点向 $[l_2,r_2]$ 中每个点连边如果一个个连复杂度是不能承受的。

但如果用线段树将 $[l,r]$ 的区间拆成 $logn$ 个区间，每个区间对应线段树上一个点，再对这些点连边，就可以在 $logn$ 时间内完成操作。

例题：CF786B

Sol:板！

树状数组

线段树完全可以实现树状数组所有功能，但树状数组代码极短且常数小，实际能用树状数组绝不用线段树。

算法流程

我们知道求一个区间的和可以转化为两个前缀和相减。

但如果暴力修改前缀和复杂度不能接受。

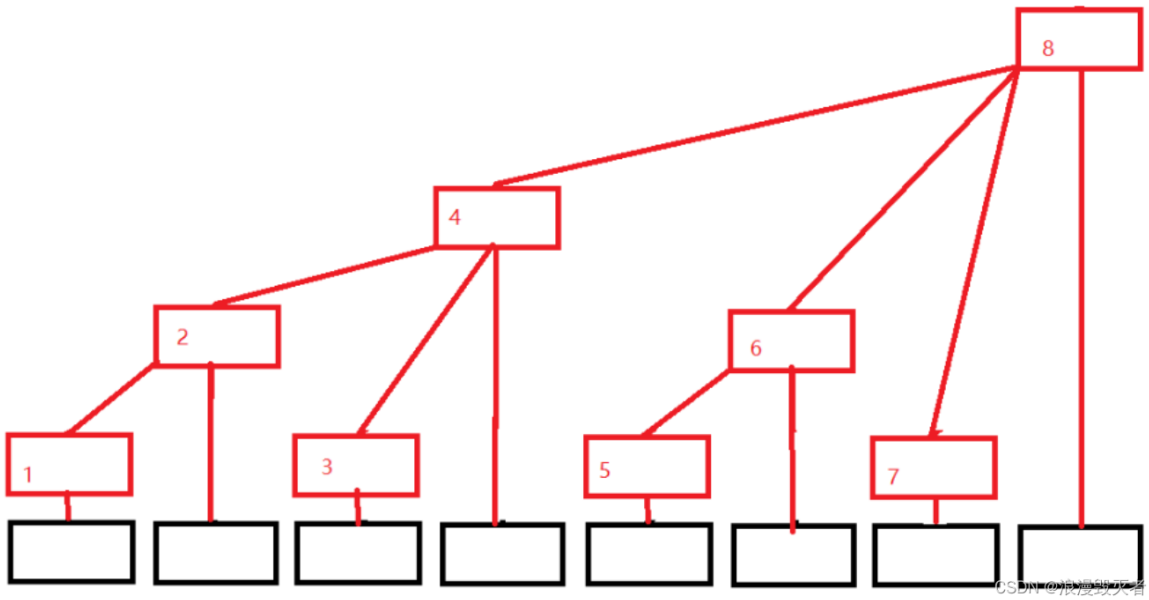
为什么慢？重复计算了某些区间。

可利用分块思想只计算当前所属的块。

如何高效分块？即是树状数组解决的问题。

树状数组本质也是和线段树差不多的二叉树，但其使用了二进制数的特性，使其十分精巧。

先上结构图：



发现这么一件事：将当前点下标用二进制表示，二进制中从小到大第一个1所表示的十进制数设为k，当前点的管辖范围即为包括当前点和它前面长度为k的区间。

例如 6的二进制为110,最后一位为中间的1，则管辖长度为2^1，管辖区间为$[5-6]$；而8的二进制为1000，则管辖长度为2^3，管辖区间为

$[1-8]$。

为什么是这样呢？我们将1-8的二进制列出：

1： 1　　5： 101

2： 10　　6： 110

3： 11　　7： 111

4：100　　8：1000

每到长度为2^k的地方，最大的区间大小会乘2，而后又开始重复2^{k-1}时的区间分割。

在二进制上则表现为 每到长度为2^k的地方，最高位后移，除了最高位的位上又开始重复2^{k-1}时的循环。

两者恰好一一对应。（据说树状数组本身是找规律找出来的）

还有另一种思考方式。

根据任意正整数的关于2的不重复次幂的唯一分解性质，即任意正整数只有唯一的二进制表示形式。

即x可以被唯一表示为$x=2^{i1}+2^{i2}+\ldots+2^m$，则可以将$1-x$划分为m个区间，分别为：

$[1-2^{i1}],[2^{i1}+1,2^{i2}],\ldots,[2^{m-1}+1,2^m]$。

考虑构造这样一个数组，发现将当前点下标用二进制表示，二进制中从小到大第一个1所表示的十进制数设为k，当前点的管辖范围即为包括当前点和它前面长度为k的区间。

例如$x=7=2^2+2^1+2^0$，那么区间$[1,7]$就可以分成$[1,4],[5,6],[7,7]$三个小区间，长度分别为$lowbit(4)=4$、$lowbit(6)=2$、$lowbit(7)=1$

由此可以得到上述结构图。

先来看下代码：

```
inline int lowbit(int x){return x&-x;}
inline void add(int x,int y){
    while(x<=n)tr[x]+=y,x+=lowbit(x);
    return ;
}
inline int get(int x){
    int ansn=0;
    while(x)ansn+=tr[x],x-=lowbit(x);
    return ansn;
}
```

注意：**树状数组只能够支持前缀和查询。**调用一次get函数得到的是1-x的前缀和。

lowbit(x)函数的作用是得到x二进制从小到大第一位为1的位所对应的十进制数。

函数add中 “x+=lowbit(x)” ： 由于上述 “在二进制上则表现为 每到长度为2^k的地方，最高位后移，除了最高位的位上又开始重复2^{k-1}时的循环。”，x+lowbit(x)等同于将x的最低为1的位进位。

函数get中“x-=lowbit(x)"同理，即为跳出当前的块，跳入前面第一个和当前块无交的整块。

lowbit中的"x&-x"：众所周知负数在计算机中是由补码的形式储存，”x&-x"相当于将x的01反转（设反转后的数为y），这样原来x从第1位开始连续的0都变成1，再加1后相当于将y第1位开始连续的1依次进位直到第一个0，也就是x第一个1的位置。设x第一个1的位置为k，y则1-(k-1)的位置都是0，而(k+1)-最高位上x和y都不相同，因此只有x&y只有第k位上是1。

复杂度证明

每次加lowbit，都会使最低位至少后移一位；每次减lowbit，都会使1减少一个，总复杂度$nlog(n)$。

实际应用

树状数组应用不如线段树广，不具有可减性的信息都不能用树状数组（例如求最大值），同时它也只能做到求前缀和，但实在是太好写了！

在一些毒瘤卡常题不能用线段树时，也会考虑用树状数组减小常数。

给你N个数，有两种操作
1：给区间[a,b]的所有数都增加X
2：询问第i个数是什么?

题解：差分前缀和QAQ

P1908 逆序对

题解：值域树状数组QAQ

维护三个序列，支持分别区间加区间乘，查询三个序列对应位置的数之积之和。

题解：先从小规模入手。考虑只有一个序列，显然给线段树加tag且先乘后加。
多个序列？
给某个位置加 x，相当于给答案增加了 x*一个常数，这个常数是相同位置另两个数的乘积。
于是维护七个元素即可。

P1966 [NOIP2013 提高组] 火柴排队双倍经验

P1972 [SDOI2009] HH的项链

树状数组卡常的题要用到cdq分治QAQ

多写几遍就熟练了（确信）

选讲

树状数组上二分/树状数组区间加区间求和/二维树状数组

咕了，有时间就讲吧。

完结撒花——

博客宣传

以上是在役时给初中讲课的ppt，反观现在的讲课水平...QAQ

线段树合并

算法流程

显然如果用节点开满的线段树，单次合并复杂度就是$O(nlog^2n)$的。（线段树有 $nlogn$ 个节点，插入一个节点复杂度是 $logn$ 的）

因此需要上面的动态开点线段树。

假设两颗线段树为 A 和 B，我们从 1 号节点开始递归合并。

递归到某个节点时，如果 A 树或者 B 树上的对应节点为空，直接返回另一个树上对应节点，这里运用了动态开点线段树的特性。

如果递归到叶子节点，我们合并两棵树上的对应节点。

最后，根据子节点更新当前节点并且返回。

不对吧！这复杂度不妥妥超预算吗？且听我细细道来。

复杂度分析

可以证明将 n 棵只有一个元素的线段树合并成一棵线段树的复杂度是 $O(nlogn)$ 的，证明如下。

如果一个位置被递归到了，就说明两棵线段树上相对应的位置都有节点，这两个节点会被合并为一个节点，视作其中一个节点被dfs后就删掉了。

每次dfs都会删掉路径上的节点，一共有 $nlogn$ 个点，总复杂度就是 $nlogn$

tips:容易被卡空间的选手，建议把删掉的节点回收利用。

实际应用

P4556 [Vani有约会] 雨天的尾巴 /【模板】线段树合并

Sol:

树上路径覆盖？树上差分！

每个点开棵线段树（下标为救济粮的类型），线段树维护数量最大的救济粮。如果要在 $u->v$ 的路径上发放 z 类型的救济粮，在 u 的线段树上 z 的位置加一，v 同理。lca 处对应位置减一。要求某个点救济粮和相应的数量，只要把子树内所有的线段树合并起来就好啦！于是从叶子往根依次dfs合并，当某个节点的子树dfs完了就计算这个节点。

练习题

P3224 [HNOI2012] 永无乡

线段树分裂

算法流程

如果要将线段树中 $[l,r]$ 的树分裂出来，从上往下递归，如果节点被完全包含在区间内就直接拆出来；如果有部分重叠，就新建一个节点。将拆出来的节点连在新建的节点上得到新树。

复杂度分析

递归就相当于线段树区间查询的过程嘛！单只 log。

实际应用

P5494 【模板】线段树分裂

给出一个可重集 a（编号为 1），它支持以下操作：

`0 p x y`：将可重集 p 中大于等于 x 且小于等于 y 的值移动到一个新的可重集中（新可重集编号为从 2 开始的正整数，是上一次产生的新可重集的编号+1）。

`1 p t`：将可重集 t 中的数放入可重集 p，且清空可重集 t（数据保证在此后的操作中不会出现可重集 t）。

`2 p x q`：在 p 这个可重集中加入 x 个数字 q。

`3 p x y`：查询可重集 p 中大于等于 x 且小于等于 y 的值的个数。

`4 p k`：查询在 p 这个可重集中第 k 小的数，不存在时输出 `-1`。

可持久化线段树

可以保存线段树的历史版本。

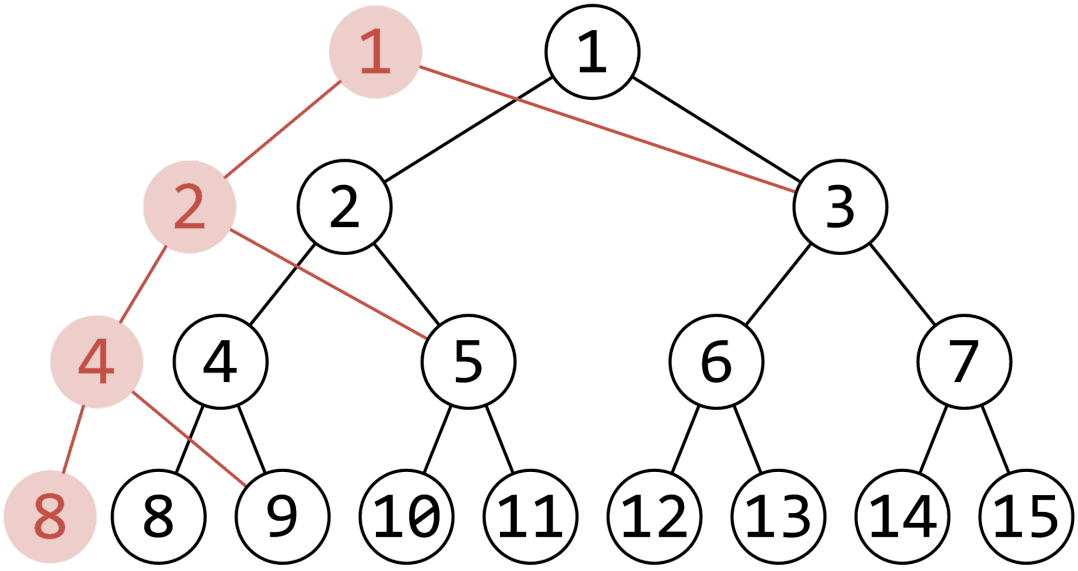
算法流程

对线段树做修改操作，怎么维护它的历史版本呢?

每次都把线段树复制一份再修改TLEMLE一片。

考虑单点修改每次只修改了一条链，只要把修改的链单独存起来就行。

对应到线段树上，若现在修改的是下标为1位置上的值，建出的树是：



访问红色1得到修改后的树，白色1得到修改前的树。

区间修改咋搞啊?

如果对 红色2 打上标记，标记会下传到 白色5，从而影响到历史版本，这并不是我们想要的。

因此标记不能下传，要将标记永久化在节点上。

比如，如果对修改后的树 8~11 都加上 k ，就给 红2 打上一个加k的标记，查询 红8 的值时不仅要找到红8 的值，还要将它所有祖先节点上的标记加起来。

复杂度分析

每次相当于改的是一条长度为 $logn$ 的链，复杂度是单log的。

空间嘛，建议多开亿点数组。

实际应用

P3834 【模板】可持久化线段树 2

给定 n 个整数构成的序列 a，将对于指定的闭区间 $[l,r]$ 查询其区间内的第 k 小值。

- 对于 100% 的数据，满足 $1\leq n,m\leq 2\times 10^5$，$0\leq a_i\leq 10^9$，$1\leq l\leq r\leq n$，$1\leq k\leq r-l+1$。

Sol：

从左到右依次往值域线段树里加数，同时维护线段树的历史版本。

在第 i 个数加入后的线段树，保存了 $a_1,a_2,\dots a_i$ 的信息。

如果将 r 个数加入后的线段树（设为A）减去 $l-1$ 时的线段树（设为B），得到的信息就是 $a_l,a_{l+1}\dots a_r$的信息。

实操时并不需要真正执行线段树相减的操作，只需要用A的对应节点减去B的对应节点的信息，就能得到目标线段树对应节点的信息。

得到了对应的值域线段树，区间第k小怎么求？线段树上二分一下嘛。

P1972 [SDOI2009] HH的项链

给一个序列，每次询问[l, r]中不同的数的个数。

离线可以用树状数组做，考虑下在线的可持久化线段树做法。

Sol：

对于一个区间，只在某个数第一次出现的时候计数
设$pre[i]$表示$a[i]$这个数上一次出现的位置，没有则为0
那么答案就是$[l,r]$中满足$(pre[i]<l)$的个数

练习题

P2839 [国家集训队] middle

小拓展：

求支持单点修改的区间第k小？

Sol：

修改一个点，实际上会对后面所有线段树造成影响，一个一个修改不太现实。

那么用类似树状数组单点修改的方式，用 $root[x]$ 维护子序列 $[x-lowbit(x)+1,x]$ 的变化量。

查询的时候同时维护 log 棵树就好了。

总复杂度两只log。

~~（这真的还是可持久化吗）~~

---

tips：类似地如果将并查集的 fa 数组用线段树维护并可持久化，这就叫可持久化并查集（）

李超线段树

要求在平面直角坐标系下维护两个操作：
在平面上加入一条线段。记第 i 条被插入的线段的标号为 i。
给定一个数 k，询问与直线 x=k 相交的线段中，交点纵坐标最大的线段的编号。

坐标均为整数，且规模较小。

算法流程

用线段树对于每个区间维护在 m=(l+r)/2 处取值最大的直线的信息。

现在我们需要插入一条线段 f，在这条线段完整覆盖的线段树节点代表的区间中，某些区间的最优线段可能发生改变。

考虑某个被新线段 f 完整覆盖的区间，若该区间无最优线段，则该线段可以直接成为最优线段。

否则，设该区间的中点为 m，我们拿新线段 f 在中点处的值与原最优线段 g 在中点处的值作比较。

如果新线段 f 更优，则将 f 和 g 交换并做下述操作。那么现在考虑在中点处 f 不如 g 优的情况：
若在左端点处 f 更优，那么 f 和 g 必然在左半区间中产生了交点，递归到左儿子中进行插入；
若在右端点处 f 更优，那么 f 和 g 必然在右半区间中产生了交点，递归到右儿子中进行插入。
若在左右端点处 g 都更优，那么 f 不可能成为答案，不需要继续下传。

为什么要交换后判断下传？若 f 与 g 在左半区间有交点，说明 g 有可能成为左半区间中某个点的答案。

因此如果要查某个点的答案，要递归到叶子节点并将路径上的线段都比较一下。

复杂度分析

因为要比较左右端点哪条线段更优，节点上的线段必须要完全覆盖对应的区间。

因此加入一条线段要将它拆成 $logn$ 条长度为 2^k 的线段，每条线段都要递归执行上述过程。

总复杂度两只log。

实际应用

P4254 [JSOI2008] Blue Mary 开公司

Sol：

这不就是板子题嘛

P3081 [USACO13MAR] Hill Walk G

Sol：

令当前位置为 (x_{now},y_{now})，要找线段满足在x_{now}处 $y_i \le y_{now}$ 且 y_i 最大。

李超树仅能满足 y_i 最大，无法满足 $y_i \le y_{now}$。

因为线段不交且单调上升，某一次在 x_{now}上方的线段以后也永远用不到了。

所以只要一直删用不到的线段直到找到能用的。

关于李超树的删除：考虑加一条线段是什么过程。

是将在节点上的tag替换为更优的线段，并把原线段下传递归。

传到最后最劣的线段会被扔掉，因此无法保留劣线段的信息。

那么要被扔掉时开个堆，将线段存堆里，等在它上面的线段都删了再拿出来。

对于此题李超双log套堆为三log爆炸。

已知**如果思路方向对，做不出来大概是条件没挖够**

线段不交还没用，对李超树线段交时才下传，因此少个log。

这种做法空间常数超大（动态开点李超+超多堆），考虑离散化+手写堆。。。

吸吸氧吧。

考虑还有没有其他简洁的做法。

还是从性质入手，设当前线段为 $(x_1,y_1)->(x_2,y_2)$，下一条线段为$(x_3,y_3)->(x_4,y_4)$。

要满足$x_1 <= x3 < x_2 < x_4$，且在 $x = x_2$ 处下一条线段的 y 要小于 y_2。

先将所有线段按照 x 排序，符合上述限制的线段是在一个区间（$x_1 <= x_i < x_2$）内的。

依次将区间内符合限制的线段加入李超树，易知没加入的线段永远不会再用，且一条线段只被加一次就够了。

从前向后依次扫直到找不到答案为止。

要离散化，复杂度两只log。