高级数据结构

wzj52501

Peking University

2018 年 10 月 22 日

前言

数据结构作为计算机科学重要的一个分支，在信息学竞赛中也有着广泛的应用。

前言

数据结构作为计算机科学重要的一个分支，在信息学竞赛中也有着广泛的应用。
熟练掌握并应用数据结构知识，是在 NOI 等高水平比赛中制胜的关键一环。

树状数组

Fenwick Tree

树状数组在竞赛中是一种主要用来实现以下操作的数据结构。

Fenwick Tree

树状数组在竞赛中是一种主要用来实现以下操作的数据结构。
维护一个长度为 n 的数组，要求支持 2 种操作。

1. 更改一个位置的权值 $A_x = v$。
2. 询问前缀和 $\sum_{i=1}^{x} A_i$。

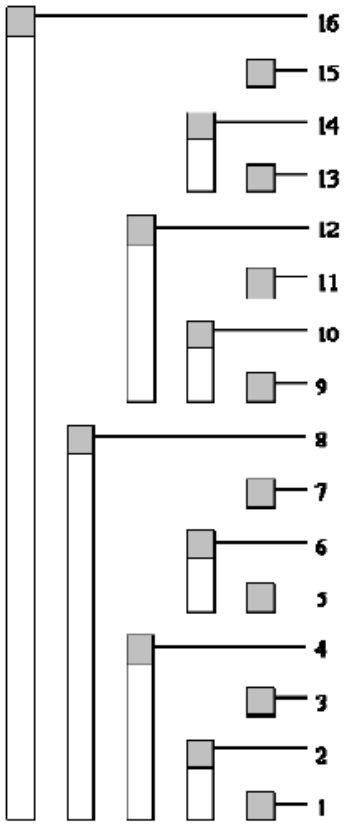
Fenwick Tree

树状数组在竞赛中是一种主要用来实现以下操作的数据结构。
维护一个长度为 n 的数组，要求支持 2 种操作。

1. 更改一个位置的权值 $A_x = v$。
2. 询问前缀和 $\sum_{i=1}^{x} A_i$。

两种操作的时间复杂度均为 $O(\log n)$。

原理



$$\sum_{i=1}^{15} A_i = C(15) + C(14) + C(12) + C(8)$$

Code

```
int n,C[maxn];
int query(int x) {
    int res=0;
    for(;x;x-=x&-x) res+=C[x];
    return res;
}
void update(int x,int v) {
    for(;x<=n;x+=x&-x) C[x]+=v;
}
```

优劣

- 树状数组支持的操作确实被线段树完全包含：(

优劣

- 树状数组支持的操作确实被线段树完全包含：(
- 树状数组简单好写，高效便捷。

优劣

- 树状数组支持的操作确实被线段树完全包含：(
- 树状数组简单好写，高效便捷。
- 值得一提的是，树状数组可以维护前缀最值和单点更新（形如 $A_x = \max(A_x, v)$ ），另外也经常作为嵌套数据结构的第一层。

例题 1

矩形数点

给定 n 个点的坐标 (x_i, y_i)，你需要回答 q 次询问。每次询问给出了一个以 (x_1, y_1) 为左上角，(x_2, y_2) 为右下角的矩形，你需要回答被这个矩形包含的点（可以在矩形边界上）的个数。

$n, q \leq 10^5, 0 \leq x_i, y_i \leq 10^9, 0 \leq x_1 \leq x_2 \leq 10^9, 0 \leq y_1 \leq y_2 \leq 10^9$

例题 1

矩形数点

给定 n 个点的坐标 (x_i, y_i)，你需要回答 q 次询问。每次询问给出了一个以 (x_1, y_1) 为左上角，(x_2, y_2) 为右下角的矩形，你需要回答被这个矩形包含的点（可以在矩形边界上）的个数。

$n, q \leq 10^5, 0 \leq x_i, y_i \leq 10^9, 0 \leq x_1 \leq x_2 \leq 10^9, 0 \leq y_1 \leq y_2 \leq 10^9$

- 首先我们可以根据容斥原理把矩形询问拆成四组形如 $\sum [x_i \leq x, y_i \leq y]$ 的询问。

例题 1

矩形数点

给定 n 个点的坐标 (x_i, y_i)，你需要回答 q 次询问。每次询问给出了一个以 (x_1, y_1) 为左上角，(x_2, y_2) 为右下角的矩形，你需要回答被这个矩形包含的点（可以在矩形边界上）的个数。

$n, q \leq 10^5, 0 \leq x_i, y_i \leq 10^9, 0 \leq x_1 \leq x_2 \leq 10^9, 0 \leq y_1 \leq y_2 \leq 10^9$

- 首先我们可以根据容斥原理把矩形询问拆成四组形如 $\sum [x_i \leq x, y_i \leq y]$ 的询问。
- 把点的坐标和询问的坐标放在一起排序，x 坐标自然满足偏序结构，离散 y 坐标后用树状数组维护前缀和即可。

例题 1

矩形数点

给定 n 个点的坐标 (x_i, y_i)，你需要回答 q 次询问。每次询问给出了一个以 (x_1, y_1) 为左上角，(x_2, y_2) 为右下角的矩形，你需要回答被这个矩形包含的点（可以在矩形边界上）的个数。

$n, q \leq 10^5, 0 \leq x_i, y_i \leq 10^9, 0 \leq x_1 \leq x_2 \leq 10^9, 0 \leq y_1 \leq y_2 \leq 10^9$

- 首先我们可以根据容斥原理把矩形询问拆成四组形如 $\sum [x_i \leq x, y_i \leq y]$ 的询问。
- 把点的坐标和询问的坐标放在一起排序，x 坐标自然满足偏序结构，离散 y 坐标后用树状数组维护前缀和即可。
- 时间复杂度为 $O((n+q)\log n)$。

例题 2

逆序对

给定一个长度为 n 的数列 A，定义其逆序对数为满足 $i < j, A_i > A_j$ 的 (i, j) 二元组数。
现在你可以将其中任意个数字变成其原来的相反数，要求数列的逆序对数尽量少。请输出最少的逆序对数。
$n \leq 10^5, |A_i| \leq 10^9$

例题 2

逆序对

给定一个长度为 n 的数列 A，定义其逆序对数为满足 $i < j, A_i > A_j$ 的 (i, j) 二元组数。
现在你可以将其中任意个数字变成其原来的相反数，要求数列的逆序对数尽量少。请输出最少的逆序对数。
$n \leq 10^5, |A_i| \leq 10^9$

- 对于每个数字 A_i，考虑其符号的正负会对逆序对数有怎样的影响。

例题 2

逆序对

给定一个长度为 n 的数列 A，定义其逆序对数为满足 $i < j, A_i > A_j$ 的 (i, j) 二元组数。
现在你可以将其中任意个数字变成其原来的相反数，要求数列的逆序对数尽量少。请输出最少的逆序对数。
$n \leq 10^5, |A_i| \leq 10^9$

- 对于每个数字 A_i，考虑其符号的正负会对逆序对数有怎样的影响。
- 我们发现，对于其前后绝对值 $\geq |A_i|$ 的数字，更改 A_i 的符号不会改变逆序对的数量。
- 而对于那些绝对值 $< |A_i|$ 的数字，设在 i 之前的有 a 个，在 i 之后的有 b 个。若 A_i 符号为正，则逆序对数增加 b 个，否则增加 a 个。这也恰好不重不漏包含了所有的逆序对情况。

例题 2

逆序对

给定一个长度为 n 的数列 A，定义其逆序对数为满足 $i < j, A_i > A_j$ 的 (i, j) 二元组数。
现在你可以将其中任意个数字变成其原来的相反数，要求数列的逆序对数尽量少。请输出最少的逆序对数。
$n \leq 10^5, |A_i| \leq 10^9$

- 对于每个数字 A_i，考虑其符号的正负会对逆序对数有怎样的影响。
- 我们发现，对于其前后绝对值 $\geq |A_i|$ 的数字，更改 A_i 的符号不会改变逆序对的数量。
- 而对于那些绝对值 $< |A_i|$ 的数字，设在 i 之前的有 a 个，在 i 之后的有 b 个。若 A_i 符号为正，则逆序对数增加 b 个，否则增加 a 个。这也恰好不重不漏包含了所有的逆序对情况。
- 将 $|A_i|$ 离散后，用树状数组求出 a, b 数组，就可以计算出答案。时间复杂度为 $O(n \log n)$。

线段树

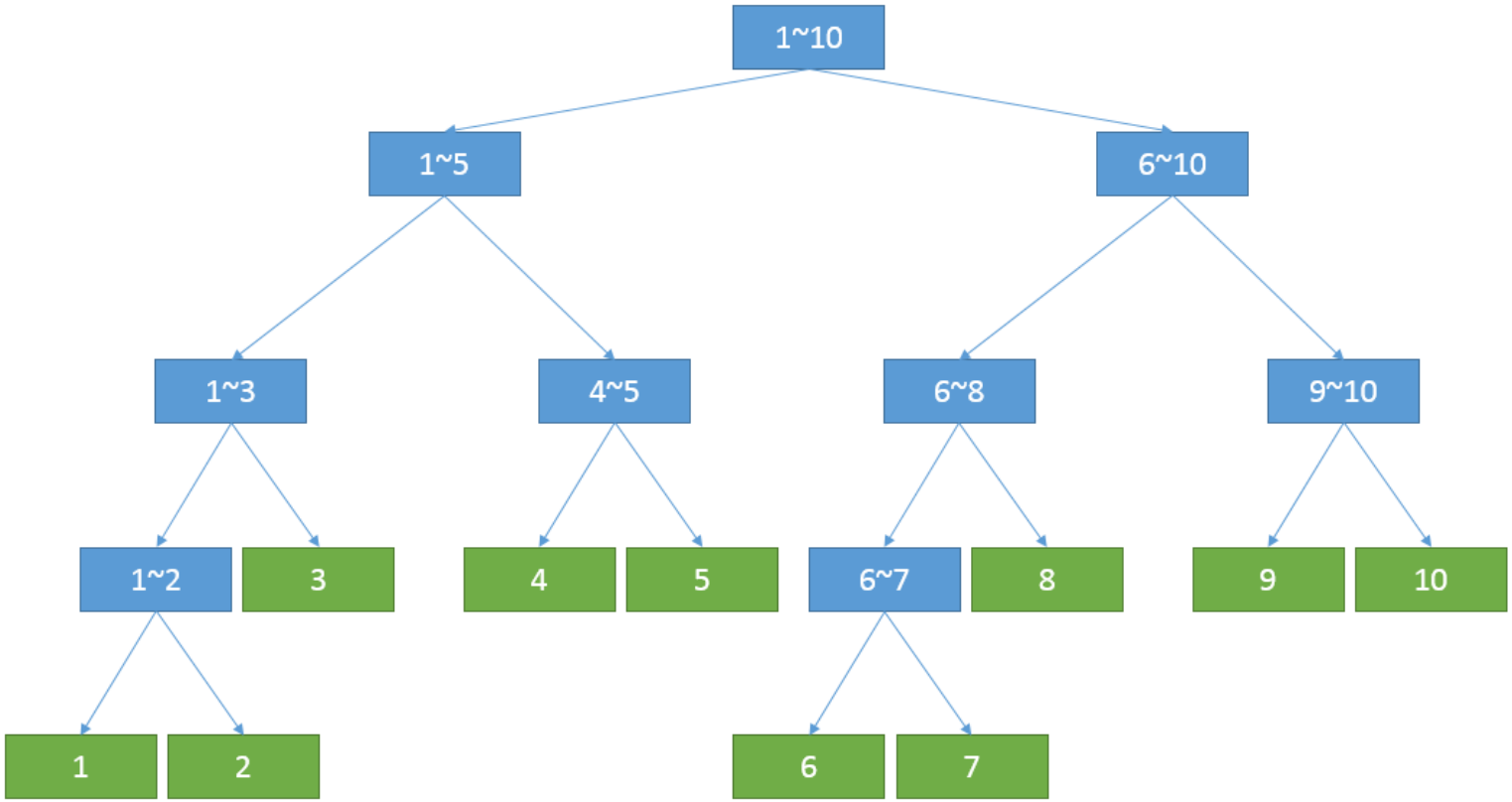
Segment Tree

线段树在 OI 中是一种功能强大、应用灵活的工具，是数据结构学习的重点。

Segment Tree

线段树在 OI 中是一种功能强大、应用灵活的工具，是数据结构学习的重点。
线段树可以对一个静态的序列（不会在中间插入或删除）实现各种区间操作，并且有很多的拓展和变形。

原理



Code

```
void update(int o,int l,int r,int ql,int qr,Data v) {
    if(ql<=l&&r<=qr) updateleaf(o,v);
    else {
        pushdown(o);
        int mid=l+r>>1,lc=o<<1,rc=lc|1;
        if(ql<=mid) update(lc,l,mid,ql,qr,v);
        if(qr>mid) update(rc,mid+1,r,ql,qr,v);
        maintain(o,l,r);
    }
}
Data ans;
void query(int o,int l,int r,int ql,int qr) {
    if(ql<=l&&r<=qr) ans=merge(ans,data[o]);
    else {
        pushdown(o);
        int mid=l+r>>1,lc=o<<1,rc=lc|1;
        if(ql<=mid) query(lc,l,mid,ql,qr);
        if(qr>mid) query(rc,mid+1,r,ql,qr);
    }
}
```

例题 3

序列操作

给定一个长度为 n 的序列 A，你需要执行 q 次操作。

1. 给定 l, r，求 $A_l, A_{l+1}, ..., A_r$ 的极值与区间和。
2. 给定 l, r, v，将 $A_l, A_{l+1}, ..., A_r$ 每个数增加 v。
3. 给定 l, r, v，将 $A_l, A_{l+1}, ..., A_r$ 每个数赋值为 v。

$n, q \leq 10^5, 1 \leq l \leq r \leq n, \quad |v|, |A_i| \leq 10^8$

例题 3

序列操作

给定一个长度为 n 的序列 A，你需要执行 q 次操作。

1. 给定 l, r，求 $A_l, A_{l+1}, ..., A_r$ 的极值与区间和。
2. 给定 l, r, v，将 $A_l, A_{l+1}, ..., A_r$ 每个数增加 v。
3. 给定 l, r, v，将 $A_l, A_{l+1}, ..., A_r$ 每个数赋值为 v。

$n, q \leq 10^5, 1 \leq l \leq r \leq n, \quad |v|, |A_i| \leq 10^8$

- 在每个线段树结点上维护区间极值和区间和，并维护 2 个懒标记 *setv*、*addv*。
- 在标记下传时需要注意 *setv* 会覆盖掉原来的 *addv*。
- 时间复杂度为 $O(n \log n)$。

例题 4

环上最大连续和

给定一个长度为 n 的环形序列 A，其中 A_1 与 A_n 是相邻的。
现在有 q 次修改操作，每次操作会更改 $A_x = v$。请对于每次修改输出修改后的最大连续和。
$n, q \leq 10^5, 1 \leq x \leq n, \quad |A_i|, |v| \leq 10^9$

例题 4

环上最大连续和

给定一个长度为 n 的环形序列 A，其中 A_1 与 A_n 是相邻的。
现在有 q 次修改操作，每次操作会更改 $A_x = v$。请对于每次修改输出修改后的最大连续和。
$n, q \leq 10^5, 1 \leq x \leq n, \quad |A_i|, |v| \leq 10^9$

- 在线段树上的每个结点维护区间内最大前缀和、最大后缀和与最大连续和，这样就可以很容易地解决非环上问题。

例题 4

环上最大连续和

给定一个长度为 n 的环形序列 A，其中 A_1 与 A_n 是相邻的。
现在有 q 次修改操作，每次操作会更改 $A_x = v$。请对于每次修改输出修改后的最大连续和。
$n, q \leq 10^5, 1 \leq x \leq n, \quad |A_i|, |v| \leq 10^9$

- 在线段树上的每个结点维护区间内最大前缀和、最大后缀和与最大连续和，这样就可以很容易地解决非环上问题。
- 而本题的序列为环形，意味着最大连续和可能是两段互不相交的前缀与后缀，那么未选择的肯定是序列中连续的一段（可能为空）

例题 4

环上最大连续和

给定一个长度为 n 的环形序列 A，其中 A_1 与 A_n 是相邻的。
现在有 q 次修改操作，每次操作会更改 $A_x = v$。请对于每次修改输出修改后的最大连续和。
$n, q \leq 10^5, 1 \leq x \leq n, \quad |A_i|, |v| \leq 10^9$

- 在线段树上的每个结点维护区间内最大前缀和、最大后缀和与最大连续和，这样就可以很容易地解决非环上问题。
- 而本题的序列为环形，意味着最大连续和可能是两段互不相交的前缀与后缀，那么未选择的肯定是序列中连续的一段（可能为空）
- 所以我们再维护一下区间内最小前缀和、最小后缀和与最小连续和就好了（ $>\_<$ ）

例题 5

游戏（SDOI2016）

Alice 和 Bob 在玩一个游戏。
游戏在一棵有 n 个点的树上进行，树边带权且非负。最初，每个点上都只有一个数字，那个数字是 12345678912345678^9。
有时，Alice 会选择一条从 s 到 t 的路径，在这条路径上的每一个点上都添加一个数字：对于路径上的一个点 r，若 r 与 s 的距离是 dis，那么 Alice 在点 r 上添加的数字是 $a \times dis + b$。
有时，Bob 会选择一条从 s 到 t 的路径。他需要先从这条路径上选择一个点，再从那个点上选择一个数字。Bob 选择的数字越小越好，但大量的数字让 Bob 眼花缭乱。Bob 需要你帮他找出他能够选择的最小的数字。
$n, q \leq 10^5, |a| \leq 10000, |b| \leq 10^9$

李超树

- 首先进行树链剖分，那么我们只要解决序列的问题就可以在时间复杂度乘一个 $O(\log n)$ 的代价下解决树上问题了。

李超树

- 首先进行树链剖分，那么我们只要解决序列的问题就可以在时间复杂度乘一个 $O(\log n)$ 的代价下解决树上问题了。
- 经过转化后问题可以变成如下的简单形式：
  1. 在区间 $[l, r]$ 内添加一条直线 $y = k \times x + b$。
  2. 询问区间 $[l, r]$ 内所有直线的最小值。

李超树

- 首先进行树链剖分，那么我们只要解决序列的问题就可以在时间复杂度乘一个 $O(\log n)$ 的代价下解决树上问题了。
- 经过转化后问题可以变成如下的简单形式：
  1. 在区间 $[l, r]$ 内添加一条直线 $y = k \times x + b$。
  2. 询问区间 $[l, r]$ 内所有直线的最小值。
- 考虑使用线段树来维护这些直线，用懒标记在每个节点上标记一条覆盖整个区间的直线，表示可能作为答案的候选直线。

李超树

- 首先进行树链剖分，那么我们只要解决序列的问题就可以在时间复杂度乘一个 $O(\log n)$ 的代价下解决树上问题了。
- 经过转化后问题可以变成如下的简单形式：
  1. 在区间 $[l, r]$ 内添加一条直线 $y = k \times x + b$。
  2. 询问区间 $[l, r]$ 内所有直线的最小值。
- 考虑使用线段树来维护这些直线，用懒标记在每个节点上标记一条覆盖整个区间的直线，表示可能作为答案的候选直线。
- 当一个线段树节点 $[l, r]$ 上已经标记了一条直线，而我们又在上面添加了一条新直线，我们应该怎么维护这两条直线的并呢？

- 分别计算出两条直线左右端点的值：如果存在一个直线整体在另一条直线上的情况，直接保留最优直线即可。否则肯定存在一个交点 x，满足 $[l, x]$ 区间内直线 1 更优，而 $[x, r]$ 区间内直线 2 更优。计算两条直线在中间端点处的函数值，在这个节点保留最优区间长的一条直线，把另一条直线向下传递即可。

cont'd

- 分别计算出两条直线左右端点的值：如果存在一个直线整体在另一条直线上的情况，直接保留最优直线即可。否则肯定存在一个交点 x，满足 $[l, x]$ 区间内直线 1 更优，而 $[x, r]$ 区间内直线 2 更优。计算两条直线在中间端点处的函数值，在这个节点保留最优区间长的一条直线，把另一条直线向下传递即可。
- 对于每个修改操作，$[l, r]$ 区间会拆成 $O(\log n)$ 个线段树节点，每个节点因为每次下传区间长度至少减半，所以最多传递 $O(\log n)$ 次，时间复杂度为 $O(\log^2 n)$。

cont'd

- 分别计算出两条直线左右端点的值：如果存在一个直线整体在另一条直线上的情况，直接保留最优直线即可。否则肯定存在一个交点 x，满足 $[l, x]$ 区间内直线 1 更优，而 $[x, r]$ 区间内直线 2 更优。计算两条直线在中间端点处的函数值，在这个节点保留最优区间长的一条直线，把另一条直线向下传递即可。
- 对于每个修改操作，$[l, r]$ 区间会拆成 $O(\log n)$ 个线段树节点，每个节点因为每次下传区间长度至少减半，所以最多传递 $O(\log n)$ 次，时间复杂度为 $O(\log^2 n)$。
- 对于每个询问操作，每个结点维护一下子树内直线最小值，查询时在每个经过的线段树节点和被完整包含的结点上查询即可，时间复杂度为 $O(\log n)$。

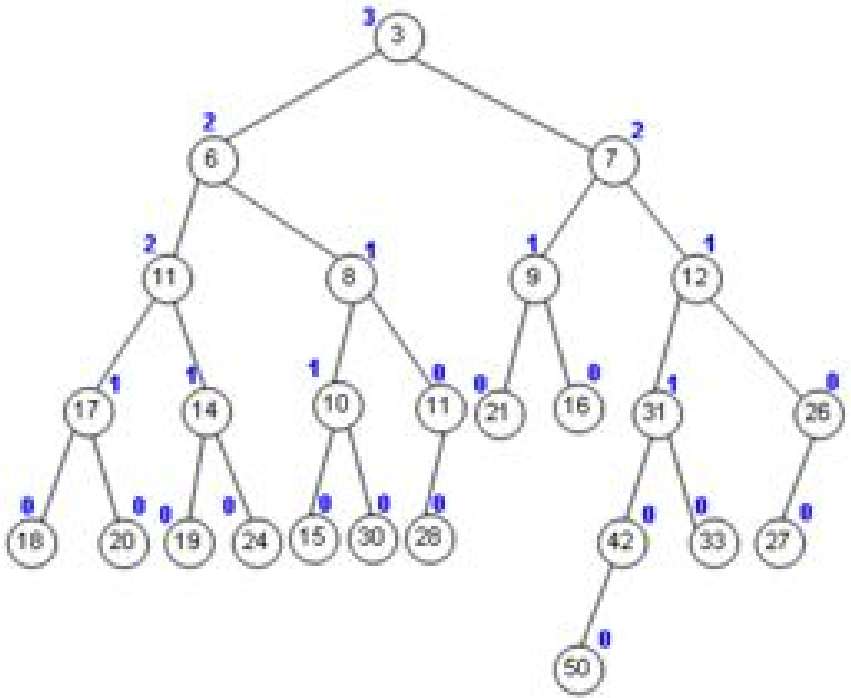
cont'd

- 分别计算出两条直线左右端点的值：如果存在一个直线整体在另一条直线上的情况，直接保留最优直线即可。否则肯定存在一个交点 x，满足 $[l, x]$ 区间内直线 1 更优，而 $[x, r]$ 区间内直线 2 更优。计算两条直线在中间端点处的函数值，在这个节点保留最优区间长的一条直线，把另一条直线向下传递即可。
- 对于每个修改操作，$[l, r]$ 区间会拆成 $O(\log n)$ 个线段树节点，每个节点因为每次下传区间长度至少减半，所以最多传递 $O(\log n)$ 次，时间复杂度为 $O(\log^2 n)$。
- 对于每个询问操作，每个结点维护一下子树内直线最小值，查询时在每个经过的线段树节点和被完整包含的结点上查询即可，时间复杂度为 $O(\log n)$。
- 总时间复杂度为 $O(n + m \log^3 n)$。

可并堆

左偏树

左偏树在竞赛中是支持 $O(\log n)$ 合并的堆。
左偏树呈二叉树结构，除维护权值信息外，还需要维护子树内最近叶结点的距离 d_x。
以小根堆为例，要求根的权值 $\leq$ 左右儿子权值，且 $d_{lc} \geq d_{rc}$。



左偏树（Leftist Tree）

Code

```
int merge(int x,int y) { //小根堆
    if(!x||!y) return x+y;
    if(v[x]>v[y]) swap(x,y);
    rs[x]=merge(rs[x],y);
    if(d[ls[x]]<d[rs[x]]) swap(ls[x],rs[x]);
    d[x]=d[rs[x]]+1;
    return x;
}
```

合并的复杂度与树高相同，而左偏树的树高是 $O(\log n)$ 级别的。

Code

```
int merge(int x,int y) { //小根堆
    if(!x||!y) return x+y;
    if(v[x]>v[y]) swap(x,y);
    rs[x]=merge(rs[x],y);
    if(d[ls[x]]<d[rs[x]]) swap(ls[x],rs[x]);
    d[x]=d[rs[x]]+1;
    return x;
}
```

合并的复杂度与树高相同，而左偏树的树高是 $O(\log n)$ 级别的。
其实左偏树的所有操作都可以被平衡树的启发式合并取代，参见树状数组与线段树之间的关系。

例题 6

Dispatching (APIO2012)

在一个忍者的帮派里，n 个忍者们被选中派遣给顾客，然后依据自己的工作获取报偿。在这个帮派里，有一名忍者被称之为 Master。除了 Master 以外，每名忍者都有且仅有一个上级 f_i。
为保密，同时增强忍者们的领导力，所有与他们工作相关的指令总是由上级发送给他的直接下属，而不允许通过其他的方式发送。
现在你要招募一批忍者，并把它们派遣给顾客。你需要为每个被派遣的忍者支付一定的薪水 c_i，同时使得支付的薪水总额不超过你的预算 m。另外，为了发送指令，你需要选择一名忍者作为管理者，要求这个管理者可以向所有被派遣的忍者发送指令，在发送指令时，任何忍者 (不管是否被派遣) 都可以作为消息的传递人。管理者自己可以被派遣，也可以不被派遣。当然，如果管理者没有被排遣，就不需要支付管理者的薪水。
你的目标是在预算内使顾客的满意度最大。这里定义顾客的满意度为派遣的忍者总数乘以管理者的领导力 l_i。
$n \leq 10^5, 0 \leq f_i < i, 1 \leq c_i \leq m \leq 10^9, 1 \leq l_i \leq 10^9$

cont'd

- 枚举管理者 x，那么我们肯定优先派遣子树内需求薪水最少的忍者。

cont'd

- 枚举管理者 x，那么我们肯定优先派遣子树内需求薪水最少的忍者。
- 考虑使用左偏树维护以 x 为根子树内派遣忍者的名单，每次先合并子树信息与 x 结点信息，再不停删除堆中最大值直到预算 $\leq m$。

cont'd

- 枚举管理者 x，那么我们肯定优先派遣子树内需求薪水最少的忍者。
- 考虑使用左偏树维护以 x 为根子树内派遣忍者的名单，每次先合并子树信息与 x 结点信息，再不停删除堆中最大值直到预算 $\leq m$。
- 时间复杂度为 $O(n \log n)$。

线段树的合并

由于线段树具有很整齐的结构，它也可以支持均摊 $O(\log n)$ 的合并。

```
int merge(int x,int y,int l,int r) {
    if(!x||!y) return x+y;
    if(l==r) return mergeleaf(x,y);
    int mid=l+r>>1;
    ls[x]=merge(ls[x],ls[y],l,mid);
    rs[x]=merge(rs[x],rs[y],mid+1,r);
    maintain(x);
}
```

线段树的合并

由于线段树具有很整齐的结构，它也可以支持均摊 $O(\log n)$ 的合并。

```
int merge(int x,int y,int l,int r) {
    if(!x||!y) return x+y;
    if(l==r) return mergeleaf(x,y);
    int mid=l+r>>1;
    ls[x]=merge(ls[x],ls[y],l,mid);
    rs[x]=merge(rs[x],rs[y],mid+1,r);
    maintain(x);
}
```

时间复杂度可以这么分析，将 n 棵单结点线段树合并在一起所花费的时间不会超过将 n 个结点依次插入一棵线段树的时间。

例题 6 的线段树合并解法

- 对于每个忍者，建立一棵权值线段树（即下标等于其薪水），维护区间和信息。

例题 6 的线段树合并解法

- 对于每个忍者，建立一棵权值线段树（即下标等于其薪水），维护区间和信息。
- 枚举管理者 x，通过合并子树的所有线段树与 x 忍者对应的线段树来得到以 x 为根子树的所有忍者的信息。

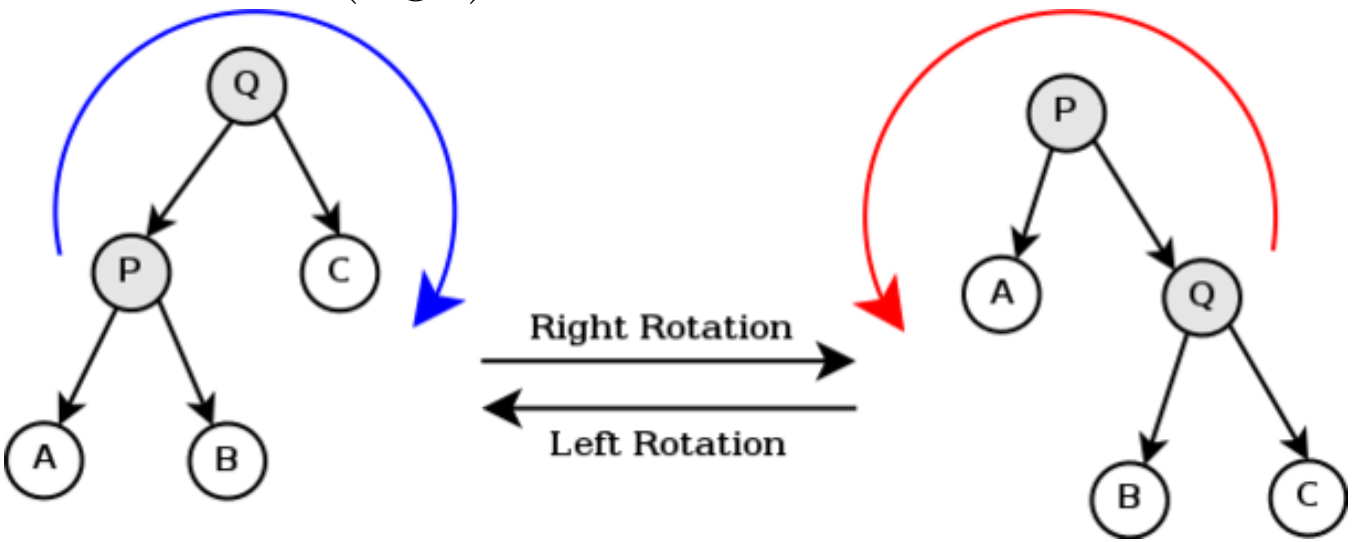
例题 6 的线段树合并解法

- 对于每个忍者，建立一棵权值线段树（即下标等于其薪水），维护区间和信息。
- 枚举管理者 x，通过合并子树的所有线段树与 x 忍者对应的线段树来得到以 x 为根子树的所有忍者的信息。
- 询问时在线段树上二分，可以做到 $O(n \log n)$ 的时间复杂度。

平衡树

Treap

Treap $=$ Tree $+$ Heap。
普通的二叉搜索树会因为数据分布不均造成树高危机，Treap 则给每个结点定义了一个随机权值 r_x，利用旋转保持 r_x 的堆性质，从而将树高维持在期望 $O(\log n)$。



Code

```
int v[maxn],r[maxn],ch[maxn][2],ToT;
void rotate(int& o,int d) {
    int k=ch[o][d^1];ch[o][d^1]=ch[k][d];
    ch[k][d]=o;o=k;
}
void insert(int& o,int val) {
    if(!o) o=++ToT,v[o]=val,r[o]=rand();
    else {
        int d=val>v[o];
        insert(ch[o][d],val);
        if(r[ch[o][d]]>r[o]) rotate(o,d^1);
    }
}
void remove(int& o,int val) {
    if(v[o]==val) {
        if(!ch[o][0]) o=ch[o][1];
        else if(!ch[o][1]) o=ch[o][0];
        else {
            int d=r[ch[o][0]]>r[ch[o][1]];
            rotate(o,d);remove(ch[o][d],val);
        }
    }
    else remove(ch[o][val>v[o]],val);
}
```

启发式合并

- 合并两棵 Treap 的时候，只需要暴力遍历较小的 Treap 提取所有结点，然后依次插入到较大的 Treap 之中就好了。

启发式合并

- 合并两棵 Treap 的时候，只需要暴力遍历较小的 Treap 提取所有结点，然后依次插入到较大的 Treap 之中就好了。
- 由于每次被新插入的结点所在 Treap 大小至少翻倍，所以将 n 个结点按照任意顺序全部合并的总时间复杂度为 $O(n \log^2 n)$。

启发式合并

- 合并两棵 Treap 的时候，只需要暴力遍历较小的 Treap 提取所有结点，然后依次插入到较大的 Treap 之中就好了。
- 由于每次被新插入的结点所在 Treap 大小至少翻倍，所以将 n 个结点按照任意顺序全部合并的总时间复杂度为 $O(n \log^2 n)$。
- 启发式合并的思想也可以用于很多数据结构的合并之中。

例题 7

Peaks (BZOJ3545)

在 Bytemountains 有 n 座山峰，每座山峰有他的高度 h_i。有些山峰之间有双向道路相连，共 m 条道路，每条道路有一个困难值，这个值越大表示越难走。
现在有 q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出 -1。
$n \leq 10^5$，$m, q \leq 5 \times 10^5$

例题 7

Peaks (BZOJ3545)

在 Bytemountains 有 n 座山峰，每座山峰有他的高度 h_i。有些山峰之间有双向道路相连，共 m 条道路，每条道路有一个困难值，这个值越大表示越难走。
现在有 q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出 -1。
$n \leq 10^5$，$m, q \leq 5 \times 10^5$

- 将道路和询问混在一起排序，进行 Kruskal 算法的同时用 Treap 维护连通分量内所有山峰的高度。

例题 7

Peaks (BZOJ3545)

在 Bytemountains 有 n 座山峰，每座山峰有他的高度 h_i。有些山峰之间有双向道路相连，共 m 条道路，每条道路有一个困难值，这个值越大表示越难走。
现在有 q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出 -1。
$n \leq 10^5$，$m, q \leq 5 \times 10^5$

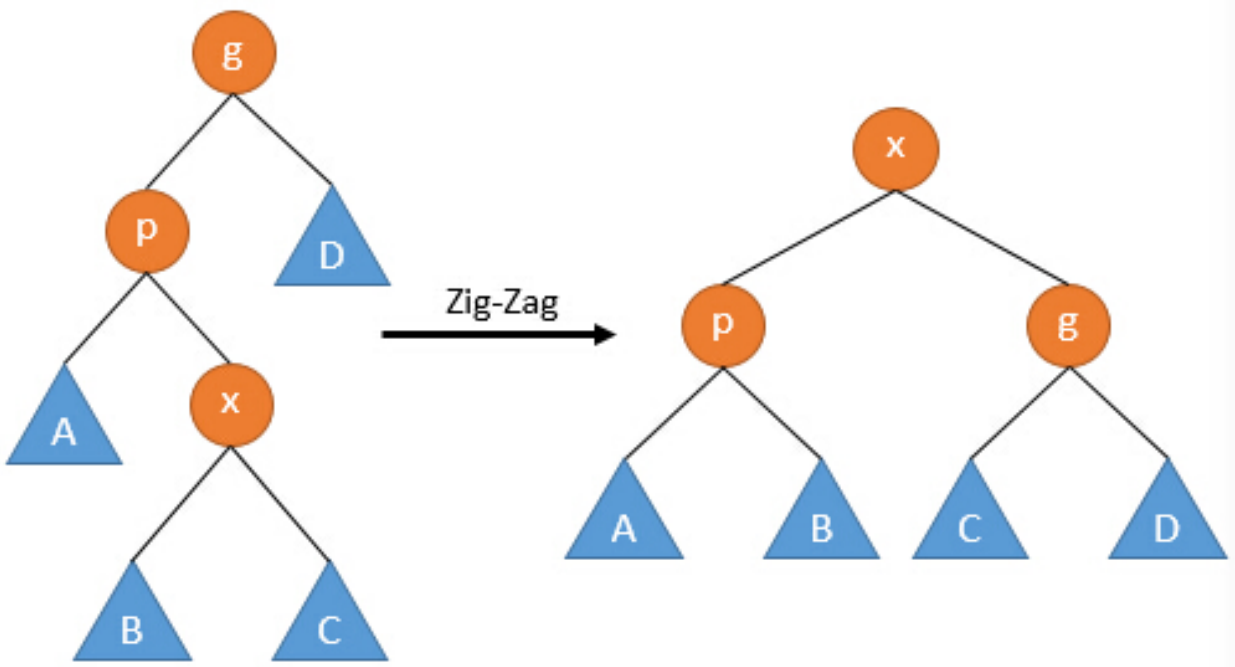
- 将道路和询问混在一起排序，进行 Kruskal 算法的同时用 Treap 维护连通分量内所有山峰的高度。
- 合并连通分量时使用启发式合并，时间复杂度为 $O(n \log^2 n + q \log n)$。

Splay

Splay 也是一种常用的平衡树，它借助双旋操作，可以在均摊 $O(\log n)$ 的时间复杂度内处理许多平衡树的操作。

Splay

Splay 也是一种常用的平衡树，它借助双旋操作，可以在均摊 $O(\log n)$ 的时间复杂度内处理许多平衡树的操作。
时间复杂度证明请参见《Splay 复杂度分析》传送门



Code

```
int splay(int x) {
    while(fa[x]) {
        int y=fa[x],z=fa[y];
        if(z) {
            if(ch[y][0]==x^ch[z][0]==y) rotate(x);
            else rotate(y);
        }
        rotate(x);
    }
}
```

Code

```
int splay(int x) {
    while(fa[x]) {
        int y=fa[x],z=fa[y];
        if(z) {
            if(ch[y][0]==x^ch[z][0]==y) rotate(x);
            else rotate(y);
        }
        rotate(x);
    }
}
```

Splay 主要用于维护可快速提取区间的数列，以及实现 Link Cut Tree。

例题 8

序列终结者 (BZOJ1251)

给定一个长度为 n 的序列，每个序列的元素是一个整数，初始均为 0。要求支持 q 次以下三种操作：

1. 输入 l, r, v，将 $[l, r]$ 这个区间内的所有数加上 v。
2. 输入 l, r，将 $[l, r]$ 这个区间翻转，如 $(1, 2, 3, 4)$ 变成 $(4, 3, 2, 1)$。
3. 输入 l, r，求 $[l, r]$ 这个区间的最大值。

$n, q \leq 10^5, |v| \leq 10^8$

例题 8

序列终结者 (BZOJ1251)

给定一个长度为 n 的序列，每个序列的元素是一个整数，初始均为 0。要求支持 q 次以下三种操作：

1. 输入 l, r, v，将 $[l, r]$ 这个区间内的所有数加上 v。
2. 输入 l, r，将 $[l, r]$ 这个区间翻转，如 $(1, 2, 3, 4)$ 变成 $(4, 3, 2, 1)$。
3. 输入 l, r，求 $[l, r]$ 这个区间的最大值。

$n, q \leq 10^5, |v| \leq 10^8$

- Splay 提取区间 $[l, r]$ 时，先将 $l-1$ 位旋转至根节点，然后断开右儿子。在右儿子为根的树中将第 $r-l+1$ 位旋转至根节点并断开右儿子后，分别得到了区间 $[1, l-1]$、$[l, r]$、$[r+1, n]$ 构成的 Splay 树。

例题 8

序列终结者 (BZOJ1251)

给定一个长度为 n 的序列，每个序列的元素是一个整数，初始均为 0。要求支持 q 次以下三种操作：

1. 输入 l, r, v，将 $[l, r]$ 这个区间内的所有数加上 v。
2. 输入 l, r，将 $[l, r]$ 这个区间翻转，如 $(1, 2, 3, 4)$ 变成 $(4, 3, 2, 1)$。
3. 输入 l, r，求 $[l, r]$ 这个区间的最大值。

$n, q \leq 10^5, |v| \leq 10^8$

- Splay 提取区间 $[l, r]$ 时，先将 $l-1$ 位旋转至根节点，然后断开右儿子。在右儿子为根的树中将第 $r-l+1$ 位旋转至根节点并断开右儿子后，分别得到了区间 $[1, l-1]$、$[l, r]$、$[r+1, n]$ 构成的 Splay 树。
- 考虑在每个结点加上一个区间翻转标记，下传时交换左右子树就可以完成操作 2 了。

例题 8

序列终结者 (BZOJ1251)

给定一个长度为 n 的序列，每个序列的元素是一个整数，初始均为 0。要求支持 q 次以下三种操作：

1. 输入 l, r, v，将 $[l, r]$ 这个区间内的所有数加上 v。
2. 输入 l, r，将 $[l, r]$ 这个区间翻转，如 $(1, 2, 3, 4)$ 变成 $(4, 3, 2, 1)$。
3. 输入 l, r，求 $[l, r]$ 这个区间的最大值。

$n, q \leq 10^5, |v| \leq 10^8$

- Splay 提取区间 $[l, r]$ 时，先将 $l-1$ 位旋转至根节点，然后断开右儿子。在右儿子为根的树中将第 $r-l+1$ 位旋转至根节点并断开右儿子后，分别得到了区间 $[1, l-1]$、$[l, r]$、$[r+1, n]$ 构成的 Splay 树。
- 考虑在每个结点加上一个区间翻转标记，下传时交换左右子树就可以完成操作 2 了。
- 用与线段树类似的方式维护 *addv* 和子树最小值，问题可以在 $O(q \log n)$ 的时间复杂度内解决。

补充例题 1

三维偏序

给定三个长度为 n 的排列 A, B, C，你需要统计有多少对 (i, j) 满足 $A_i < A_j$ 且 $B_i < B_j$ 且 $C_i < C_j$。

$n \leq 5 \times 10^6$

多维线段树

时代的眼泪

- 二维线段树主要的任务是维护平面区域的信息，可以做到的事情包括单点修改区域查询、区域修改单点查询等，但无法支持区间修改区域查询（无法很容易打标记）。

时代的眼泪

- 二维线段树主要的任务是维护平面区域的信息，可以做到的事情包括单点修改区域查询、区域修改单点查询等，但无法支持区间修改区域查询（无法很容易打标记）。
- 前两件事情会被嵌套数据结构完美取代，后一件事情可以被 kd 树胜任。

时代的眼泪

- 二维线段树主要的任务是维护平面区域的信息，可以做到的事情包括单点修改区域查询、区域修改单点查询等，但无法支持区间修改区域查询（无法很容易打标记）。
- 前两件事情会被嵌套数据结构完美取代，后一件事情可以被 kd 树胜任。
- 故二维线段树已经成为了时代的眼泪，如同可持久化线段树普及后的划分树一样被淘汰。

嵌套数据结构

树套树

- 嵌套数据结构的思想是将普通的数据结构维护的信息拓展成另一种数据结构（可能与第一维数据结构相同）

树套树

- 嵌套数据结构的思想是将普通的数据结构维护的信息拓展成另一种数据结构（可能与第一维数据结构相同）
- 常见的树套树有树状数组套平衡树，线段树套权值线段树等。

树套树

- 嵌套数据结构的思想是将普通的数据结构维护的信息拓展成另一种数据结构（可能与第一维数据结构相同）
- 常见的树套树有树状数组套平衡树，线段树套权值线段树等。
- 注意在嵌套数据结构的时候尽量让第一维的数据结构尽量简单，可以起到节约空间减小时间复杂度的作用（如树状数组套平衡树不要写成平衡树套树状数组）

例题 9

Mokia (BZOJ1176)

维护一个 $w \times w$ 的矩阵，初始值均为 s。每次操作可以增加某格子的权值，或询问某个以 (x_1, y_1) 为左上角，(x_2, y_2) 为右下角的矩形格子的权值之和（保证答案绝对值不超过 10^9）
设修改数为 m, 询问数为 q，则 $m \leq 1.5 \times 10^5, q \leq 10^4$，强制在线。

例题 9

Mokia (BZOJ1176)

维护一个 $w \times w$ 的矩阵，初始值均为 s。每次操作可以增加某格子的权值，或询问某个以 (x_1, y_1) 为左上角，(x_2, y_2) 为右下角的矩形格子的权值之和（保证答案绝对值不超过 10^9）
设修改数为 m, 询问数为 q，则 $m \leq 1.5 \times 10^5, q \leq 10^4$，强制在线。

- 考虑使用树状数组套 Treap 解决，把树状数组中的 C 数组替换成用 Treap 维护的包含区间内 y 坐标的集合。

例题 9

Mokia (BZOJ1176)

维护一个 $w \times w$ 的矩阵，初始值均为 s。每次操作可以增加某格子的权值，或询问某个以 (x_1, y_1) 为左上角，(x_2, y_2) 为右下角的矩形格子的权值之和（保证答案绝对值不超过 10^9）
设修改数为 m, 询问数为 q，则 $m \leq 1.5 \times 10^5, q \leq 10^4$，强制在线。

- 考虑使用树状数组套 Treap 解决，把树状数组中的 C 数组替换成用 Treap 维护的包含区间内 y 坐标的集合。
- 时间复杂度为 $O((q+m)\log^2 w)$，空间复杂度为 $O(m \log w)$。

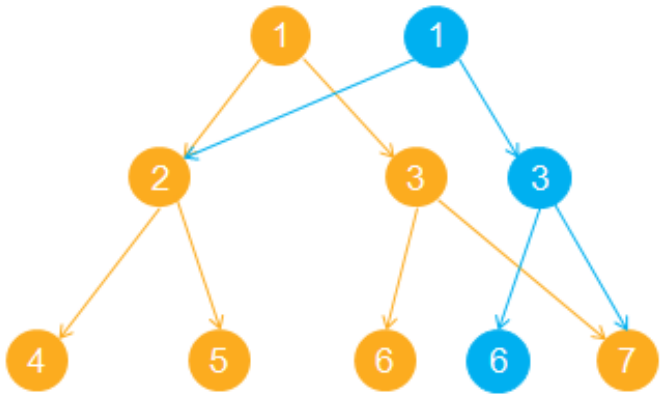
可持久化线段树

可持久化数据结构

可持久化数据结构就是利用函数式编程的思想使其支持询问历史版本、同时充分利用它们之间的共同数据来减少时间和空间消耗。

可持久化数据结构

可持久化数据结构就是利用函数式编程的思想使其支持询问历史版本、同时充分利用它们之间的共同数据来减少时间和空间消耗。
大致的思想就是只新建不修改，保存历史版本————即每次加入新结点后都返回一颗包含新结点的新树保存起来（什么？这么多树空间消耗太大？————充分利用历史版本）



例题 10

静态区间第 k 小数

给定一个长度为 n 的序列 A，你需要回答 q 次询问。每次询问给定 l, r, k，你需要回答 A 序列 $[l, r]$ 区间内第 k 小的数字。
$n, q \leq 10^5, 1 \leq k \leq r - l + 1, 1 \leq l \leq r \leq n, |A_i| \leq 10^9$

例题 10

静态区间第 k 小数

给定一个长度为 n 的序列 A，你需要回答 q 次询问。每次询问给定 l, r, k，你需要回答 A 序列 $[l, r]$ 区间内第 k 小的数字。
$n, q \leq 10^5, 1 \leq k \leq r - l + 1, 1 \leq l \leq r \leq n, |A_i| \leq 10^9$

- 我们对于每个前缀 i，求出包含 $A_1, A_2, ..., A_i$ 所有元素的权值线段树。
- 询问时只需要在线段树上二分最大的位置 v，使得 $A_l, A_{l+1} ..., A_r$ 区间内 $\leq v$ 的个数 $\leq k$ 即可。
- 时间复杂度均为 $O((n+q)\log n)$，空间复杂度为 $O(n \log n)$。

Code

```
void update(int& y,int x,int l,int r,int p) {
    sum[y=++ToT]=sum[x]+1;if(l==r) return;
    int mid=l+r>>1;ls[y]=ls[x];rs[y]=rs[x];
    if(p<=mid) update(ls[y],ls[x],l,mid,p);
    else update(rs[y],rs[x],mid+1,r,p);
}
int query(int y,int x,int l,int r,int k) {
    if(l==r) return l;
    int s=sum[ls[y]]-sum[ls[x]];
    if(k<=s) return query(ls[y],ls[x],l,mid,k);
    else return query(rs[y],rs[x],mid+1,r,k-s);
}
```

补充例题 2

数颜色

给定一个长度为 n 的序列 A，你需要回答 q 次询问。每次询问给定 l, r，你需要回答 A 序列 $[l, r]$ 区间内有多少不同的数字。
$n, q \leq 10^5, 1 \leq l \leq r \leq n, |A_i| \leq 10^9$

动态树

Link Cut Tree

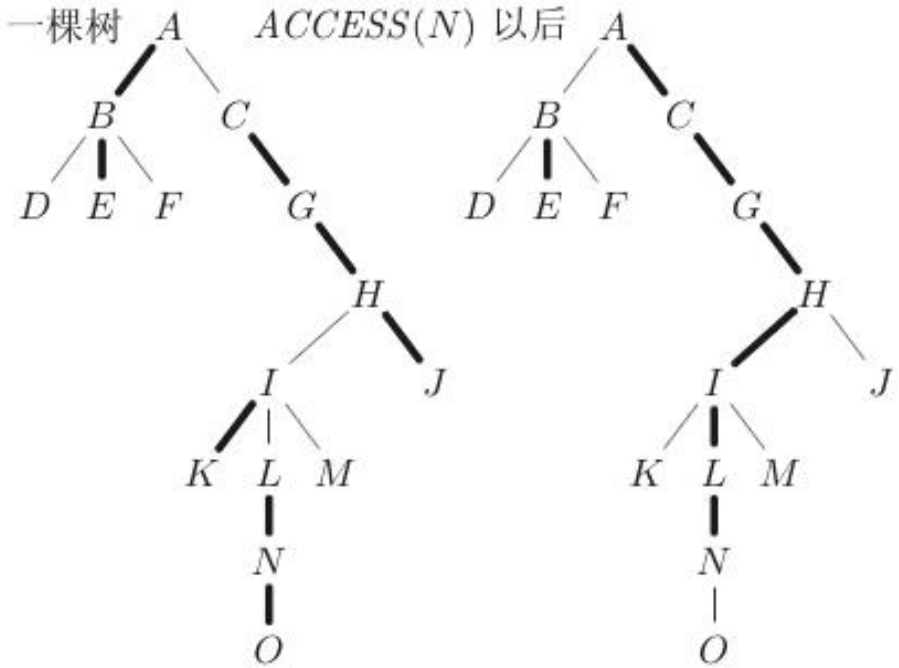
动态树问题要求我们维护一个由若干棵子结点无序的有根树组成的森林，支持对树的分裂、合并、换根，以及对某个点到它的根的路径的某些操作。

Link Cut Tree

动态树问题要求我们维护一个由若干棵子结点无序的有根树组成的森林，支持对树的分裂、合并、换根，以及对某个点到它的根的路径的某些操作。
一般通过 Splay 维护的 Link Cut Tree 实现。

Access

LCT 通过动态维护结点到根的路径从而实现动态树操作，其中最重要的操作为 Access。



Code

```
void rotate(int x) {
    int y=pre[x],z=pre[y],d=(ch[y][0]==x);
    ch[y][d^1]=ch[x][d];pre[ch[x][d]]=y;
    ch[z][ch[z][1]==y]=x;pre[x]=z;
    ch[x][d]=y;pre[y]=x;maintain(y);
}
void pushdown(int x) {
    if(flip[x]) {
        flip[lc]^=1;flip[rc]^=1;
        swap(lc,rc);flip[x]=0;
    }
}
int sta[maxn];
void splay(int x) {
    int top=0;for(int i=x;i;i=pre[i]) sta[++top]=i;
    if(top!=1) fa[x]=fa[sta[top]];
    while(top) pushdown(sta[top--]);
    while(pre[x]) {
        int y=pre[x],z=pre[y];
        if(z) rotate((ch[y][0]==x^ch[z][0]==y)?x:y);
        rotate(x);
    }
    maintain(x);
}
void access(int x) {
    for(int y=0;x;x=fa[x]) {
        splay(x);pre[ch[x][1]]=0;fa[ch[x][1]]=x;
        ch[x][1]=y;pre[y]=x;maintain(y=x);
    }
}
void makeroot(int x) {access(x);splay(x);flip[x]^=1;}
void link(int x,int y) {makeroot(x);fa[x]=y;}
void cut(int x,int y) {makeroot(x);access(y);splay(y);pre[x]=ch[y][0]=0;maintain(y);}
```

例题 11

动态图连通性

给定一个 n 个结点 m 条边的无向图，你需要对其执行 q 次操作：

1. 加入一条无向边 (u, v)。
2. 删除一条无向边 (u, v)，保证这样的边一定存在。
3. 询问结点 u 和 v 之间的连通性。

$n, m, q \leq 10^5$，允许离线。

- 由于问题离线，我们可以预处理每一条边被删除的时刻，记为 t_i。

- 由于问题离线，我们可以预处理每一条边被删除的时刻，记为 t_i。
- 询问连通性的时候我们考虑在图上行走，设法让所经路径上 t_i 最小的边尽量大，这其实就是最大瓶颈路问题。

- 由于问题离线，我们可以预处理每一条边被删除的时刻，记为 t_i。
- 询问连通性的时候我们考虑在图上行走，设法让所经路径上 t_i 最小的边尽量大，这其实就是最大瓶颈路问题。
- 所以我们只需要动态维护整张图按删除时间为权值的最大生成树，就可以实现题目所要求的操作。

- 由于问题离线，我们可以预处理每一条边被删除的时刻，记为 t_i。
- 询问连通性的时候我们考虑在图上行走，设法让所经路径上 t_i 最小的边尽量大，这其实就是最大瓶颈路问题。
- 所以我们只需要动态维护整张图按删除时间为权值的最大生成树，就可以实现题目所要求的操作。
- 考虑使用 Link Cut Tree 来动态维护无向图的最大生成树：

- 由于问题离线，我们可以预处理每一条边被删除的时刻，记为 t_i。
- 询问连通性的时候我们考虑在图上行走，设法让所经路径上 t_i 最小的边尽量大，这其实就是最大瓶颈路问题。
- 所以我们只需要动态维护整张图按删除时间为权值的最大生成树，就可以实现题目所要求的操作。
- 考虑使用 Link Cut Tree 来动态维护无向图的最大生成树：
- 当加入一条边 (u, v, t_i) 的时候先判断 u 和 v 是否连通，若不连通直接加入。否则找到 u 到 v 路径上最小的 t_j，若 $t_i > t_j$ 则用新边替换旧边。

- 由于问题离线，我们可以预处理每一条边被删除的时刻，记为 t_i。
- 询问连通性的时候我们考虑在图上行走，设法让所经路径上 t_i 最小的边尽量大，这其实就是最大瓶颈路问题。
- 所以我们只需要动态维护整张图按删除时间为权值的最大生成树，就可以实现题目所要求的操作。
- 考虑使用 Link Cut Tree 来动态维护无向图的最大生成树：
- 当加入一条边 (u, v, t_i) 的时候先判断 u 和 v 是否连通，若不连通直接加入。否则找到 u 到 v 路径上最小的 t_j，若 $t_i > t_j$ 则用新边替换旧边。
- 当删除一条边 (u, v, t_i) 的时候判断该边是否在当前最大生成树中，若在则将其删除。

- 由于问题离线，我们可以预处理每一条边被删除的时刻，记为 t_i。
- 询问连通性的时候我们考虑在图上行走，设法让所经路径上 t_i 最小的边尽量大，这其实就是最大瓶颈路问题。
- 所以我们只需要动态维护整张图按删除时间为权值的最大生成树，就可以实现题目所要求的操作。
- 考虑使用 Link Cut Tree 来动态维护无向图的最大生成树：
- 当加入一条边 (u, v, t_i) 的时候先判断 u 和 v 是否连通，若不连通直接加入。否则找到 u 到 v 路径上最小的 t_j，若 $t_i > t_j$ 则用新边替换旧边。
- 当删除一条边 (u, v, t_i) 的时候判断该边是否在当前最大生成树中，若在则将其删除。
- 当询问 u 和 v 连通性时，直接查询 u 和 v 在最大生成树上的连通性即可。
- 时间复杂度为 $O(m + q \log n)$。

综合题选讲

例题 12

动态区间第 k 小数

给定一个长度为 n 的序列 A，你需要执行 q 次操作。操作有两种：第一种是给定 l, r, k，你需要回答 A 序列 $[l, r]$ 区间内第 k 小的数字。第二种是给定 l, v，你需要将 A_l 改成 v。

$n, q \leq 10^5, 1 \leq k \leq r - l + 1, 1 \leq l \leq r \leq n, \quad |A_i|, |v| \leq 10^9$

例题 12

动态区间第 k 小数

给定一个长度为 n 的序列 A，你需要执行 q 次操作。操作有两种：第一种是给定 l, r, k，你需要回答 A 序列 $[l, r]$ 区间内第 k 小的数字。第二种是给定 l, v，你需要将 A_l 改成 v。

$n, q \leq 10^5, 1 \leq k \leq r - l + 1, 1 \leq l \leq r \leq n, \quad |A_i|, |v| \leq 10^9$

- 使用树状数组套可持久化线段树解决。

例题 12

动态区间第 k 小数

给定一个长度为 n 的序列 A，你需要执行 q 次操作。操作有两种：第一种是给定 l, r, k，你需要回答 A 序列 $[l, r]$ 区间内第 k 小的数字。第二种是给定 l, v，你需要将 A_l 改成 v。

$n, q \leq 10^5, 1 \leq k \leq r - l + 1, 1 \leq l \leq r \leq n, \quad |A_i|, |v| \leq 10^9$

- 使用树状数组套可持久化线段树解决。
- 可持久化线段树维护的信息类似例题 10，询问第 k 小数时根据树状数组拆分成 $O(\log n)$ 棵可持久化线段树，并在上面同时进行二分。

例题 12

动态区间第 k 小数

给定一个长度为 n 的序列 A，你需要执行 q 次操作。操作有两种：第一种是给定 l, r, k，你需要回答 A 序列 $[l, r]$ 区间内第 k 小的数字。第二种是给定 l, v，你需要将 A_l 改成 v。

$n, q \leq 10^5, 1 \leq k \leq r - l + 1, 1 \leq l \leq r \leq n, \quad |A_i|, |v| \leq 10^9$

- 使用树状数组套可持久化线段树解决。
- 可持久化线段树维护的信息类似例题 10，询问第 k 小数时根据树状数组拆分成 $O(\log n)$ 棵可持久化线段树，并在上面同时进行二分。
- 时空复杂度均为 $O(n \log n + q \log^2 n)$。

例题 13

GERALD07 加强版 (BZOJ3514)

给定 n 个点 m 条边的无向图，q 次询问保留图中编号在 $[l, r]$ 的边的时候图中的连通块个数。
$n, m, q \leq 2 \times 10^5$，强制在线。

例题 13

GERALD07 加强版 (BZOJ3514)

给定 n 个点 m 条边的无向图，q 次询问保留图中编号在 $[l, r]$ 的边的时候图中的连通块个数。
$n, m, q \leq 2 \times 10^5$，强制在线。

- 首先把边依次加到图中，用 LCT 维护加入时间的最大生成树。若当前这条边与图中的边形成了环，那么把这个环中最早加进来的边弹出去，并记录每条边把哪条边弹了出去，设为 p_i。特别地，要是没有弹出边，$p_i = 0$。

例题 13

GERALD07 加强版 (BZOJ3514)

给定 n 个点 m 条边的无向图，q 次询问保留图中编号在 $[l, r]$ 的边的时候图中的连通块个数。
$n, m, q \leq 2 \times 10^5$，强制在线。

- 首先把边依次加到图中，用 LCT 维护加入时间的最大生成树。若当前这条边与图中的边形成了环，那么把这个环中最早加进来的边弹出去，并记录每条边把哪条边弹了出去，设为 p_i。特别地，要是没有弹出边，$p_i = 0$。
- 对于每个询问，答案即为 $n - \sum_{i=l}^{r} [f_i < l]$，这可以使用可持久化线段树在 $O(\log n)$ 的时间内求出。

例题 13

GERALD07 加强版 (BZOJ3514)

给定 n 个点 m 条边的无向图，q 次询问保留图中编号在 $[l, r]$ 的边的时候图中的连通块个数。
$n, m, q \leq 2 \times 10^5$，强制在线。

- 首先把边依次加到图中，用 LCT 维护加入时间的最大生成树。若当前这条边与图中的边形成了环，那么把这个环中最早加进来的边弹出去，并记录每条边把哪条边弹了出去，设为 p_i。特别地，要是没有弹出边，$p_i = 0$。
- 对于每个询问，答案即为 $n - \sum_{i=l}^{r} [f_i < l]$，这可以使用可持久化线段树在 $O(\log n)$ 的时间内求出。
- 正确性：如果一条边 $f_i \geq l$，那么它可显然以与 $[l, r]$ 中的边形成环，对答案没有贡献。反之它与 $[l, r]$ 中的边是不能形成环的，那么它对答案的贡献为 -1。

例题 13

GERALD07 加强版 (BZOJ3514)

给定 n 个点 m 条边的无向图，q 次询问保留图中编号在 $[l, r]$ 的边的时候图中的连通块个数。
$n, m, q \leq 2 \times 10^5$，强制在线。

- 首先把边依次加到图中，用 LCT 维护加入时间的最大生成树。若当前这条边与图中的边形成了环，那么把这个环中最早加进来的边弹出去，并记录每条边把哪条边弹了出去，设为 p_i。特别地，要是没有弹出边，$p_i = 0$。
- 对于每个询问，答案即为 $n - \sum_{i=l}^{r} [f_i < l]$，这可以使用可持久化线段树在 $O(\log n)$ 的时间内求出。
- 正确性：如果一条边 $f_i \geq l$，那么它可显然以与 $[l, r]$ 中的边形成环，对答案没有贡献。反之它与 $[l, r]$ 中的边是不能形成环的，那么它对答案的贡献为 -1。
- 总时间复杂度为 $O((n + m + q) \log n)$。

例题 14

Peaks 加强版 (BZOJ3551)

在 Bytemountains 有 n 座山峰，每座山峰有他的高度 h_i。有些山峰之间有双向道路相连，共 m 条道路，每条道路有一个困难值，这个值越大表示越难走。
现在有 q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出 -1。
$n \leq 10^5$，$m, q \leq 5 \times 10^5$，强制在线。

例题 14

Peaks 加强版 (BZOJ3551)

在 Bytemountains 有 n 座山峰，每座山峰有他的高度 h_i。有些山峰之间有双向道路相连，共 m 条道路，每条道路有一个困难值，这个值越大表示越难走。
现在有 q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出 -1。
$n \leq 10^5$，$m, q \leq 5 \times 10^5$，强制在线。

- 离线的问题我们已经分析过了。

例题 14

Peaks 加强版 (BZOJ3551)

在 Bytemountains 有 n 座山峰，每座山峰有他的高度 h_i。有些山峰之间有双向道路相连，共 m 条道路，每条道路有一个困难值，这个值越大表示越难走。
现在有 q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出 -1。
$n \leq 10^5$，$m, q \leq 5 \times 10^5$，强制在线。

- 离线的问题我们已经分析过了。
- 对于在线算法，可以改造生成树，在添加边 (u, v, w) 时不直接将 u 在并查集上的根 x 父亲设为 v 在并查集上的根 y，而是新建结点 z 并连接 $z \to x$ 和 $z \to y$，同时将 z 的权值赋为 w。

例题 14

Peaks 加强版 (BZOJ3551)

在 Bytemountains 有 n 座山峰，每座山峰有他的高度 h_i。有些山峰之间有双向道路相连，共 m 条道路，每条道路有一个困难值，这个值越大表示越难走。
现在有 q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出 -1。
$n \leq 10^5$，$m, q \leq 5 \times 10^5$，强制在线。

- 离线的问题我们已经分析过了。
- 对于在线算法，可以改造生成树，在添加边 (u, v, w) 时不直接将 u 在并查集上的根 x 父亲设为 v 在并查集上的根 y，而是新建结点 z 并连接 $z \to x$ 和 $z \to y$，同时将 z 的权值赋为 w。
- 这样我们发现从 v 开始只经过权值小于等于 x 的路径能走到的结点集合就对应生成树上的一棵子树，其根节点为距离 v 最远的权值不超过 x 的结点。

例题 14

Peaks 加强版 (BZOJ3551)

在 Bytemountains 有 n 座山峰，每座山峰有他的高度 h_i。有些山峰之间有双向道路相连，共 m 条道路，每条道路有一个困难值，这个值越大表示越难走。
现在有 q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出 -1。
$n \leq 10^5$，$m, q \leq 5 \times 10^5$，强制在线。

- 离线的问题我们已经分析过了。
- 对于在线算法，可以改造生成树，在添加边 (u, v, w) 时不直接将 u 在并查集上的根 x 父亲设为 v 在并查集上的根 y，而是新建结点 z 并连接 $z \to x$ 和 $z \to y$，同时将 z 的权值赋为 w。
- 这样我们发现从 v 开始只经过权值小于等于 x 的路径能走到的结点集合就对应生成树上的一棵子树，其根节点为距离 v 最远的权值不超过 x 的结点。
- 使用 DFS 序和可持久化线段树维护答案信息，就可以在 $O((n+m)\log n) - O(\log n)$ 的时间复杂度解决该问题。

例题 15

线路规划（原创）

给定一棵 n 个结点的有根树，1 号结点为根节点。你需要根据 m 条线路重构一棵生成树。
其中第 i 条线路可以用四元组 (x_i, y_i, k_i, w_i) 来表达：

- 你可以在 x_i 和 y_i、x_i 的父亲和 y_i 的父亲、……、x_i 的 $k_i - 1$ 级祖先和 y_i 的 $k_i - 1$ 级祖先之间任意添加边，每一条的费用都是 w_i。

求总费用之和最小的方案，保证题目有解且 x_i 和 y_i 均至少有 $k_i - 1$ 个祖先。
$n, m \leq 152501$

cont'd

- 先从简单情况入手，考虑解决链上问题。

cont'd

- 先从简单情况入手，考虑解决链上问题。
- 按照 Kruskal 算法先将边权进行排序，那么我们需要做的其实是这样一件事情：
- 将区间 $[l_1, r_1]$ 里每一个结点向区间 $[l_2, r_2]$ 里对应的结点连边。

cont'd

- 考虑使用倍增 ST 表的思想，维护 $\log n$ 层并查集。其中第 i 层的并查集 $pa_i(x)$ 维护的是以 x 为起点的长度为 2^i 的一段区间的连通信息。

cont'd

- 考虑使用倍增 ST 表的思想，维护 $\log n$ 层并查集。其中第 i 层的并查集 $pa_i(x)$ 维护的是以 x 为起点的长度为 2^i 的一段区间的连通信息。
- 当我们合并一对长度为 2^i 的区间 $[x, x+2^i-1]$ 与 $[y, y+2^i-1]$ 时，先检查 $pa_i(x)$ 是否已经和 $pa_i(y)$ 连通，否则将 $pa_i(x)$ 与 $pa_i(y)$ 合并，并且暴力递归第 $i-1$ 层的 (x, y) 和 $(x+2^{i-1}, y+2^{i-1})$ 进行合并。这样的合并显然至多会发生 $O(n \log n)$ 次。

cont'd

- 考虑使用倍增 ST 表的思想，维护 $\log n$ 层并查集。其中第 i 层的并查集 $pa_i(x)$ 维护的是以 x 为起点的长度为 2^i 的一段区间的连通信息。
- 当我们合并一对长度为 2^i 的区间 $[x, x+2^i-1]$ 与 $[y, y+2^i-1]$ 时，先检查 $pa_i(x)$ 是否已经和 $pa_i(y)$ 连通，否则将 $pa_i(x)$ 与 $pa_i(y)$ 合并，并且暴力递归第 $i-1$ 层的 (x, y) 和 $(x+2^{i-1}, y+2^{i-1})$ 进行合并。这样的合并显然至多会发生 $O(n \log n)$ 次。
- 这样在合并一对普通区间时可以类似 RMQ 问题将其拆成两对长度为 2 的整数次幂的区间，问题就能得到解决。时间复杂度为 $O(m + n \log n \alpha(n))$。

cont'd

- 考虑使用倍增 ST 表的思想，维护 $\log n$ 层并查集。其中第 i 层的并查集 $pa_i(x)$ 维护的是以 x 为起点的长度为 2^i 的一段区间的连通信息。
- 当我们合并一对长度为 2^i 的区间 $[x, x+2^i-1]$ 与 $[y, y+2^i-1]$ 时，先检查 $pa_i(x)$ 是否已经和 $pa_i(y)$ 连通，否则将 $pa_i(x)$ 与 $pa_i(y)$ 合并，并且暴力递归第 $i-1$ 层的 (x, y) 和 $(x+2^{i-1}, y+2^{i-1})$ 进行合并。这样的合并显然至多会发生 $O(n \log n)$ 次。
- 这样在合并一对普通区间时可以类似 RMQ 问题将其拆成两对长度为 2 的整数次幂的区间，问题就能得到解决。时间复杂度为 $O(m + n \log n\alpha(n))$。
- 然后树上问题只需要使用树链剖分将两条路径分解成 $O(\log n)$ 对 DFS 序上的区间，套用链上算法即可。总时间复杂度为 $O(m \log n + n \log n\alpha(n))$。

课后练习

- [BeiJing2016] 回转寿司 传送门
- [NOI2014] 魔法森林 传送门
- [Tjoi2016] 序列 传送门
- [BeiJing2015] 骑士的旅行 传送门

Q & A