Shu 据结构乱讲

qwq123

CJ 2020 信息组

2024 年 1 月 26 日

前言

数据结构内容其实挺多的，还要加上树的部分，所以不会特别具体，建议在课后多写些题，加深对算法的理解，也积累一些经典套路。

由于本人已经变成 ACMer 了，接触的题目可能有些过时，欢迎补充。

太过高端的科技本人也不会（还是太菜了）

今天的任务

1. 数据结构回顾
   - 并查集，树状数组，线段树，trie 树，平衡树，可并堆
   - 维护信息的一些技巧
   - 偏序关系
   - 分块，莫队
2. 树论
   - 重链剖分，长链剖分
   - 树分治
   - 题目

并查集

2021.9.27T4 部落冲突 Part

在有向图中，定义好的连通块为：

- 将所有点染成红黄蓝三种颜色，存在一种染色方式，使红点的直接相连的点是黄点，黄点指向蓝点，蓝点指向红点。

初始共 n 个点，没有边，一共 q 次加边操作，求每次加边后好的连通块的数量。数据范围：$n, q \leq 10^6$

2021.9.27T4 部落冲突 Part

用并查集维护连通块。怎么维护颜色？

2021.9.27T4 部落冲突 Part

用并查集维护连通块。怎么维护颜色？

实际上就是一个带权并查集，把红色看成 0，黄色 1，蓝色 2，那么一个点的颜色就是距离根节点的距离 $\mathrm{mod}3$，记为 d_x。

考虑加边操作 $x \to y$

- 若 x, y 在同一连通块内，那么判断一下 $d_x + 1 == d_y$
- 若 x, y 在不在同一连通块内，那么 fa_x 到 fa_y 边的距离就是 $d_y + 1 - d_x$ 。

平衡树

还没有好好讲平衡树呢，但其实近几年没有什么平衡树的好题，所以不仔细讲了。

用哪一种平衡树？只要你不学 lct，那么掌握 fhq treap 就可以了。

记住只能平衡树能维护的几个操作：

- 区间翻转。
- 涉及元素在位置上的改变（插入，删除，提前。。。）

一般难一点的题目就会考你反转之后对信息的改变情况，这里基本要具体问题具体分析，就不展开讲了。

平衡树

GYM 104787 Phony

长度为 n 的序列，有 m 次操作：

- C t，表示进行 t 次给最大的数减 k (k 为定值)
- A x，求第 x 大的数。

$n, m \leq 5 \times 10^5$ 。

平衡树

GYM 104787 Phony

长度为 n 的序列，有 m 次操作：

- C t，表示进行 t 次给最大的数减 k (k 为定值)
- A x，求第 x 大的数。

$n, m \leq 5 \times 10^5$ 。

题解

应该是比较直接的题目，用平衡树维护最大的那一坨数，整体处理 C 操作，直到有其他数成为最大的数。

平衡树

GYM 104787 Phony

长度为 n 的序列，有 m 次操作：

- C t，表示进行 t 次给最大的数减 k (k 为定值)
- A x，求第 x 大的数。

$n, m \leq 5 \times 10^5$ 。

题解

应该是比较直接的题目，用平衡树维护最大的那一坨数，整体处理 C 操作，直到有其他数成为最大的数。

但是写起来比较复杂（但也贴合当下 OI 题的特色），建议想好再写。

比如直接计算其他数加入平衡树需要多少次操作比较复杂，可以考虑执行过一次操作后看当前平衡树维护的最小的数是否小于其他数中最大的数。

Trie

除了 01trie 处理亦或最大值还有什么运用吗？(确信

那么我们就讲讲 01trie，

Trie

P5283 [十二省联考 2019] 异或粽子

区间 $[l, r]$ 的权值定义为 $\mathrm{Xor}_{i=l}^{r} a_i$，求前 k 大区间的权值和。
$(n \leq 5 \times 10^5, k \leq 2 \times 10^5)$

Trie

P5283 [十二省联考 2019] 异或粽子

区间 $[l, r]$ 的权值定义为 $\text{Xor}_{i=l}^{r} a_i$，求前 k 大区间的权值和。$(n \leq 5 \times 10^5, k \leq 2 \times 10^5)$

题解

前缀和处理过后，题目就变成了给定 n 个整数，求两两异或值前 k 大的和。

Trie

P5283 [十二省联考 2019] 异或粽子

区间 $[l, r]$ 的权值定义为 $\text{Xor}_{i=l}^{r} a_i$，求前 k 大区间的权值和。$(n \leq 5 \times 10^5, k \leq 2 \times 10^5)$

题解

前缀和处理过后，题目就变成了给定 n 个整数，求两两异或值前 k 大的和。

区间的选择必然包括一个左端点小于右端点的条件，但我们只需要将 k 乘 2 就可以不用考虑这个限制了。

之后就很简单了，给 n 个数建立 01trie，让各个 a_i 去找最大异或值，将这个值存入堆中。

Trie

P5283 [十二省联考 2019] 异或粽子

区间 $[l, r]$ 的权值定义为 $\text{Xor}_{i=l}^{r} a_i$，求前 k 大区间的权值和。$(n \leq 5 \times 10^5, k \leq 2 \times 10^5)$

题解

前缀和处理过后，题目就变成了给定 n 个整数，求两两异或值前 k 大的和。

区间的选择必然包括一个左端点小于右端点的条件，但我们只需要将 k 乘 2 就可以不用考虑这个限制了。

之后就很简单了，给 n 个数建立 01trie，让各个 a_i 去找最大异或值，将这个值存入堆中。

怎么继续找次大值呢？记录一下上一次结束时的节点标号，向上回溯继续遍历即可。

Trie

CF241B Friends

和上题一样 $(n \leq 5 \times 10^4, k \leq n * (n+1)/2)$

Trie

CF241B Friends

和上题一样 $(n \leq 5 \times 10^4, k \leq n * (n+1)/2)$

题解

二分答案 +01trie 上的简单查找，应该是比较显然的。

对顶堆

不记得是什么题

一个集合，每次加入一个数，求每次操作后的中位数

对顶堆

不记得是什么题

一个集合，每次加入一个数，求每次操作后的中位数

维护一个大根堆，维护最小值 $\sim$ 中位数，一个小根堆，维护中位数 $+1 \sim$ 最大值。

然后处理一下即可，比平衡树应该会好写一些。

可并堆（左偏树）

先简单过一下可并堆这个东西。

顾名思义，可并队就是可以合并的堆，可以完全套用标记系统，只是由于堆维护的信息或许有些单一，所以你只要看到有关最值和合并的题目就可以想到可并堆。

它是怎么实现的呢？

在堆的基础上引入 di_x 为 x 节点距离最近且在 x 子树中的叶子节点的距离（从左右儿子处转移），同时保证左儿子的 di 大于等于右儿子，合并的时候就直接递归右儿子与该堆继续合并，可以证明一直向右儿子递归深度不会超过 $\log n$ 层，所以可以保证复杂度。

可并堆（左偏树）

直接看代码吧

```
int merge(int x, int y) {
    if (!x || !y) return x | y;  // 若一个堆为空则返回另一个堆
    if (t[x].val > t[y].val) swap(x, y);  // 取值较小的作为根
    t[x].rs = merge(t[x].rs, y);// 递归合并右儿子与另一个堆
    if (t[t[x].rs].d > t[t[x].ls].d)
    swap(t[x].ls, t[x].rs);//若不满足左偏性质则交换左右儿子
    t[x].d = t[t[x].rs].d + 1;  // 更新dist
    return x;
}
```

可并堆（左偏树）

想要删除任意节点，直接合并左右儿子，然后 pushup 就可以了。

```
void pushup(int x) {
    if (!x) return;
    if (t[t[x].son[0]].d > t[t[x].son[1]].d)
    swap(t[x].son[0], t[x].son[1]);
    if (t[x].d != t[t[x].son[1]].d + 1) {
        t[x].d = t[t[x].son[1]].d + 1;
        pushup(t[x].fa);
    }
}
void delete(int x){
    int fa=t[x].fa;
    t[fa].son[t[fa].son[1]==x]=merge(t[x].son[0],t[x].son[1]);
    pushup(fa);
}
```

因为会带来持续 pushup 的一定是从右儿子上来的（要么是左儿子被交换了），所以复杂度是 $O(\log n)$

可并堆（左偏树）

P3261 [JLOI2015] 城池攻占

一棵树，m 个骑士在 u_i 一直向上走，如果当前骑士权值大于节点权值继续走，并且权值会变化（加，减，乘正数），否则停止。问最终状态。

题解

涉及不断删除最小值，堆可以做到，树结构，可并堆。

可并堆（左偏树）

P3261 [JLOI2015] 城池攻占

一棵树，m 个骑士在 u_i 一直向上走，如果当前骑士权值大于节点权值继续走，并且权值会变化（加，减，乘正数），否则停止。问最终状态。

题解

涉及不断删除最小值，堆可以做到，树结构，可并堆。

现在就剩如何处理修改了，打标记！修改并不会改变值的相对大小。

可并堆（左偏树）

P3261 [JLOI2015] 城池攻占

一棵树，m 个骑士在 u_i 一直向上走，如果当前骑士权值大于节点权值继续走，并且权值会变化（加，减，乘正数），否则停止。问最终状态。

题解

涉及不断删除最小值，堆可以做到，树结构，可并堆。

现在就剩如何处理修改了，打标记！修改并不会改变值的相对大小。

加强：如果需要乘负数怎么办？

可并堆（左偏树）

P3261 [JLOI2015] 城池攻占

一棵树，m 个骑士在 u_i 一直向上走，如果当前骑士权值大于节点权值继续走，并且权值会变化（加，减，乘正数），否则停止。问最终状态。

题解

涉及不断删除最小值，堆可以做到，树结构，可并堆。

现在就剩如何处理修改了，打标记！修改并不会改变值的相对大小。

加强：如果需要乘负数怎么办？

再开一个大根堆，乘上负数就转换堆，还需要一个删除任意节点的操作。

维护信息的一些技巧

这一部分就算是一些套路了…

注意一个盲点：以下技巧一般会在线段树中出现，但是要记住他不一定就只在线段树中出现。

维护较复杂的信息

题目需要维护的信息可能需要维护多个变量，第一次做很难一下看出，所以平常练习中的积累很重要

维护较复杂的信息

题目需要维护的信息可能需要维护多个变量，第一次做很难一下看出，所以平常练习中的积累很重要

动态最大子段和

共 n 个数，两种操作

- 操作 0 x y，把 A_x 修改为 y。
- 操作 1 l r，询问区间 $[l, r]$ 的最大子段和。

维护较复杂的信息

题目需要维护的信息可能需要维护多个变量，第一次做很难一下看出，所以平常练习中的积累很重要

动态最大子段和

共 n 个数，两种操作

- 操作 0 x y，把 A_x 修改为 y。
- 操作 1 l r，询问区间 $[l, r]$ 的最大子段和。

不用多说了吧，每个区间维护包括左端点，右端点的最大子段和，目的是在两区间合并的时候维护包含两个区间的最大子段和。这种维护包含左右端点的信息，合并时处理的思想很常见。

维护较复杂的信息

P6845 [CEOI2019] Dynamic Diameter

一颗树，有 q 次修改，每次修改要修改某一条边的边权，之后查询整树直径。

$n, q \leq 10^5$

维护较复杂的信息

P6845 [CEOI2019] Dynamic Diameter

一颗树，有 q 次修改，每次修改要修改某一条边的边权，之后查询整树直径。

$n, q \leq 10^5$

题解

算是一个比较经典的题目。

给树标上欧拉序（到达一个点就加入一次，包括从子节点回溯到当前节点），这样两个点 x, y 的 lca 就是 $x \sim y$ 中 deep 最小的点，而距离就是 $d_x + d_y - 2 \times \min_{x \leq i \leq y}\{dep_i\}$。我们就记区间 $[x, y]$ 的权值为这个。

所以树的直径就是所有区间中权值最大的。

维护较复杂的信息

题解

怎么维护区间的信息？这个区间上有哪些信息？

维护较复杂的信息

题解

怎么维护区间的信息？这个区间上有哪些信息？

两端端点上的值，中间的最小值。如果确定了一个端点和中间的最小值，我们就可以选一个另一边的最大值。为什么？

维护较复杂的信息

题解

怎么维护区间的信息？这个区间上有哪些信息？

两端端点上的值，中间的最小值。如果确定了一个端点和中间的最小值，我们就可以选一个另一边的最大值。为什么？

所以大致的思路就出来了：选取一边的 mi 和另一边的 ma，合并之后成了向左扩展/向右扩展，即缺了哪个端点。又因为我们要贡献最大，所以最小值不小并不会影响我们最终的结果，所以正常合并就行了。

维护较复杂的信息

合并代码

```
void up(int p){
    tr[p].ma=max(tr[p1].ma,tr[p2].ma);
    tr[p].mi=min(tr[p1].mi,tr[p2].mi);
    tr[p].lma=max(tr[p1].lma,tr[p2].lma,tr[p2].ma-tr[p1].mi*2);
    tr[p].rma=max(tr[p1].rma,tr[p2].rma,tr[p1].ma-tr[p2].mi*2);
    tr[p].ans=max(tr[p1].ans,tr[p2].ans,tr[p1].rma+tr[p2].ma,tr
        [p2].lma+tr[p1].ma);
}
```

不要觉得这里有 lma 和 rma 就和最大子段和类似，这里是确定端点，上面是可以向左右扩展!

区间信息量随修改不断减少

在数据结构的处理中，常常有一些操作区间操作看起来很高级，无法通过懒标记记录和下传，那么好好想一想，这个操作会不会减少区间的信息量，如果可以那么我们就可以试一试暴力修改。

区间信息量随修改不断减少

在数据结构的处理中，常常有一些操作区间操作看起来很高级，无法通过懒标记记录和下传，那么好好想一想，这个操作会不会减少区间的信息量，如果可以那么我们就可以试一试暴力修改。

最经典的问题莫过于区间开根，查询区间和。此处的信息量就是数的大小，一个数在经过 $\log\log S$ 次开根之后就会变为 1，信息量就保持不变了，之后我们就可以忽略开根带来的影响。

区间信息量随修改不断减少

在数据结构的处理中，常常有一些操作区间操作看起来很高级，无法通过懒标记记录和下传，那么好好想一想，这个操作会不会减少区间的信息量，如果可以那么我们就可以试一试暴力修改。

最经典的问题莫过于区间开根，查询区间和。此处的信息量就是数的大小，一个数在经过 $\log\log S$ 次开根之后就会变为 1，信息量就保持不变了，之后我们就可以忽略开根带来的影响。

loj6029. 「雅礼集训 2017 Day1」市场

一个序列，几个操作：

- 对于 $i \in [l, r], a_i \leftarrow \lfloor \frac{a_i}{d} \rfloor$
- 对于 $i \in [l, r], a_i \leftarrow a_i + c$
- 对于 $i \in [l, r]$，求 $\sum_{i=l}^{r} a_i$ 和 $\max_{i=l}^{r} a_i$

区间信息量随修改不断减少

题解

由于区间加的存在，区间的数并不会一直减小，那么什么信息是会不断减少的呢？

区间信息量随修改不断减少

题解

由于区间加的存在，区间的数并不会一直减小，那么什么信息是会不断减少的呢？

区间的极差！对一个区间加是不改变它的极差的。但最多进行 $\log S$ 次就会让极差变为 0，一旦极差变成了 0，我们就可以转换为区间减了。

但这样是不完整的，由于向下取整，可能会极差不变，怎么办？

区间信息量随修改不断减少

题解

由于区间加的存在，区间的数并不会一直减小，那么什么信息是会不断减少的呢？

区间的极差！对一个区间加是不改变它的极差的。但最多进行 $\log S$ 次就会让极差变为 0，一旦极差变成了 0，我们就可以转换为区间减了。

但这样是不完整的，由于向下取整，可能会极差不变，怎么办？

观察得到下取整最多带来 1 的误差，所以极差为 1 时才有可能除完极差不变，此时两个数的减小量是一样的，我们也可以通过区间减来处理。

区间信息量随修改不断减少

P3747 [六省联考 2017] 相逢是问候

一共有 m 个操作，可以分为两种：

- 对于 $i \in [l, r]$, $a_i \leftarrow c^{a_i}$。
- 求 $\sum_{i=l}^{r} a_i \bmod p$

区间信息量随修改不断减少

P3747 [六省联考 2017] 相逢是问候

一共有 m 个操作，可以分为两种：

- 对于 $i \in [l, r], a_i \leftarrow c^{a_i}$。
- 求 $\sum_{i=l}^{r} a_i \bmod p$

题解

看上去完全没法处理，这时候就需要考虑处理 $c^{c^{c^{\cdots}}} \bmod p$，根据扩展欧拉定理，$c^{c^{c^{\cdots}}} \bmod p = c^{(c^{c^{\cdots}} \bmod \varphi(p)+\varphi(p))} \bmod p$（前提是 $c^{c^{\cdots}} \geq p$）

而一个数最多进行 $\log$ 次 $n \leftarrow \varphi(n)$ 操作，所以最多修改 $\log$ 次之后 $n = 1$，之后就不会再产生修改了。

修改的时候也按照上面的公式，只是需要注意，若 $c^{c^{\cdots}} < \varphi(p)$ 的时候，是不能用这个公式的，所以我们可以强制修改取模操作。

李超线段树

用来维护线段/直线的强力工具。

标记结构类似标记永久化。

一定要注意：在加入线段的情况下，不要认为一个节点的 id 为 0 表示次区间内没有线段!!!

李超线段树

CF932F Escape Through Leaf

给定以 1 为根的树，每个节点有 2 个权值 ai 和 bi。你可以花费 $a_u \times b_v$ 的代价从 u 跳到它的任意一个其子树中的节点 v 上。对每个节点计算它跳到某个叶子节点的最小花费。

李超线段树

CF932F Escape Through Leaf

给定以 1 为根的树，每个节点有 2 个权值 ai 和 bi。你可以花费 $a_u \times b_v$ 的代价从 u 跳到它的任意一个其子树中的节点 v 上。对每个节点计算它跳到某个叶子节点的最小花费。

题解

有转移:$f[u] = \min_{\text{v 在 u 的子树中}} f[v] + a_u \times b_v$。

还是李超线段树维护直线，问题是在树上怎么搞？

发现需要子树不断合并，启发式合并？$O(n \log^2 n)$

李超线段树

题解

然而可以直接在线段树合并之后加入线段。

```
int hb(int p1,int p2,int l,int r){
    if(!p1||!p2)return p1|p2;
    if(l^r){
        int mid=(l+r)>>1;
        tr[p1].ls=hb(tr[p1].ls,tr[p2].ls,l,mid);
        tr[p1].rs=hb(tr[p1].rs,tr[p2].rs,mid+1,r);
    }
    p1=change(p1,tr[p2].id,l,r);//这就是正常的加入线段操作
    return p1;
}
```

复杂度是 $O(n\log n)$，为什么呢？

因为一条线段有且只会存在一个区间内，假如我们将这 n 条线段一次性加入，复杂度是 $O(n\log n)$ 的，线段树合并只是相当于分布进行。

吉老师线段树

区间取 min/max

问题形式：给定 l, r, x，**要求对** $[l, r]$，**使** $A_i \leftarrow \min(A_i, x)$

吉老师线段树

区间取 min/max

问题形式：给定 l, r, x，要求对 $[l, r]$，使 $A_i \leftarrow \min(A_i, x)$

做法：对于线段树的每一个节点都额外维护三个值：最大值 ma、严格次大值 $ma2$、最大值的个数 cnt。对于和节点 x 取 min 操作：

吉老师线段树

区间取 min/max

问题形式：给定 l, r, x，要求对 $[l, r]$，使 $A_i \leftarrow \min(A_i, x)$

做法：对于线段树的每一个节点都额外维护三个值：最大值 ma、严格次大值 $ma2$、最大值的个数 cnt。对于和节点 x 取 min 操作：

- 若当前区间内 $x \geq ma$，那么无法产生修改，直接返回
- 若当前区间内 $ma > x > ma2$，那么我们知道这个区间内有 cnt 个数会由 ma 变为 x，之后的修改依题面来。
- 若当前区间内 $ma2 \geq x$，那么继续递归，直到满足上面的条件。

吉老师线段树

区间取 min/max

问题形式：给定 l, r, x，要求对 $[l, r]$，使 $A_i \leftarrow \min(A_i, x)$

做法：对于线段树的每一个节点都额外维护三个值：最大值 ma、严格次大值 $ma2$、最大值的个数 cnt。对于和节点 x 取 min 操作：

- 若当前区间内 $x \geq ma$，那么无法产生修改，直接返回
- 若当前区间内 $ma > x > ma2$，那么我们知道这个区间内有 cnt 个数会由 ma 变为 x，之后的修改依题面来。
- 若当前区间内 $ma2 \geq x$，那么继续递归，直到满足上面的条件。

核心：将区间取 max 的问题转化为了区间部分数的加减问题。

吉老师线段树

区间取 min/max

复杂度：$O(n\log n)$，带上修改之后就是近似于 $O(n\log n)$ 的 $O(n\log^2 n)$

搭配：可以搭配上所有除了需要依赖**数组下标**的信息。原因是我们不知道对于区间修改的最大值究竟是来着哪里的。

下面是一些典型问题

吉老师线段树

CF1290E Cartesian Tree

给出一个排列 P，对于每个 $1 \leq k \leq n$，询问当保留 $\leq k$ 的数时，该序列笛卡尔树 (大根堆) 的子树大小和。

吉老师线段树

CF1290E Cartesian Tree

给出一个排列 P，对于每个 $1 \le k \le n$，询问当保留 $\le k$ 的数时，该序列笛卡尔树 (大根堆) 的子树大小和。

题解

对于一个 k，一个节点 i 所在子树的大小就是在只保留 $\le k$ 的数组成的排列里，右边第一个大于它的位置为 ne_i，减去左边第一个大于它的位置 pr_i。

吉老师线段树

CF1290E Cartesian Tree

给出一个排列 P，对于每个 $1 \le k \le n$，询问当保留 $\le k$ 的数时，该序列笛卡尔树 (大根堆) 的子树大小和。

题解

对于一个 k，一个节点 i 所在子树的大小就是在只保留 $\le k$ 的数组成的排列里，右边第一个大于它的位置为 ne_i，减去左边第一个大于它的位置 pr_i。

接下来只考虑维护 ne_i（pr 就是序列翻转之后的 ne）

考虑从小到大在 pos_k 插入 k，那么 pos_k 左边的 $ne_i \leftarrow \min(ne_i, pos_k)$，右边的 $ne_i \leftarrow ne_i + 1$。修改之后统计全局和。

吉老师线段树

CF1290E Cartesian Tree

给出一个排列 P，对于每个 $1 \le k \le n$，询问当保留 $\le k$ 的数时，该序列笛卡尔树 (大根堆) 的子树大小和。

题解

对于一个 k，一个节点 i 所在子树的大小就是在只保留 $\le k$ 的数组成的排列里，右边第一个大于它的位置为 ne_i，减去左边第一个大于它的位置 pr_i。

接下来只考虑维护 ne_i（pr 就是序列翻转之后的 ne）

考虑从小到大在 pos_k 插入 k，那么 pos_k 左边的 $ne_i \leftarrow \min(ne_i, pos_k)$，右边的 $ne_i \leftarrow ne_i + 1$。修改之后统计全局和。

因为要考虑区间加的时候这个区间有几个位置加入了数，还要维护一个变量 su_b 记录插入的个数。代码

吉老师线段树

「JOISC 2021 Day1」饮食区

题面

吉老师线段树

「JOISC 2021 Day1」饮食区

题面

题解

首先考虑没有离开操作，那么对于每个询问，我们只需要知道最早在哪一次操作队列 A 的大小 $\geq B$。

这可以对所有询问离线，然后将每个询问用 vector 记录在对应的位置，用线段树维护区间中询问的最小值。一个加入操作，等价于区间减，当某个位置 ≤ 0 时，该位置对应询问的答案就是当前操作的标号 C 。

吉老师线段树

「JOISC 2021 Day1」饮食区

题面

题解

现在考虑存在离开操作，如果我们知道当前询问之前队列删除了多少个数，则可以把问题转换为没有离开操作。因为假定删除了 k 个数，相当于查询第 B+k 个加入队列的数。

而如果我们知道当前队列的大小，还知道有多少数入过队，就能得到出队的数的个数。

后者可以直接树状数组维护区间加，单点查询。

前者需要支持区间加、区间减、区间取 max。

吉老师线段树

历史最值的区间最值

问题形式：额外设一个辅助数组 B，在每次修改后 $B_i = \max(B_i, A_i)$，求 $\max_{i=l}^{r} B_i$。

做法（修改只有区间加）：

吉老师线段树

历史最值的区间最值

问题形式：额外设一个辅助数组 B，在每次修改后 $B_i = \max(B_i, A_i)$，求 $\max_{i=l}^{r} B_i$。

做法（修改只有区间加）：

我们考虑暴力下传所有懒标记，设 ma, hma 分别表示：区间最值，区间历史最值。每次有一个区间需要加 t 时，$ma \leftarrow ma + t$ 然后 $hma \leftarrow \max(hma, ma)$。

吉老师线段树

历史最值的区间最值

问题形式：额外设一个辅助数组 B，在每次修改后 $B_i = \max(B_i, A_i)$，求 $\max_{i=l}^{r} B_i$。

做法（修改只有区间加）：

我们考虑暴力下传所有懒标记，设 ma, hma 分别表示：区间最值，区间历史最值。每次有一个区间需要加 t 时，$ma \leftarrow ma + t$ 然后 $hma \leftarrow \max(hma, ma)$。

设 C 为依次传来的标记序列，发现我们真正需要的只是 $\max_{i=1}^{n}\{ma_0 + \sum_{j=1}^{i} C_j\} = ma_0 + \max \sum_{j=1}^{i} C_j$，即标记序列的最大前缀和，而且这个最大前缀和的合并也很好维护。

吉老师线段树

所以，我们可以设 bj 为当前标记序列的总和，mbj 为最大前缀和。

合并过程：

```
1  mbj=max(mbj,bj+mbj2);
2  hma=max(hma,ma+mbj2);
3  ma+=bj2,bj+=bj2;
```

吉老师线段树

所以，我们可以设 bj 为当前标记序列的总和，mbj 为最大前缀和。

合并过程：

```
1 mbj=max(mbj,bj+mbj2);
2 hma=max(hma,ma+mbj2);
3 ma+=bj2,bj+=bj2;
```

核心：合并懒标记序列带来的影响。

复杂度：$O(n\log n)$

吉老师线段树

SP1557 GSS2

给出 n 个数，q 次询问，求最大子段和，相同的数只算一次。

吉老师线段树

SP1557 GSS2

给出 n 个数，q 次询问，求最大子段和，相同的数只算一次。

题解

考虑将询问离线，按右端点升序，线段树维护以该点为左端点的后缀和。随着右端点的递增，每次在 $[last_{a_i}+1, i]$ 加上 a_i，最后在 $[l, r]$ 查询历史最值。

此题我们可以知道，可以用历史最值**离线**维护区间维护最大子段和。

吉老师线段树

加入区间赋值操作

发现在一个赋值操作之后的加操作等同于赋值操作，额外开一个 $bj2, mbj2$ 表示当前赋值，所有赋值中最大的。合并代码：

代码

吉老师线段树

P6349 [PA2011]Kangaroos

给定每个元素是一个区间的序列，多次询问，给出 A,B，求出 a 中最长的区间（即这个序列中的一段），使得这个区间内每个区间都与 $[A, B]$ 有交集。

吉老师线段树

P6349 [PA2011]Kangaroos

给定每个元素是一个区间的序列，多次询问，给出 A,B，求出 a 中最长的区间（即这个序列中的一段），使得这个区间内每个区间都与 $[A, B]$ 有交集。

题解

考虑将询问离线。然后让区间元素去找符合条件的询问，符合条件则 $+1$，否则赋值为 0。

怎么找符合条件的询问？

吉老师线段树

P6349 [PA2011]Kangaroos

给定每个元素是一个区间的序列，多次询问，给出 A,B，求出 a 中最长的区间（即这个序列中的一段），使得这个区间内每个区间都与 $[A, B]$ 有交集。

题解

考虑将询问离线。然后让区间元素去找符合条件的询问，符合条件则 $+1$，否则赋值为 0。

怎么找符合条件的询问？

kdt! 因为 kdt 可以完美使用线段树的标签系统。

同时我们发现，这类多次询问满足 XXX 条件的最长连续区间长度的问题就可以用历史最值线段树解决。

吉老师线段树

加入区间取 max 操作

上面我们知道，可以将取 max 转化为区间加，所以本质是没有区别的，但是要注意，由于 max 与非 max 部分的懒标记序列已经不一样了，所以必须开两个上面的东西。

代码

吉老师线段树

加入区间取 max 操作

上面我们知道，可以将取 max 转化为区间加，所以本质是没有区别的，但是要注意，由于 max 与非 max 部分的懒标记序列已经不一样了，所以必须开两个上面的东西。

代码

区间取 min+ 区间赋值。

区间赋值 = 区间 +inf 和区间取 min

吉老师线段树

加入区间取 max 操作

上面我们知道，可以将取 max 转化为区间加，所以本质是没有区别的，但是要注意，由于 max 与非 max 部分的懒标记序列已经不一样了，所以必须开两个上面的东西。

代码

区间取 min+ 区间赋值。

区间赋值 = 区间 +inf 和区间取 min

扩展部分还没有遇到什么有代表性的运用场所，遇到了再补。可以把模板做一下。

什么是多维关系

多维关系基本上指偏序关系，即一个元素包含多个属性 $(a, b, ..)$，每次需要快速查找满足某一关系的所有元素，再进行整体处理（权值和，第 k 大...），如最基本的逆序对，每一个元素包含了位置和权值 (pos, val)，对于每一个 i，需要查询的是满足 $pos_j < pos_i \wedge a_j > a_i$ 的 j 的个数。

什么是多维关系

多维关系基本上指偏序关系，即一个元素包含多个属性 $(a, b, ..)$，每次需要快速查找满足某一关系的所有元素，再进行整体处理（权值和，第 k 大...），如最基本的逆序对，每一个元素包含了位置和权值 (pos, val)，对于每一个 i，需要查询的是满足 $pos_j < pos_i \wedge a_j > a_i$ 的 j 的个数。

在处理这类问题的时候，首先需要将题目包含的关系找到，再运用各种方法，不断消除偏序关系，最终使满足条件的元素在一个可以整体处理的数据结构中，就可以很好的解决了。

下面就将介绍一些消除偏序关系的方法。

排序

没错，最简单的方法就是排序。

设偏序关系是 $a_i < a_j$(下同)，那我们就按 a_i 降序排序，这样当处理 i 的时候，所有 $a_j > a_i$ 的 j 就已经被处理好了，这时候我们就消去了 a 这一偏序关系。

排序

没错，最简单的方法就是排序。

设偏序关系是 $a_i < a_j$(下同)，那我们就按 a_i 降序排序，这样当处理 i 的时候，所有 $a_j > a_i$ 的 j 就已经被处理好了，这时候我们就消去了 a 这一偏序关系。

当然，如果偏序关系是 $a_{j,1} < a_{i,2} < a_{j,3}$（保证 $a_{i,1} \le a_{i,2} \le a_{i,3}$）的时候，我们按 $a_{i,2}$ 排序后，可以满足 $a_{j,1} < a_{i,2}$，但 $a_{i,2} < a_{j,3}$ 的怎么满足呢？其实只要在删除 $a_{j,3} < a_{i,2}$ 的 j 就可以了。或许你已经听过这个思想，它还有另外一个名字——扫描线。

排序可以消除一维关系，剩下一维整体处理，但往往题目中的偏序关系不止两维，这时候就需要更高级的方法了。

可持久化

当一个问题是用数据结构处理，多次询问，如果对象是整体数列你会做，但询问关系的对象只是一个部分，这时候可以考虑一下可持久化。

可持久化在干什么？

可持久化

当一个问题是用数据结构处理，多次询问，如果对象是整体数列你会做，但询问关系的对象只是一个部分，这时候可以考虑一下可持久化。

可持久化在干什么？——前缀和!

为什么要可持久化？

可持久化

当一个问题是用数据结构处理，多次询问，如果对象是整体数列你会做，但询问关系的对象只是一个部分，这时候可以考虑一下可持久化。

可持久化在干什么？——前缀和！

为什么要可持久化？——处理区间问题！

怎么可持久化？

可持久化

当一个问题是用数据结构处理，多次询问，如果对象是整体数列你会做，但询问关系的对象只是一个部分，这时候可以考虑一下可持久化。

可持久化在干什么？——前缀和！

为什么要可持久化？——处理区间问题！

怎么可持久化？——动态开点，继承信息！

局限？

可持久化

当一个问题是用数据结构处理，多次询问，如果对象是整体数列你会做，但询问关系的对象只是一个部分，这时候可以考虑一下可持久化。

可持久化在干什么？——前缀和！

为什么要可持久化？——处理区间问题！

怎么可持久化？——动态开点，继承信息！

局限？——距离上一个版本只能是单点，建立之后无法修改后缀，维护的信息只能是可以分解的，不能分解的就只能是前后缀！

可持久化

当一个问题是用数据结构处理，多次询问，如果对象是整体数列你会做，但询问关系的对象只是一个部分，这时候可以考虑一下可持久化。

可持久化在干什么？——前缀和！

为什么要可持久化？——处理区间问题！

怎么可持久化？——动态开点，继承信息！

局限？——距离上一个版本只能是单点，建立之后无法修改后缀，维护的信息只能是可以分解的，不能分解的就只能是前后缀！

没了？但可持久化的核心就是这些，下面结合题目了解一下吧

可持久化

P4587 [FJOI2016] 神秘数

集合 S 的神秘数定义为最小的不能被 S 的子集的和表示的正整数。

给定正整数序列 a，每次询问由 $a_l, a_{l+1}, \cdots, a_r$ 所组成的可重集合的神秘数。

可持久化

P4587 [FJOI2016] 神秘数

集合 S 的神秘数定义为最小的不能被 S 的子集的和表示的正整数。

给定正整数序列 a，每次询问由 $a_l, a_{l+1}, \cdots, a_r$ 所组成的可重集合的神秘数。

题解

给你一个集合如何求出神秘数，考虑如果 $[1, x]$ 是能被表示的数，并存在一个数 $x_0 \in [1, x+1]$，那么可以被表示的数就更新为 $[1, x+x_0]$，这启发我们不断选取当前范围内的数的和，不断更新，直到和无法增加。由于只要能更新，范围至少会扩大一倍，所以复杂度是 $O(n \log^2 n)$ 的。

可持久化

P4587 [FJOI2016] 神秘数

集合 S 的神秘数定义为最小的不能被 S 的子集的和表示的正整数。

给定正整数序列 a，每次询问由 $a_l, a_{l+1}, \cdots, a_r$ 所组成的可重集合的神秘数。

题解

给你一个集合如何求出神秘数，考虑如果 $[1, x]$ 是能被表示的数，并存在一个数 $x_0 \in [1, x+1]$，那么可以被表示的数就更新为 $[1, x+x_0]$，这启发我们不断选取当前范围内的数的和，不断更新，直到和无法增加。由于只要能更新，范围至少会扩大一倍，所以复杂度是 $O(n \log^2 n)$ 的。

这样我们就会处理整体了，由于这里只需要统计和，加上可持久化就可以处理区间了。

可持久化

P2839 [国家集训队]middle

多个询问，求序列左端点在 [a,b] 之间，右端点在 [c,d] 之间的子区间中，最大的中位数。保证 $a < b < c < d$。强制在线。

可持久化

P2839 [国家集训队]middle

多个询问，求序列左端点在 [a,b] 之间，右端点在 [c,d] 之间的子区间中，最大的中位数。保证 $a < b < c < d$。强制在线。

题解

对于中位数，有一个很常见的做法，二分答案，然后给大于等于 mid 的标成 1，小于的标成-1，然后看区间和是否大于（等于）0

假如只有一次询问怎么搞？

可持久化

P2839 [国家集训队]middle

多个询问，求序列左端点在 [a,b] 之间，右端点在 [c,d] 之间的子区间中，最大的中位数。保证 $a < b < c < d$。强制在线。

题解

对于中位数，有一个很常见的做法，二分答案，然后给大于等于 mid 的标成 1，小于的标成-1，然后看区间和是否大于（等于）0

假如只有一次询问怎么搞？
暴力建树，维护区间和，前缀/后缀最大值。

可持久化

P2839 [国家集训队]middle

多个询问，求序列左端点在 [a,b] 之间，右端点在 [c,d] 之间的子区间中，最大的中位数。保证 $a < b < c < d$。强制在线。

题解

对于中位数，有一个很常见的做法，二分答案，然后给大于等于 mid 的标成 1，小于的标成-1，然后看区间和是否大于（等于）0

假如只有一次询问怎么搞？
暴力建树，维护区间和，前缀/后缀最大值。

多组询问？瓶颈在二分之后的建树过程，怎么优化？

可持久化

P2839 [国家集训队]middle

多个询问，求序列左端点在 [a,b] 之间，右端点在 [c,d] 之间的子区间中，最大的中位数。保证 $a < b < c < d$。强制在线。

题解

对于中位数，有一个很常见的做法，二分答案，然后给大于等于 mid 的标成 1，小于的标成-1，然后看区间和是否大于（等于）0

假如只有一次询问怎么搞？
暴力建树，维护区间和，前缀/后缀最大值。

多组询问？瓶颈在二分之后的建树过程，怎么优化？
可持久化！随着 mid 增加，修改的位置只有一个，而且对值域的查询只是一个前缀，不用管拆分的问题。

可持久化

P2633 Count on a tree

一棵点带权值的树。m 个询问，每次询问 u 和 v 这两个节点间第 k 小的点权。

可持久化

P2633 Count on a tree

一棵点带权值的树。m 个询问，每次询问 u 和 v 这两个节点间第 k 小的点权。

题解

你以为可持久化只能在序列上吗？其实树上也是可以的，想想你在树上是怎么差分的，可持久化就怎么用。

每个节点在父亲信息的基础上修改一个点，差分时选取 $u, v, lca_{u,v}, fa_{lca_{u,v}}$ 四棵树，之后的操作就一模一样了。

可持久化

到这里或许你能感受到开始说的那四个问题了吧

可持久化

到这里或许你能感受到开始说的那四个问题了吧

可持久化在干什么？——前缀和！

为什么要可持久化？——处理区间问题！

怎么可持久化？——动态开点，继承信息！

局限？——距离上一个版本只能是单点，建立之后无法修改后缀，维护的信息只能是可以分解的，不能分解的就只能是前后缀！

可持久化

到这里或许你能感受到开始说的那四个问题了吧

可持久化在干什么？——前缀和！

为什么要可持久化？——处理区间问题！

怎么可持久化？——动态开点，继承信息！

局限？——距离上一个版本只能是单点，建立之后无法修改后缀，维护的信息只能是可以分解的，不能分解的就只能是前后缀！

你再想想，如果需要修改怎么办？

可持久化

到这里或许你能感受到开始说的那四个问题了吧

可持久化在干什么？——前缀和！

为什么要可持久化？——处理区间问题！

怎么可持久化？——动态开点，继承信息！

局限？——距离上一个版本只能是单点，建立之后无法修改后缀，维护的信息只能是可以分解的，不能分解的就只能是前后缀！

你再想想，如果需要修改怎么办？

如果你真的理解了，就会懂得这个问题的含义，这就好比问你维护区间和，但要支持修改，怎么办？

可持久化

到这里或许你能感受到开始说的那四个问题了吧

可持久化在干什么？——前缀和！

为什么要可持久化？——处理区间问题！

怎么可持久化？——动态开点，继承信息！

局限？——距离上一个版本只能是单点，建立之后无法修改后缀，维护的信息只能是可以分解的，不能分解的就只能是前后缀！

你再想想，如果需要修改怎么办？

如果你真的理解了，就会懂得这个问题的含义，这就好比问你维护区间和，但要支持修改，怎么办？

树状数组/线段树，这其实就是树套树了。所以个人认为，可持久化其实就是树套树，只是外层的数是一个前缀和数组罢了。

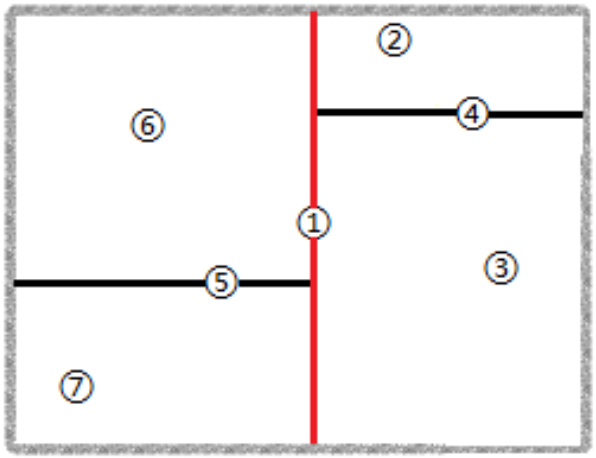
k-d tree

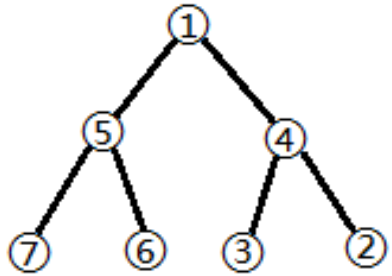
KDT 是用来处理在 k 维空间上，**维护单点**的二叉树/线段树结构。核心是将点进行不断划分，使整颗树的深度为 $O(\log n)$ 级别，对一个矩形查询/修改一次的复杂度为 $O(n^{1-\frac{1}{k}})$。

k-d tree

KDT 是用来处理在 k 维空间上，**维护单点**的二叉树/线段树结构。核心是将点进行不断划分，使整颗树的深度为 $O(\log n)$ 级别，对一个矩形查询/修改一次的复杂度为 $O(n^{1-\frac{1}{k}})$。

具体来说，KDT 每一次划分时，会择一个维度，将当前长方体按照这个维度分成两个长方体，然后左儿子后又儿子分别链接了两个长方体。所以 KDT 是一个二叉树。





洛谷

图:

k-d tree

划分的过程：

- 计算这 k 的维度的方差，找到方差最大的一维。
- 找到该维中的中位数 mid，并将在该维度上的值小于该中位数的置于中位数的左边，其余置于右边
- pushup 的时候维护包含所有点的最小矩形的边界。

代码

k-d tree

静态二维数点

二维平面中有 n 个点，有点权，有 m 次操作，要么让矩形内的点权值加，要么求矩形内权值和。

k-d tree

静态二维数点

二维平面中有 n 个点，有点权，有 m 次操作，要么让矩形内的点权值加，要么求矩形内权值和。

题解

在没有修改的情况下，我们给每个 KDT 上的点记录其子树内权值和，询问的时候若当前矩形范围被询问矩形包含即返回该权值和，若完全不包含的情况就返回 0，然后由于某种奇妙的性质，整个过程的复杂度为 $O(n^{1-\frac{1}{k}})$。

k-d tree

静态二维数点

二维平面中有 n 个点，有点权，有 m 次操作，要么让矩形内的点权值加，要么求矩形内权值和。

题解

在没有修改的情况下，我们给每个 KDT 上的点记录其子树内权值和，询问的时候若当前矩形范围被询问矩形包含即返回该权值和，若完全不包含的情况就返回 0，然后由于某种奇妙的性质，整个过程的复杂度为 $O(n^{1-\frac{1}{k}})$。

有了修改，我们给节点打上懒标记，其他的就是线段树的写法。

代码

k-d tree

动态二维数点

二维平面，有 m 次操作，要么在平面内加入一个点，要么求矩形内权值和。

k-d tree

动态二维数点

二维平面，有 m 次操作，要么在平面内加入一个点，要么求矩形内权值和。

方法一

首先暴力插入，然后模仿替罪羊树，当左右子树不平衡时重构部分子树。注意这个时候我们最好用平衡树式。

插入的过程也很简单，即判断加入点和当前点的大小关系，看是进左子树还是右子树。复杂度大约是 $O(n^{0.57})$

代码

k-d tree

动态二维数点

二维平面，有 m 次操作，要么在平面内加入一个点，要么求矩形内权值和。

方法二

设一个阈值 B，当积攒了 B 个插入后，重构整棵树。

重构复杂度为 $O(n^2 \log n/B)$ 查询复杂度为 $O(B+\sqrt{n})$，取 $B=\sqrt{n\log n}$，可得复杂度为 $O(n\sqrt{n\log n})$

代码

k-d tree

KDT 的一些特点：

- **空间** $O(n)$，但时间 $O(n\sqrt{n})$，动态插入的复杂度还要略高一些。但是建好了的点的属性是**不能改变的**！。所以一般在已知要查询的点时可以使用 KDT，需要修改点的属性的时候不能使用 KDT。
- 由于其优美的树型结构，可以完全支持线段树的所有处理办法，如吉老师线段树。所以一般在整体处理复杂**但有结合律**（这是所有可以拿标记维护的基础）的时候可以使用 KDT。
- **复杂度和值域无关**！

树套树

树套树，即在树型数据结构的**每一个节点**上再开一个树型数据结构。在第一个数据结构的处理中删掉第一个偏序关系，而第二个数据结构进行整体处理。

具体来说，如树状数组上某一个节点 x，维护的是 $[x - lowbit(x), x]$ 的信息，那么所有第一维在 $[x - lowbit(x), x]$ 的点都会在该点的所在的内层数据结构上。（线段树同理）

内层的数据结构**一定**是要能动态开点的。

树套树

动态二维数点

二维平面，有 $m \leq 10^5$ 次操作，要么在平面内加入一个点，要么求矩形内权值和，值域 $\leq 10^5$

树套树

动态二维数点

二维平面，有 $m \leq 10^5$ 次操作，要么在平面内加入一个点，要么求矩形内权值和，值域 $\leq 10^5$

题解

由于值域较小，我们可以使用树状数组套线段树，但空间仍要到 $10^5 * \log 10^5 * \log 10^5 * 2 = 57800000$ 左右，这个 512M 都开不下，所以你只能相信数据是随的。

直接看代码吧，干讲讲不出什么。

代码

树套树

动态二维数点

二维平面，有 $m \leq 10^5$ 次操作，要么在平面内加入一个点，要么求矩形内权值和，值域 $\leq 10^9$

树套树

动态二维数点

二维平面，有 $m \leq 10^5$ 次操作，要么在平面内加入一个点，要么求矩形内权值和，值域 $\leq 10^9$

题解

由于值域大了，外层还可以用 unordered_map，但内层的线段树是怎么也开不下了，所以只能改用平衡树。

同样直接看代码吧，干讲讲不出什么。

代码

树套树

动态二维数点

二维平面，有 $m \leq 10^5$ 次操作，要么在平面内加入一个点，要么删掉一个点，要么求矩形内点最大值（**无重点**），值域 $\leq 10^5$

树套树

动态二维数点

二维平面，有 $m \leq 10^5$ 次操作，要么在平面内加入一个点，要么删掉一个点，要么求矩形内点最大值（**无重点**），值域 $\leq 10^5$

题解

外层可以用树状数组吗？树状数组可以求区间 min,max 吗？

树套树

动态二维数点

二维平面，有 $m \leq 10^5$ 次操作，要么在平面内加入一个点，要么删掉一个点，要么求矩形内点最大值（**无重点**），值域 $\leq 10^5$

题解

外层可以用树状数组吗？树状数组可以求区间 min,max 吗？

所以需要外层用线段树维护。

内层用线段树维护没有问题。

树套树

动态二维数点

二维平面，有 $m \leq 10^5$ 次操作，要么在平面内加入一个点，要么删掉一个点，要么求矩形内点最大值（**有重点**），值域 $\leq 10^5$

树套树

动态二维数点

二维平面，有 $m \leq 10^5$ 次操作，要么在平面内加入一个点，要么删掉一个点，要么求矩形内点最大值（**有重点**），值域 $\leq 10^5$

题解

外层树状数组没有问题

但内层可以用线段树吗？如何处理同一位置多次加点删点？

树套树

动态二维数点

二维平面，有 $m \leq 10^5$ 次操作，要么在平面内加入一个点，要么删掉一个点，要么求矩形内点最大值（**有重点**），值域 $\leq 10^5$

题解

外层树状数组没有问题

但内层可以用线段树吗？如何处理同一位置多次加点删点？

所以只能用平衡树，代码就不展示了。

树套树

根据上面的题目你可能感受到，不同的数据结构处理的问题是不一样的，主要取决与本来这个数据结构的性质。

下面就将介绍一下各个数据结构在内层和外层的特点。

树套树

树状数组：

- 内层：树状数组放入内层的时候无法动态开点，用 unordered_map 常数巨大，所以**无法做内层**。
- 外层：可以单点加区间查，区间加单点查，无法处理区间加区间查。可处理问题少，如只能求前缀 min/max，不能区间 min/max 常数很小。

树套树

线段树（值域线段树）：

- 内层：可以处理所有单点，区间问题，在处理区间问题时注意标记永久化，可以极大减小常数。
  在有重点的情况下，处理比较困难。
  有**值域限制**，而且空间为 $O(n\log^2 n)$。
- 外层：和内层基本一致，处理此处标记无法下传，所以必须标记永久化。

树套树

平衡树：

- 内层：处理无值域限制，空间为 $O(n \log n)$
- 外层：好像要替罪羊树，没写过，唯一能多干的事就是支持动态插入。

实际问题

CF44G Shooting Gallery

给定 n 个三维空间的平面，由高度 z、x 的范围 [xl,xr] 和 y 的范围 [yl,yr] 来表示。有 m 次射击，每次射击点 (x,y)，摧毁包含此点的 z 值最小的平面，输出此平面编号，若摧毁不了任何平面，输出 0。

实际问题

CF44G Shooting Gallery

给定 n 个三维空间的平面，由高度 z、x 的范围 [xl,xr] 和 y 的范围 [yl,yr] 来表示。有 m 次射击，每次射击点 (x,y)，摧毁包含此点的 z 值最小的平面，输出此平面编号，若摧毁不了任何平面，输出 0。

题解

是不是发现不好维护矩形的删除，那么我们就反着来。

实际问题

CF44G Shooting Gallery

给定 n 个三维空间的平面，由高度 z、x 的范围 [xl,xr] 和 y 的范围 [yl,yr] 来表示。有 m 次射击，每次射击点 (x,y)，摧毁包含此点的 z 值最小的平面，输出此平面编号，若摧毁不了任何平面，输出 0。

题解

是不是发现不好维护矩形的删除，那么我们就反着来。

现在的问题就变成了：二维平面，给定一些点，点带点权，每次操作查询矩形内最小权值的点，并删除。

实际问题

CF44G Shooting Gallery

给定 n 个三维空间的平面，由高度 z、x 的范围 [xl,xr] 和 y 的范围 [yl,yr] 来表示。有 m 次射击，每次射击点 (x,y)，摧毁包含此点的 z 值最小的平面，输出此平面编号，若摧毁不了任何平面，输出 0。

题解

是不是发现不好维护矩形的删除，那么我们就反着来。

现在的问题就变成了：二维平面，给定一些点，点带点权，每次操作查询矩形内最小权值的点，并删除。

首先查询是区间最小值，外层不能用树状数组，内层线段树处理困难，平衡树常数巨大，所以推荐使用 kdt。

实际问题

P1975 [国家集训队] 排队

每次交换数列中两个数，查询整体逆序对数。

实际问题

P1975 [国家集训队] 排队

每次交换数列中两个数，查询整体逆序对数。

题解

分析一下得：求数列在 (x, y) 而且值在 (l, r) 的数的个数，然后单点修改值。

实际问题

P1975 [国家集训队] 排队

每次交换数列中两个数，查询整体逆序对数。

题解

分析一下得：求数列在 (x, y) 而且值在 (l, r) 的数的个数，然后单点修改值。

虽然是明显的二维数点，但涉及第二维值的修改，即需要维护删点和加点，难搞。不如直接树状数组套线段树。

实际问题

BZOJ4154 Generating Synergy

给定一棵有根树，支持下列操作：

- 将距离节点 u 不超过 l 的子节点权值设为 x。
- 询问点 u 的权值。

实际问题

BZOJ4154 Generating Synergy

给定一棵有根树，支持下列操作：

- 将距离节点 u 不超过 l 的子节点权值设为 x。
- 询问点 u 的权值。

题解

首先先要确定偏序关系：子树的限制是 dfs 序上的一个区间，而深度的限制是 dep 上的一个前缀。

实际问题

BZOJ4154 Generating Synergy

给定一棵有根树，支持下列操作：

- 将距离节点 u 不超过 l 的子节点权值设为 x。
- 询问点 u 的权值。

题解

首先先要确定偏序关系：子树的限制是 dfs 序上的一个区间，而深度的限制是 dep 上的一个前缀。

kdt 是肯定可以做的，打上赋值标记即可。考虑树套树怎么做。

实际问题

BZOJ4154 Generating Synergy

给定一棵有根树，支持下列操作：

- 将距离节点 u 不超过 l 的子节点权值设为 x。
- 询问点 u 的权值。

题解

首先先要确定偏序关系：子树的限制是 dfs 序上的一个区间，而深度的限制是 dep 上的一个前缀。

kdt 是肯定可以做的，打上赋值标记即可。考虑树套树怎么做。

赋值操作需要考虑的是时间的先后，给每个赋值标记加上一个时间，合并的时候取时间最大的即可。

如果加上区间加怎么搞？所以还是推荐使用 KDT。

分块

开始讲分块了啊！分块这个东西就比较尴尬，你说它不重要吧，确实还是有一堆题目和知识点的（比如 NOI2020D1T3），但你说它重要吧，那么在平常的联考的 CF、AT、NOIP、省选、近年 NOI 中又出现过吗？

所以我今天就简单过一下（主要后面还有一个树的内容）

分块

开始讲分块了啊！分块这个东西就比较尴尬，你说它不重要吧，确实还是有一堆题目和知识点的（比如 NOI2020D1T3），但你说它重要吧，那么在平常的联考的 CF、AT、NOIP、省选、近年 NOI 中又出现过吗？

所以我今天就简单过一下（主要后面还有一个树的内容）

先讲讲动态数列分块

动态数列分块可以在维护的信息无法合并的时候（也就是普通数据结构没法做的时候），通过将数列分成块，在块内用使用**离线**的算法，修改时暴力重构和整体标记，查询时散块暴力查询，整块使用离线算法，最后信息合并。

嗯，大概是这样。

动态数列分块

loj6546 简单的数列题

- 将数列 A 中区间 $[l, r]$ 内所有数加上 $w \geq 0$
- 交换 B_x 和 B_y
- 求 $\max_{i \in [l,r]} \{A_i \times B_i\}$

动态数列分块

loj6546 简单的数列题

- 将数列 A 中区间 $[l, r]$ 内所有数加上 $w \geq 0$
- 交换 B_x 和 B_y
- 求 $\max\limits_{i \in [l,r]} \{A_i \times B_i\}$

题解

发现无论怎么处理，给一段区间打上懒标记是无法避免的，但打上懒标记对 max $A_i \times B_i$ 怎么处理？

动态数列分块

loj6546 简单的数列题

- 将数列 A 中区间 $[l, r]$ 内所有数加上 $w \geq 0$
- 交换 B_x 和 B_y
- 求 $\max\limits_{i \in [l,r]} \{A_i \times B_i\}$

题解

发现无论怎么处理，给一段区间打上懒标记是无法避免的，但打上懒标记对 max $A_i \times B_i$ 怎么处理？

转化成 $(A_i + bj) \times B_i = A_i \times B_i + bj \times B_i$，这是什么式子？

动态数列分块

loj6546 简单的数列题

- 将数列 A 中区间 $[l, r]$ 内所有数加上 $w \geq 0$
- 交换 B_x 和 B_y
- 求 $\max\limits_{i \in [l,r]} \{A_i \times B_i\}$

题解

发现无论怎么处理，给一段区间打上懒标记是无法避免的，但打上懒标记对 max $A_i \times B_i$ 怎么处理？

转化成 $(A_i + bj) \times B_i = A_i \times B_i + bj \times B_i$，这是什么式子？

还有交换 B_i 的操作？在线维护直线的加入和删除？树套树好做吗？

动态数列分块

loj6546 简单的数列题

- 将数列 A 中区间 $[l,r]$ 内所有数加上 $w \geq 0$
- 交换 B_x 和 B_y
- 求 $\max\limits_{i\in[l,r]}\{A_i \times B_i\}$

题解

发现无论怎么处理，给一段区间打上懒标记是无法避免的，但打上懒标记对 max $A_i \times B_i$ 怎么处理？

转化成 $(A_i + bj) \times B_i = A_i \times B_i + bj \times B_i$，这是什么式子？

还有交换 B_i 的操作？在线维护直线的加入和删除？树套树好做吗？

离线维护凸包？貌似可以，暴力重构，线性扫描。归并可以 $O(q\sqrt{n})$

静态数列分块

除了动态的分块，还有静态的数列分块。

静态数列分块

除了动态的分块，还有静态的数列分块。

静态的数列分块是将数列每 B 个数分成一块，然后设 $g_{l,r}$ 为第 $l \sim r$ 块的答案，然后询问的时候 l, r 到对应的边界只有 $O(B)$ 个数，这些数只会带来 $O(B)$ 的影响，通过前缀和差分每种影响 $O(1)$ 处理，那么就可以通过这种分块形式解决。

静态数列分块

除了动态的分块，还有静态的数列分块。

静态的数列分块是将数列每 B 个数分成一块，然后设 $g_{l,r}$ 为第 $l \sim r$ 块的答案，然后询问的时候 l, r 到对应的边界只有 $O(B)$ 个数，这些数只会带来 $O(B)$ 的影响，通过前缀和差分每种影响 $O(1)$ 处理，那么就可以通过这种分块形式解决。

还是看题吧。

静态数列分块

P4135 作诗

查询区间内出现正偶数次数的个数，强制在线。值域在 O(n) 级别。

静态数列分块

P4135 作诗

查询区间内出现正偶数次数的个数，强制在线。值域在 O(n) 级别。

题解

考虑处理 g 数组，按照上面说的办法，发现其实需要求某个数在 $l \sim r$ 块中的答案，这个可以前缀和求，注意这里的空间达到了 n^2/B。

静态数列分块

P4135 作诗

查询区间内出现正偶数次数的个数，强制在线。值域在 O(n) 级别。

题解

考虑处理 g 数组，按照上面说的办法，发现其实需要求某个数在 $l \sim r$ 块中的答案，这个可以前缀和求，注意这里的空间达到了 n^2/B。

发现可能会影响到答案的数只会有 B 个，那么就可以单独重新考虑一下这些数的答案。查询时也是如此。代码

静态数列分块

P4135 作诗

查询区间内出现正偶数次数的个数，强制在线。值域在 O(n) 级别。

题解

考虑处理 g 数组，按照上面说的办法，发现其实需要求某个数在 $l \sim r$ 块中的答案，这个可以前缀和求，注意这里的空间达到了 n^2/B。

发现可能会影响到答案的数只会有 B 个，那么就可以单独重新考虑一下这些数的答案。查询时也是如此。代码

同样的题目还有求区间众数、区间逆序对。

分块

P4462 [CQOI2018] 异或序列

多次询问，每次问在 $[l_i, r_i]$ 区间内，有多少子段满足异或和等于 k。

分块

P4462 [CQOI2018] 异或序列

多次询问，每次问在 $[l_i, r_i]$ 区间内，有多少子段满足异或和等于 k。

题解

好好想想，这个题目怎么做。一定要想到才继续。

分块

P4462 [CQOI2018] 异或序列

多次询问，每次问在 $[l_i, r_i]$ 区间内，有多少子段满足异或和等于 k。

题解

好好想想，这个题目怎么做。一定要想到才继续。

这个例子说明了你看见一个莫队题目能不能想到是莫队。

具体解法就不说了。

分块

P4462 [CQOI2018] 异或序列

多次询问，每次问在 $[l_i, r_i]$ 区间内，有多少子段满足异或和等于 k。

题解

好好想想，这个题目怎么做。一定要想到才继续。

这个例子说明了你看见一个莫队题目能不能想到是莫队。

具体解法就不说了。

其实莫队题目大概有几个特征：可以离线、增加一个元素/删除一个元素的时间复杂度是 $O(1)$ 或 $O(\log n)$。当然，后面的一些高科技可以优化普通莫队算法。

莫队

带修的莫队？其实就是加了一个时间轴，由二维变成三维。

这里简单介绍一下树上莫队：

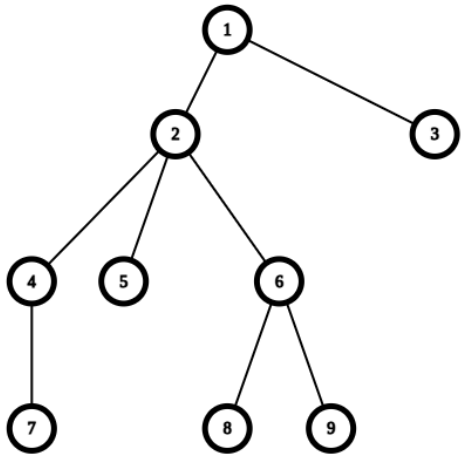
其实就是给树标上一个欧拉序（入栈和出栈的时候加入），记 in 为入栈的时间戳，out 为出栈，这样一条树上的路径会对应成一个节点的出栈到另一个节点的入栈，也就是序列上的一段区间，而我们需要的节点，也就是这条链上的节点，除了 lca 之外，其余节点全在这个区间内出现 1 次，其余点全出现 0/2 次

所以这就是加入询问的代码：

```
1  int x=read(),y=read();
2  if(out[x]<in[y])q[i]=(qq){out[x],in[y],lca(x,y),i};
3  else q[i]=(qq){out[y],in[x],lca(x,y),i};
```

莫队

举个例子：



这棵树的欧拉序就是 1,2,4,7,7,4,5,5,6,8,8,9,9,6,2,3,3,1，我们假如要询问 $9 \to 4$ 的路径，那么就对应了序列上的 4,5,5,6,8,8,9。

莫队

怎么进行莫队的转移呢？因为我们需要的节点（除 lca）全在这个区间内出现 1 次，那么在莫队指针的移动过程中，发现有出现次数为奇数的就加入，否则就删除。最后再特判 lca 的情况。

```
void calc(int x){       //对点进行加入或删除
    if(!use[x])add(x);
    else del(x);
    use[x]^=1;
}
    for (int i=1;i<=m;i++){
        while (l>q[i].l)calc(his[--l]);
        while (r<q[i].r)calc(his[++r]);
        while (l<q[i].l)calc(his[l++]);
        while (r>q[i].r)calc(his[r--]);
        if (q[i].lca)calc(q[i].lca);
        ans[q[i].id]=tmp;
        if (q[i].lca)calc(q[i].lca);
    }
```

莫队

树上莫队没有什么其他特别的地方了（记得把模板题写一下!）

接下来考虑这样一个问题。

莫队

树上莫队没有什么其他特别的地方了（记得把模板题写一下!）

接下来考虑这样一个问题。

「JOISC 2014 Day1」历史研究

多次询问，每次询问给出一个区间，定义定义事件 A 的重要度为 $A \times T_A$，其中 T_A 为该事件在区间中出现的次数。

莫队

树上莫队没有什么其他特别的地方了（记得把模板题写一下!）

接下来考虑这样一个问题。

「JOISC 2014 Day1」历史研究

多次询问，每次询问给出一个区间，定义定义事件 A 的重要度为 $A \times T_A$，其中 T_A 为该事件在区间中出现的次数。

莫队？怎么处理加入操作？又怎么处理删除操作？

莫队

树上莫队没有什么其他特别的地方了（记得把模板题写一下!）

接下来考虑这样一个问题。

「JOISC 2014 Day1」历史研究

多次询问，每次询问给出一个区间，定义定义事件 A 的重要度为 $A \times T_A$，其中 T_A 为该事件在区间中出现的次数。

莫队？怎么处理加入操作？又怎么处理删除操作？

是不是不好处理删除？那么就需要回滚莫队了，它可以保证在不改变原莫队复杂度的基础上支持不删除/不增加操作。

莫队

先按原来的莫队排序方法进行排序，但是不能使用奇偶块排序的优化，我们要求 r 单调递增。之后我们依次处理左端点在第 i 块的询问。

具体来说，我们在处理块 i 的时候，可以初始一个 $[R_i, R_i - 1]$ 的区间，在右端点可以不断向右扩展加入，而因为此时任何一个会被处理到的询问的左端点和 R_i 不超过 $\sqrt{n}$，所以我们就可以暴力加入这一段，得到答案之后也可以暴力撤回操作（就像线段树分治的撤回操作一样）。

这么一看，复杂度确实是 $O(n\sqrt{n})$，常数却大了几倍。代码

树论

树这个东西还是比较重要的，但其实不是很好讲，因为变化很多，一些 trick 需要自己积累

重链剖分

重链剖分应该是不用讲的吧

那就直接看题目吧。

重链剖分

P5305 [GXOI/GZOI2019] 旧词

n 个点的树，有 Q 次询问，每次询问给定 x, y，求 $\sum_{1 \le i \le x} \mathrm{depth}(\mathrm{lca}(i, y))^k$，其中 k 为常数。

$n, Q \le 50000, k \le 10^9$

重链剖分

P5305 [GXOI/GZOI2019] 旧词

n 个点的树，有 Q 次询问，每次询问给定 x, y，求 $\sum_{1 \le i \le x} \mathrm{depth}(\mathrm{lca}(i, y))^k$，其中 k 为常数。

$n, Q \le 50000, k \le 10^9$

题解

如果是 $k = 1$ 怎么做？

重链剖分

P5305 [GXOI/GZOI2019] 旧词

n 个点的树，有 Q 次询问，每次询问给定 x, y，求 $\sum_{1 \le i \le x} \text{depth}(\text{lca}(i, y))^k$，其中 k 为常数。

$n, Q \le 50000, k \le 10^9$

题解

如果是 $k = 1$ 怎么做？

考虑拆分 depth(lca) 的贡献，如果我们将 $x \to 1$ 路径上的每一个点的权值都加 1，那么从 $y \to 1$ 的路径上的权值和就是 $\text{depth}(\text{lca}(x, y))$

那么考虑 $k \neq 1$ 的时候怎么做？

重链剖分

P5305 [GXOI/GZOI2019] 旧词

n 个点的树，有 Q 次询问，每次询问给定 x, y，求 $\sum_{1 \le i \le x} \mathrm{depth}(\mathrm{lca}(i, y))^k$，其中 k 为常数。

$n, Q \le 50000, k \le 10^9$

题解

如果是 $k = 1$ 怎么做？

考虑拆分 depth(lca) 的贡献，如果我们将 $x \to 1$ 路径上的每一个点的权值都加 1，那么从 $y \to 1$ 的路径上的权值和就是 $\mathrm{depth}(\mathrm{lca}(x, y))$

那么考虑 $k \neq 1$ 的时候怎么做？

实际上就是做一个差分，我们只要保证某个点 z 到 1 号点路径上的权值和是 $\mathrm{depth}(z)^k$ 就行了。

重链剖分

其实你会发现，所谓的重链剖分，就是把点给重新标了一个号，之后的操作还是数据结构。

其实树上标号这个东西，用起来是很灵活的。

树上标号

CF916E Jamie and Tree

有一棵 n 个节点的有根树，标号为 $1 \sim n$，你需要维护以下三种操作

- 1 v：给定一个点 v，将整颗树的根变为 v。
- 2 u v x：给定两个点 u, v 和整数 x，将 $\mathrm{lca}(u, v)$ 为根的子树的所有点的点权都加上 x。
- 3 v：给定一个点 v，你需要回答以 v 所在的子树的所有点的权值和。

树上标号

CF916E Jamie and Tree

有一棵 n 个节点的有根树，标号为 $1 \sim n$，你需要维护以下三种操作

- 1 v：给定一个点 v，将整颗树的根变为 v。
- 2 u v x：给定两个点 u, v 和整数 x，将 $\mathrm{lca}(u, v)$ 为根的子树的所有点的点权都加上 x。
- 3 v：给定一个点 v，你需要回答以 v 所在的子树的所有点的权值和。

题解

这个换根操作看起来很高级，实际上我们只要讨论一下不同的根对我们的操作有什么影响就可以了

详情见板书。

树上标号

P3242 [HNOI2015] 接水果

给你一个树上路径集合 S，每条路径有个权值。每次询问一条路径 p:$x \to y$，问他在 S 中包含的路径中权值第 k 小的是多少。

树上标号

P3242 [HNOI2015] 接水果

给你一个树上路径集合 S，每条路径有个权值。每次询问一条路径 p:$x \to y$，问他在 S 中包含的路径中权值第 k 小的是多少。

题解

考虑一条路径 x,y，什么情况下才会有路径包含 x,y

树上标号

P3242 [HNOI2015] 接水果

给你一个树上路径集合 S，每条路径有个权值。每次询问一条路径 p:$x \to y$，问他在 S 中包含的路径中权值第 k 小的是多少。

题解

考虑一条路径 x,y，什么情况下才会有路径包含 x,y

- $lca_{x,y} = x$: 设 z 是 $x \to y$ 上的第一个点，那么就要求路径 p 满足一个节点在 y 子树内，一个节点在 z 子树外，即满足 $dfs_a \in [dfs_y, dfs_y + si_y - 1]$ 且 $dfs_b \in [1, dfs_z - 1] \cup [dfs_z + si_z, n]$
- $lca_{x,y} \neq x$: $dfs_a \in [dfs_x, dfs_x + si_x - 1], dfs_b \in [dfs_y, dfs_y + si_y - 1]$

然后就可以通过扫描线消除一维，接下来就是区间加，区间减，求第 k 大。值域线段树套线段树可以解决。

一种特殊的重链剖分

P7735 [NOI2021] 轻重边

有 n 个结点的树，树上所有边都是轻边。操作有以下两种：

- 1 a b：首先对于 a 到 b 路径上的所有点 x，将与 x 相连的所有边变为轻边。然后再将 a 到 b 路径上包含的所有边变为重边。
- 2 a b：你需要计算当前 a 到 b 的路径上一共包含多少条重边。

一种特殊的重链剖分

P7735 [NOI2021] 轻重边

有 n 个结点的树，树上所有边都是轻边。操作有以下两种：

- 1 a b：首先对于 a 到 b 路径上的所有点 x，将与 x 相连的所有边变为轻边。然后再将 a 到 b 路径上包含的所有边变为重边。
- 2 a b：你需要计算当前 a 到 b 的路径上一共包含多少条重边。

题解

主要难点就在：将 a 到 b 路径上的所有点 x 相连的所有边变为轻边。

我们可以在给一条重链标号之后，从上倒下，再给每一个重链上的点的所有轻儿子标个号。体现在代码上是这样：

一种特殊的重链剖分

```
void sous2(int x,int topp){
    if(x==topp){
        int now=son[x];
        while(now)dfs[now]=++dfss,now=son[now];
        now=x;
        while(now){
            l[now]=dfss+1;
            for(auto y:a[now])
                if(y!=son[now])dfs[y]=++dfss;
            r[now]=dfss,now=son[now];
        }
    }
    top[x]=topp;
    if(son[x])sous2(son[x],topp);
    for(auto y:a[x])
        if(y!=son[x])sous2(y,y);
}
```

一种特殊的重链剖分

这样的处理之后，发现重链上连续一段的点的轻儿子的编号都是连续的，只有重链的链顶不会和下面的点编号连续

之后的处理就只需要注意亿些细节就可以解决了。提交记录

一种特殊的重链剖分

hdu7228 Gilneas

一棵树，有 m 次操作，每次操作会把某个点到根的路径上所有边染成某个颜色，并且把这条路径上连出去的其它边的颜色清零。现在每个操作有一定的概率成功，一定的概率失败，求最终所有边上的颜色之和的期望。

一种特殊的重链剖分

hdu7228 Gilneas

一棵树，有 m 次操作，每次操作会把某个点到根的路径上所有边染成某个颜色，并且把这条路径上连出去的其它边的颜色清零。现在每个操作有一定的概率成功，一定的概率失败，求最终所有边上的颜色之和的期望。

题解

记点 x 颜色的期望是 f_x ，那么 p 概率染成 c 色的操作后，$f_x \leftarrow f_x(1-p)+p*c$ 。这是可以用线段树维护的。

那么直接上我们介绍的剖分方法，就变成了一个板子题了。

长链剖分

CF1009F Dominant Indices

给定一棵以 1 为根，n 个节点的树。设 $d(u,x)$ 为 u 子树中到 u 距离为 x 的节点数，对于每个点，求一个最小的 k，使得 $d(u,k)$ 最大。

数据范围：$n \leq 10^6$ 。

长链剖分

CF1009F Dominant Indices

给定一棵以 1 为根，n 个节点的树。设 $d(u,x)$ 为 u 子树中到 u 距离为 x 的节点数，对于每个点，求一个最小的 k，使得 $d(u,k)$ 最大。

数据范围：$n \leq 10^6$ 。

题解

首先有 dp，记 $f_{x,i}$，表示 x 的子树中与 x 距离为 i 的点的个数。转移方程如下:

$$f_{x,i} = \sum_{y \in son(x)} f_{y,i-1}$$

想优化这个转移，我们可以尝试让 f_x 继承一个子树的 f_y ，由于 f_y 和 $f_{y'}$ 合并的复杂度是 $\min(dep(y), dep(y'))$ ，那么我们就可以继承 x 子树中 dep 最大的那个，这样就保证了合并的总复杂度是 $O(n)$ 。

点分治

关于点分治和边分治，个人感觉就是将序列上的分治问题转化为了树上的分治问题，为了每次让划分的子问题更平均，所以才有了点分治的找重心、边分治的边三度化。剩下的部分基本都可以看出序列分治的影子。

想想序列的分治是怎么样的？找中点，处理跨过中点的贡献，之后在递归处理两边的子问题，既然说点分治就是树上的序列分治，那么同样的，我们找到重心，然后处理跨过重心的路径，然后递归处理子问题。

点分治有在线和离线两种写法，需要结合问题去看。

点分治

P7565 [JOISC 2021 Day3] ビFバFの会合 2

给定一棵有 N 个点的树，每一个点上有一个人，这些人要开会，假设一次会议有 P 个人参加，这第 i 个人在第 p_i 个点上。如果点 $k \in [1, n]$ 满足下面这个值最小。

$$\sum_{i=1}^{P} dis(k, p_i)$$

那么就称第 k 个点为可期待的，这场会议的期待值即为所有点中中可期待点的个数。

对于每个 $j \in [1, N]$，求当会议里有 j 个人的时候，会议的期待值的最大值是多少。

P7565 [JOISC 2021 Day3] ビFバFの会合 2

题解

分析一下，我们需要对每一个 j，求最长的链的长度，使两端链底各带有 j 个节点。链的长度就是 $2 \times j$ 的答案。

P7565 [JOISC 2021 Day3] ビFバFの会合 2

题解

分析一下，我们需要对每一个 j，求最长的链的长度，使两端链底各带有 j 个节点。链的长度就是 $2 \times j$ 的答案。

看起来，我们在递归子树的时候无法忽视在当前处理问题外的节点（也就是当前子树外的节点），但实际上，由于重心的性质（子树大小不会超过整树的一半），一旦有节点要以处理完的重心作为链底的节点，那么这条链链底的节点大小将超过当前树大小的一半，肯定不会对答案产生贡献。

现在再来考虑处理答案，开一个桶，维护当前某一链底大小为 i 的链到当前根的最大长度，这个可以直接在扫描子树的时候更新单点，之后维护后缀和处理答案就行。

注意单链也可以作为答案，就是某一链底为根的情况，这时候由于重心的性质，一定是满足条件的。

点分治

ICPC2023 西安站热身赛 C 题

给定一颗树和常数 k，边有边权，求有多少对 (x, y) ，满足 $x \to y$ 路径的最大值 $-$ 最小值 $\leq k$。

数据范围 $n \leq 10^5$

点分治

ICPC2023 西安站热身赛 C 题

给定一颗树和常数 k，边有边权，求有多少对 (x, y) ，满足 $x \to y$ 路径的最大值 $-$ 最小值 $\leq k$。

数据范围 $n \leq 10^5$

题解

考虑在线的维护怎么搞？发现要来一个动态加点的二维数点，复杂度达到了惊人的 $O(n \log^3 n)$。

点分治

ICPC2023 西安站热身赛 C 题

给定一颗树和常数 k，边有边权，求有多少对 (x, y) ，满足 $x \to y$ 路径的最大值 $-$ 最小值 $\leq k$。

数据范围 $n \leq 10^5$

题解

考虑在线的维护怎么搞？发现要来一个动态加点的二维数点，复杂度达到了惊人的 $O(n \log^3 n)$。

但实际上我们如果采用容斥 $+$ 离线的写法，就可以通过排序消除一维偏序关系，之后就是一个树状数组的单 $\log$ 就可以解决的东西。

边分治

和点分治其实差不多，只不过在某些特殊的情况下需要边分治来让合并变得简单一点。

但是需要边三度化，所以在某些不好设边权为 0 的情况下不能用边分治。

边分治

P4565 [CTSC2018] 暴力写挂

给两棵树 $T1, T2$ ，求
$\max_{x,y}\{\mathsf{depth}(x)+\mathsf{depth}(y)-\mathsf{depth}(lca_{x,y})-\mathsf{depth}'(lca'_{x,y})\}$

P4565 [CTSC2018] 暴力写挂

题解

考虑化一下式子，得到

$$\frac{1}{2}(\mathrm{depth}(x)+\mathrm{depth}(y)+dis(x,y))-\mathrm{depth}'(lca'_{x,y})$$

。

我们先考虑点分治，对于分治中心，我们遍历其子树，设 $val_x=\mathrm{depth}(x)+$ 到分治重心的距离，答案就是 $\frac{1}{2}(val_x+val_y)-\mathrm{depth}'(lca'_{x,y})$。

我们可以对这些遍历到的点在 $T2$ 上建一虚树，对于虚树上的点 x 考虑以 x 为 lca 时的答案，但是这样对于多个子树的点我们无法排除，只能用边分治，这样在只有两边的点之后我们就可以用 $g_{x,0/1}$ 来排除同树的情况了。

点分树

将普通点分治的划分过程提前处理，并且将在每次分治的时候让各个子树的分治中心连接到当前中心上

根据点分治的过程，我们重新建出来的树的树高只有 $O(\log n)$ 层，这就给了我们暴力修改/查找的机会。

点分树

hdu7358 Landmine

给定一颗树，边有边权，每个节点上有一个炸弹，爆炸范围是 a_i ，一个炸弹爆炸后，所有爆炸范围内的炸弹会在 1s 后爆炸，对每一个节点求出，引爆这个炸弹后，1 号节点的炸弹什么时候会爆炸。$n \leq 10^5$

点分树

hdu7358 Landmine

给定一颗树，边有边权，每个节点上有一个炸弹，爆炸范围是 a_i ，一个炸弹爆炸后，所有爆炸范围内的炸弹会在 1s 后爆炸，对每一个节点求出，引爆这个炸弹后，1 号节点的炸弹什么时候会爆炸。$n \leq 10^5$

题解

考虑 BFS，每次要找所有未被扫过的点中能够炸到当前点 i 的所有点，相当于找所有未被标记的点 j 使得 $dist(i,j) \leq a_j$。建立点分树，那么就相当于对所有 i 在点分树上的祖先 u，找 u 点分树子树中所有未被标记的点 j，使得 $dist(i,u) \leq a_j - dist(j,u)$。直接对每个点 u 按照 $a_j - dist(j,u)$ 将所有 j 排序即可。每次扫指针找到所有需要求的点。

题目

CF768G The Winds of Winter

给一个有根树。若我们删去一个节点和所有与他相连的边，则会得到一个森林。你希望这个森林中节点最多的树的节点个数尽量少，于是你可以进行至多一次如下操作：删除一个节点和其父亲之间的边，把这个节点连到某个节点上。这个操作不得改变森林中树的个数。

对于每个节点，输出它作为删去的节点时，进行至多一次操作后的最大树大小的最小值。$n \leq 10^5$

题目

CF768G The Winds of Winter

给一个有根树。若我们删去一个节点和所有与他相连的边，则会得到一个森林。你希望这个森林中节点最多的树的节点个数尽量少，于是你可以进行至多一次如下操作：删除一个节点和其父亲之间的边，把这个节点连到某个节点上。这个操作不得改变森林中树的个数。

对于每个节点，输出它作为删去的节点时，进行至多一次操作后的最大树大小的最小值。$n \leq 10^5$

题解

我们首先扫一遍树，对于每个点，得到删除这个点之后最大的连通块，次大的和最小的。

因为要操作的这个点肯定是在最大的那个连通块里，而这个联通块既可以是子树，也可以是父亲，所以要分开考虑。

CF768G The Winds of Winter

记 x 为当前点，y 为操作点，有以下三种情况。

- y 是 x 的子树。
- y 是 x 的父亲。
- x 是 y 没有直系关系。

三种情况对应这三种不同的 dfs 方式，什么时候加入、删除、查找会有所不同，查找的过程都是一样的，就是尽量让 k 接近 $\frac{1}{2}(ma - mi)$ ，这样可以更平均。

现在一下子没有明白可以看代码

题目

BZOJ4771 七彩树

一个每个点有颜色的树，多组询问，每次询问给出 u, d，求 u 的子树内深度不超过 $dep_u + d$ 的点集中的颜色种类数，强制在线。

题目

BZOJ4771 七彩树

一个每个点有颜色的树，多组询问，每次询问给出 u, d，求 u 的子树内深度不超过 $dep_u + d$ 的点集中的颜色种类数，强制在线。

题解

怎么处理颜色？

题目

BZOJ4771 七彩树

一个每个点有颜色的树，多组询问，每次询问给出 u, d，求 u 的子树内深度不超过 $dep_u + d$ 的点集中的颜色种类数，强制在线。

题解

怎么处理颜色？

对于对每种颜色考虑两个 dfs 序相邻的点，记为 x, y，发现 x, y 对 $x, y, lca_{x,y}$ 分别产生 $1, 1, -1$ 的贡献。

题目

BZOJ4771 七彩树

一个每个点有颜色的树，多组询问，每次询问给出 u, d，求 u 的子树内深度不超过 $dep_u + d$ 的点集中的颜色种类数，强制在线。

题解

怎么处理颜色？

对于对每种颜色考虑两个 dfs 序相邻的点，记为 x, y，发现 x, y 对 $x, y, lca_{x,y}$ 分别产生 $1, 1, -1$ 的贡献。

处理了这个问题之后就是一个二维偏序问题，强制在线的话我们可以使用可持久化消除 dep 的偏序。

ARC159E Difference Sum Query

给定 N, M 以及 M 个二元组 $(a_0, b_0), ..., (a_{M-1}, b_{M-1})$。

生成序列 $\{X_N\}$，生成方式如下：

- 首先有 $l = 1, r = N, t = 0$；
- 令 $m = \lfloor \frac{a_{t \bmod M} \times l + b_{t \bmod M} \times r}{a_{t \bmod M} + b_{t \bmod M}} \rfloor$，若 $m = i$ 则令 $X_i = t$ 并结束；
- 若 $l \leqslant i < m$，则令 $r = m - 1$，否则 $l = m + 1$，并且 $t \leftarrow t + 1$ 并返回第二步。

有 Q 个询问，每次给定区间 (c_i, d_i)，询问 $\sum_{j=c_i}^{d_i-1} |X_j - X_{j+1}|$。

数据范围：$2 \leq N \leq 10^{15}, 1 \leq Q \leq 10^4, \max\left(\frac{a_i}{b_i}, \frac{b_i}{a_i}\right) \leq 2$

ARC159E Difference Sum Query

你模拟以下题目中的过程，会发现这就是一个类似于二分查找的东西，我们给他建立二分查找树，由于题目的性质，这个树的深度为 $O(\log N)$ 的。

现在考虑 $ans = \sum_{j=c_i}^{d_i-1} |X_j - X_{j+1}|$，由于两个编号相邻的点 $i, i+1$ 一定有祖先后代关系。所以这一坨就是树上一堆连续点的距离和。

而一堆连续点的距离和这个问题有没有很熟悉，实际上 $ans + dis(c_i, d_i)$ 就是这些点构成的虚树的边 $\times 2$ 。

这个虚树上的点还有哪些？怎么求 $dis(c_i, d_i)$ ？这些都可以暴力求解。

2021.9.13 T3 没有上司的涨薪舞会

有一课树。每个点加入集合 T 的概率按以下方式计算（b 已知）。

$$p(u)=\begin{cases}0 & \forall v\in u.son\&u\in T\\ b[u] & other\end{cases}$$

每个加入集合 T 的点 u 会产生贡献，具体为

$$\sum_{u\in T}c[u]\sum_{v\in T\&v\in subtree(u)}s[v]$$

你要确定 01 序列 S 的值，使上面的式子的期望最大。

数据范围:$n\le 5\times 10^5$

2021.9.13 T3 没有上司的涨薪舞会

考虑期望的线性性，我们实际上可以对每个点的 s_i 单独判断，看看 $s_i = 1$ 时带来的贡献是否 > 0。

所以暴力就是枚举 i，把 i 放入 T 中，看 i 的所有祖先的 $\sum p(j) \times c_j$ 是多少。

考虑优化，注意到我们每次重新求一次都只更改了一个点的转移系数，仿照动态 dp，我们可以将计算 $\sum p_j \cdot c_j$ 的转移用矩阵来维护，利用矩阵的结合律，用栈维护根到父亲的转移的矩阵积，这样我们对每一个点快速得到答案。

P6773 [NOI2020] 命运

给定一棵树，你需要给树上每条边确定染色。

同时给定 m 个限制，每条限制形如 u 到 v 路径上的边中存在一个黑色边，保证 u 是 v 的祖先。

问所有染色方案数模 998244353 。

数据范围： $n, m \leq 5 \times 10^5$ 。

P6773 [NOI2020] 命运

设 dp，设 $f_{i,j}$ 表示考虑以 i 为根的子树，其中未满足的限制中 u 深度最大的为 j。这样我们把每个 $u \to v$ 的限制存在 v 上，我们只保留深度最大的（深度大的满足了，那么深度小的一定满足）。

考虑转移，有：

- 若 (x, y) 染成黑色，那么 y 子树内的限制一定都能满足，除了 $f_{y,j}$ ，其中 $j > de_x$ ，那么这些就在 y 这就不能满足了。
  即 $f_{x,i} \leftarrow f_{x,i} \sum_{j \le de_x} f_{y,j}$ 。
- 若 (x, y) 染成白色，那么相当于 y 子树内的是咋样还是咋样，即 $f_{x,i} \leftarrow f_{x,i} \sum_{j \le i} f_{y,j} + f_{y,i} \sum_{j < i} f_{x,j}$ 。

P6773 [NOI2020] 命运

现在就有 n^2 的 DP 了，即

$$f_{x,i} \leftarrow f_{x,i}\left(\sum_{j\le de_x} f_{y,j}+\sum_{j\le i} f_{y,j}\right)+f_{y,i}\sum_{j<i} f_{x,j}$$

考虑优化，我们给每个节点 x 开一个线段树，用来维护 $f_{x,i}$ 。

虽然这个 DP 式子看起来很复杂，但是我们发现只有 $\sum_{j\le de_x} f_{y,j}$ 不和 i 相关，而这个就是一个前缀和，可以在线段树上直接查询。

而后面的每一个都是一个前缀和的形式，我们可以在线段树合并的时候记录一下两棵树的前缀和，同时更新。

hdu7219 Rock Tree

给定一颗带点权的树，求点权和最大的且直径不超过 k 的联通块权值和。

数据范围：$k \leq n \leq 10^5$ 。

hdu7219 Rock Tree

首先有 dp，设 $f_{x,i}$ 表示连通块最高点是 x，此时最深的点到 x 的距离不超过 i 的最大权值和，有转移：

- $i+i\le k$，此时 $f_{x,i}=f'_{x,i}+f_{y,i-1}$。
- $i+i>k$，此时 $f_{x,i}=\max(f'_{x,i}+f_{y,k-i-1},f'_{x,k-i}+f_{y,i-1})$。

现在要考虑优化这个转移。

注意到 $f_{x,i}$ 是关于深度的，所以可以长链剖分优化。但这貌似解决不了第二种转移，准确的说，是只有第一部分的形式是可以直接转移的。

hdu7219 Rock Tree

注意我们是如何保证长链剖分的复杂度的，是在把 y 合并到 x 时，使用了一个关于 $O(len_y)$ 的算法。

那么我们就可以暴力处理每一个 $f_{y,i}$：

- 若 $2(i+1) \le k$，那么就给 $f_{x,i+1\sim k} \leftarrow f_{x,i+1\sim k} + f_{y,i}$
- 若 $2(i+1) > k$，那么就 $f_{x,i+1} \leftarrow \max(f_{x,i+1}, f_{y,i} + f_{x,k-i-1})$

注意当计算上 val_x 之后，会让 f_x 的一段前缀 < 0 ，这时候我们就需要把这些部分变成 0 。

所以我们需要一个维护区间加，单点修改，区间赋值的东西，那么线段树就是可以的。整体的复杂度是 $O(n \log n)$ 的。

The End