分治乱讲

qwq123

CJ 2020 信息组

2024 年 1 月 29 日

前言

分治处理算是一个很重要的思想吧，

其实应该放在数据结构里的，不过分治也不算是一种数据结构，只能说是一种运用数据结构的思想，所以单独拿出来也是可以的。

今天的任务

1. 线段树分治
2. cdq 分治
3. 整体二分
4. wqs 二分
5. 题目

线段树分治

适用问题：考虑某种奇怪的动态问题，我们有查询 Q，加入元素 I，和删除元素 D 三种操作。但是，如果没有 D 操作，取而代之的是撤回操作，我们就会做。

建一棵以询问编号为下标的线段树，每个叶子节点上挂上对应的询问，然后将修改挂到会影响到的询问区间上（类似于线段树区间修改）。

然后在线段树上递归，每次把挂在这个节点上的所有修改执行，这样子递归到叶子节点时就执行了所有会影响到这个询问的修改。记得要在返回上一层时撤销当前节点的修改。

线段树分治

P5787 二分图 / 【模板】线段树分治

给出一个无向图，每条边都有一个存在时间区间，总时间为 $[1,r]$，询问每个时刻该图是不是二分图。

线段树分治

P5787 二分图 / 【模板】线段树分治

给出一个无向图，每条边都有一个存在时间区间，总时间为 [1,r]，询问每个时刻该图是不是二分图。

题解

将边的出现区间对应到线段树区间上，现在想办法维护二分图：

线段树分治

P5787 二分图 / 【模板】线段树分治

给出一个无向图，每条边都有一个存在时间区间，总时间为 [1,r]，询问每个时刻该图是不是二分图。

题解

将边的出现区间对应到线段树区间上，现在想办法维护二分图：

可以像食物链一样维护和 x 同边和异边，若在合并的时候 x 和 y 在一个联通块里了，那么就不是，否则连上 $x+n \leftrightarrow y$ 和 $x \leftrightarrow y+n$。代码

线段树分治

P5787 二分图 / 【模板】线段树分治

给出一个无向图，每条边都有一个存在时间区间，总时间为 [1,r]，询问每个时刻该图是不是二分图。

题解

将边的出现区间对应到线段树区间上，现在想办法维护二分图：

可以像食物链一样维护和 x 同边和异边，若在合并的时候 x 和 y 在一个联通块里了，那么就不是，否则连上 $x+n \leftrightarrow y$ 和 $x \leftrightarrow y+n$。代码

同时也可以通过判断没有奇环来解决。通过带权并查集在每次加入边的时候，如果在联通块内部，查看两个点到根路径长度的奇偶性情况，如果相同，加上新边则形成奇环。如果不在同一联通块，那么计算出两个根之间的路径奇偶性，赋权并合并。

线段树分治

P3247 [HNOI2016] 最小公倍数

给定一张无向图，每条边上有权值，都可以分解成 $2^a \times 3^b$。

现在有 q 个询问，请你求出是否存在一条顶点 u 到 v 之间的路径(可以不是简单路径)，使路径经过的边上的权值的最小公倍数为 $2^a \times 3^b$。

线段树分治

P3247 [HNOI2016] 最小公倍数

给定一张无向图，每条边上有权值，都可以分解成 $2^a \times 3^b$。

现在有 q 个询问，请你求出是否存在一条顶点 u 到 v 之间的路径 (可以不是简单路径)，使路径经过的边上的权值的最小公倍数为 $2^a \times 3^b$。

题解

对于一个询问，考虑满足条件的情况为：在只保留 $a_i \le a \wedge b_i \le b$ 的边的情况下，u 与 v 联通，且该联通块内存在 $a_i = a$ 和 $b_i = b$ 的边。

那么我们线段树分治，但似乎还是无法处理二维偏序的情况，那么我们不用线段树，而用 kdt！

建立询问的 kdt，对于一条边会贡献的区间是 $\sqrt{m}$ 个，复杂度就是 $n\sqrt{m}\log n$。

线段树分治

CF576E Painting Edges

一个无向图，共有 k($k \leq 50$) 种颜色，初始每条边都没有颜色。定义合法状态为仅保留染成 k 种颜色中的任何一种颜色的边，图都是一张二分图。

有 q 次操作，第 i 次操作将第 e_i 条边的颜色染成 c_i。只有当执行后仍然合法，才会执行本次操作。你需要判断每次操作是否会被执行。

线段树分治

CF576E Painting Edges

一个无向图，共有 k($k \le 50$) 种颜色，初始每条边都没有颜色。定义合法状态为仅保留染成 k 种颜色中的任何一种颜色的边，图都是一张二分图。

有 q 次操作，第 i 次操作将第 e_i 条边的颜色染成 c_i。只有当执行后仍然合法，才会执行本次操作。你需要判断每次操作是否会被执行。

题解

k 种颜色那么就开 k 个并查集，难点是如何处理只有当执行后仍然合法，才会执行本次操作？

线段树分治

CF576E Painting Edges

一个无向图，共有 k($k \leq 50$) 种颜色，初始每条边都没有颜色。定义合法状态为仅保留染成 k 种颜色中的任何一种颜色的边，图都是一张二分图。

有 q 次操作，第 i 次操作将第 e_i 条边的颜色染成 c_i。只有当执行后仍然合法，才会执行本次操作。你需要判断每次操作是否会被执行。

题解

k 种颜色那么就开 k 个并查集，难点是如何处理只有当执行后仍然合法，才会执行本次操作？

考虑在遍历的时候在什么时候可以判断合法？合法之后会产生的修改是什么？

线段树分治

CF576E Painting Edges

一个无向图，共有 k($k \le 50$) 种颜色，初始每条边都没有颜色。定义合法状态为仅保留染成 k 种颜色中的任何一种颜色的边，图都是一张二分图。

有 q 次操作，第 i 次操作将第 e_i 条边的颜色染成 c_i。只有当执行后仍然合法，才会执行本次操作。你需要判断每次操作是否会被执行。

题解

k 种颜色那么就开 k 个并查集，难点是如何处理只有当执行后仍然合法，才会执行本次操作？

考虑在遍历的时候在什么时候可以判断合法？合法之后会产生的修改是什么？

是不是修改和遍历并不冲突，那么就可以做。

线段树分治

CF603E Pastoral Oddities

给定一张 n 个点的无向图，初始没有边。

依次加入 m 条带权的边，每次加入后询问是否存在一个边集，满足每个点的度数均为奇数，若存在，则还需要最小化边集中的最大边权。

线段树分治

CF603E Pastoral Oddities

给定一张 n 个点的无向图，初始没有边。

依次加入 m 条带权的边，每次加入后询问是否存在一个边集，满足每个点的度数均为奇数，若存在，则还需要最小化边集中的最大边权。

题解

首先有一个结论：某个图合法当且仅当所有连通块的大小都是偶数。而且两个块合并，一定不会让奇数的块增多，所以可以不断加边（尽管我们最后可能不会去选他们作为边集）

线段树分治

CF603E Pastoral Oddities

给定一张 n 个点的无向图，初始没有边。
依次加入 m 条带权的边，每次加入后询问是否存在一个边集，满足每个点的度数均为奇数，若存在，则还需要最小化边集中的最大边权。

题解

首先有一个结论：某个图合法当且仅当所有连通块的大小都是偶数。而且两个块合并，一定不会让奇数的块增多，所以可以不断加边（尽管我们最后可能不会去选他们作为边集）

所以暴力可以这么做：每一次询问把当前边按边权小到大排序，不断加边，直到满足条件。

线段树分治

CF603E Pastoral Oddities

给定一张 n 个点的无向图，初始没有边。

依次加入 m 条带权的边，每次加入后询问是否存在一个边集，满足每个点的度数均为奇数，若存在，则还需要最小化边集中的最大边权。

题解

首先有一个结论：某个图合法当且仅当所有连通块的大小都是偶数。而且两个块合并，一定不会让奇数的块增多，所以可以不断加边（尽管我们最后可能不会去选他们作为边集）

所以暴力可以这么做：每一次询问把当前边按边权小到大排序，不断加边，直到满足条件。

现在我们要动态加边，我们想象一条边什么时候会在决策集合里面：加入的时候很优，进入了决策集合，后来随着新边加入被抛弃。

线段树分治

题解

据此发现，每条边在决策集合中的时间是一个区间，但我们只知道左端点（加入的时间），怎么求右端点？

感觉还是好复杂，不如化简一下：如何求出现时间右端点为 m 的边？

线段树分治

题解

据此发现，每条边在决策集合中的时间是一个区间，但我们只知道左端点（加入的时间），怎么求右端点？

感觉还是好复杂，不如化简一下：如何求出现时间右端点为 m 的边？

只需要按上面排序的方法，到满足条件之前加入的边我们都可以说他们的出现时间右端点是 m。

线段树分治

题解

据此发现，每条边在决策集合中的时间是一个区间，但我们只知道左端点（加入的时间），怎么求右端点？

感觉还是好复杂，不如化简一下：如何求出现时间右端点为 m 的边？

只需要按上面排序的方法，到满足条件之前加入的边我们都可以说他们的出现时间右端点是 m。

那么我们逆着来考虑，随着时间到了某个边出现时间的左端点之前，还有哪些边可以作为补救的选项？

线段树分治

题解

据此发现，每条边在决策集合中的时间是一个区间，但我们只知道左端点（加入的时间），怎么求右端点？

感觉还是好复杂，不如化简一下：如何求出现时间右端点为 m 的边？

只需要按上面排序的方法，到满足条件之前加入的边我们都可以说他们的出现时间右端点是 m。

那么我们逆着来考虑，随着时间到了某个边出现时间的左端点之前，还有哪些边可以作为补救的选项？

一直这么进行下去，我们好像就求出了每条边的出现区间，那么我们就可以在求出的时候插入，代码

cdq 分治

cdq 分治是将动态（在线）的问题转化为静态（离线）的问题。

怎么理解这句话？

cdq 分治

例子

有一个数组，数轴上有一些点，多次询问，每次询问给定一个区间，求区间所包含的点数。

cdq 分治

例子

有一个数组，数轴上有一些点，多次询问，每次询问给定一个区间，求区间所包含的点数。

如果提前告诉你所有的点，然后再去询问。这时候你知道，所有的点**都可能**会在区间中，就可以将点按坐标排序，在区间左端点后、右端点前的就在区间内。

cdq 分治

例子

有一个数组，数轴上有一些点，多次询问，每次询问给定一个区间，求区间所包含的点数。

如果提前告诉你所有的点，然后再去询问。这时候你知道，所有的点**都可能**会在区间中，就可以将点按坐标排序，在区间左端点后、右端点前的就在区间内。

但如果在询问之后又加了一些点，并且后面还有询问... 这时候上面的办法就没用了，因为在询问后加入的点**不可能**前面的询问矩形内。这时候你按坐标排序，在区间左端点后、右端点前的点就**不一定**在区间内。

cdq 分治

上面这两种情况就分别对应了静态和动态

你也可以理解为动态情况多了**时间轴的偏序关系**，即只有时间前的才能贡献时间后的，所以 cdq 分治也是一种消除偏序关系的方法。

cdq 分治

算法流程：

套用上面的问题，有一个时间先后的操作序列，C 为加点，Q 为查询。如 $\{C\ \ C\ \ Q\ \ C\ \ Q\ \ C\ \ Q\ \ C\}$。

cdq 分治

算法流程：

套用上面的问题，有一个时间先后的操作序列，C 为加点，Q 为查询。如 $\{C\ \ C\ \ Q\ \ C\ \ Q\ \ C\ \ Q\ \ C\}$。

考虑从正中间划分 $\{C\ \ C\ \ Q\ \ C \mid Q\ \ C\ \ Q\ \ C\}$

cdq 分治

算法流程:

套用上面的问题，有一个时间先后的操作序列，C 为加点，Q 为查询。如 $\{\mathrm{C\ \ C\ \ Q\ \ C\ \ Q\ \ C\ \ Q\ \ C}\}$。

考虑从正中间划分 $\{\mathrm{C\ \ C\ \ Q\ \ C} \mid \mathrm{Q\ \ C\ \ Q\ \ C}\}$

这时候你发现，在分割线左边的加点操作一定先与右边的查询操作 $\{\mathrm{C\ \ C\ \ .\ \ C} \mid \mathrm{Q\ \ .\ \ Q\ \ .}\}$。

cdq 分治

算法流程：

套用上面的问题，有一个时间先后的操作序列，C 为加点，Q 为查询。如 {C C Q C Q C Q C}。

考虑从正中间划分 {C C Q C | Q C Q C}

这时候你发现，在分割线左边的加点操作一定先与右边的查询操作 {C C . C | Q . Q .}。

那么这就转化为了静态的问题，在给左边的 C 和右边的 Q 打上标记之后，就可以按照上面引入部分的方法解决了（处理的时候我们只处理带标记的加点和查询）。

cdq 分治

算法流程：

处理完之后再去递归处理分割线左边的 {C C Q C} 和右边的 {Q C Q C}。

当然，你可能会说排序之后分割线左右的已经不是原来那样了，没有关系，我们先递归再处理也是可以的。

cdq 分治

算法流程：

处理完之后再去递归处理分割线左边的 $\{\mathsf{C}\quad\mathsf{C}\quad\mathsf{Q}\quad\mathsf{C}\}$ 和右边的 $\{\mathsf{Q}\quad\mathsf{C}\quad\mathsf{Q}\quad\mathsf{C}\}$。

当然，你可能会说排序之后分割线左右的已经不是原来那样了，没有关系，我们先递归再处理也是可以的。

复杂度：

cdq 的分治树是 $O(\log n)$ 层的，所以复杂度基本上就是内层操作加上一个 $\log$。

cdq 分治

上面所介绍的是最基本的 cdq 处理问题方法，其实还有很多情况是 cdq 可以处理的，下面是一些 cdq 的技巧（或许是）

cdq 分治

- 偏序降维 $\to$ 时间轴:
  我们可以将一个静态的问题，将某一偏序关系排序之后，就变成了”**只有前面才会贡献后面**”的动态问题。

cdq 分治

- 偏序降维 $\rightarrow$ 时间轴:
  我们可以将一个静态的问题，将某一偏序关系排序之后，就变成了”**只有前面才会贡献后面**”的动态问题。
- 查询与修改的相互转化:
  题目中，我们没有修改，而查询满足条件的点对数。看似与上面我们介绍的修改和查询分开不同，但实际我们在处理的时候，就可以把分割线左边的点当做修改，右边的点当做查询。这样在查询到的点全是左边的点，满足题面的要求。

cdq 分治

P3810 【模板】三维偏序

有 n 个元素，第 i 个元素有 a_i, b_i, c_i 三个属性，设 f(i) 表示满足 $a_j \le a_i$ 且 $b_j \le b_i$ 且 $c_j \le c_i$ 且 $j \ne i$ 的 j 的数量。

cdq 分治

P3810 【模板】三维偏序

有 n 个元素，第 i 个元素有 a_i, b_i, c_i 三个属性，设 f(i) 表示满足 $a_j \le a_i$ 且 $b_j \le b_i$ 且 $c_j \le c_i$ 且 $j \ne i$ 的 j 的数量。

题解

不管之前是怎么写的，但我推荐按上面所讲的思路来再写一遍，因为这样能更好理解这种思路，而且之后的题都要依靠这种思路。

代码
归并排序版

cdq 分治

【瞎搞】四维偏序

有 n 个元素，第 i 个元素有 a_i, b_i, c_i, d_i 四个属性，设 f(i) 表示满足 $a_j \le a_i$ 且 $b_j \le b_i$ 且 $c_j \le c_i$ 且 $d_i \le d_j$ 且 $j \ne i$ 的 j 的数量。

cdq 分治

【瞎搞】四维偏序

有 n 个元素，第 i 个元素有 a_i, b_i, c_i, d_i 四个属性，设 f(i) 表示满足 $a_j \le a_i$ 且 $b_j \le b_i$ 且 $c_j \le c_i$ 且 $d_i \le d_j$ 且 $j \ne i$ 的 j 的数量。

题解

cdq 嵌套！

在消除第一维偏序之后，其实还可以继续 cdq 消除第二维偏序，此时与三维偏序不同的是这里我们已经明确了修改和查询。

代码

cdq 分治

P3769 [CH 弱省胡策 R2]TATT

有 n 个元素，第 i 个元素有 a_i, b_i, c_i, d_i 四个属性，$dp_i = \max\{a_i \geq a_j \wedge b_i \geq b_j \wedge c_i \geq c_j \wedge d_i \geq d_j |\ dp_j\} + 1$ 求 $\max dp_i$

cdq 分治

P3769 [CH 弱省胡策 R2]TATT

有 n 个元素，第 i 个元素有 a_i, b_i, c_i, d_i 四个属性，$dp_i = \max\{a_i \geq a_j \wedge b_i \geq b_j \wedge c_i \geq c_j \wedge d_i \geq d_j |\ dp_j\} + 1$ 求 $\max dp_i$

题解

同样，先排序消除一维偏序关系，转化为动态问题 cdq 分治后，发现这个 dp 转移好像又有所不同，其实我们可以将左边点当做修改（加入一个值为 dp_i 的点），右边的查询满足条件的最大值。

注意：由于一个点在转移时必须先知道其 dp 值，所以遍历分治树时必须按中序遍历，并且为了避免破坏原来的顺序，需要额外开一个数组去处理。代码

cdq 分治

关于 cdq 与树套树和 kdt 的比较：

- cdq 常数吊打另外两个！
- cdq 的处理方法类似树套树，在消除偏序关系的时候分割了修改，所以无法处理如区间赋值，区间加的操作。
- cdq 必须离线。

cdq 分治

P4169 [Violet] 天使玩偶/SJY 摆棋子

维护 n 个二维坐标点，要求支持以下操作：

- 加入一个点
- 查询这些点中离指定点的曼哈顿距离的最小值。

cdq 分治

P4169 [Violet] 天使玩偶/SJY 摆棋子

维护 n 个二维坐标点，要求支持以下操作：

- 加入一个点
- 查询这些点中离指定点的曼哈顿距离的最小值。

题解

若 a_i 和 a_j 的大小确定了，那么 $|a_i - a_j|$ 的正负也确定了。

所以我们可以先处理有 $a_i \leq a_j$ 关系的，再处理有 $a_i > a_j$ 关系的。

二维也是如此，为了方便，可以通过翻转坐标轴达到只用一个排序函数解决。

这题好像也可以用玄学 kdt 做，但好像也有 hack 数据。

cdq 分治

CF1045G AI robots

有 N 个机器人排成一行，第 i 个机器人的位置为 x_i 视野为 r_i，智商为 q_i。我们认为第 i 个机器人可以看到的位置是 $[x_i - r_i, x_i + r_i]$。如果一对机器人相互可以看到，且它们的智商 q_i 的差距不大于 K，那么它们会开始聊天，请计算有多少对机器人可能会聊天。

cdq 分治

CF1045G AI robots

有 N 个机器人排成一行，第 i 个机器人的位置为 x_i 视野为 r_i，智商为 q_i。我们认为第 i 个机器人可以看到的位置是 $[x_i - r_i, x_i + r_i]$。如果一对机器人相互可以看到，且它们的智商 q_i 的差距不大于 K，那么它们会开始聊天，请计算有多少对机器人可能会聊天。

题解

发现就是求 $(x_i - x_j) \le \min(r_i, r_j)$ 且 $|q_i - q_j| \le K$ 的对数。

cdq 分治

CF1045G AI robots

有 N 个机器人排成一行，第 i 个机器人的位置为 x_i 视野为 r_i，智商为 q_i。我们认为第 i 个机器人可以看到的位置是 $[x_i - r_i, x_i + r_i]$。如果一对机器人相互可以看到，且它们的智商 q_i 的差距不大于 K，那么它们会开始聊天，请计算有多少对机器人可能会聊天。

题解

发现就是求 $(x_i - x_j) \le \min(r_i, r_j)$ 且 $|q_i - q_j| \le K$ 的对数。

发现 min 不好处理怎么办？我们可以以 i 去考虑所有 $r_i < r_j$ 的 j，这样就可以将 min 去掉。

整体二分

整体二分是用来处理 n 个决策单调的问题，直接干讲不太好讲，我们就一结合着题目来讲。

查询区间第 k 小

给定 $a[1...n]$，q 次询问，每次询问查询 $a[l...r]$ 中的第 k 小的数。

考虑处理单次查询，可以二分答案 mid ，然后将 $a_l \sim a_r$ 中 $\le mid$ 的 i 统计起来，看个数是否 $\ge k$ 还是 $< k$ 。

或许你会想单次处理就是 $O(n \log n)$ 了，多次询问不会复杂度爆炸？

整体二分

那是因为，我们可以用数据结构维护多次 $+1$ 操作和查询。

比如在此题中，我们可以把 $\leq mid$ 的每一个 a_i 看成 1 ，然后对每次询问统计 $[l,r]$ 内有多少个 1 ，这个可以通过树状数组优化到 $O(\log n)$ 单次处理，所以我们如果对于一个分治层的操作次数为 $O(n)$ 的话，那整个复杂度就是 $O(n\log^2 n)$ 了。

整体二分

P2617 Dynamic Rankings

给定一个含有 n 个数的序列 $a_1, a_2 \dots a_n$，需要支持两种操作：

- Q l r k 表示查询下标在区间 $[l, r]$ 中的第 k 小的数
- C x y 表示将 a_x 改为 y

整体二分

P2617 Dynamic Rankings

给定一个含有 n 个数的序列 $a_1, a_2 \dots a_n$，需要支持两种操作：

- Q l r k 表示查询下标在区间 $[l, r]$ 中的第 k 小的数
- C x y 表示将 a_x 改为 y

题解

考虑加入的修改操作有什么影响？

整体二分

P2617 Dynamic Rankings

给定一个含有 n 个数的序列 $a_1, a_2 \ldots a_n$，需要支持两种操作：

- Q l r k 表示查询下标在区间 $[l, r]$ 中的第 k 小的数
- C x y 表示将 a_x 改为 y

题解

考虑加入的修改操作有什么影响？

对于 C x y ，实际上就是多了一个 $(a_x, -1)$ 和 $(y, 1)$ 的操作。

而且因为我们整体二分的处理是按照时间顺序来处理的，所以这种随时间偏序的关系我们是可以处理的。

整体二分

Gym104651 L. Partially Free Meal

有 n 个物品，属性为 a_i, b_i ，对于每一个 $k, 1 \le k \le n$ ，你要在 n 个物品中选出 k 个组成 S，使 $\sum_{x \in S} a_x + \max_{x \in S} b_x$ 最小，求出每个最小值。$n \le 2 \times 10^5$ 。

整体二分

Gym104651 L. Partially Free Meal

有 n 个物品，属性为 a_i, b_i ，对于每一个 $k, 1 \le k \le n$ ，你要在 n 个物品中选出 k 个组成 S，使 $\sum_{x \in S} a_x + \max_{x \in S} b_x$ 最小，求出每个最小值。$n \le 2 \times 10^5$ 。

题解

首先考虑如何计算固定的 k 的答案。将所有物品按照 b 从小到大排序，枚举第 $x(k \le x \le n)$ 个物品作为选中的 b 最大的物品，那么剩下的 $k-1$ 个物品显然是贪心选择前 $x-1$ 个物品中 a 最小的 $k-1$ 个。可以通过可持久线段树在 $O(\log n)$ 的时间内求出对应方案的值。记这个最优的 x 为 $f(k)$。

整体二分

Gym104651 L. Partially Free Meal

有 n 个物品，属性为 a_i, b_i ，对于每一个 $k, 1 \le k \le n$ ，你要在 n 个物品中选出 k 个组成 S，使 $\sum_{x\in S} a_x + \max_{x\in S} b_x$ 最小，求出每个最小值。$n \le 2\times 10^5$ 。

题解

首先考虑如何计算固定的 k 的答案。将所有物品按照 b 从小到大排序，枚举第 $x(k \le x \le n)$ 个物品作为选中的 b 最大的物品，那么剩下的 $k-1$ 个物品显然是贪心选择前 $x-1$ 个物品中 a 最小的 $k-1$ 个。可以通过可持久线段树在 $O(\log n)$ 的时间内求出对应方案的值。记这个最优的 x 为 $f(k)$。

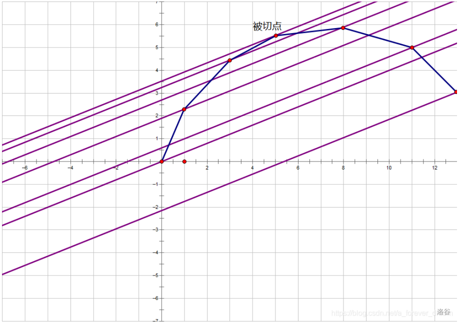
随着 k 增大，容易发现 $f(k)$ 会不断增大, 这是因为，对于 $x_1, x_2(x_1 > x_2)$ ，随着 k 增大，x_1 的可选择范围严格包含了 x_2 的可选择范围，因此 x_2 新选的 a 值一定不大于 x_1 所选的。因此最优决策具有单调性，可以分治求解。

wqs 二分

快速解决“有若干个物品，要求你选出 m 个，选的时候带有限制，要你求出最优的方案”。

用的时候有一个大前提，就是，设 $g(i)$ 表示选 i 个物品的最优方案，那么将所有点 $(i, g(i))$ 画出来，他们一定要组成一个凸包（上凸下凸皆可），这样就有一个性质：斜率单调递增或递减。

wqs 二分



通过二分斜率 k, 可以求出该斜率切于凸包的点，因为该斜率在切于凸包时一定会截距最大，即 $\max\{g(i) - ik\}$，只需在每次计算选一个物品的时候减去 k，这样 DP 所求的就是最大值就是截距最大值。

此时我们求出当前 $g(i)$ 选了多少个物品，从而二分找到 k 切凸包于 $(m, g(m))$ 的时候就能找到所求答案了。

wqs 二分

[ABC218H] Red and Blue Lamps

有 N 盏灯排成一列。你想让其中的 R 盏灯变成红色，$N-R$ 盏灯变成蓝色。当第 i 和 $i+1$ 盏灯以不同的颜色发光时，你将获得 A_i 点报酬。求你能获得的报酬和的最大值。$n \leq 2 \times 10^5$。

wqs 二分

[ABC218H] Red and Blue Lamps

有 N 盏灯排成一列。你想让其中的 R 盏灯变成红色，$N-R$ 盏灯变成蓝色。当第 i 和 $i+1$ 盏灯以不同的颜色发光时，你将获得 A_i 点报酬。求你能获得的报酬和的最大值。$n \leq 2 \times 10^5$。

题解

首先要注意答案随 R 是上凸的，所以可 wqs 二分。

wqs 二分

[ABC218H] Red and Blue Lamps

有 N 盏灯排成一列。你想让其中的 R 盏灯变成红色，$N-R$ 盏灯变成蓝色。当第 i 和 $i+1$ 盏灯以不同的颜色发光时，你将获得 A_i 点报酬。求你能获得的报酬和的最大值。$n \le 2 \times 10^5$。

题解

首先要注意答案随 R 是上凸的，所以可 wqs 二分。

只不过在使用 wqs 二分处理的时候会遇到几个相邻的点斜率相同的情况，需要根据你的写法特殊处理一下，见板书，也可以见代码

wqs 二分

[ARC168E] Subsegments with Large Sums

给定长度为 n 的数列 $\{a_i\}$ 和两个参数 k, s，将 $\{a_i\}$ 划分成 k 段，最大化和 $\geq s$ 的段数。$1 \leq k \leq n \leq 250000$。

wqs 二分

[ARC168E] Subsegments with Large Sums

给定长度为 n 的数列 $\{a_i\}$ 和两个参数 k, s，将 $\{a_i\}$ 划分成 k 段，最大化和 $\geq s$ 的段数。$1 \leq k \leq n \leq 250000$。

题解

首先注意到如果当前划分的一段 $sum < s$ ，那么这种段的长度肯定是 1 。

wqs 二分

[ARC168E] Subsegments with Large Sums

给定长度为 n 的数列 $\{a_i\}$ 和两个参数 k, s，将 $\{a_i\}$ 划分成 k 段，最大化和 $\geq s$ 的段数。$1 \leq k \leq n \leq 250000$。

题解

首先注意到如果当前划分的一段 $sum < s$ ，那么这种段的长度肯定是 1 。

那么我们就只考虑 $sum \geq s$ 的段，我们就可以把原问题转化为：我要在原序列中选择 m 个区间（区间和划分段是等价的），满足以下条件，问 m 最大是多少。

- 每一个区间的 $sum \geq s$
- 选出的区间不能有交
- 记一个选出来的区间的代价为 $len - 1$ ，那么代价和要小于 $n - k$

[ARC168E] Subsegments with Large Sums

注意随着你选的区间增加，代价肯定是越来越大的，也就是记 $f(x)$ 表示选 x 个区间的最小代价，$f(x)$ 是上凹的。（证明可以看官方题解）

[ARC168E] Subsegments with Large Sums

注意随着你选的区间增加，代价肯定是越来越大的，也就是记 $f(x)$ 表示选 x 个区间的最小代价，$f(x)$ 是上凹的。（证明可以看官方题解）

所以可以用 wqs 二分！但我们一般用 wqs 二分是来找到 $(x, f(x))$ 的值，这里是要找到 $\max(x|f(x) \leq n-k)$ ，可以用吗？

[ARC168E] Subsegments with Large Sums

注意随着你选的区间增加，代价肯定是越来越大的，也就是记 $f(x)$ 表示选 x 个区间的最小代价，$f(x)$ 是上凹的。（证明可以看官方题解）

所以可以用 wqs 二分！但我们一般用 wqs 二分是来找到 $(x, f(x))$ 的值，这里是要找到 $\max(x|f(x) \leq n-k)$ ，可以用吗？

当然可以，我们只要二分斜率 mid ，找到切点，看切点的 $f(x)$ 大于还是小于 $n-k$ ，若大于，则 mid 太大了，切到的点在我们想要的 x 点的右边了，反之亦然。这样复杂度就是 $O(n \log a_i)$。

和上题一样，处理斜率相等的问题上要处理一下。

题目选讲

题目

[ARC127D] Sum of Min of Xor

给定序列 A 和序列 B，求 $\sum_{1\le i<j\le N}\min(A_i\oplus A_j, B_i\oplus B_j)$，$\oplus$ 为异或。$1\le n\le 250000, a_i\le 2^{18}$。

题目

[ARC127D] Sum of Min of Xor

给定序列 A 和序列 B，求 $\sum_{1\le i<j\le N}\min(A_i\oplus A_j, B_i\oplus B_j)$，$\oplus$ 为异或。$1\le n\le 250000, a_i\le 2^{18}$。

题解

先考虑怎么求 $\sum_{1\le i<j\le N} A_i\oplus A_j$

题目

[ARC127D] Sum of Min of Xor

给定序列 A 和序列 B，求 $\sum_{1\le i<j\le N}\min(A_i\oplus A_j, B_i\oplus B_j)$，$\oplus$ 为异或。$1\le n\le 250000, a_i\le 2^{18}$。

题解

先考虑怎么求 $\sum_{1\le i<j\le N} A_i\oplus A_j$

如何去除 min 呢？考虑比较两个二进制数，肯定是前面几位相等，之后有一位不相等。

此时我们设 $C_i=A_i\oplus B_i$，若 $A_i\oplus A_j$ 与 $B_i\oplus B_j$ 在第 k 位不同，那么 $C_i\oplus C_j$ 第 k 位就为 1，此时我们就可以通过分析 A_i, B_i, A_j, B_j 是 0 还是 1 来看 $\min(A_i\oplus A_j, B_i\oplus B_j)$ 是 $A_i\oplus A_j$ 还是 $B_i\oplus B_j$。

[ARC127D] Sum of Min of Xor

我们设 $C_i=0,C_j=1$ ，那么有以下四种情况。

- A_i,B_i 这一位是 $0,0$， A_j,B_j 这一位是 $0,1$， 产生贡献的是 $A_i\oplus A_j$。
- A_i,B_i 这一位是 $0,0$， A_j,B_j 这一位是 $1,0$， 产生贡献的是 $B_i\oplus B_j$。
- A_i,B_i 这一位是 $1,1$， A_j,B_j 这一位是 $1,0$， 产生贡献的是 $A_i\oplus A_j$。
- A_i,B_i 这一位是 $1,1$， A_j,B_j 这一位是 $0,1$， 产生贡献的是 $B_i\oplus B_j$。

[ARC127D] Sum of Min of Xor

我们设 $C_i=0, C_j=1$ ，那么有以下四种情况。

- A_i, B_i 这一位是 $0,0$，A_j, B_j 这一位是 $0,1$，产生贡献的是 $A_i \oplus A_j$。
- A_i, B_i 这一位是 $0,0$，A_j, B_j 这一位是 $1,0$，产生贡献的是 $B_i \oplus B_j$。
- A_i, B_i 这一位是 $1,1$，A_j, B_j 这一位是 $1,0$，产生贡献的是 $A_i \oplus A_j$。
- A_i, B_i 这一位是 $1,1$，A_j, B_j 这一位是 $0,1$，产生贡献的是 $B_i \oplus B_j$。

如何处理？

因为要从大到小去枚举位数，所以可以将序列按 C_i 排序，然后对于位数，求出该位 0 和 1 的分界 x，即 $[l, x-1]$ 的 c_i 是 0，$[x, r]$ 的 c_i 是 1，统计完答案后分治处理。

CF1648D Serious Business

给你 $3\times n$ 的矩阵，i 行 j 列的权值为 $a_{i,j}$ ，你需要从 $(1,1)$ 只向右、向下走到 $(3,n)$ 。

最开始第 2 行所有点都不能走，有 m 个解锁方法，形如 l_i,r_i,z_i ，表示将 $a_{2,l_i}\sim a_{2,r_i}$ 解锁需要花费 z_i 的代价（你可以选多个解锁方法。)。

一条路径的价值为路径中所有点的权值之和 $-$ 解锁花费的代价之和，求权值最大的路径权值大小。

数据范围：$n,m\le 5\times 10^5$ 。

CF1648D Serious Business

一开始肯定会想确定到第 2 行的起点 l ，终点 r ，然后这样一条路径的权值就是 $su_{1,l}+su_{2,r}-su_{2,l-1}+ne_{3,r}-cost(l,r)$ ，其中 $cost(i,j)$ 表示将 $a_{2,l}\sim a_{2,r}$ 都解锁需要花费的代价。

CF1648D Serious Business

一开始肯定会想确定到第 2 行的起点 l ，终点 r ，然后这样一条路径的权值就是 $su_{1,l} + su_{2,r} - su_{2,l-1} + ne_{3,r} - cost(l,r)$ ，其中 $cost(i,j)$ 表示将 $a_{2,l} \sim a_{2,r}$ 都解锁需要花费的代价。

然后可能会考虑分治，对于 mid 考虑所有会经过 mid 的方案，这样选前缀中最大的和后缀中最大的就是答案。

CF1648D Serious Business

一开始肯定会想确定到第 2 行的起点 l ，终点 r ，然后这样一条路径的权值就是 $su_{1,l}+su_{2,r}-su_{2,l-1}+ne_{3,r}-cost(l,r)$ ，其中 $cost(i,j)$ 表示将 $a_{2,l}\sim a_{2,r}$ 都解锁需要花费的代价。

然后可能会考虑分治，对于 mid 考虑所有会经过 mid 的方案，这样选前缀中最大的和后缀中最大的就是答案。

但是因为有 $cost(i,j)$ ，前缀和后缀的答案是无法合并的，所以就 G 了。

CF1648D Serious Business

我们设 f_i 表示走到了 $(2, i)$ 的最大权值，就有转移：

$$f_i = \max_{j<i}\{f_j + su_i - su_j + cost(i+1, j)\}$$

这和上面的有区别吗？

CF1648D Serious Business

我们设 f_i 表示走到了 $(2,i)$ 的最大权值，就有转移：

$$f_i=\max_{j<i}\{f_j+su_i-su_j+cost(i+1,j)\}$$

这和上面的有区别吗？

有！因为我们 DP 是一步一步来的，所以这里的 $cost(i,j)$ 就可以表示为所有解锁区间可以覆盖 $(2,i)\sim(2,j)$ 的代价最小值。

因为此时 $cost(i,j)$ 变成了好处理的东西，所以我们可以考虑 cdq 去优化 dp 的转移。

CF1648D Serious Business

基本的分治，先处理左边不用多说，现在我们假设 $[l, mid]$ 的所有值已经求出来了。

对于一个跨过 mid 的解锁方法 l_i, r_i, z_i，我们可以给满足：$\max(l_i, l) \le i \le mid < j \le \min(r, r_i)$ 的 (i, j) 作上面的转移。

然后就可以给 mid 前的用后缀 $\max$ 得到最大值，转移之后再用向前去更新。

但是这样复杂度还是 nm 的（每个解锁方案会被遍历 $O(len)$ 次）。

CF1648D Serious Business

基本的分治，先处理左边不用多说，现在我们假设 $[l, mid]$ 的所有值已经求出来了。

对于一个跨过 mid 的解锁方法 l_i, r_i, z_i，我们可以给满足：$\max(l_i, l) \le i \le mid < j \le \min(r, r_i)$ 的 (i, j) 作上面的转移。

然后就可以给 mid 前的用后缀 $\max$ 得到最大值，转移之后再用向前去更新。

但是这样复杂度还是 nm 的（每个解锁方案会被遍历 $O(len)$ 次）。

但是，我们可以对当前分治的区间 $[l, r]$ 记 mi 表示会覆盖 $[l, r]$ 的所有解锁方案中费用最小的，这样我们就不用把所有解锁方案都递归给左右儿子中。

分析一下现在的复杂度，发现每一个解锁方案只会被遍历 $O(\log n)$ 次了（可以类比线段树），然后这样复杂度就是正确的了。

P5617 [MtOI2019] 不可视境界线

有 n 个半径为 r 的圆，第 i 个在 $(0, x_i)$，你需要选出 k 个圆，使圆的并面积最大。

数据范围：$n \leq 10^5$。

P5617 [MtOI2019] 不可视境界线

算是一个有点套路的题。

设 $f[i][j]$ 为前 i 个圆选取了 j 个，并且选了第 i 个，所得到的最大并面积。

转移有：$f[i][j] = \max_{k=1}^{i-1} f[k][j-1] + c(k, i)$，其中 $c(i, j)$ 表示第 j 个圆 $-$ 第 i 个圆与第 j 个圆相交的面积。这是可以计算出来的。

然后套路的认为这个是凸的，wqs 二分后可以把第二维去除。

再套路的认为这个 $c(k, i)$ 是随 i 增长而增长得越来越慢的，得到前面的决策会被后面的反超，后面的不会被前面的反超。用一个队列 $+$ 二分去维护就可以了。

CF938G Shortest Path Queries

给出一个连通带权无向图, 边有边权, 要求支持 q 个操作:

$1\ x\ y\ d$ 在原图中加入一条 x 到 y 权值为 d 的边。

$2\ x\ y$ 把图中 x 到 y 的边删掉。

$3\ x\ y$ 表示询问 x 到 y 的异或最短路。

CF938G Shortest Path Queries

可能先要会 P4151 [WC2011] 最大 XOR 和路径，如果你不会，时间也还充裕的话是可以解释一下做法的。

先考虑没有修改的情况，因为异或的自反性，我只要从 $x \to y$，之后 $y \to x$，发现带来的贡献为 0 ，所以我们可以找到一个环，然后找环上的一个点 y ，从当前的点 x 到 y ，之后再环上转一圈，记录上环的贡献，再回到 x 。

所以我们可以找到所有图里面的上的环，加入一个线性基，之后随便找一条 $x \to y$ 的路径就可以了，但是环可以有很多，我们不能一个个找。

那么我们建立一棵生成树，之后对于一条非树边，就对应了一个环。而包含多条非树边的环一定会被多条这样的环给表示出来。

现在考虑带修改的情况，注意线性基加入简单但是删除困难，所以我们可以使用线段树分治，同时用带权并查集维护生成树。

P5163 WD 与地图

给定一个有向图，点初始的点权是 a_i，定义关联集合为一个强连通分量内的所有点，有 q 次操作：

- 删除掉一条边。
- 修改某个点的点权。
- 求与 x 所在的关联集合中，点权最大的 k 个点权值之和。

数据范围：$n \leq 10^5, m, q \leq 2 \times 10^5$ 。

P5163 WD 与地图

考虑如果图是无向的怎么做？

那么我们反这来，删边就变成了加边，对于每个连通块，维护一个线段树，然后每次加边就合并两个连通块。

现在考虑有向。注意到随着加边之后，也是几个关联集合并到了一起，如果我们可以记录在当前时刻，哪几个关联集合会合并，就可以按上面的做法实现了。所以我们实际上是希望找到什么时候两个强连通分量会合并在一起。

于是就有了暴力做法，就是每次加边之后暴力寻找强连通分量。现在考虑如何优化。

P5163 WD 与地图

首先我们要知道，两个关联集合 x, y 会合并，一定是存在一条边 $x \to y$(或 $y \to x$，但肯定会有一条)，并且在这个时刻 x, y 会同属一个关联集合，所以我们就可以对每条边，求出 t_i 表示最大的时间（正着来）使得 x, y 在同一个关联集合。

怎么求？发现是可二分的，但要求多个边的... 那么我们就可以用**整体二分**！

对于当前 (l, r, mid)，我们求出在 mid 这个时刻的关联集合的情况，然后根据定义看把每条边往左还是往右。

但是你会发现这个东西实现不了，因为我们判断一次关联集合的情况就是 $O(m)$ 的，一共要二分 n 次复杂度还是不对。

P5163 WD 与地图

但是，注意到如果确定了一条边 (x, y) 的 t_i ，那么在 t_i 之前的时刻，我们都是可以把 x, y 看成是一个点的！

这样的化，我们在求 mid 时刻之前的关联集合的时候，t_i 在 r 之后的边我们就可以用并查集维护其带来的影响就可以了。

但是还有 t_i 在 l 之前的啊？——这个你稍微想一想就会发现其对 mid 时刻之后的关联集合是无影响的。

所以我们就可以在**一个分治层** $O(m)$ 的复杂度内求出答案了。

实现上要注意，因为一个 t_i 的边会影响所有小于 t_i 的时刻，所以我们应该先递归 (mid, r) 这一部分，然后在递归 $(l, mid-1)$ 的部分。

P5979 [PA2014] Druzyny

体育课上，n 个小朋友排成一行（从 1 到 n 编号），老师想把他们分成若干组，每一组都包含编号连续的一段小朋友，每个小朋友属于且仅属于一个组。

第 i 个小朋友希望它所在的组的人数不多于 d_i，不少于 c_i，否则他就会不满意。

在所有小朋友都满意的前提下，求可以分成的组的数目的最大值，以及有多少种分组方案能达到最大值。

数据范围：$n \leq 10^6$。

P5979 [PA2014] Druzyny

首先肯定有 n^2 dp，设 f_i 表示只考虑 $1 \sim i$ 小朋友的情况下的最大值和方案，这里定义 $+$ 为 f 的二元运算，有转移：

$$f_i = \sum_{j=1}^{i-1} (f_j + 1) \qquad \left(\max_{j<k\le i} l_k \le i - j + 1 \le \min_{j<k\le i} r_k \right)$$

考虑优化这个 dp。

P5979 [PA2014] Druzyny

首先要注意 $\leq (\min_{j<k\leq i} r_k)$ 这个限制是好做的，因为设 g_i 表示最小的 j 使 j 满足限制，那么 g_i 是单调不降的。

所以我们可以预处理出 g_i ，那么 dp 就成了：

$$f_i = \sum_{j=g_i}^{i-1} (f_j + 1) \qquad \left(\max_{j<k\leq i} l_k \leq i - j + 1 \right)$$

因为有 max 这种东西，我们可以考虑 cdq 分治优化 dp，然后就是一个二维偏序，这就是好做的。

所以就有 $n \log^2 n$ 的做法。但是 $n = 10^6$ ，这还不够优秀。

P5979 [PA2014] Druzyny

因为有区间 $\max$ 的存在，我们可以考虑笛卡尔树分治！这样对于一组 (l, r, mid)，$\forall x \in [l, mid), y \in [mid, r], f_x \to f_y$ 时，$\max_{j<k\le i} l_k = l_{mid}$！

但是笛卡尔树分治我们不能保证 mid 两边的区间大小尽量一致啊，这样怎么保证复杂度？

除了保证大小一致，我们还可以让一次分治 $x + y \to x, y$ 的复杂度达到 $\min(x, y)$，那么总复杂度也是 $n \log n$ 的。

P5979 [PA2014] Druzyny

现在考虑一下我们的转移，记 $k = l_{mid}$，那么对于 $y \in [mid, r]$ ，合法的 x 的范围是 $[\max(l, g_y), \min(mid-1, y-k)]$ 。考虑把 $\max, \min$ 去掉：

- $mid \le g_y$：无合法转移。
- $l < g_y < mid$：注意到一个这样的 y 只会有一次这样的转移。
- $g_y \le l$ ：现在值要考虑 $x \le \min(mid-1, x-k)$ ，我们先求出 $j \le mid-k$ 的所有 j 的贡献，然后每次随着 $y \leftarrow y+1$，x 也会 $\leftarrow x+1$，所以只会贡献一个新的 x ，因此直接扫描一遍就可以完成更新。最后如果 y 没有到 r ，则还需要整体修改 $[y+1, r]$ 一次。

所以对于操作二和操作三中的整体查询/修改，我们可以通过线段树维护，一次分治只会操作一次，所以是 $O(n \log n)$ 的，而操作三中的扫描一遍的复杂度是 $\min(x, y)$ 的，所以这部分也是 $O(n \log n)$ 的，所以就有 $O(n \log n)$ 的做法。

The End