分治基础

LC

ASDFZ

2019 年 3 月 19 日

浅谈分治

浅谈分治

分治，即分而治之，将一个问题分成若干个子问题处理并合并。

浅谈分治

分治，即分而治之，将一个问题分成若干个子问题处理并合并。

分治是一种思想，相信大家都掌握的非常熟练……

浅谈分治

分治，即分而治之，将一个问题分成若干个子问题处理并合并。

分治是一种思想，相信大家都掌握的非常熟练……

分治涉及的分支也不少：根号分治、树分治、cdq 分治、线段树分治、整体二分……

浅谈分治

分治，即分而治之，将一个问题分成若干个子问题处理并合并。

分治是一种思想，相信大家都掌握的非常熟练……

分治涉及的分支也不少：根号分治、树分治、cdq 分治、线段树分治、整体二分……

今天主要来简单看一下这些内容以及一些题目。

浅谈分治

分治，即分而治之，将一个问题分成若干个子问题处理并合并。

分治是一种思想，相信大家都掌握的非常熟练……

分治涉及的分支也不少：根号分治、树分治、cdq 分治、线段树分治、整体二分……

今天主要来简单看一下这些内容以及一些题目。

题目主要来源于校内 OJ 、BZOJ 和 codeforces, 之前上课的课件上的题目应该不会出现，希望能给大家一个不错的体验 ~

浅谈分治

分治，即分而治之，将一个问题分成若干个子问题处理并合并。

分治是一种思想，相信大家都掌握的非常熟练……

分治涉及的分支也不少：根号分治、树分治、cdq 分治、线段树分治、整体二分……

今天主要来简单看一下这些内容以及一些题目。

题目主要来源于校内 OJ 、BZOJ 和 codeforces, 之前上课的课件上的题目应该不会出现，希望能给大家一个不错的体验 ~

题目大多比较简单基础，欢迎大家上来秒切!

浅谈根号分治

浅谈根号分治

根号分治相信大家都会，但还是要简单介绍一下。

浅谈根号分治

根号分治相信大家都会，但还是要简单介绍一下。

根号分治主要就是对一个图，根据每个点的度数将其分为大点和小点，对两种点单独处理。

浅谈根号分治

根号分治相信大家都会，但还是要简单介绍一下。

根号分治主要就是对一个图，根据每个点的度数将其分为大点和小点，对两种点单独处理。

一般的处理方法为：对于每个点，枚举周围的大点并更新答案；

浅谈根号分治

根号分治相信大家都会，但还是要简单介绍一下。

根号分治主要就是对一个图，根据每个点的度数将其分为大点和小点，对两种点单独处理。

一般的处理方法为：对于每个点，枚举周围的大点并更新答案；

同时对于每个小点，枚举周围的所有点，更新周围点的信息。

浅谈根号分治

根号分治相信大家都会，但还是要简单介绍一下。

根号分治主要就是对一个图，根据每个点的度数将其分为大点和小点，对两种点单独处理。

一般的处理方法为：对于每个点，枚举周围的大点并更新答案；

同时对于每个小点，枚举周围的所有点，更新周围点的信息。

考虑平衡两种操作，我们一般取度数的分界线为 $\sqrt{n}$

浅谈根号分治

根号分治相信大家都会，但还是要简单介绍一下。

根号分治主要就是对一个图，根据每个点的度数将其分为大点和小点，对两种点单独处理。

一般的处理方法为：对于每个点，枚举周围的大点并更新答案；

同时对于每个小点，枚举周围的所有点，更新周围点的信息。

考虑平衡两种操作，我们一般取度数的分界线为 $\sqrt{n}$

在 zzy 的模拟赛中就出现过根号分治的题目，相信大家都会做了，我就不说了。

浅谈根号分治

根号分治相信大家都会，但还是要简单介绍一下。

根号分治主要就是对一个图，根据每个点的度数将其分为大点和小点，对两种点单独处理。

一般的处理方法为：对于每个点，枚举周围的大点并更新答案；

同时对于每个小点，枚举周围的所有点，更新周围点的信息。

考虑平衡两种操作，我们一般取度数的分界线为 $\sqrt{n}$

在 zzy 的模拟赛中就出现过根号分治的题目，相信大家都会做了，我就不说了。

接下来看两道根号分治题。

FZOJ2993 灯

有一排 n 个灯，每个灯有一个 1 到 m 的颜色。

一开始所有灯都是关着的。

有 q 次操作，每次改变某种颜色的灯的状态，每次操作后你需要求出有多少个极长的开着的灯的连续段。

$1 \leq n, q \leq 10^5$.

Solution

首先将相邻同颜色的灯合并，然后我们来构图。

Solution

首先将相邻同颜色的灯合并，然后我们来构图。

每种颜色的灯两两连边，边权为两种颜色的灯相邻的对数。

Solution

首先将相邻同颜色的灯合并，然后我们来构图。

每种颜色的灯两两连边，边权为两种颜色的灯相邻的对数。

这样答案就等于亮着的灯数减掉同时亮着的两灯的边权。

Solution

首先将相邻同颜色的灯合并，然后我们来构图。

每种颜色的灯两两连边，边权为两种颜色的灯相邻的对数。

这样答案就等于亮着的灯数减掉同时亮着的两灯的边权。

知道构图之后就比较容易了，直接根号分治，方法上上一页讲了。

Solution

首先将相邻同颜色的灯合并，然后我们来构图。

每种颜色的灯两两连边，边权为两种颜色的灯相邻的对数。

这样答案就等于亮着的灯数减掉同时亮着的两灯的边权。

知道构图之后就比较容易了，直接根号分治，方法上上一页讲了。

复杂度 $O(n\sqrt{n})$.

CF1039D You Are Given a Tree

给你一棵 n 个节点的树，对于每个正整数 $k \in [1, n]$ ，求出最多能找出多少条恰好包含 k 个点的互不相交的路径。

$1 \leq n \leq 10^5$.

Solution

对于 $O(n)$ 求单个整数 k 的答案相信大家都会做，每个点记录向下的最长链，能合并就直接贪心合并即可。

Solution

对于 $O(n)$ 求单个整数 k 的答案相信大家都会做，每个点记录向下的最长链，能合并就直接贪心合并即可。

我们考虑根据 k 大小分两种情况处理。

Solution

对于 $O(n)$ 求单个整数 k 的答案相信大家都会做，每个点记录向下的最长链，能合并就直接贪心合并即可。

我们考虑根据 k 大小分两种情况处理。

对于 $k \leq S$ ，我们直接按上面的方法对每个 k 暴力跑一遍。

Solution

对于 $O(n)$ 求单个整数 k 的答案相信大家都会做，每个点记录向下的最长链，能合并就直接贪心合并即可。

我们考虑根据 k 大小分两种情况处理。

对于 $k \leq S$ ，我们直接按上面的方法对每个 k 暴力跑一遍。

对于 $k > S$ ，剩下的答案最多只有 $\frac{n}{S}$ 种取值，直接二分每个答案所在区间即可。

Solution

对于 $O(n)$ 求单个整数 k 的答案相信大家都会做，每个点记录向下的最长链，能合并就直接贪心合并即可。

我们考虑根据 k 大小分两种情况处理。

对于 $k \leq S$ ，我们直接按上面的方法对每个 k 暴力跑一遍。

对于 $k > S$ ，剩下的答案最多只有 $\frac{n}{S}$ 种取值，直接二分每个答案所在区间即可。

时间复杂度为 $O(nS + \frac{n^2 \log n}{S})$ ，取 $S = \sqrt{n \log n}$ 时间复杂度最优为 $O(n\sqrt{n \log n})$ 。

浅谈树分治

浅谈树分治

树分治之前 zzt 都讲的差不多了，相信大家都非常熟悉了。

浅谈树分治

树分治之前 zzt 都讲的差不多了，相信大家都非常熟悉了。

点分治的核心是找一个点作为根，而找出来的这个点就是我们所说的“重心”。

浅谈树分治

树分治之前 zzt 都讲的差不多了，相信大家都非常熟悉了。

点分治的核心是找一个点作为根，而找出来的这个点就是我们所说的“重心”。

每次找出一个根以后，所有点对就只有两种可能了：

浅谈树分治

树分治之前 zzt 都讲的差不多了，相信大家都非常熟悉了。

点分治的核心是找一个点作为根，而找出来的这个点就是我们所说的“重心”。

每次找出一个根以后，所有点对就只有两种可能了：

- 两个点都在根的某一棵子树中，即路径不过根；

浅谈树分治

树分治之前 zzt 都讲的差不多了，相信大家都非常熟悉了。

点分治的核心是找一个点作为根，而找出来的这个点就是我们所说的“重心”。

每次找出一个根以后，所有点对就只有两种可能了：

- 两个点都在根的某一棵子树中，即路径不过根；
- 两个点在根的不同子树中，或其中一个点就是根，此时路径必过根。

浅谈树分治

树分治之前 zzt 都讲的差不多了，相信大家都非常熟悉了。

点分治的核心是找一个点作为根，而找出来的这个点就是我们所说的“重心”。

每次找出一个根以后，所有点对就只有两种可能了：

- 两个点都在根的某一棵子树中，即路径不过根；
- 两个点在根的不同子树中，或其中一个点就是根，此时路径必过根。

对于 case1 显然递归处理，对于 case2 直接在此处计算。

浅谈树分治

树分治之前 zzt 都讲的差不多了，相信大家都非常熟悉了。

点分治的核心是找一个点作为根，而找出来的这个点就是我们所说的“重心”。

每次找出一个根以后，所有点对就只有两种可能了：

- 两个点都在根的某一棵子树中，即路径不过根；
- 两个点在根的不同子树中，或其中一个点就是根，此时路径必过根。

对于 case1 显然递归处理，对于 case2 直接在此处计算。

每次统计经过根的路径，注意 case1 不要重复计算。这就是树分治的基本套路。

简单题抢答环节

接下来会有若干道树分治板子题。你会了后可以立刻冲上来抢答。

简单题抢答环节

接下来会有若干道树分治板子题。你会了后可以立刻冲上来抢答。

抢答成功的同学课后会获得辣鸡讲课人的小礼物（0.5￥红包）！

简单题抢答环节

接下来会有若干道树分治板子题。你会了后可以立刻冲上来抢答。

抢答成功的同学课后会获得辣鸡讲课人的小礼物（0.5￥红包）！

你需要回答：对于每个重心统计什么信息，怎么统计答案。

树分治经典例题 - 1

树分治经典例题 - 1

（以下例题默认树节点数 $\leq 10^5$）

给你一棵树，询问长度为 $\leq L$ 的路径有多少条。

树分治经典例题 - 1

（以下例题默认树节点数 $\leq 10^5$）

给你一棵树，询问长度为 $\leq L$ 的路径有多少条。

把所有深度拉下来排个序，双指针统计答案。非常简单。

树分治经典例题 - 2

树分治经典例题 - 2

给你一棵树，询问长度为 L 的路径有多少条。

树分治经典例题 - 2

给你一棵树，询问长度为 L 的路径有多少条。

统计每个深度个数，统计答案直接两两相乘即可。非常简单。

树分治经典例题 - 3

树分治经典例题 - 3

给你一棵树， q 次询问，每次询问是否存在长度为 L 的路径。

$q \leq 50$.

树分治经典例题 - 3

给你一棵树， q 次询问，每次询问是否存在长度为 L 的路径。

$q \leq 50$.

询问数很少。每个询问离线存下来，直接枚举每个点，对于每个点到重心的距离枚举每个询问统计答案，并记录深度。注意不要重复统计即可。非常简单。

树分治经典例题 - 4

树分治经典例题 - 4

给你一棵树， q 次询问，每次询问长度为 L 的路径有多少条。

$q \leq 10^5$.

树分治经典例题 - 4

给你一棵树， q 次询问，每次询问长度为 L 的路径有多少条。

$q \leq 10^5$.

前两题拼在一起？还是非常简单。

树分治经典例题 - 4

给你一棵树， q 次询问，每次询问长度为 L 的路径有多少条。

$q \leq 10^5$.

前两题拼在一起？还是非常简单。

依然统计每个深度个数。最后统计答案是卷积形式，直接 FFT 优化即可。

浅谈动态树分治

接下来简单谈谈动态树分治。

浅谈动态树分治

接下来简单谈谈动态树分治。

点分治的重心连接形成的树就叫点分树。对于每棵点分树我们用数据结构维护信息，这就是动态点分治。

浅谈动态树分治

接下来简单谈谈动态树分治。

点分治的重心连接形成的树就叫点分树。对于每棵点分树我们用数据结构维护信息，这就是动态点分治。

我们修改、查询都只影响点分树从根到当前重心的一条链，由于点分树树高不超过 $\log n$ ，所以我们暴力往上跳复杂度有保证。

浅谈动态树分治

接下来简单谈谈动态树分治。

点分治的重心连接形成的树就叫点分树。对于每棵点分树我们用数据结构维护信息，这就是动态点分治。

我们修改、查询都只影响点分树从根到当前重心的一条链，由于点分树树高不超过 $\log n$ ，所以我们暴力往上跳复杂度有保证。

一般的套路是，我们用数据结构维护每棵点分树内节点到当前根的信息和到根的父亲的信息（避免重复统计，这也是点分治的套路）。

浅谈动态树分治

接下来简单谈谈动态树分治。

点分治的重心连接形成的树就叫点分树。对于每棵点分树我们用数据结构维护信息，这就是动态点分治。

我们修改、查询都只影响点分树从根到当前重心的一条链，由于点分树树高不超过 $\log n$ ，所以我们暴力往上跳复杂度有保证。

一般的套路是，我们用数据结构维护每棵点分树内节点到当前根的信息和到根的父亲的信息（避免重复统计，这也是点分治的套路）。

动态点分治题目的特点一般是：码量大，套路化~~，考场上一般不可写~~。

最最经典的动态点分治例题

给定一棵树，在线支持两种操作：

- 查询与某点距离不超过 k 的点的权值和；
- 修改某点权值。

最最经典的动态点分治例题

给定一棵树，在线支持两种操作：

- 查询与某点距离不超过 k 的点的权值和；
- 修改某点权值。

Source: BZOJ3730 震波

最最经典的动态点分治例题

给定一棵树，在线支持两种操作：

- 查询与某点距离不超过 k 的点的权值和；
- 修改某点权值。

Source: BZOJ3730 震波

相信大家都做过，哪位同学上来说一下？

最最经典的动态点分治例题

给定一棵树，在线支持两种操作：

- 查询与某点距离不超过 k 的点的权值和；
- 修改某点权值。

Source: BZOJ3730 震波

相信大家都做过，哪位同学上来说一下？

每棵点分树内用线段树维护子节点到当前根的距离和到根的父亲的距离。查询修改直接按套路就可以了。

浅谈 cdq 分治

浅谈 cdq 分治

cdq 分治简单来说就是将修改/查询分治处理并统计左区间对右区间的影响/贡献。

浅谈 cdq 分治

cdq 分治简单来说就是将修改/查询分治处理并统计左区间对右区间的影响/贡献。

cdq 分治主要分三个专题，接下来将配合例题带大家一一复习！

复习 - 三维偏序问题

三维偏序问题相信大家都非常熟练!

复习 - 三维偏序问题

三维偏序问题相信大家都非常熟练!

具体的做法就简单说一下：我们先对第一关键字排序，然后分治处理。对于区间 $[l, r]$ ，我们考虑我们对区间 $[l, mid], [mid + 1, r]$ 分别按照第二关键字排序，然后统计 $[l, mid]$ 和 $[mid + 1, r]$ 两个区间之间第三关键字对答案的贡献，用数据结构维护。

（表达能力不太行……大概就这个意思）

复习 - 三维偏序问题

三维偏序问题相信大家都非常熟练!

具体的做法就简单说一下：我们先对第一关键字排序，然后分治处理。对于区间 $[l, r]$ ，我们考虑我们对区间 $[l, mid], [mid + 1, r]$ 分别按照第二关键字排序，然后统计 $[l, mid]$ 和 $[mid + 1, r]$ 两个区间之间第三关键字对答案的贡献，用数据结构维护。

（表达能力不太行……大概就这个意思）

每次排序太慢啦！怎么办？

复习 - 三维偏序问题

三维偏序问题相信大家都非常熟练!

具体的做法就简单说一下：我们先对第一关键字排序，然后分治处理。

对于区间 $[l, r]$ ，我们考虑我们对区间 $[l, mid], [mid + 1, r]$ 分别按照第二关键字排序，然后统计 $[l, mid]$ 和 $[mid + 1, r]$ 两个区间之间第三关键字对答案的贡献，用数据结构维护。

（表达能力不太行……大概就这个意思）

每次排序太慢啦！怎么办？

每次分治结束以后进行归并，相当于进行了归并排序。

FZOJ1859 动态逆序对

给 1 到 n 的一个排列，按照某种顺序依次删除 m 个元素，你的任务是在每次删除一个元素之前统计整个序列的逆序对数。

$N \leq 100000, M \leq 50000$.

FZOJ1859 动态逆序对

给 1 到 n 的一个排列，按照某种顺序依次删除 m 个元素，你的任务是在每次删除一个元素之前统计整个序列的逆序对数。

$N \leq 100000, M \leq 50000$.

（只讨论分治做法）

Solution

将操作离线反转，将删除操作看做插入操作。

Solution

将操作离线反转，将删除操作看做插入操作。

设一个点 i 的位置为 p_i，插入的时间为 t_i，权值为 w_i.

Solution

将操作离线反转，将删除操作看做插入操作。

设一个点 i 的位置为 p_i，插入的时间为 t_i，权值为 w_i.

i 的贡献即为

$\sum_j [t_j < t_i, p_j < p_i, w_j > w_i] + \sum_j [t_j > t_i, p_j > p_i, w_j < w_i]$

Solution

将操作离线反转，将删除操作看做插入操作。

设一个点 i 的位置为 p_i，插入的时间为 t_i，权值为 w_i.

i 的贡献即为

$\sum_j [t_j < t_i, p_j < p_i, w_j > w_i] + \sum_j [t_j > t_i, p_j > p_i, w_j < w_i]$

我们发现这就是一个裸的三维偏序问题，直接用 cdq 做就可以了！

FZOJ1850 拦截导弹

给定一个长度为 n 的二元组序列 (h_i, v_i)，从其中选出尽可能多的元素使得 $\forall i < j, h_i \geq h_j, v_i \geq v_j$. 求最多能选出的元素的个数和每个元素被选中的概率。

$n \leq 5 \cdot 10^4$.

FZOJ1850 拦截导弹

给定一个长度为 n 的二元组序列 (h_i, v_i)，从其中选出尽可能多的元素使得 $\forall i < j, h_i \geq h_j, v_i \geq v_j$. 求最多能选出的元素的个数和每个元素被选中的概率。

$n \leq 5 \cdot 10^4$.

tds 已经讲过这题的二维线段树做法了，所以这里我们仍然不讨论其他做法。

Solution

题意即为求二元组的 LIS .

Solution

题意即为求二元组的 LIS .

考虑 DP . 设 $f(i)$ 表示以 i 结尾的 LIS 长度， $g(i)$ 表示以 i 结尾的 LIS 有多少个。

Solution

题意即为求二元组的 LIS .

考虑 DP . 设 $f(i)$ 表示以 i 结尾的 LIS 长度， $g(i)$ 表示以 i 结尾的 LIS 有多少个。

则有：

$$f(i) = \max\{f(j)\} + 1(j \le i, h_j \ge h_i, v_j \ge v_i)$$

$$g(i) = \sum\{g(j)\}(j \le i, h_j \ge h_i, v_j \ge v_i, f(i) = f(j) + 1)$$

Solution

题意即为求二元组的 LIS .

考虑 DP . 设 $f(i)$ 表示以 i 结尾的 LIS 长度， $g(i)$ 表示以 i 结尾的 LIS 有多少个。

则有：

$$f(i) = \max\{f(j)\} + 1(j \le i, h_j \ge h_i, v_j \ge v_i)$$

$$g(i) = \sum\{g(j)\}(j \le i, h_j \ge h_i, v_j \ge v_i, f(i) = f(j) + 1)$$

同样，我们考虑 $f(i), g(i)$ 反过来的情况 $f'(i), g'(i)$，即将以 i 结尾改成从 i 开始。 dp 式类似。

Solution

考虑最后的概率计算，如果当前 i 两边 LIS 加起来正好等于答案， i 的答案就为以 i 结尾和开始的方案数的乘积除以方案总数。

Solution

考虑最后的概率计算，如果当前 i 两边 LIS 加起来正好等于答案， i 的答案就为以 i 结尾和开始的方案数的乘积除以方案总数。

接下来的任务就是计算 dp 式。我们发现上面的 dp 都是三维偏序的模型，直接用 cdq 分治 + 树状数组维护.

Solution

考虑最后的概率计算，如果当前 i 两边 LIS 加起来正好等于答案， i 的答案就为以 i 结尾和开始的方案数的乘积除以方案总数。

接下来的任务就是计算 dp 式。我们发现上面的 dp 都是三维偏序的模型，直接用 cdq 分治 + 树状数组维护.

当然实现也有一点小技巧：比如计算 $f'(i), g'(i)$ ，可以直接将原序列反转并将权值反转，可以直接套用原来的 cdq .

复习 - cdq 的二维数点

复习 - cdq 的二维数点

离线的二维数点（支持修改）大家都非常熟悉吧!

复习 - cdq 的二维数点

离线的二维数点（支持修改）大家都非常熟悉吧!

但有的时候树套树TLE/MLE用不了怎么办?

复习 - cdq 的二维数点

离线的二维数点（支持修改）大家都非常熟悉吧!

但有的时候树套树TLE/MLE用不了怎么办?

这时候就轮到 cdq 分治出场了!

复习 - cdq 的二维数点

离线的二维数点（支持修改）大家都非常熟悉吧!

但有的时候树套树TLE/MLE用不了怎么办?

这时候就轮到 cdq 分治出场了!

虽然 cdq 不常用于二维数点，但在某些题目中，cdq 还是管用的!

复习 - cdq 的二维数点

cdq 怎么做二维数点啊？

复习 - cdq 的二维数点

cdq 怎么做二维数点啊？

二维差分相信大家都会。

复习 - cdq 的二维数点

cdq 怎么做二维数点啊？

二维差分相信大家都会。

那么直接对询问的矩阵二维差分一下就可以啦！

CF848C Goodbye Souvenir

给你一个序列 $\{a_n\}$ 和 m 个操作。操作有两种：

- 1 p x：将 a_p 修改为 x
- 2 l r：询问 $[l, r]$ 中每种元素最后出现位置-最先出现位置之和。

$n, m \leq 10^5$.

Solution

一眼二维数点。

Solution

一眼二维数点。

每个点记为 (i, pre_i) (i 为下标， pre_i 为与 i 相同元素的上一个位置)，

权值即为 $i - pre_i$.

（这个套路相信大家都比较熟悉）

Solution

一眼二维数点。

每个点记为 (i, pre_i) (i 为下标， pre_i 为与 i 相同元素的上一个位置)，权值即为 $i - pre_i$.

（这个套路相信大家都比较熟悉）

那么答案所求即为坐标范围在 $[l, r]$ 内的元素之和，裸的二维数点。

Solution

一眼二维数点。

每个点记为 (i, pre_i) (i 为下标， pre_i 为与 i 相同元素的上一个位置)，权值即为 $i - pre_i$.

（这个套路相信大家都比较熟悉）

那么答案所求即为坐标范围在 $[l, r]$ 内的元素之和，裸的二维数点。

对于修改操作，直接用 set 维护每种元素即可，相信大家都会。

Solution

一眼二维数点。

每个点记为 (i, pre_i) (i 为下标， pre_i 为与 i 相同元素的上一个位置)，

权值即为 $i - pre_i$.

（这个套路相信大家都比较熟悉）

那么答案所求即为坐标范围在 $[l, r]$ 内的元素之和，裸的二维数点。

对于修改操作，直接用 set 维护每种元素即可，相信大家都会。

树套树？

Solution

一眼二维数点。

每个点记为 (i, pre_i) (i 为下标， pre_i 为与 i 相同元素的上一个位置)，权值即为 $i - pre_i$.

（这个套路相信大家都比较熟悉）

那么答案所求即为坐标范围在 $[l, r]$ 内的元素之和，裸的二维数点。

对于修改操作，直接用 set 维护每种元素即可，相信大家都会。

树套树？

这题写树套树不仅常数大，而且容易爆空间。

Solution

一眼二维数点。

每个点记为 (i, pre_i) (i 为下标， pre_i 为与 i 相同元素的上一个位置)，

权值即为 $i - pre_i$.

（这个套路相信大家都比较熟悉）

那么答案所求即为坐标范围在 $[l, r]$ 内的元素之和，裸的二维数点。

对于修改操作，直接用 set 维护每种元素即可，相信大家都会。

树套树？

这题写树套树不仅常数大，而且容易爆空间。

那就用 cdq 维护吧!

BZOJ4237 稻草人

给定平面上 N 个关键点，询问有多少个矩形满足左下和右上各有一个关键点，且矩形中间没有关键点。

$N \leq 2 \cdot 10^5$.

Solution

这应该是今天难度最大的题吧，而且这也不算是二维数点类的题……

Solution

这应该是今天难度最大的题吧，而且这也不算是二维数点类的题……

我们按 x 排序分治，对于左右两边的区间按 y 排序。

Solution

这应该是今天难度最大的题吧，而且这也不算是二维数点类的题……

我们按 x 排序分治，对于左右两边的区间按 y 排序。

考虑左边的点对右边的每个点产生的贡献。

Solution

这应该是今天难度最大的题吧，而且这也不算是二维数点类的题……

我们按 x 排序分治，对于左右两边的区间按 y 排序。

考虑左边的点对右边的每个点产生的贡献。

比较容易发现，产生贡献的点的 x 一定单减，我们维护一个单调栈，每个右边的点的答案即为当前单调栈的大小

Solution

这应该是今天难度最大的题吧，而且这也不算是二维数点类的题……

我们按 x 排序分治，对于左右两边的区间按 y 排序。

考虑左边的点对右边的每个点产生的贡献。

比较容易发现，产生贡献的点的 x 一定单减，我们维护一个单调栈，每个右边的点的答案即为当前单调栈的大小……吗？

Solution

这应该是今天难度最大的题吧，而且这也不算是二维数点类的题……

我们按 x 排序分治，对于左右两边的区间按 y 排序。

考虑左边的点对右边的每个点产生的贡献。

比较容易发现，产生贡献的点的 x 一定单减，我们维护一个单调栈，每个右边的点的答案即为当前单调栈的大小……吗？

要注意的是：之前的点也会对当前点产生影响：他们会被包含在矩形里。我们考虑处理这样的情况。

Solution

我们注意到，如果左边的点能和之前统计的右边的点形成矩形，那么这个点一定不会对当前点产生贡献。

Solution

我们注意到，如果左边的点能和之前统计的右边的点形成矩形，那么这个点一定不会对当前点产生贡献。

那么做法就比较显然了：我们对于离当前点最近的横坐标比它小的点，在左边二分找纵坐标比该点小的点数，统计答案时减掉这部分点即可。

Solution

我们注意到，如果左边的点能和之前统计的右边的点形成矩形，那么这个点一定不会对当前点产生贡献。

那么做法就比较显然了：我们对于离当前点最近的横坐标比它小的点，在左边二分找纵坐标比该点小的点数，统计答案时减掉这部分点即可。

怎么找这样的点？我们对右边维护一个横坐标单增的单调栈即可。

Solution

我们注意到，如果左边的点能和之前统计的右边的点形成矩形，那么这个点一定不会对当前点产生贡献。

那么做法就比较显然了：我们对于离当前点最近的横坐标比它小的点，在左边二分找纵坐标比该点小的点数，统计答案时减掉这部分点即可。

怎么找这样的点？我们对右边维护一个横坐标单增的单调栈即可。

代码非常好写。

复习 - cdq 和斜率优化 dp

复习 - cdq 和斜率优化 dp

斜率优化 dp 相信大家都很熟悉。（我就不熟悉）

复习 - cdq 和斜率优化 dp

斜率优化 dp 相信大家都很熟悉。（我就不熟悉）

简单来说，就是将 dp 式子转化成斜率的形式，利用单调栈维护斜率的单调性，即维护一个凸壳。

复习 - cdq 和斜率优化 dp

斜率优化 dp 相信大家都很熟悉。（我就不熟悉）

简单来说，就是将 dp 式子转化成斜率的形式，利用单调栈维护斜率的单调性，即维护一个凸壳。

但是有的题目斜率并不随 x 单调递增，因此我们需要动态维护凸壳。

复习 - cdq 和斜率优化 dp

斜率优化 dp 相信大家都很熟悉。（我就不熟悉）

简单来说，就是将 dp 式子转化成斜率的形式，利用单调栈维护斜率的单调性，即维护一个凸壳。

但是有的题目斜率并不随 x 单调递增，因此我们需要动态维护凸壳。

动态维护凸壳有两种方法：splay 和 cdq 。这里只谈 cdq 。

复习 - cdq 和斜率优化 dp

斜率优化 dp 相信大家都很熟悉。（我就不熟悉）

简单来说，就是将 dp 式子转化成斜率的形式，利用单调栈维护斜率的单调性，即维护一个凸壳。

但是有的题目斜率并不随 x 单调递增，因此我们需要动态维护凸壳。

动态维护凸壳有两种方法：splay 和 cdq 。这里只谈 cdq 。

具体的操作就是，我们先递归左区间，单调栈维护出左区间的凸壳，然后再将右区间的点依次在凸壳上跳就可以了。

FZOJ1197 任务安排

机器上有 N 个需要处理的任务。这 N 个任务被分成若干批，每批包含相邻的若干任务。从时刻 0 开始，这些任务被分批加工，第 i 个任务单独完成所需的时间是 T_i。在每批任务开始前，机器需要启动时间 S ，而完成这批任务所需的时间是各个任务需要时间的总和。

注意，同一批任务将在同一时刻完成。每个任务的费用是它的完成时刻乘以一个费用系数 F_i 。请确定一个分组方案，使得总费用最小。

$0 < N \leq 100000, -2^8 \leq T_i \leq 2^8, 0 \leq S \leq 2^8, 0 \leq F_i \leq 2^8$

Solution

设 $f[i]$ 表示前 i 个任务的最小费用。不难列出dp方程：

Solution

设 $f[i]$ 表示前 i 个任务的最小费用。不难列出dp方程：

$$f[i] = \min_{1 \le j < i} \{f[j] + (F[n] - F[j]) \cdot (T[i] - T[j] + S)\}$$

（这里 F, T 是前缀和）

Solution

设 $f[i]$ 表示前 i 个任务的最小费用。不难列出dp方程：

$$f[i] = \min_{1 \le j < i} \{f[j] + (F[n] - F[j]) \cdot (T[i] - T[j] + S)\}$$

（这里 F, T 是前缀和）

转化成斜率的形式：

$$f[i] = \min_{1 \le j < i} \{-T[i] \cdot F[j] + (f[j] - F[n] \cdot T[j] + F[j] \cdot T[j] - S \cdot F[j])\}$$

Solution

设 $f[i]$ 表示前 i 个任务的最小费用。不难列出dp方程：

$$f[i] = \min_{1 \le j < i} \{f[j] + (F[n] - F[j]) \cdot (T[i] - T[j] + S)\}$$

（这里 F, T 是前缀和）

转化成斜率的形式：

$$f[i] = \min_{1 \le j < i} \{-T[i] \cdot F[j] + (f[j] - F[n] \cdot T[j] + F[j] \cdot T[j] - S \cdot F[j])\}$$

因为 T_i 可能是负数，所以不能单调栈维护斜率。

Solution

设 $f[i]$ 表示前 i 个任务的最小费用。不难列出dp方程：

$$f[i] = \min_{1 \le j < i} \{f[j] + (F[n] - F[j]) \cdot (T[i] - T[j] + S)\}$$

（这里 F, T 是前缀和）

转化成斜率的形式：

$$f[i] = \min_{1 \le j < i} \{-T[i] \cdot F[j] + (f[j] - F[n] \cdot T[j] + F[j] \cdot T[j] - S \cdot F[j])\}$$

因为 T_i 可能是负数，所以不能单调栈维护斜率。

因此我们使用 cdq 动态维护凸壳。

浅谈整体二分

浅谈整体二分

嘛…… 整体二分，没啥好说的，上次 jhr 已经讲得差不多啦

浅谈整体二分

嘛…… 整体二分，没啥好说的，上次 jhr 已经讲得差不多啦

这个东西什么时候用？有个神仙在他的博客里这么说：

浅谈整体二分

嘛…… 整体二分，没啥好说的，上次 jhr 已经讲得差不多啦

这个东西什么时候用？有个神仙在他的博客里这么说：

- 当你发现多组询问可以离线的时候

浅谈整体二分

嘛…… 整体二分，没啥好说的，上次 jhr 已经讲得差不多啦

这个东西什么时候用？有个神仙在他的博客里这么说：

- 当你发现多组询问可以离线的时候
- 当你发现询问可以二分答案而且check复杂度对于单组询问可以接受的时候

浅谈整体二分

嘛…… 整体二分，没啥好说的，上次 jhr 已经讲得差不多啦

这个东西什么时候用？有个神仙在他的博客里这么说：

- 当你发现多组询问可以离线的时候
- 当你发现询问可以二分答案而且check复杂度对于单组询问可以接受的时候
- 当你发现询问的操作都是一样的的时候

浅谈整体二分

嘛…… 整体二分，没啥好说的，上次 jhr 已经讲得差不多啦

这个东西什么时候用？有个神仙在他的博客里这么说：

- 当你发现多组询问可以离线的时候
- 当你发现询问可以二分答案而且check复杂度对于单组询问可以接受的时候
- 当你发现询问的操作都是一样的的时候

整体二分简单来说就是对所有询问二分一个答案，然后根据check结果将询问分成两半继续二分。

浅谈整体二分

嘛…… 整体二分，没啥好说的，上次 jhr 已经讲得差不多啦

这个东西什么时候用？有个神仙在他的博客里这么说：

- 当你发现多组询问可以离线的时候
- 当你发现询问可以二分答案而且check复杂度对于单组询问可以接受的时候
- 当你发现询问的操作都是一样的的时候

整体二分简单来说就是对所有询问二分一个答案，然后根据check结果将询问分成两半继续二分。

相信大家都已经熟练掌握整体二分了，所以我们直接看题8!

浅谈整体二分

嘛…… 整体二分，没啥好说的，上次 jhr 已经讲得差不多啦

这个东西什么时候用？有个神仙在他的博客里这么说：

- 当你发现多组询问可以离线的时候
- 当你发现询问可以二分答案而且check复杂度对于单组询问可以接受的时候
- 当你发现询问的操作都是一样的的时候

整体二分简单来说就是对所有询问二分一个答案，然后根据check结果将询问分成两半继续二分。

相信大家都已经熟练掌握整体二分了，所以我们直接看题8!

(PS：整体二分的题目真的不多，我也就找到这么几题，大家凑合着看吧！)

再探：动态排名 (FZOJ1784)

给定一个含有 n 个数的序列 $a_1, a_2, ..., a_n$ ， 有 m 次操作：

- 单点修改
- 区间查询第 k 大

$n, m \leq 10000, a_i \leq 10^9$.

再探：动态排名 (FZOJ1784)

给定一个含有 n 个数的序列 $a_1, a_2, ..., a_n$ ，有 m 次操作：

- 单点修改
- 区间查询第 k 大

$n, m \leq 10000, a_i \leq 10^9$.

相信大家都已经用各种方法过了这题了。这题用整体二分怎么做？

Solution

将修改和询问放在一起，注意这里不能排序 .

Solution

将修改和询问放在一起，注意这里不能排序 .

对所有询问二分一个答案 mid ，用树状数组维护有多少个数 $\leq mid$.

Solution

将修改和询问放在一起，注意这里不能排序 .

对所有询问二分一个答案 mid ，用树状数组维护有多少个数 $\leq mid$.

然后就直接整体二分就可以了。

BZOJ3833 Solar lamps

有 n 盏灯，所有灯面向同一方向，每盏灯各有照明范围。比如有盏灯位于 (x, y)，则它会照亮在 (x, y) 到 $(x + X1, y + Y1)$ 这条射线，(x, y) 到 $(x + X2, y + Y2)$ 这条射线之间的区域。

第 i 盏灯会在时刻 i 被点亮发光。当第 i 盏灯被 k_i 盏灯照射时也会发光。

求每盏灯在哪个时刻发光。

$n \leq 2 \cdot 10^5, |x|, |y| \leq 10^9$.

Solution

向量相信大家都非常熟练!

Solution

向量相信大家都非常熟练!

因为方向相同，我们将所有的射线以向量 $(X1, Y1), (X2, Y2)$ 为基底，转化一下坐标。

Solution

向量相信大家都非常熟练!

因为方向相同，我们将所有的射线以向量 $(X1, Y1), (X2, Y2)$ 为基底，转化一下坐标。

（注意特判一条直线的情况，要对一个向量的旋转一个角度，对答案不产生影响）。

Solution

向量相信大家都非常熟练!

因为方向相同，我们将所有的射线以向量 $(X1, Y1), (X2, Y2)$ 为基底，转化一下坐标。

（注意特判一条直线的情况，要对一个向量的旋转一个角度，对答案不产生影响）。

那么一个灯点亮的范围即为其右上角（第一象限）。

Solution

向量相信大家都非常熟练!

因为方向相同，我们将所有的射线以向量 $(X1, Y1), (X2, Y2)$ 为基底，转化一下坐标。

（注意特判一条直线的情况，要对一个向量的旋转一个角度，对答案不产生影响）。

那么一个灯点亮的范围即为其右上角（第一象限）。

考虑整体二分，二分答案 mid ，如果当前灯编号 $i < mid$ 或左下角有 $\geq k_i$ 盏灯就放到左边，否则放到右边。

Solution

向量相信大家都非常熟练!

因为方向相同，我们将所有的射线以向量 $(X1, Y1), (X2, Y2)$ 为基底，转化一下坐标。

（注意特判一条直线的情况，要对一个向量的旋转一个角度，对答案不产生影响）。

那么一个灯点亮的范围即为其右上角（第一象限）。

考虑整体二分，二分答案 mid ，如果当前灯编号 $i < mid$ 或左下角有 $\geq k_i$ 盏灯就放到左边，否则放到右边。

维护统计灯直接离散化 $+$ 按 x 排序 $+$ 树状数组即可。

「HNOI2015」接水果

给你一棵 n 个节点的树和 P 条树上的路径，每条路径有一个权值。给出 Q 个询问，每次询问包含 $u_i \to v_i$ 子路径的所有路径中第 k 小的权值。

$n, P, Q \leq 40000$.

Solution

考虑路径 (u, v) 是路径 (a, b) 的子路径要满足的条件。

Solution

考虑路径 (u, v) 是路径 (a, b) 的子路径要满足的条件。

如果 u 和 v 不是祖孙关系很简单，只要满足 a, b 分别在 u, v 的子树里即可。

Solution

考虑路径 (u, v) 是路径 (a, b) 的子路径要满足的条件。

如果 u 和 v 不是祖孙关系很简单，只要满足 a, b 分别在 u, v 的子树里即可。

如果 u 和 v 是祖孙关系也不难。设 u 是 v 的祖先， w 是 $u \to v$ 路径上 u 的孩子。那么要满足 a, b 中有一个节点在 v 的子树中，另一个节点不在 w 的子树中。

Solution

考虑路径 (u, v) 是路径 (a, b) 的子路径要满足的条件。

如果 u 和 v 不是祖孙关系很简单，只要满足 a, b 分别在 u, v 的子树里即可。

如果 u 和 v 是祖孙关系也不难。设 u 是 v 的祖先， w 是 $u \to v$ 路径上 u 的孩子。那么要满足 a, b 中有一个节点在 v 的子树中，另一个节点不在 w 的子树中。

将上述条件用 dfs 序描述出来，其实是一个矩形覆盖问题。即，将每条给定路径抽象成 $1 \sim 2$ 个矩形，每个询问的路径抽象成一个点，其实就是判断一个点在何时被覆盖 k 次。

Solution

考虑路径 (u,v) 是路径 (a,b) 的子路径要满足的条件。

如果 u 和 v 不是祖孙关系很简单，只要满足 a,b 分别在 u,v 的子树里即可。

如果 u 和 v 是祖孙关系也不难。设 u 是 v 的祖先， w 是 $u \to v$ 路径上 u 的孩子。那么要满足 a,b 中有一个节点在 v 的子树中，另一个节点不在 w 的子树中。

将上述条件用 dfs 序描述出来，其实是一个矩形覆盖问题。即，将每条给定路径抽象成 $1 \sim 2$ 个矩形，每个询问的路径抽象成一个点，其实就是判断一个点在何时被覆盖 k 次。

整体二分，每次覆盖矩形用树状数组维护即可。

「CTSC2018」混合果汁

商店里有 n 种果汁，编号为 $0, 1, 2, ..., n-1$。i 号果汁的美味度是 d_i，每升价格为 p_i。小 R 在制作混合果汁时，还有一些特殊的规定，即在一瓶混合果汁中，i 号果汁最多只能添加 l_i 升。

现在有 m 个小朋友过来找小 R 要混合果汁喝，他们都希望小 R 用商店里的果汁制作成一瓶混合果汁。其中，第 j 个小朋友希望他得到的混合果汁总价格不大于 g_j，体积不小于 L_j。在上述这些限制条件下，小朋友们还希望混合果汁的美味度尽可能地高，一瓶混合果汁的美味度等于所有参与混合的果汁的美味度的最小值。请你计算每个小朋友能喝到的最美味的混合果汁的美味度。

$n, m \leq 100000,\ 1 \leq d_i, p_i, l_i \leq 10^5, 1 \leq g_j, L_j \leq 10^{18}$.

「CTSC2018」混合果汁

商店里有 n 种果汁，编号为 $0, 1, 2, ..., n-1$。i 号果汁的美味度是 d_i，每升价格为 p_i。小 R 在制作混合果汁时，还有一些特殊的规定，即在一瓶混合果汁中，i 号果汁最多只能添加 l_i 升。

现在有 m 个小朋友过来找小 R 要混合果汁喝，他们都希望小 R 用商店里的果汁制作成一瓶混合果汁。其中，第 j 个小朋友希望他得到的混合果汁总价格不大于 g_j，体积不小于 L_j。在上述这些限制条件下，小朋友们还希望混合果汁的美味度尽可能地高，一瓶混合果汁的美味度等于所有参与混合的果汁的美味度的最小值。请你计算每个小朋友能喝到的最美味的混合果汁的美味度。

$n, m \leq 100000$, $1 \leq d_i, p_i, l_i \leq 10^5$, $1 \leq g_j, L_j \leq 10^{18}$.

PS：欢迎来分享本题的各种做法!

Solution

既然我们现在讲整体二分，那么就往整体二分上想。

Solution

既然我们现在讲整体二分，那么就往整体二分上想。

我们对所有的需求整体二分一个答案 mid ，表示最小美味度为 mid 。

Solution

既然我们现在讲整体二分，那么就往整体二分上想。

我们对所有的需求整体二分一个答案 mid ，表示最小美味度为 mid 。

将所有美味度 $\geq mid$ 的果汁用线段树维护。

Solution

既然我们现在讲整体二分，那么就往整体二分上想。

我们对所有的需求整体二分一个答案 mid ，表示最小美味度为 mid 。

将所有美味度 $\geq mid$ 的果汁用线段树维护。

线段树以单价为下标维护价格和数量，查询线段树二分即可。

Solution

既然我们现在讲整体二分，那么就往整体二分上想。

我们对所有的需求整体二分一个答案 mid ，表示最小美味度为 mid 。

将所有美味度 $\geq mid$ 的果汁用线段树维护。

线段树以单价为下标维护价格和数量，查询线段树二分即可。

当然也可以用 set 维护，代码复杂度优于线段树，时间效率劣于线段树。（我比较懒也比较菜，就用了 set）

Solution

既然我们现在讲整体二分，那么就往整体二分上想。

我们对所有的需求整体二分一个答案 mid ，表示最小美味度为 mid 。

将所有美味度 $\geq mid$ 的果汁用线段树维护。

线段树以单价为下标维护价格和数量，查询线段树二分即可。

当然也可以用 set 维护，代码复杂度优于线段树，时间效率劣于线段树。（我比较懒也比较菜，就用了 set）

这里也有细节要注意。比如我们不用每次都清空，用一个指针记录当前的位置，直接左右移动修改即可。

Solution

既然我们现在讲整体二分，那么就往整体二分上想。

我们对所有的需求整体二分一个答案 mid ，表示最小美味度为 mid 。

将所有美味度 $\geq mid$ 的果汁用线段树维护。

线段树以单价为下标维护价格和数量，查询线段树二分即可。

当然也可以用 set 维护，代码复杂度优于线段树，时间效率劣于线段树。（我比较懒也比较菜，就用了 set）

这里也有细节要注意。比如我们不用每次都清空，用一个指针记录当前的位置，直接左右移动修改即可。

时间复杂度 $O(n \log^2 n)$ ，和二分主席树的复杂度相同，可以通过此题。

浅谈线段树分治

浅谈线段树分治

今天最后一个专题：线段树分治!

浅谈线段树分治

今天最后一个专题：线段树分治!

众所周知，线段树是一个优秀分治结构，线段树作为数据结构最优秀地方的就是能支持快速修改和查询。

浅谈线段树分治

今天最后一个专题：线段树分治!

众所周知，线段树是一个优秀分治结构，线段树作为数据结构最优秀地方的就是能支持快速修改和查询。

那么我们利用（或模拟）这一优秀的分治结构，解决一些可能 cdq 分治不太能解决的问题。

浅谈线段树分治

今天最后一个专题：线段树分治!

众所周知，线段树是一个优秀分治结构，线段树作为数据结构最优秀地方的就是能支持快速修改和查询。

那么我们利用（或模拟）这一优秀的分治结构，解决一些可能 cdq 分治不太能解决的问题。

线段树分治最适用于解决一类修改影响的是一段询问区间的题目。

浅谈线段树分治

一般我们是以操作时间为下标建立线段树，将一段操作的区间放到线段树上的若干连续区间。

浅谈线段树分治

一般我们是以操作时间为下标建立线段树，将一段操作的区间放到线段树上的若干连续区间。

统计答案我们一般遍历线段树，将遍历到的节点加入进来。一般遍历到叶子就得到了和该询问相关的所有操作。

FZOJ1841 IHHH

已知 n 个同学编号是 $1 \sim n$，给出他们的名字。老师进行 q 次询问，每次询问对于一个字符串 a，编号从 l 到 r 的同学的名字是 a 的子串的名字最长是多少。

$n, q \leq 10^5$，询问串总长度和名字总长度 $\leq 5 \cdot 10^5$。

Solution

将所有的询问区间 $[l, r]$ 放到线段树上。

Solution

将所有的询问区间 $[l, r]$ 放到线段树上。

遍历线段树，将线段树节点区间对应所有字符串建立AC自动机，将询问点放进去跑匹配。

Solution

将所有的询问区间 $[l, r]$ 放到线段树上。

遍历线段树，将线段树节点区间对应所有字符串建立AC自动机，将询问点放进去跑匹配。

每个询问所能得到的匹配的最大长度即为询问答案。

Solution

将所有的询问区间 $[l, r]$ 放到线段树上。

遍历线段树，将线段树节点区间对应所有字符串建立AC自动机，将询问点放进去跑匹配。

每个询问所能得到的匹配的最大长度即为询问答案。

实现中清空AC自动机的细节也需要注意。

FZOJ3371 平行宇宙初探Ⅳ

（离线动态图连通性）

你要维护一张无向简单图。你被要求加入、删除一条边及查询两个点是否连通。

节点数 ≤ 5000 ，询问数 $\leq 5 \cdot 10^5$ 。

Solution

对于每个加边维护其时间区间，删边就修改其结束时间。

Solution

对于每个加边维护其时间区间，删边就修改其结束时间。

将每个时间区间投到线段树上。

Solution

对于每个加边维护其时间区间，删边就修改其结束时间。

将每个时间区间投到线段树上。

遍历线段树，将遍历到的边用并查集维护。

Solution

对于每个加边维护其时间区间，删边就修改其结束时间。

将每个时间区间投到线段树上。

遍历线段树，将遍历到的边用并查集维护。

这题还要撤销加进来的边，所以并查集要支持撤销。（用栈维护操作即可）

「HAOI2017」八纵八横

给你一个 n 个点 m 条边的无向连通图。每条边有一个边权，以不超过 1000 位的二进制数形式给出。给出 P 个操作。操作类型如下：

- Add x y z：添加一条 $x \Leftrightarrow y$ 的无向边，权值为 z 。
- Cancel k：删除第 k 条加入的无向边。
- Change k z：将加入的第 k 条无向边边权修改为 z 。

每进行一次操作后，你需要找一条从 1 点出发经过若干条边回到 1 号点的路径，最大化经过的边的权值的异或和。

$n, m \leq 500, q \leq 1000$.

pre-Solution

从 1 开始的任意路径的异或值就等价于从 1 开始的一条树链异或上一些非树边对应的环。

pre-Solution

从 1 开始的任意路径的异或值就等价于从 1 开始的一条树链异或上一些非树边对应的环。

那么这是从 1 出发再回来，就是若干环的异或最大值。

pre-Solution

从 1 开始的任意路径的异或值就等价于从 1 开始的一条树链异或上一些非树边对应的环。

那么这是从 1 出发再回来，就是若干环的异或最大值。

相信大家都会线性基!

pre-Solution

从 1 开始的任意路径的异或值就等价于从 1 开始的一条树链异或上一些非树边对应的环。

那么这是从 1 出发再回来，就是若干环的异或最大值。

相信大家都会线性基!

如果有人~~不会~~稍有遗忘，那我们也没时间讲线性基，你只需要知道线性基可以 $\log w$ 维护异或最大值就可以了。

pre-Solution

从 1 开始的任意路径的异或值就等价于从 1 开始的一条树链异或上一些非树边对应的环。

那么这是从 1 出发再回来，就是若干环的异或最大值。

相信大家都会线性基!

如果有人~~不会~~稍有遗忘，那我们也没时间讲线性基，你只需要知道线性基可以 $\log w$ 维护异或最大值就可以了。

当然我们都知道，线性基好像不太能做撤销操作。

Solution

我们先把生成树建出来，找出所有环边及其权值。因为我们有了环边就有了环的异或和。

Solution

我们先把生成树建出来，找出所有环边及其权值。因为我们有了环边就有了环的异或和。

那么每次操作相当于：加一条环边，删一条环边，修改一条环边的权值。

Solution

我们先把生成树建出来，找出所有环边及其权值。因为我们有了环边就有了环的异或和。

那么每次操作相当于：加一条环边，删一条环边，修改一条环边的权值。

和上题套路一样，对于每个加边维护其时间区间，删边就修改其结束时间，修改=删除+加入。

Solution

我们先把生成树建出来，找出所有环边及其权值。因为我们有了环边就有了环的异或和。

那么每次操作相当于：加一条环边，删一条环边，修改一条环边的权值。

和上题套路一样，对于每个加边维护其时间区间，删边就修改其结束时间，修改=删除+加入。

然后把这些区间投到线段树上。

Solution

我们先把生成树建出来，找出所有环边及其权值。因为我们有了环边就有了环的异或和。

那么每次操作相当于：加一条环边，删一条环边，修改一条环边的权值。

和上题套路一样，对于每个加边维护其时间区间，删边就修改其结束时间，修改=删除+加入。

然后把这些区间投到线段树上。

遍历线段树，线段树叶子节点的答案即为叶子到根路径所有环的线性基异或和最大值。

Solution

我们先把生成树建出来，找出所有环边及其权值。因为我们有了环边就有了环的异或和。

那么每次操作相当于：加一条环边，删一条环边，修改一条环边的权值。

和上题套路一样，对于每个加边维护其时间区间，删边就修改其结束时间，修改=删除+加入。

然后把这些区间投到线段树上。

遍历线段树，线段树叶子节点的答案即为叶子到根路径所有环的线性基异或和最大值。

细节：有 1000 位怎么办？用 bitset 就好了。

BZOJ3237 连通图

给定一个无向连通图和若干个小集合，每个小集合包含一些边，对于每个集合，你需要确定将集合中的边删掉后改图是否保持联通。集合间的询问相互独立。

节点数、集合数 $\leq 10^5$ ，边数 $\leq 2 \cdot 10^5$ ，每个集合大小不超过 4 .

Solution

这题最简单的分治方法，既不是 cdq 分治也不是线段树分治，但有类似之处。

Solution

这题最简单的分治方法，既不是 cdq 分治也不是线段树分治，但有类似之处。

离线，变删除为加入。

Solution

这题最简单的分治方法，既不是 cdq 分治也不是线段树分治，但有类似之处。

离线，变删除为加入。

我们分治区间 $[l, r]$，处理左半区间，我们将右半区间中左半区间没有的边连上，递归左半区间。右半区间同理。

Solution

这题最简单的分治方法，既不是 cdq 分治也不是线段树分治，但有类似之处。

离线，变删除为加入。

我们分治区间 $[l, r]$，处理左半区间，我们将右半区间中左半区间没有的边连上，递归左半区间。右半区间同理。

然后就做完了……维护还是用可撤销并查集。

Solution

这题最简单的分治方法，既不是 cdq 分治也不是线段树分治，但有类似之处。

离线，变删除为加入。

我们分治区间 $[l, r]$，处理左半区间，我们将右半区间中左半区间没有的边连上，递归左半区间。右半区间同理。

然后就做完了……维护还是用可撤销并查集。

这也算是一类经典分治方法，不过还没有名字……

Solution

这题最简单的分治方法，既不是 cdq 分治也不是线段树分治，但有类似之处。

离线，变删除为加入。

我们分治区间 $[l,r]$，处理左半区间，我们将右半区间中左半区间没有的边连上，递归左半区间。右半区间同理。

然后就做完了……维护还是用可撤销并查集。

这也算是一类经典分治方法，不过还没有名字……

本题加强版：FZOJ3370 平行宇宙初探 III。

Thanks!