基础算法水题选讲

GongChen

雅礼中学

August 12, 2019

写在前面

这份课件旨在以题目的形式回顾与贪心，倍增，二分，分治有关的知识点。

主要是普及组别的简单技巧巩固吧。

因为作者水平有限，接下来你会见到很多道千百年前切过的原题。

而且在不可抗因素的干扰下，即使最基础的东西，所述也远称不上完善。

不过如你所见，它已经是我能做到的最好了。

愿食用愉快！

贪心

- 经典问题
- 反悔型贪心
- 倒着考虑
- 涉及位运算
- 枚举一部分
- 对数列排序
- 其它

流水作业调度问题

有 n 个工件需要加工，其中第 i 个工件需要先花费 $a(i)$ 的时间在机器 1 上加工，再花费 $b(i)$ 的时间在机器 2 上加工. 你需要安排加工顺序使得从第 1 个工件开始加工到第 n 个工件结束加工的时间跨度最短.

流水作业调度问题

有 n 个工件需要加工，其中第 i 个工件需要先花费 $a(i)$ 的时间在机器 1 上加工，再花费 $b(i)$ 的时间在机器 2 上加工. 你需要安排加工顺序使得从第 1 个工件开始加工到第 n 个工件结束加工的时间跨度最短.

将 $a < b$ 的工件归为 $g1$ 类，$a \geq b$ 的工件归为 $g2$ 类. $g1$ 内按 a 非降排序，$g2$ 内按 b 非增排序. 最后把 $g2$ 接到 $g1$ 后面即可. 非常好感性理解，理性证明见 $Johnson$ 法则.

Huffman 树

构造一棵含 n 个叶子结点的 k 叉树，其中第 i 个叶子结点权值 $w(i)$，要求最小化 $\sum w(i) \times d(i)$，$d(i)$ 表示 i 结点的深度.

Huffman 树

构造一棵含 n 个叶子结点的 k 叉树, 其中第 i 个叶子结点权值 $w(i)$, 要求最小化 $\sum w(i) \times d(i)$, $d(i)$ 表示 i 结点的深度.

增加一些叶子结点为 0 的结点, 使得 $(k-1)|(n-1)$. 将 n 个 w 丢进小根堆, 每次取前 k 小的 w, 和为 s, 建立权值为 s 的树节点 p, 令 p 成为这 k 个结点的父亲, 并把 s 重新丢入堆.

CF867E Buy Low Sell High

你预言了每天的股票价格 v_i，从第 1 天开始，你每天可以选择卖一支股票，买一支股票，或者什么也不做。问直到第 n 天结束你最多可以获得多少收益。

$n \leq 3 \times 10^5$

CF867E Buy Low Sell High

提示：

如何在从前往后扫的同时进行抉择？什么情况下会买股票？有没有可能不优？怎么反悔？

CF867E Buy Low Sell High

我们可以把每天的价格抽象为一个保留”选项”丢到小根堆里，从堆里拿出元素就表示在对应的日期进行买股票的操作。

从前往后扫。若扫到第 i 天时，当前堆中存在元素 $v_j < v_i$，那么在第 j 天肯定得买股票。

CF867E Buy Low Sell High

我们可以把每天的价格抽象为一个保留”选项”丢到小根堆里，从堆里拿出元素就表示在对应的日期进行买股票的操作。

从前往后扫。若扫到第 i 天时，当前堆中存在元素 $v_j < v_i$，那么在第 j 天肯定得买股票。

唯一的干扰因素在于，我们不知道应不应该在第 i 天卖。

CF867E Buy Low Sell High

我们可以把每天的价格抽象为一个保留”选项”丢到小根堆里，从堆里拿出元素就表示在对应的日期进行买股票的操作。

从前往后扫。若扫到第 i 天时，当前堆中存在元素 $v_j < v_i$，那么在第 j 天肯定得买股票。

唯一的干扰因素在于，我们不知道应不应该在第 i 天卖。

更具体地说，如果把原价为 v_j 的股票在第 i 天卖出，但存在 k 满足 $k > i$ 且 $v_k > v_i$，那么这个方案就很有可能是不够优秀的。

CF867E Buy Low Sell High

我们可以把每天的价格抽象为一个保留”选项”丢到小根堆里，从堆里拿出元素就表示在对应的日期进行买股票的操作。

从前往后扫。若扫到第 i 天时，当前堆中存在元素 $v_j < v_i$，那么在第 j 天肯定得买股票。

唯一的干扰因素在于，我们不知道应不应该在第 i 天卖。

更具体地说，如果把原价为 v_j 的股票在第 i 天卖出，但存在 k 满足 $k > i$ 且 $v_k > v_i$，那么这个方案就很有可能是不够优秀的。

而且，收益最大的方案也可能要求了在第 i 天按 v_i 的价格买下一支股票。

CF867E Buy Low Sell High

实际上我们还是可以直接把 $v_i - v_j$ 的贡献加到 ans 。

CF867E Buy Low Sell High

实际上我们还是可以直接把 $v_i - v_j$ 的贡献加到 ans 。

不过为了消除后效性对当前决策的影响，我们还需要再丢一个 v_i 到堆里。

CF867E Buy Low Sell High

实际上我们还是可以直接把 $v_i - v_j$ 的贡献加到 ans 。

不过为了消除后效性对当前决策的影响，我们还需要再丢一个 v_i 到堆里。

这时堆中就有了两个 v_i 的选项。第 2 个 v_i 不再意味着" 购买"，而是" 反悔"。

CF867E Buy Low Sell High

实际上我们还是可以直接把 $v_i - v_j$ 的贡献加到 ans 。

不过为了消除后效性对当前决策的影响，我们还需要再丢一个 v_i 到堆里。

这时堆中就有了两个 v_i 的选项。第 2 个 v_i 不再意味着" 购买"，而是" 反悔"。

当之后从堆顶取出其中一个并售出时，我们理解为是回溯到了第 i 天，把当时卖出的那支股票按 v_i 的价格买回来。换句话说，取消第 i 天卖股票的决策，推迟到现在售卖。

CF867E Buy Low Sell High

实际上我们还是可以直接把 $v_i - v_j$ 的贡献加到 ans 。

不过为了消除后效性对当前决策的影响，我们还需要再丢一个 v_i 到堆里。

这时堆中就有了两个 v_i 的选项。第 2 个 v_i 不再意味着" 购买"，而是" 反悔"。

当之后从堆顶取出其中一个并售出时，我们理解为是回溯到了第 i 天，把当时卖出的那支股票按 v_i 的价格买回来。换句话说，取消第 i 天卖股票的决策，推迟到现在售卖。

而第二次取出就是普通的购买了。

CF3D Least Cost Bracket Sequence

给出一个由 ( ) ? 组成的序列，第 i 个 ? 可以花费 a_i 的代价变成 (，花费 b_i 的代价变成)，问将其变为合法括号序列的最小代价。

$n \leq 10^6$

CF3D Least Cost Bracket Sequence

提示：

使得在某一位放）合法的条件是？

CF3D Least Cost Bracket Sequence

一个）合法当且仅当在它与一个（配对的同时前面的所有）都能配对相应的（。

CF3D Least Cost Bracket Sequence

一个）合法当且仅当在它与一个（配对的同时前面的所有）都能配对相应的（。

先把所有的？设为），并把换成左右括号所需代价的差值丢进堆里，然后从左往右扫，如果扫到某处发现（不够用，就从堆里取一个原是？的）出来改成（。

CF3D Least Cost Bracket Sequence

一个）合法当且仅当在它与一个（配对的同时前面的所有）都能配对相应的（。

先把所有的？设为），并把换成左右括号所需代价的差值丢进堆里，然后从左往右扫，如果扫到某处发现（不够用，就从堆里取一个原是？的）出来改成（。

堆空时无解。

BZOJ2151 种树

你要在首尾相连的 n 个位置中选择 m 个互不相邻的位置，其中选择第 i 个位置即可获得 v_i 的价值，最大化价值之和.

$n \leq 2 \times 10^5$

BZOJ2151 种树

提示：

v 中间大两边小的三个相邻位置，它们所有可能的选择情况有？

BZOJ2151 种树

对于任意三个相邻位置 $i-1, i, i+1$ 而言，若 $v_i > v_{i-1}$ 并且 $v_i > v_{i+1}$，最优解不可能单选 $i-1$ 或 $i+1$ 中的任意一个.

BZOJ2151 种树

对于任意三个相邻位置 $i-1, i, i+1$ 而言，若 $v_i > v_{i-1}$ 并且 $v_i > v_{i+1}$，最优解不可能单选 $i-1$ 或 $i+1$ 中的任意一个.

否则，若选 $i-1$ 又不选 $i+1$，换成选 i 必然不冲突而且会产生更大的价值.

BZOJ2151 种树

对于任意三个相邻位置 $i-1, i, i+1$ 而言，若 $v_i > v_{i-1}$ 并且 $v_i > v_{i+1}$，最优解不可能单选 $i-1$ 或 $i+1$ 中的任意一个.

否则，若选 $i-1$ 又不选 $i+1$，换成选 i 必然不冲突而且会产生更大的价值.

把所有 v 丢进堆里，每次取堆顶元素 i，然后更新左右指针 l_i, r_i，删除原来 l_i, r_i 指向的元素，并把 v_i 更新为 $v_{l_i} + v_{r_i} - v_i$ 并丢进堆里.

BZOJ2151 种树

对于任意三个相邻位置 $i-1, i, i+1$ 而言，若 $v_i > v_{i-1}$ 并且 $v_i > v_{i+1}$，最优解不可能单选 $i-1$ 或 $i+1$ 中的任意一个.

否则，若选 $i-1$ 又不选 $i+1$，换成选 i 必然不冲突而且会产生更大的价值.

把所有 v 丢进堆里，每次取堆顶元素 i，然后更新左右指针 l_i, r_i，删除原来 l_i, r_i 指向的元素，并把 v_i 更新为 $v_{l_i} + v_{r_i} - v_i$ 并丢进堆里.

这样，下次再取到 i 就相当于是反悔这一次的决策从选择 i 换成了选择 $i-1, i+1$，这同时也解释了为何要更新 l_i 和 r_i .

51nod1053 最大 M 子段和 V2

你有一个长度为 n 的序列。你需要从中划分出 m 个互不相交的子段，使得它们的和最大。子段允许为空。

$n, m \leq 5 \times 10^4$

51nod1053 最大 M 子段和 V2

提示：

假设没有 m 的限制？在此基础上应如何操作使得 m 的限制得到满足？

51nod1053 最大 M 子段和 V2

我们将序列拆成许多个交错的均为负 or 正的连续子段，并约定 $sum(i)$ 表示第 i 段的和。

51nod1053 最大 M 子段和 V2

我们将序列拆成许多个交错的均为负 or 正的连续子段，并约定 $sum(i)$ 表示第 i 段的和。

如果没有 m 的限制，最好的选法显然是把所有的正数段全部选中。

51nod1053 最大 M 子段和 V2

我们将序列拆成许多个交错的均为负 or 正的连续子段，并约定 $sum(i)$ 表示第 i 段的和。

如果没有 m 的限制，最好的选法显然是把所有的正数段全部选中。

在此基础上，考虑如何使选择的总段数减少到 m。

51nod1053 最大 M 子段和 V2

我们将序列拆成许多个交错的均为负 or 正的连续子段，并约定 $sum(i)$ 表示第 i 段的和。

如果没有 m 的限制，最好的选法显然是把所有的正数段全部选中。

在此基础上，考虑如何使选择的总段数减少到 m。

- 不取一个正数段 i 。

51nod1053 最大 M 子段和 V2

我们将序列拆成许多个交错的均为负 or 正的连续子段，并约定 $sum(i)$ 表示第 i 段的和。

如果没有 m 的限制，最好的选法显然是把所有的正数段全部选中。

在此基础上，考虑如何使选择的总段数减少到 m。

- 不取一个正数段 i。
- 再取一个负数段 i。

51nod1053 最大 M 子段和 V2

我们将序列拆成许多个交错的均为负 or 正的连续子段，并约定 $sum(i)$ 表示第 i 段的和。

如果没有 m 的限制，最好的选法显然是把所有的正数段全部选中。

在此基础上，考虑如何使选择的总段数减少到 m。

- 不取一个正数段 i。
- 再取一个负数段 i。

它们的共同点在于，都是付出 $|sum(i)|$ 的代价，将 3 个连续的段合并，使总段数减少 1。

51nod1053 最大 M 子段和 V2

所以我们可以把所有段丢进按 $|sum|$ 排序的小根堆，每次取出堆顶的子段 i，合并它和它的左右指针所指向的两段，直到总段数减少到 m 为止。

51nod1053 最大 M 子段和 V2

所以我们可以把所有段丢进按 $|sum|$ 排序的小根堆，每次取出堆顶的子段 i，合并它和它的左右指针所指向的两段，直到总段数减少到 m 为止。

这里所说的合并，具体而言就是修改 i 的 sum 和左右指针再丢回堆，使得 i 成为一个新的段。

51nod1053 最大 M 子段和 V2

所以我们可以把所有段丢进按 $|sum|$ 排序的小根堆，每次取出堆顶的子段 i，合并它和它的左右指针所指向的两段，直到总段数减少到 m 为止。

这里所说的合并，具体而言就是修改 i 的 sum 和左右指针再丢回堆，使得 i 成为一个新的段。

从堆顶取段 i 的时候注意判 i 是否已经被合并以及是否是位于两端的负数段。

NOI2017 蔬菜

你的仓库里有 n 种蔬菜，每天最多销售 m 个单位的蔬菜。

第 i 种蔬菜有 c_i 单位的库存，每天固定会有 x_i 个单位变质从而不能再用于销售。这种蔬菜的单位收益为 a_i，同时，对其进行的第一次销售会产生 s_i 的额外收益。

你想知道销售 p 天的最大收益。多组询问。

$n \leq 10^5, m \leq 10, p \leq 10^5, a, c, x, s \leq 10^9$

NOI2017 蔬菜

提示：

如果每种蔬菜仅有一个单位？尝试离线回答？

NOI2017 蔬菜

若每种蔬菜仅有一个单位，那显然就是”在限制内尽量靠后”的策略。

NOI2017 蔬菜

若每种蔬菜仅有一个单位，那显然就是"在限制内尽量靠后"的策略。

这里也倒过来考虑。这样蔬菜从每天减少 x 变成了每天增多 x 。

NOI2017 蔬菜

尝试离线处理多组询问。

NOI2017 蔬菜

尝试离线处理多组询问。

能在第 i 天售卖的蔬菜也必然能在第 $i-1$ 天售卖。

NOI2017 蔬菜

尝试离线处理多组询问。

能在第 i 天售卖的蔬菜也必然能在第 $i-1$ 天售卖。

于是乎如果能够知道 $[1,p]$ 天内售卖了哪些蔬菜，就可以通过取消售卖收益最小的 m 单位蔬菜来得到 $p-1$ 天的最优售卖方案。

NOI2017 蔬菜

尝试离线处理多组询问。

能在第 i 天售卖的蔬菜也必然能在第 $i-1$ 天售卖。

于是乎如果能够知道 $[1,p]$ 天内售卖了哪些蔬菜，就可以通过取消售卖收益最小的 m 单位蔬菜来得到 $p-1$ 天的最优售卖方案。

这样，问题就简化为了如何求 $[1,\max p]$ 的蔬菜售卖方案。

NOI2017 蔬菜

尝试离线处理多组询问。

能在第 i 天售卖的蔬菜也必然能在第 $i-1$ 天售卖。

于是乎如果能够知道 $[1,p]$ 天内售卖了哪些蔬菜，就可以通过取消售卖收益最小的 m 单位蔬菜来得到 $p-1$ 天的最优售卖方案。

这样，问题就简化为了如何求 $[1,\max p]$ 的蔬菜售卖方案。

预处理出每种蔬菜最终腐烂的日期，按收益的从多到少排序，每次将蔬菜放在尽量靠后的日期。

NOI2017 蔬菜

尝试离线处理多组询问。

能在第 i 天售卖的蔬菜也必然能在第 $i-1$ 天售卖。

于是乎如果能够知道 $[1,p]$ 天内售卖了哪些蔬菜，就可以通过取消售卖收益最小的 m 单位蔬菜来得到 $p-1$ 天的最优售卖方案。

这样，问题就简化为了如何求 $[1,\max p]$ 的蔬菜售卖方案。

预处理出每种蔬菜最终腐烂的日期，按收益的从多到少排序，每次将蔬菜放在尽量靠后的日期。

具体实现可以用并查集去维护，合并已经放满 m 单位蔬菜的相邻日期，从而快速找到能够售卖的日期。

CF967E Big Secret

对 n 个数排序使得它们的前缀异或和从 1 到 n 递增.

$n \leq 10^5, v \leq 2^{60}$

CF967E Big Secret

提示：

使得 $s \oplus k > s$ 的 k 会有什么性质？

CF967E Big Secret

所有满足 $s \oplus k > s$ 的 k 必然满足，在二进制表示下 k 不为 0 的最高位在 s 上为 0.

CF967E Big Secret

所有满足 $s \oplus k > s$ 的 k 必然满足，在二进制表示下 k 不为 0 的最高位在 s 上为 0.

于是把所有数按二进制最高位存起来，每次对当前前缀和从低到高 chk 每个为 0 的位是否是某个未选数最高位的落点.

CF967E Big Secret

所有满足 $s \oplus k > s$ 的 k 必然满足，在二进制表示下 k 不为 0 的最高位在 s 上为 0.

于是把所有数按二进制最高位存起来，每次对当前前缀和从低到高 chk 每个为 0 的位是否是某个未选数最高位的落点.

如果有，用它更新当前前缀和，再去填下一个数，重新找. 如果扫到最高位都没有，那就 -1.

CF967E Big Secret

所有满足 $s \oplus k > s$ 的 k 必然满足，在二进制表示下 k 不为 0 的最高位在 s 上为 0.

于是把所有数按二进制最高位存起来，每次对当前前缀和从低到高 chk 每个为 0 的位是否是某个未选数最高位的落点.

如果有，用它更新当前前缀和，再去填下一个数，重新找. 如果扫到最高位都没有，那就 -1.

至于为什么不是从高到低，可以感性理解那样最多只能放 60 个数.

CF1054D Changing Array

你有一个由 n 个 k 位二进制数组成的序列。允许多次对任意数异或上 2^k-1 。你要最大化异或和不为 0 的子串个数。

$n \leq 2 \times 10^5, k \leq 30$

CF1054D Changing Array

提示：

考虑将问题转化为前缀和相关？前缀和又会怎么变化？

CF1054D Changing Array

约定 s 表示异或前缀和。

补集转化，区间 $[l,r]$ 异或和为 0 ，等价于 $s_r = s_{l-1}$ 。

CF1054D Changing Array

约定 s 表示异或前缀和。

补集转化，区间 $[l,r]$ 异或和为 0 ，等价于 $s_r=s_{l-1}$ 。

无论对哪些数异或 2^k-1 ，最终对于 $i\in[1,n]$, s'_i 的取值只会是 s_i 或 $s_i\oplus(2^k-1)$ ，而且一定存在操作方案使其变成两者间任意一个。

CF1054D Changing Array

约定 s 表示异或前缀和。

补集转化，区间 $[l,r]$ 异或和为 0 ，等价于 $s_r = s_{l-1}$ 。

无论对哪些数异或 2^k-1 ，最终对于 $i \in [1,n]$, s'_i 的取值只会是 s_i 或 $s_i \oplus (2^k-1)$ ，而且一定存在操作方案使其变成两者间任意一个。

于是可以按 $min(s_i, s_i \oplus (2^k-1))$ 对所有 i 进行分类。

CF1054D Changing Array

约定 s 表示异或前缀和。

补集转化，区间 $[l,r]$ 异或和为 0 ，等价于 $s_r=s_{l-1}$ 。

无论对哪些数异或 2^k-1 ，最终对于 $i\in[1,n]$, s'_i 的取值只会是 s_i 或 $s_i\oplus(2^k-1)$ ，而且一定存在操作方案使其变成两者间任意一个。

于是可以按 $min(s_i,s_i\oplus(2^k-1))$ 对所有 i 进行分类。

位于不同类的 i,j 无论怎么异或都肯定不会出现 $s_i=s_j$ 的情况。

CF1054D Changing Array

约定 s 表示异或前缀和。

补集转化，区间 $[l,r]$ 异或和为 0 ，等价于 $s_r = s_{l-1}$ 。

无论对哪些数异或 2^k-1 ，最终对于 $i \in [1,n]$, s'_i 的取值只会是 s_i 或 $s_i \oplus (2^k-1)$ ，而且一定存在操作方案使其变成两者间任意一个。

于是可以按 $min(s_i, s_i \oplus (2^k-1))$ 对所有 i 进行分类。

位于不同类的 i,j 无论怎么异或都肯定不会出现 $s_i = s_j$ 的情况。

因此我们只需要对每一类单独考虑，应当将其中哪些元素的 s 变为 $s \oplus (2^k-1)$ 才能使得类内 s 两两相等的元素对数量最少即可。

CF1054D Changing Array

位于同一类的元素在 $s \neq 0$ 的情况下可视为无差别。因此对这些类而言最优操作显然是将类内 $1/2$ 的 s 异或上 $2^k - 1$。

CF1054D Changing Array

位于同一类的元素在 $s \neq 0$ 的情况下可视为无差别。因此对这些类而言最优操作显然是将类内 $1/2$ 的 s 异或上 $2^k - 1$。

但在 $s = 0$ 的类中，需要注意到 s_0 不可被转化。

CF1054D Changing Array

位于同一类的元素在 $s \neq 0$ 的情况下可视为无差别。因此对这些类而言最优操作显然是将类内 $1/2$ 的 s 异或上 $2^k - 1$。

但在 $s = 0$ 的类中，需要注意到 s_0 不可被转化。

不过考虑完就会发现它对计算并不造成影响。

CF1054D Changing Array

位于同一类的元素在 $s \neq 0$ 的情况下可视为无差别。因此对这些类而言最优操作显然是将类内 $1/2$ 的 s 异或上 $2^k - 1$。

但在 $s = 0$ 的类中，需要注意到 s_0 不可被转化。

不过考虑完就会发现它对计算并不造成影响。

总之，统计出每一类的 $size$ ，然后像这样计算，拿总区间数减一减就可以了。

51nod1302 矩形面积交

n 个矩形，你要把它们分成两组，使得两组最大重叠面积之和最大。矩形可旋转，可移动。

$n \leq 10^5$

51nod1302 矩形面积交

提示：

真正影响答案的变量是哪几个？其中的一些是否能被固定？

51nod1302 矩形面积交

会影响答案的变量有 4 个：A 组最短短边 Ax，最短长边 Ay，B 组最短短边 Bx，最短长边 By.

51nod1302 矩形面积交

会影响答案的变量有 4 个：A 组最短短边 Ax，最短长边 Ay，B 组最短短边 Bx，最短长边 By.

先考虑短边的话，总有一个短边是最短的，那么把它放到 A 组之后，Ax 就固定了. 再去枚举 By，这个时候 Ay 也就确定了.

51nod1302 矩形面积交

会影响答案的变量有 4 个：A 组最短短边 Ax，最短长边 Ay，B 组最短短边 Bx，最短长边 By.

先考虑短边的话，总有一个短边是最短的，那么把它放到 A 组之后，Ax 就固定了. 再去枚举 By，这个时候 Ay 也就确定了.

但是长边长度在 By 及以上的可能有不止 n 个矩形. 这时把短边最短的那几个矩形也分配给 A，从而得到 Ax, Ay, By 固定前提下最优的 Bx. 最终答案对按所有枚举的 By 求出的最大面积取 max.

51nod1302 矩形面积交

会影响答案的变量有 4 个：A 组最短短边 Ax，最短长边 Ay，B 组最短短边 Bx，最短长边 By.

先考虑短边的话，总有一个短边是最短的，那么把它放到 A 组之后，Ax 就固定了. 再去枚举 By，这个时候 Ay 也就确定了.

但是长边长度在 By 及以上的可能有不止 n 个矩形. 这时把短边最短的那几个矩形也分配给 A，从而得到 Ax, Ay, By 固定前提下最优的 Bx. 最终答案对按所有枚举的 By 求出的最大面积取 max.

具体实现用优先队列来维护.

算法框架

这类题是让你对一个序列排序从而最大/最小化某个式子的值。值跟元素顺序有关。

算法框架

这类题是让你对一个序列排序从而最大/最小化某个式子的值。值跟元素顺序有关。

入手点是假设已经排好了前 $n-2$ 个元素的位置，考虑如何安排 i 和 j 的相对顺序。

算法框架

这类题是让你对一个序列排序从而最大/最小化某个式子的值。值跟元素顺序有关。

入手点是假设已经排好了前 $n-2$ 个元素的位置，考虑如何安排 i 和 j 的相对顺序。

直接令 i 放在 j 前更优然后推式子推出比较函数。

算法框架

这类题是让你对一个序列排序从而最大/最小化某个式子的值。值跟元素顺序有关。

入手点是假设已经排好了前 $n-2$ 个元素的位置，考虑如何安排 i 和 j 的相对顺序。

直接令 i 放在 j 前更优然后推式子推出比较函数。

重点是比较函数必须具有传递性，也就是若 i 放在 j 前更优，j 放在 k 前更优，i 放在 k 前必然更优

luogu2123 皇后游戏

给定 n 个元素, 其中元素 i 有属性 $a(i), b(i)$. 你需要通过对元素排序最小化

$$max_{i=1}^{n} c_i, c_i = max(c_{i-1}, \sum_{j=1}^{i} a_j) + b_i$$

$n \leq 2 \times 10^4, a, b \leq 10^9$

luogu2123 皇后游戏

提示：

化简式子时可能会需要以下几个手段：

1. 两个 max 套在一起时，把里面那个 max 拆开.
2. 尝试把 max 内相同的元素提出来，移项相消.
3. $max(a_j, b_i) - a_j - b_i$ 可以化简

luogu2123 皇后游戏

列式子，假设 i 放在 j 前更优，最终得到 $min(b_i, a_j) > min(a_i, b_j)$.

luogu2123 皇后游戏

列式子，假设 i 放在 j 前更优，最终得到 $min(b_i, a_j) > min(a_i, b_j)$.

这个比较条件是否具有传递性很难看出。不过事实上是有的，只是不能够直接写到 $sort$ 里面. ouuan 的博客对此进行了详细分析，感兴趣的话可以看一看.

另一个更加简单粗暴的办法是分类讨论.

1. 对于 $a > b$ 的元素，按 b 从大到小排序
2. 对于 $a = b$ 的元素，随便排序
3. 对于 $a < b$ 的元素，按 a 从小到大排序

luogu2123 皇后游戏

列式子，假设 i 放在 j 前更优，最终得到 $min(b_i, a_j) > min(a_i, b_j)$.

这个比较条件是否具有传递性很难看出。不过事实上是有的，只是不能够直接写到 $sort$ 里面. ouuan 的博客对此进行了详细分析，感兴趣的话可以看一看.

另一个更加简单粗暴的办法是分类讨论.

1. 对于 $a > b$ 的元素，按 b 从大到小排序

2. 对于 $a = b$ 的元素，随便排序

3. 对于 $a < b$ 的元素，按 a 从小到大排序

而第 1 类也可以是 $a \geq b$，第 3 类也可以是 $a \leq b$，所以这三大类之间可以按 123 的顺序放.

51nod1164 最高的奖励 V2

你有 n 个任务可以接。每个任务需要 1 个单位的时间。第 i 个任务可产生 v_i 的价值，但必须在 $[l_i, r_i]$ 的时间段里去做。你要最大化总价值。

$n \leq 5000$

51nod1164 最高的奖励 V2

提示：

强行为下午 EndSaH 的图论课堂做铺垫。

51nod1164 最高的奖励 V2

正解是二分图匹配。

51nod1164 最高的奖励 V2

正解是二分图匹配。

先把所有区间按左端点排序，而不管右端点。

51nod1164 最高的奖励 V2

正解是二分图匹配。

先把所有区间按左端点排序，而不管右端点。

存储从每个区间的左端点开始不断往后走所到达的最早的一个没有被存储过的时间点。

51nod1164 最高的奖励 V2

正解是二分图匹配。

先把所有区间按左端点排序，而不管右端点。

存储从每个区间的左端点开始不断往后走所到达的最早的一个没有被存储过的时间点。

这样我们会得到 n 个时间点，性质是只有它们最终有可能会被安排上工作。

51nod1164 最高的奖励 V2

正解是二分图匹配。

先把所有区间按左端点排序，而不管右端点。

存储从每个区间的左端点开始不断往后走所到达的最早的一个没有被存储过的时间点。

这样我们会得到 n 个时间点，性质是只有它们最终有可能会被安排上工作。

然后再把区间按奖励从大到小排序，每个区间尝试去匹配它的左端点。

51nod1164 最高的奖励 V2

正解是二分图匹配。

先把所有区间按左端点排序，而不管右端点。

存储从每个区间的左端点开始不断往后走所到达的最早的一个没有被存储过的时间点。

这样我们会得到 n 个时间点，性质是只有它们最终有可能会被安排上工作。

然后再把区间按奖励从大到小排序，每个区间尝试去匹配它的左端点。

如果时间点已经被匹配，让两个区间中右端点更靠右的一个离开去尝试匹配下一个时间点。

51nod1164 最高的奖励 V2

正解是二分图匹配。

先把所有区间按左端点排序，而不管右端点。

存储从每个区间的左端点开始不断往后走所到达的最早的一个没有被存储过的时间点。

这样我们会得到 n 个时间点，性质是只有它们最终有可能会被安排上工作。

然后再把区间按奖励从大到小排序，每个区间尝试去匹配它的左端点。

如果时间点已经被匹配，让两个区间中右端点更靠右的一个离开去尝试匹配下一个时间点。

如果当前时间点已经小于所匹配区间的右端点，或者没有下一个时间点了，就说明匹配失败。

倍增

- 区间覆盖问题
- 预处理距离

SCOI2015 国旗计划

模板题。

给出一个大小为 m 的环和环上的 n 个区间。每个区间覆盖了序列的一部分，保证互不包含，首尾落在整点。对于每一个区间，你要求出在它必须被选中的前提下，至少需要多少个区间才能完全覆盖整个序列。考虑非整点。

$n \leq 2 \times 10^5, m \leq 10^9$

SCOI2015 国旗计划

先把环拆成序列，坐标轴长度翻倍，士兵数量翻倍。

SCOI2015 国旗计划

先把环拆成序列，坐标轴长度翻倍，士兵数量翻倍。

从区间没有交集可以推出，一个士兵的起点坐标越大，终点坐标也就越大。

SCOI2015 国旗计划

先把环拆成序列，坐标轴长度翻倍，士兵数量翻倍。

从区间没有交集可以推出，一个士兵的起点坐标越大，终点坐标也就越大。

于是可以把所有士兵按起点坐标排序，再一遍 $O(n)$ 对每个士兵求出与其区间交最小的士兵，也即它的最优后继。

SCOI2015 国旗计划

先把环拆成序列，坐标轴长度翻倍，士兵数量翻倍。

从区间没有交集可以推出，一个士兵的起点坐标越大，终点坐标也就越大。

于是可以把所有士兵按起点坐标排序，再一遍 $O(n)$ 对每个士兵求出与其区间交最小的士兵，也即它的最优后继。

用 $ne(i,j)$ 表示 i 的第 2^j 后继，每次 $O(\log n)$ 求走完一圈回到每个士兵的最小代价。

SCOI2016 萌萌哒

一个长度为 n 的大数，每次告诉你两个长度相同的区间数字完全对应相同，问可能的数的个数。

$n \leq 10^5$

SCOI2016 萌萌哒

提示：

暴力怎么做？如何用倍增优化？

SCOI2016 萌萌哒

暴力想法是每次对两区间的对应点合并并查集，答案即为 $10^{cnt-1} \times 9$ ，因为最高位不为 0 。

SCOI2016 萌萌哒

暴力想法是每次对两区间的对应点合并并查集，答案即为 $10^{cnt-1} \times 9$ ，因为最高位不为 0 。

考虑用倍增优化合并的过程，每次合并一整个区间。

SCOI2016 萌萌哒

暴力想法是每次对两区间的对应点合并并查集，答案即为 $10^{cnt-1} \times 9$ ，因为最高位不为 0 。

考虑用倍增优化合并的过程，每次合并一整个区间。

具体而言，对每一个 i 幂次的区间长度建立倍增数组 $f(i,j)$ ，分别记录区间 $[j, j+2^i-1]$ 所属的并查集。

SCOI2016 萌萌哒

暴力想法是每次对两区间的对应点合并并查集，答案即为 $10^{cnt-1} \times 9$ ，因为最高位不为 0 。

考虑用倍增优化合并的过程，每次合并一整个区间。

具体而言，对每一个 i 幂次的区间长度建立倍增数组 $f(i,j)$ ，分别记录区间 $[j, j+2^i-1]$ 所属的并查集。

每次合并就像 ST 表那样将区间首尾分开合，最后再从大到小把 2^i 的信息分开成 2 段下传到 2^{i-1} ，就能通过查询 2^0 长度，也就是单点，得到总共的并查集个数。

PKUSC2018 星际穿越

坐标轴上有 n 个星球，第 i 个星球坐标为 i ，与星球 $[l_i, i-1]$ 之间都有双向通道。其中，经过每个通道都需要 1 的单位时间。

定义 $dist(x, y)$ 为从星球 x 出发到达星球 y 的最短时间。求 $\sum_{i=l}^{r} dist(x, i)$ 。

多组询问，保证 $l < r < x$ 。

$n, q \leq 3 \times 10^5$

PKUSC2018 星际穿越

提示：

$i \to j(j < i)$ 的最短路都有什么共性？要求的式子如何化成能够利用倍增的形式？

PKUSC2018 星际穿越

分析性质可以发现 $i \to j (j < i)$ 的最短路一定是一路向左跳或者向右跳 1 步再一路向左跳。

PKUSC2018 星际穿越

分析性质可以发现 $i \to j (j < i)$ 的最短路一定是一路向左跳或者向右跳 1 步再一路向左跳。

因为不可能向右跳多步，也不可能向左跳之后再向右跳。

PKUSC2018 星际穿越

我们把 $\sum_{y=l}^{r} dist(x,y)$ 拆成 $\sum_{y=l}^{x} dist(x,y) - \sum_{y=r+1}^{x} dist(x,y)$ 。

PKUSC2018 星际穿越

我们把 $\sum_{y=l}^{r} dist(x,y)$ 拆成 $\sum_{y=l}^{x} dist(x,y) - \sum_{y=r+1}^{x} dist(x,y)$ 。

$dist(x,y)=1$ ，当且仅当 $y \in [l_x, x]$ 。

PKUSC2018 星际穿越

我们把 $\sum_{y=l}^{r} dist(x,y)$ 拆成 $\sum_{y=l}^{x} dist(x,y) - \sum_{y=r+1}^{x} dist(x,y)$ 。

$dist(x,y)=1$ ，当且仅当 $y \in [l_x, x]$ 。

在考虑 $dist(x,y) \leq 2$ ，这时它可以选择先往右跳一步再往左跳一步。

PKUSC2018 星际穿越

我们把 $\sum_{y=l}^{r} dist(x,y)$ 拆成 $\sum_{y=l}^{x} dist(x,y) - \sum_{y=r+1}^{x} dist(x,y)$ 。

$dist(x,y)=1$ ，当且仅当 $y \in [l_x, x]$ 。

在考虑 $dist(x,y) \leq 2$ ，这时它可以选择先往右跳一步再往左跳一步。

于是必须存在一个 $[l_x, x]$ 中的点到 y 的 $dist \leq 1$ 。

PKUSC2018 星际穿越

我们把 $\sum_{y=l}^{r} dist(x,y)$ 拆成 $\sum_{y=l}^{x} dist(x,y) - \sum_{y=r+1}^{x} dist(x,y)$ 。

$dist(x,y)=1$ ，当且仅当 $y \in [l_x, x]$ 。

在考虑 $dist(x,y) \le 2$ ，这时它可以选择先往右跳一步再往左跳一步。

于是必须存在一个 $[l_x, x]$ 中的点到 y 的 $dist \le 1$ 。

推广得到，$dist(x,y) \le k, k \ge 2$ 的条件是，存在 $i \in [l_x, n]$ 满足 $dist(y,i) \le k-1$ 。

PKUSC2018 星际穿越

我们把 $\sum_{y=l}^{r} dist(x,y)$ 拆成 $\sum_{y=l}^{x} dist(x,y) - \sum_{y=r+1}^{x} dist(x,y)$ 。

$dist(x,y)=1$ ，当且仅当 $y \in [l_x, x]$ 。

在考虑 $dist(x,y) \le 2$ ，这时它可以选择先往右跳一步再往左跳一步。

于是必须存在一个 $[l_x, x]$ 中的点到 y 的 $dist \le 1$ 。

推广得到，$dist(x,y) \le k, k \ge 2$ 的条件是，存在 $i \in [l_x, n]$ 满足 $dist(y,i) \le k-1$ 。

PKUSC2018 星际穿越

推广得到，$dist(x,y) \leq k, k \geq 2$ 的条件是，存在 $i \in [l_x, n]$ 满足 $dist(y,i) \leq k-1$ 。

PKUSC2018 星际穿越

推广得到，$dist(x,y) \leq k, k \geq 2$ 的条件是，存在 $i \in [l_x, n]$ 满足 $dist(y,i) \leq k-1$ 。

我们维护一个 $t(i,j)$ 数组，记录从 $[i,n]$ 出发跳 j 步所能到达的最左点。

PKUSC2018 星际穿越

推广得到，$dist(x,y) \le k, k \ge 2$ 的条件是，存在 $i \in [l_x, n]$ 满足 $dist(y,i) \le k-1$ 。

我们维护一个 $t(i,j)$ 数组，记录从 $[i,n]$ 出发跳 j 步所能到达的最左点。

那么，求 $\sum_{y=l}^{x} dis(x,y)$ 的时候先特判一下 $dist=1$ 的情况，再一步跳到 l_x ，就能通过 $t(i,j)$ 求出答案了。

PKUSC2018 星际穿越

然而不幸的是，预处理 t 数组需要 $O(n^2)$ 的时间复杂度。

PKUSC2018 星际穿越

然而不幸的是，预处理 t 数组需要 $O(n^2)$ 的时间复杂度。

所以考虑对 j 倍增化，得到：$t(i,j) = t(t(i,j-2),j-1)$

PKUSC2018 星际穿越

然而不幸的是，预处理 t 数组需要 $O(n^2)$ 的时间复杂度。

所以考虑对 j 倍增化，得到：$t(i,j) = t(t(i,j-2),j-1)$

同时因为做了倍增处理，我们还要预处理出 $s(i,j)$ ，表示 $\sum_{k=t(i,j)}^{i} dist(i,k)$ 。

PKUSC2018 星际穿越

然而不幸的是，预处理 t 数组需要 $O(n^2)$ 的时间复杂度。

所以考虑对 j 倍增化，得到：$t(i,j)=t(t(i,j-2),j-1)$

同时因为做了倍增处理，我们还要预处理出 $s(i,j)$ ，表示 $\sum_{k=t(i,j)}^{i} dist(i,k)$ 。

$$s(i,j)=s(i,j-1)+s(t(i,j-1),j-1)+(t(i,j-1)-t(i,j))\times 2^{j-1}$$

PKUSC2018 星际穿越

然而不幸的是，预处理 t 数组需要 $O(n^2)$ 的时间复杂度。

所以考虑对 j 倍增化，得到：$t(i,j) = t(t(i,j-2),j-1)$

同时因为做了倍增处理，我们还要预处理出 $s(i,j)$ ，表示 $\sum_{k=t(i,j)}^{i} dist(i,k)$ 。

$$s(i,j) = s(i,j-1) + s(t(i,j-1),j-1) + (t(i,j-1) - t(i,j)) \times 2^{j-1}$$

每次询问先大步再小步地往前跳即可。

二分

- 查找第 k 大
- 最大化最小
- 三分找单峰

BZOJ2653 Middle

模板题。

给定一个长为 n 的序列，每次限制合法区间左端点落在 $[a, b]$ ，右端点落在 $[c, d]$ ，问合法区间排序后的最大中位数。

强制在线。

$n\,q \leq 2 \times 10^4$

BZOJ2653 Middle

二分中位数，把大于它的赋值为 1 ，小则 -1 ，判定是否存在合法区间和大于 0 。

BZOJ2653 Middle

二分中位数，把大于它的赋值为 1 ，小则 -1 ，判定是否存在合法区间和大于 0 。

不能对每个询问都暴力赋值一次，所以用主席树辅助。

51nod1671 货物运输

在坐标轴上有 n 个城市，第 i 个城市坐标为 i ，从 i 到 $i+1$ 需要耗费 1 点时间，反之亦然。

现在有 m 个计划，第 i 个计划要从城市 x_i 走到 y_i 。

你可以选择任意两个城市建立双向传送门，使得在它们之间的移动耗时为 0 。

所有计划同时开始，你要最小化它们的最晚结束时间。

$n \leq 5 \times 10^5$

51nod1671 货物运输

提示：

拆式子。

51nod1671 货物运输

因为一切都是双向的，我们强制 $r > l$ 。

51nod1671 货物运输

因为一切都是双向的，我们强制 $r > l$ 。

考虑二分最长用时，检验 md 是否可取。

51nod1671 货物运输

因为一切都是双向的，我们强制 $r > l$ 。

考虑二分最长用时，检验 md 是否可取。

忽略 $l + md \geq r$ 的计划，对于 $l + md < r$ 的计划，传送门 x, y 必须要满足 $|l - x| + |r - y| \leq md$ 。

51nod1671 货物运输

因为一切都是双向的，我们强制 $r > l$ 。

考虑二分最长用时，检验 md 是否可取。

忽略 $l + md \geq r$ 的计划，对于 $l + md < r$ 的计划，传送门 x, y 必须要满足 $|l - x| + |r - y| \leq md$ 。

把这个式子拆开，可以得到 $l + r - md \leq x + y \leq l + r + md$ 和 $l - r - md \leq x - y \leq l - r + md$ 。

51nod1671 货物运输

因为一切都是双向的，我们强制 $r > l$ 。

考虑二分最长用时，检验 md 是否可取。

忽略 $l + md \geq r$ 的计划，对于 $l + md < r$ 的计划，传送门 x, y 必须要满足 $|l - x| + |r - y| \leq md$ 。

把这个式子拆开，可以得到 $l + r - md \leq x + y \leq l + r + md$ 和 $l - r - md \leq x - y \leq l - r + md$ 。

每次从 1 扫到 m，对 $x + y$ 和 $x - y$ 的上下界进行维护，最终若下界大于上界或者不存在整点则 md 不可取。

AHOI2014 宅男计划

外卖店有 n 种食物，第 i 种食物价格为 p_i ，保质期 s_i 天。点一次外卖要付 F 元的配送费，一次外卖可以点无限多份食物。保质期从点外卖的那天起开始生效。

你每天需要至少吃一份食物，但你只有 m 元。你想知道自己最多能活到第几天。

$n \leq 200, p, s, f, m \leq 10^{18}$

AHOI2014 宅男计划

提示：

感性认识一下什么变量关于什么变量的图象有可能会呈单峰？

AHOI2014 宅男计划

可以发现，如果点外卖的次数少，你必须要多买价格贵的食品；来的多了，你就可以把钱更多地花在便宜的食品上，但是要付更多的快递费。

AHOI2014 宅男计划

可以发现，如果点外卖的次数少，你必须要多买价格贵的食品；来的多了，你就可以把钱更多地花在便宜的食品上，但是要付更多的快递费。

于是我们感性理解存活天数与点外卖次数呈单峰。

AHOI2014 宅男计划

可以发现，如果点外卖的次数少，你必须要多买价格贵的食品；来的多了，你就可以把钱更多地花在便宜的食品上，但是要付更多的快递费。

于是我们感性理解存活天数与点外卖次数呈单峰。

先按价格排序，维护一个单调栈，贪心地把价格贵而且保质期短的食品筛掉。

AHOI2014 宅男计划

可以发现，如果点外卖的次数少，你必须要多买价格贵的食品；来的多了，你就可以把钱更多地花在便宜的食品上，但是要付更多的快递费。

于是我们感性理解存活天数与点外卖次数呈单峰。

先按价格排序，维护一个单调栈，贪心地把价格贵而且保质期短的食品筛掉。

对于确定的点外卖次数，必然是从单调栈内最便宜的食品买起。

AHOI2014 宅男计划

可以发现，如果点外卖的次数少，你必须要多买价格贵的食品；来的多了，你就可以把钱更多地花在便宜的食品上，但是要付更多的快递费。

于是我们感性理解存活天数与点外卖次数呈单峰。

先按价格排序，维护一个单调栈，贪心地把价格贵而且保质期短的食品筛掉。

对于确定的点外卖次数，必然是从单调栈内最便宜的食品买起。

除了最后一次，单次购买量应当恰好卡在每种食物的保质期。

AHOI2014 宅男计划

可以发现，如果点外卖的次数少，你必须要多买价格贵的食品；来的多了，你就可以把钱更多地花在便宜的食品上，但是要付更多的快递费。

于是我们感性理解存活天数与点外卖次数呈单峰。

先按价格排序，维护一个单调栈，贪心地把价格贵而且保质期短的食品筛掉。

对于确定的点外卖次数，必然是从单调栈内最便宜的食品买起。

除了最后一次，单次购买量应当恰好卡在每种食物的保质期。

一次性计算总购买量即可。有点类似于蔬菜那题的计算。

AHOI2014 宅男计划

可以发现，如果点外卖的次数少，你必须要多买价格贵的食品；来的多了，你就可以把钱更多地花在便宜的食品上，但是要付更多的快递费。

于是我们感性理解存活天数与点外卖次数呈单峰。

先按价格排序，维护一个单调栈，贪心地把价格贵而且保质期短的食品筛掉。

对于确定的点外卖次数，必然是从单调栈内最便宜的食品买起。

除了最后一次，单次购买量应当恰好卡在每种食物的保质期。

一次性计算总购买量即可。有点类似于蔬菜那题的计算。

总之三分找单峰，注意计算过程中可能会爆 $longlong$ 。

分治

- CDQ 分治
- 整体二分
- 线段树分治（不归我
- 分治优化 DP（也不归我

算法基本框架

以三维偏序问题为例。

算法基本框架

以三维偏序问题为例。

将元素按第一维排序，需要保证位于 i 的元素不会对 $[1, i-1]$ 造成影响。

算法基本框架

以三维偏序问题为例。

将元素按第一维排序，需要保证位于 i 的元素不会对 $[1, i-1]$ 造成影响。

进行分治，中心思想是在每一层只计算左区间对右区间的影响。

算法基本框架

以三维偏序问题为例。

将元素按第一维排序，需要保证位于 i 的元素不会对 $[1, i-1]$ 造成影响。

进行分治，中心思想是在每一层只计算左区间对右区间的影响。

递归处理 $[l, mid]$ 和 $[mid+1, r]$ 。

算法基本框架

以三维偏序问题为例。

将元素按第一维排序，需要保证位于 i 的元素不会对 $[1, i-1]$ 造成影响。

进行分治，中心思想是在每一层只计算左区间对右区间的影响。

递归处理 $[l, mid]$ 和 $[mid+1, r]$ 。

在每一层左右区间第一维的大小关系恒定，问题简化为二维偏序。

算法基本框架

以三维偏序问题为例。

将元素按第一维排序，需要保证位于 i 的元素不会对 $[1, i-1]$ 造成影响。

进行分治，中心思想是在每一层只计算左区间对右区间的影响。

递归处理 $[l, mid]$ 和 $[mid+1, r]$ 。

在每一层左右区间第一维的大小关系恒定，问题简化为二维偏序。

于是对其中一维排序，再用数据结构维护第三维的信息，完成计算。

算法基本框架

经典的三维偏序组合是时间，位置，大小。

算法基本框架

经典的三维偏序组合是时间，位置，大小。

因此，CDQ 分治也常用于将动态问题转化为静态问题。

算法基本框架

经典的三维偏序组合是时间，位置，大小。

因此，CDQ 分治也常用于将动态问题转化为静态问题。

具体而言就是以时间为第一维进行分治，在每一层仅计算左区间的修改操作对右区间的询问操作造成的影响。

算法基本框架

经典的三维偏序组合是时间，位置，大小。

因此，CDQ 分治也常用于将动态问题转化为静态问题。

具体而言就是以时间为第一维进行分治，在每一层仅计算左区间的修改操作对右区间的询问操作造成的影响。

因为每次计算时整个左区间的修改可以视作静态，所以会方便处理很多。

CDQ 分治复杂度的计算

介绍一些无脑手段。

CDQ 分治复杂度的计算

介绍一些无脑手段。

可以把分治的过程表现为线段树的形式，称每一个结点所表示的区间为一层，所有长度相同的区间位于一级上，一级总长 n ，总共有 $\log n$ 级。

CDQ 分治复杂度的计算

介绍一些无脑手段。

可以把分治的过程表现为线段树的形式，称每一个结点所表示的区间为一层，所有长度相同的区间位于一级上，一级总长 n ，总共有 $\log n$ 级。

如果每一层的时间复杂度是 $O($ 区间长度 $)$ 的，每一级就是 $O(n)$ ，总复杂度就是 $O(n \log n)$ 。

CDQ 分治复杂度的计算

介绍一些无脑手段。

可以把分治的过程表现为线段树的形式，称每一个结点所表示的区间为一层，所有长度相同的区间位于一级上，一级总长 n ，总共有 $\log n$ 级。

如果每一层的时间复杂度是 $O($ 区间长度 $)$ 的，每一级就是 $O(n)$ ，总复杂度就是 $O(n \log n)$ 。

如果每一层 $O(\log$ 区间长度 $)$，总复杂度为 $O(n \log^2 n)$ 。

CDQ 分治复杂度的计算

介绍一些无脑手段。

可以把分治的过程表现为线段树的形式，称每一个结点所表示的区间为一层，所有长度相同的区间位于一级上，一级总长 n ，总共有 $\log n$ 级。

如果每一层的时间复杂度是 $O($ 区间长度 $)$ 的，每一级就是 $O(n)$ ，总复杂度就是 $O(n\log n)$ 。

如果每一层 $O(\log$ 区间长度 $)$，总复杂度为 $O(n\log^2 n)$ 。

如果每一层 $O(n)$ ，总复杂度为 $O(n\sum_{i=1}^{\frac{n}{\log n}})$，约为 $O(n^2)$ 。

TJOI2016 序列

模板题。

给定一个长为 n 的序列，序列上某些值会发生改变，求最长子序列的长度，满足在任何一种改变的情况下它都不降。改变不同时发生。

$n \leq 10^5$

TJOI2016 序列

提示：

j 能转移到 i 的前提是？

TJOI2016 序列

j 能转移到 i 的前提是 $\max v_j \leq v_i$ 并且 $v_j \leq \min v_i$ 。

TJOI2016 序列

j 能转移到 i 的前提是 $\max v_j \leq v_i$ 并且 $v_j \leq \min v_i$ 。

再加上位置就是典型的三维偏序了，直接上 CDQ 。

TJOI2016 序列

j 能转移到 i 的前提是 $\max v_j \leq v_i$ 并且 $v_j \leq \min v_i$ 。

再加上位置就是典型的三维偏序了，直接上 CDQ 。

因为是 DP ，所以按照左中右的顺序处理区间。

TJOI2016 序列

j 能转移到 i 的前提是 $\max v_j \leq v_i$ 并且 $v_j \leq \min v_i$ 。

再加上位置就是典型的三维偏序了，直接上 CDQ 。

因为是 DP ，所以按照左中右的顺序处理区间。

复杂度是 $O(n \log^2 n)$

JOISC2014 稻草人

平面上有 n 个点，你要求出有多少个平行与坐标轴的矩形满足左下角和右上角各有一个点，并且内部没有点。

$n \leq 2 \times 10^5$

JOISC2014 稻草人

提示：

什么情况下两点形成的矩形合法？计算时最好能让什么静态？

JOISC2014 稻草人

CDQ 分治，按 y 排序，考虑对于确定的界线如何计算下区间对上区间的贡献。

JOISC2014 稻草人

CDQ 分治，按 y 排序，考虑对于确定的界线如何计算下区间对上区间的贡献。

将两区间内部的点各自按 x 升序排序。

JOISC2014 稻草人

CDQ 分治，按 y 排序，考虑对于确定的界线如何计算下区间对上区间的贡献。

将两区间内部的点各自按 x 升序排序。

对于上区间的点 i 而言，若 j 是上区间在 i 前面的点中满足 $y_j < y_i$ 且坐标最大的一个，能与 i 匹配的下区间的点必然要满足横坐标在 x_j 和 x_i 之间。

JOISC2014 稻草人

CDQ 分治，按 y 排序，考虑对于确定的界线如何计算下区间对上区间的贡献。

将两区间内部的点各自按 x 升序排序。

对于上区间的点 i 而言，若 j 是上区间在 i 前面的点中满足 $y_j < y_i$ 且坐标最大的一个，能与 i 匹配的下区间的点必然要满足横坐标在 x_j 和 x_i 之间。

除此之外，若下区间新增点 u 合法，它左下角的所有点都成了非法点。

JOISC2014 稻草人

CDQ 分治，按 y 排序，考虑对于确定的界线如何计算下区间对上区间的贡献。

将两区间内部的点各自按 x 升序排序。

对于上区间的点 i 而言，若 j 是上区间在 i 前面的点中满足 $y_j < y_i$ 且坐标最大的一个，能与 i 匹配的下区间的点必然要满足横坐标在 x_j 和 x_i 之间。

除此之外，若下区间新增点 u 合法，它左下角的所有点都成了非法点。

于是上区间维护上升单调栈，下区间维护下降单调栈，每次用上区间单调栈内的第二个点的横坐标在下区间的单调栈内二分查找大于它的点数即可。

算法基本框架

其实就是一次性对所有询问进行二分。

算法基本框架

其实就是一次性对所有询问进行二分。

将多组询问放在一起，每次查询每个询问的答案与 mid 的关系，并以此为界将它们划分为两类，进行递归分治。

算法基本框架

其实就是一次性对所有询问进行二分。

将多组询问放在一起，每次查询每个询问的答案与 mid 的关系，并以此为界将它们划分为两类，进行递归分治。

值域缩小到单点时记录询问答案并返回。

算法基本框架

其实就是一次性对所有询问进行二分。

将多组询问放在一起，每次查询每个询问的答案与 mid 的关系，并以此为界将它们划分为两类，进行递归分治。

值域缩小到单点时记录询问答案并返回。

如果有修改操作，同样以 mid 为界跟询问一起分类。

算法基本框架

其实就是一次性对所有询问进行二分。

将多组询问放在一起，每次查询每个询问的答案与 mid 的关系，并以此为界将它们划分为两类，进行递归分治。

值域缩小到单点时记录询问答案并返回。

如果有修改操作，同样以 mid 为界跟询问一起分类。

比较经典的问题就是区间带修第 k 大。

网格图分治

多组询问，求点对在网格图上的最短路。

$n \times m \leq 10^4, q \leq 10^5$

网格图分治

多组询问，求点对在网格图上的最短路。

$n \times m \leq 10^4, q \leq 10^5$

分治矩形区域，每次对区域较长的边划一条中线，以中线上的每个点为起点做一次最短路，更新矩形内的所有询问。

网格图分治

多组询问，求点对在网格图上的最短路。

$n \times m \leq 10^4, q \leq 10^5$

分治矩形区域，每次对区域较长的边划一条中线，以中线上的每个点为起点做一次最短路，更新矩形内的所有询问。

端点分别落在中线两侧的询问必然会被更新到最优解，而落在同侧的询问可能不会。于是把这样的询问以中线为界分治下去。

网格图分治

多组询问，求点对在网格图上的最短路。

$n \times m \leq 10^4, q \leq 10^5$

分治矩形区域，每次对区域较长的边划一条中线，以中线上的每个点为起点做一次最短路，更新矩形内的所有询问。

端点分别落在中线两侧的询问必然会被更新到最优解，而落在同侧的询问可能不会。于是把这样的询问以中线为界分治下去。

复杂度 $O((nm)^{1.5})$ 。

HNOI2015 接水果

模板题。

给定一棵大小为 n 的树，树上有 p 条带权路径，它们中的一些完全覆盖了从 x 到 y 的唯一路径，而你要求出这些路径中权值第 k 大的权值。q 组询问，每次给定 x, y, k 。

$n, p, q \leq 4 \times 10^4$

HNOI2015 接水果

若 $lca(x,y)!=x$ ，路径 (x,y) 被 (a,b) 完全覆盖需要满足的条件是 a,b 分别在 x,y 的子树内。写成表达式就是

$dfn_a \in [dfn_x, dfn_x+siz_x-1]$, $dfn_b \in [dfn_y, dfn_y+siz_y-1]$ 或者反过来。

HNOI2015 接水果

若 $lca(x,y)!=x$ ，路径 (x,y) 被 (a,b) 完全覆盖需要满足的条件是 a,b 分别在 x,y 的子树内。写成表达式就是

$dfn_a \in [dfn_x, dfn_x+siz_x-1], dfn_b \in [dfn_y, dfn_y+siz_y-1]$ 或者反过来。

再换个说法，要满足点 (dfn_a, dfn_b) 或 (dfn_b, dfn_a) 在上式围成的矩形区域内。

HNOI2015 接水果

若 $lca(x,y)!=x$ ，路径 (x,y) 被 (a,b) 完全覆盖需要满足的条件是 a,b 分别在 x,y 的子树内。写成表达式就是

$dfn_a \in [dfn_x, dfn_x+siz_x-1], dfn_b \in [dfn_y, dfn_y+siz_y-1]$ 或者反过来。

再换个说法，要满足点 (dfn_a, dfn_b) 或 (dfn_b, dfn_a) 在上式围成的矩形区域内。

若 $lca(x,y)=x$ ，则要满足一个点在 y 子树内，而另一个点不在从 y 往 x 的方向走，父节点为 x 的结点子树内。这种情况应当拆成 2 个矩形区域。

HNOI2015 接水果

若 $lca(x,y)!=x$ ，路径 (x,y) 被 (a,b) 完全覆盖需要满足的条件是 a,b 分别在 x,y 的子树内。写成表达式就是

$dfn_a \in [dfn_x, dfn_x+siz_x-1], dfn_b \in [dfn_y, dfn_y+siz_y-1]$ 或者反过来。

再换个说法，要满足点 (dfn_a, dfn_b) 或 (dfn_b, dfn_a) 在上式围成的矩形区域内。

若 $lca(x,y)=x$ ，则要满足一个点在 y 子树内，而另一个点不在从 y 往 x 的方向走，父节点为 x 的结点子树内。这种情况应当拆成 2 个矩形区域。

二分路径权值，每次树状数组 $+$ 扫描线求每个点被多少个矩形包含。

写在最后

本来是有想着挑几道毒题好好讲一讲的，但是做题量太少，就连放上来的这些水题都基本是上个月临时做的。

题目谈的比较浅，尤其是分治，建议课后再仔细研究一下，它们都特别有趣。

Thanks