



广州大学附属中学  
HIGH SCHOOL AFFILIATED TO GUANGZHOU UNIVERSITY



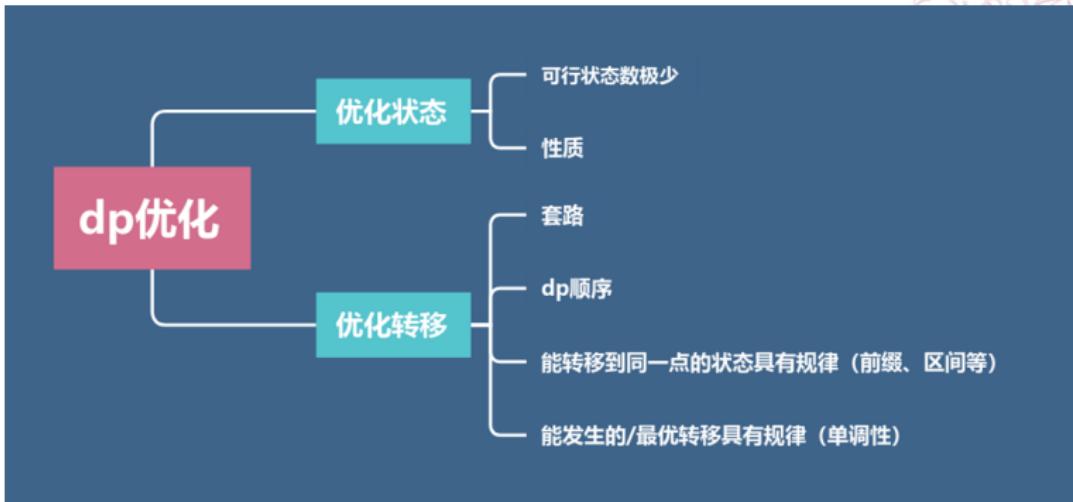
# dp 优化专题

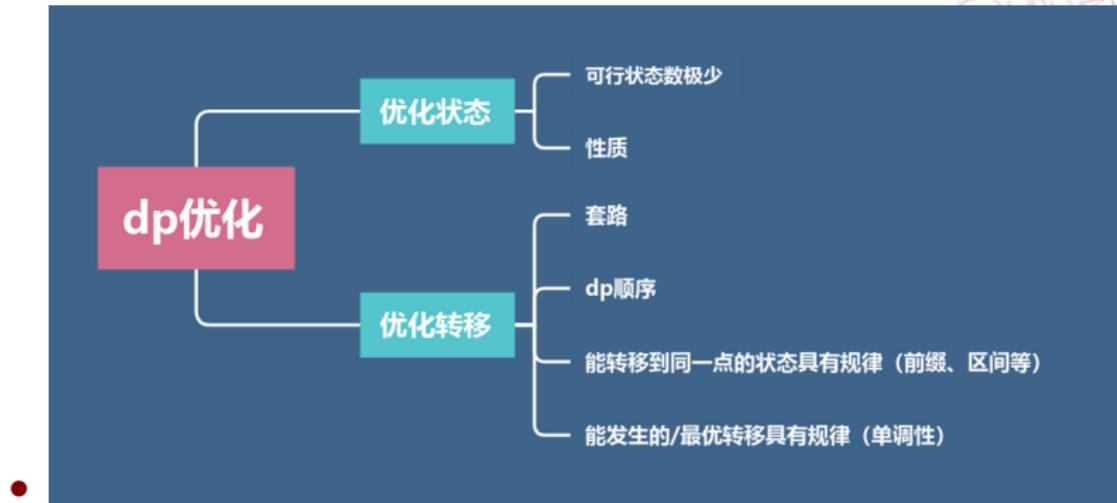
信息竞赛组

广州大学附属中学

2025 年 7 月

做最好的自己



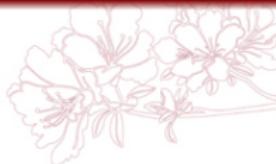


## ● 【8】动态规划的常用优化



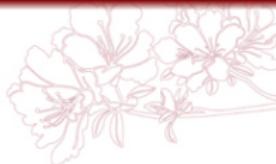
- $f_i = \sum\{f_j\} (l_i < j < r_i)$ , 或是类似更复杂的形式, 如

做最好的自己



- $f_i = \sum\{f_j\} (l_i < j < r_i)$ , 或是类似更复杂的形式, 如
  - $f_i = \sum\{g(f_j)\} (l_i < j < r_i)$ ,  $g(x)$  可以  $O(1)$  求

做最好的自己



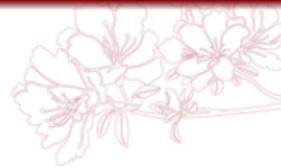
- $f_i = \sum\{f_j\} (l_i < j < r_i)$ , 或是类似更复杂的形式, 如
  - $f_i = \sum\{g(f_j)\} (l_i < j < r_i)$ ,  $g(x)$  可以  $O(1)$  求
  - $f_{i,j} = \sum\{f_{i-k,j-k}\}$ , 记  $s_{i,j} = f_{i',j'}$  的前缀和 ( $i-j=i'-j'$ ) (Queen on Grid)

做最好的自己



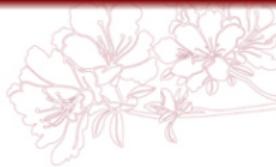
- $f_i = \sum\{f_j\} (l_i < j < r_i)$ , 或是类似更复杂的形式, 如
  - $f_i = \sum\{g(f_j)\} (l_i < j < r_i)$ ,  $g(x)$  可以  $O(1)$  求
  - $f_{i,j} = \sum\{f_{i-k,j-k}\}$ , 记  $s_{i,j} = f_{i',j'}$  的前缀和 ( $i-j=i'-j'$ ) (Queen on Grid)
  - $f_i = \sum\{f_j\} (l_i < j < r_i, a_i = a_j)$ , 对每个  $a_i$  开一个桶,  $x$  处记录  $j < i, a_j = x$  的所有  $f_j$  之和 ( $\text{Mod } l$ )

做最好的自己



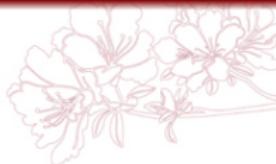
- $j$  对  $i$  的贡献只和  $j$  有关，和  $i$  无关，或是可化成和  $i$  无关的形式

做最好的自己



- $j$  对  $i$  的贡献只和  $j$  有关，和  $i$  无关，或是可化成和  $i$  无关的形式
- 转移通常需要枚举可转移到  $i$  的状态  $j$ ，然而转移通常是简单的求区间和，或是稍微复杂一点的求区间内属于某种类别 的状态之和

做最好的自己



- $j$  对  $i$  的贡献只和  $j$  有关，和  $i$  无关，或是可化成和  $i$  无关的形式
- 转移通常需要枚举可转移到  $i$  的状态  $j$ ，然而转移通常是简单的求区间和，或是稍微复杂一点的求区间内属于某种类别 的状态之和
- 转移复杂度  $O(n) \rightarrow O(1)$

做最好的自己



- Bessie 有一个数  $x + 0.5$ , 其中  $0 \leq x \leq N$  ( $1 \leq N \leq 5000$ )。Elsie 正试着猜这个数。她可以以如下形式提问: 「 $i$  是大了还是小了?」如果  $i$  大于  $x + 0.5$ , Bessie 会回答 “HI!”, 如果  $i$  小于  $x + 0.5$  则回答 “LO!”。

在进行任何猜测之前, Elsie 创建了长为  $N$  的一个排列。然后她遍历这一排列, 按排列中的数的顺序依次猜数。然而, Elsie 会跳过所有不必要的猜测。也就是说, 如果 Elsie 将要猜某个数  $i$ , 而 Elsie 之前已经猜过了某个  $j < i$  并且 Bessie 回答 “HI！”, Elsie 不会再猜  $i$ , 而是继续猜序列中的下一个数。“LO!” 同理。

Bessie 知道 Elsie 将要使用这一策略, 并且已经选定了值  $x$ , 但她不知道 Elsie 会使用什么排列。如果我们将同一排列所有 Bessie 回答的 “HI” 或 “LO” 拼接成一个字符串  $S$ , 你的目标是对于所有 Elsie 可能选用的排列, 计算 Bessie 说 “HILO”的次数之和, 对  $10^9 + 7$  取模。

做最好的自己

- Bessie 有一个数  $x + 0.5$ , 其中  $0 \leq x \leq N$  ( $1 \leq N \leq 5000$ )。Elsie 正试着猜这个数。她可以以如下形式提问: 「 $i$  是大了还是小了?」如果  $i$  大于  $x + 0.5$ , Bessie 会回答 “HI!”, 如果  $i$  小于  $x + 0.5$  则回答 “LO!”。

在进行任何猜测之前, Elsie 创建了长为  $N$  的一个排列。然后她遍历这一排列, 按排列中的数的顺序依次猜数。然而, Elsie 会跳过所有不必要的猜测。也就是说, 如果 Elsie 将要猜某个数  $i$ , 而 Elsie 之前已经猜过了某个  $j < i$  并且 Bessie 回答 “HI！”, Elsie 不会再猜  $i$ , 而是继续猜序列中的下一个数。“LO!” 同理。

Bessie 知道 Elsie 将要使用这一策略, 并且已经选定了值  $x$ , 但她不知道 Elsie 会使用什么排列。如果我们将同一排列所有 Bessie 回答的 “HI” 或 “LO” 拼接成一个字符串  $S$ , 你的目标是对于所有 Elsie 可能选用的排列, 计算 Bessie 说 “HILO”的次数之和, 对  $10^9 + 7$  取模。

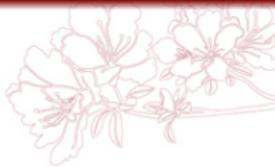
- 题目来源: 洛谷 P7986

做最好的自己



- Bessie 说的话肯定是一段 HI 一段 LO 交替出现，产生贡献的时刻只有一段 HI 即将结束，后面紧接着一段 LO 的时刻，故考虑按 HI, LO 连续段划分阶段

做最好的自己



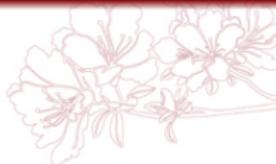
- Bessie 说的话肯定是一段 HI 一段 LO 交替出现，产生贡献的时刻只有一段 HI 即将结束，后面紧接着一段 LO 的时刻，故考虑按 HI, LO 连续段划分阶段
- 由于要设计 dp 状态，显然不能设计为每个数字是否用过，故需找到某种连续性以简化状态

做最好的自己



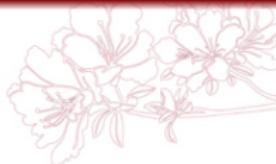
- Bessie 说的话肯定是一段 HI 一段 LO 交替出现，产生贡献的时刻只有一段 HI 即将结束，后面紧接着一段 LO 的时刻，故考虑按 HI, LO 连续段划分阶段
- 由于要设计 dp 状态，显然不能设计为每个数字是否用过，故需找到某种连续性以简化状态
- 考虑第一个 LO 段的最后一个数字  $y$ ，我们已经知道  $y \leq x$ ，故小于  $y$  的数字放在  $y$  后没有任何影响，放在  $y$  前也只会作为这个 LO 段的一部分，不影响贡献，故可以在考虑这一段时就确认好所有  $\leq y$  数字的位置，后续即为子问题

做最好的自己



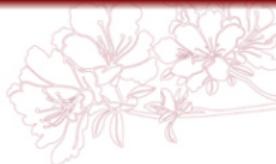
- 故可设  $f_{i,j,0/1}$  表示还有  $[i,j]$  数字未考虑，下一段是 LO/HI 的总贡献，则有

做最好的自己



- 故可设  $f_{i,j,0/1}$  表示还有  $[i,j]$  数字未考虑，下一段是 LO/HI 的总贡献，则有
  - $f_{i,j,0} = \sum_{i < k \leq j+1} (f_{k,j,1} + (j - k + 1)!)$  (无论后续  $[k,j]$  这些数如何排列，当前这个 HILO 都会产生贡献)  $(j - i + 1)^{\frac{k-i-1}{2}}$

做最好的自己



- 故可设  $f_{i,j,0/1}$  表示还有  $[i,j]$  数字未考虑，下一段是 LO/HI 的总贡献，则有
  - $f_{i,j,0} = \sum_{i < k \leq j+1} (f_{k,j,1} + (j - k + 1)!)$  (无论后续  $[k,j]$  这些数如何排列，当前这个 HILO 都会产生贡献)  $(j - i + 1)^{\underline{k-i-1}}$
  - $f_{i,j,1} = \sum_{i \leq k < j} f_{i,k,0} (j - i + 1)^{\underline{j-k-1}}$

做最好的自己



- 把下降幂化成阶乘的形式，则有

做最好的自己



- 把下降幂化成阶乘的形式，则有

$$\bullet f_{i,j,0} = \sum_{i < k \leq x+1} (f_{k,j,1} + (j - k + 1)!) \frac{(j-i+1)!}{(j-k+2)!}$$

做最好的自己



- 把下降幂化成阶乘的形式，则有

$$\begin{aligned} \bullet \quad f_{i,j,0} &= \sum_{i < k \leq x+1} (f_{k,j,1} + (j - k + 1)!) \frac{(j-i+1)!}{(j-k+2)!} \\ \bullet \quad f_{i,j,1} &= \sum_{x \leq k < j} f_{i,k,0} \frac{(j-i+1)!}{(k-i+2)!} \end{aligned}$$

做最好的自己



- 分离参数，把和  $j-i$  有关的除到左边，和  $j-k/k-i$  有关的留在右边，则有

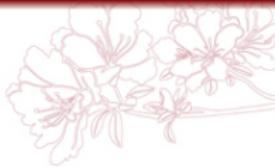
做最好的自己



- 分离参数，把和  $j-i$  有关的除到左边，和  $j-k/k-i$  有关的留在右边，则有

- $\frac{f_{i,j,0}}{(j-i+1)!} = \sum_{i < k \leq x+1} \frac{f_{k,j,1} + (j-k+1)!}{(j-k+2)!}$

做最好的自己



- 分离参数，把和  $j-i$  有关的除到左边，和  $j-k/k-i$  有关的留在右边，则有

$$\frac{f_{i,j,0}}{(j-i+1)!} = \sum_{i < k \leq x+1} \frac{f_{k,j,1} + (j-k+1)!}{(j-k+2)!}$$

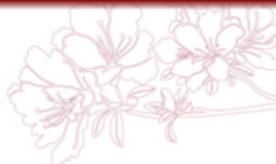
$$\frac{f_{i,j,1}}{(j-i+1)!} = \sum_{x \leq k < j} \frac{f_{i,k,0}}{(k-i+2)!}$$

做最好的自己



- 设  $g_{i,j,0/1} = \frac{f_{i,j,0/1}}{(j-i+1)!}$ , 则

做最好的自己



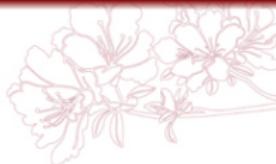
- 设  $g_{i,j,0/1} = \frac{f_{i,j,0/1}}{(j-i+1)!}$ , 则
  - $g_{i,j,0} = \sum_{i < k \leq x+1} \frac{g_{k,j,1} + 1}{(j-k+2)}$

做最好的自己



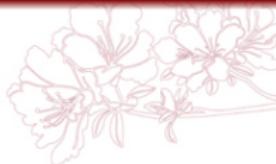
- 设  $g_{i,j,0/1} = \frac{f_{i,j,0/1}}{(j-i+1)!}$ , 则
  - $g_{i,j,0} = \sum_{i < k \leq x+1} \frac{g_{k,j,1} + 1}{(j-k+2)}$
  - $g_{i,j,1} = \sum_{x \leq k < j} \frac{g_{i,k,0}}{(k-i+2)}$

做最好的自己



- 设  $g_{i,j,0/1} = \frac{f_{i,j,0/1}}{(j-i+1)!}$ , 则
  - $g_{i,j,0} = \sum_{i < k \leq x+1} \frac{g_{k,j,1} + 1}{(j-k+2)}$
  - $g_{i,j,1} = \sum_{x \leq k < j} \frac{g_{i,k,0}}{(k-i+2)}$
- 右边只和区间长度有关, 在计算出 g 时计算并记录其前缀和即可

做最好的自己



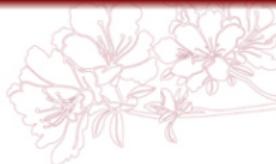
- 设  $g_{i,j,0/1} = \frac{f_{i,j,0/1}}{(j-i+1)!}$ , 则
  - $g_{i,j,0} = \sum_{i < k \leq x+1} \frac{g_{k,j,1} + 1}{(j-k+2)}$
  - $g_{i,j,1} = \sum_{x \leq k < j} \frac{g_{i,k,0}}{(k-i+2)}$
- 右边只和区间长度有关, 在计算出 g 时计算并记录其前缀和即可
- 时间复杂度  $O(n^2)$

做最好的自己



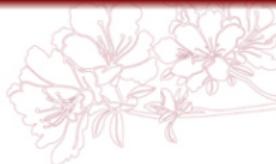
- $f_i = \max\{f_j\} + g(j, i)$ , 可转移到  $i$ /最优的  $j$  随着  $i$  的递增而递增

做最好的自己



- $f_i = \max\{f_j\} + g(j, i)$ , 可转移到  $i$ /最优的  $j$  随着  $i$  的递增而递增
- 用两个指针维护  $i, j$ , 每一次把  $i$  或  $j$  前推一格, 用  $O(1)$  的时间计算出  $g(i, j)$  的变化量

做最好的自己



- $f_i = \max\{f_j\} + g(j, i)$ , 可转移到  $i$ /最优的  $j$  随着  $i$  的递增而递增
- 用两个指针维护  $i, j$ , 每一次把  $i$  或  $j$  前推一格, 用  $O(1)$  的时间计算出  $g(i, j)$  的变化量
- 由于  $i, j$  递增, 所以它们都只会前推  $n$  格, 所以总时间复杂度是  $O(n)$  的

做最好的自己



- 例如：题目的限制条件以区间的形式出现，而且有

做最好的自己



- 例如：题目的限制条件以区间的形式出现，而且有
- 若  $b$  是  $a$  的子区间，满足  $a$  则必满足  $b$ ，或满足  $b$  则必满足  $a$ ，以把区间套区间的情况去掉

做最好的自己



- 例如：题目的限制条件以区间的形式出现，而且有
- 若  $b$  是  $a$  的子区间，满足  $a$  则必满足  $b$ ，或满足  $b$  则必满足  $a$ ，以把区间套区间的情况去掉
- 余下的情况，各区间的右端点  $r$  必然随着左端点  $l$  递增，这就有了使用双指针的机会

做最好的自己



- 给出  $n$  个整数  $a_1, a_2, \dots, a_n$ , 现在需要对其进行分组, 使其满足以下条件:

请判断是否存在满足条件的分组方案, 若有请输出"YES", 否则输出"NO"。

做最好的自己



- 给出  $n$  个整数  $a_1, a_2, \dots, a_n$ , 现在需要对其进行分组, 使其满足以下条件:
  - 每个数都必须恰好分入一组中

请判断是否存在满足条件的分组方案, 若有请输出"YES", 否则输出"NO"。

做最好的自己



- 给出  $n$  个整数  $a_1, a_2, \dots, a_n$ , 现在需要对其进行分组, 使其满足以下条件:
  - 每个数都必须恰好分入一组中
  - 每一组中必须至少包含  $k$  个数

请判断是否存在满足条件的分组方案, 若有请输出"YES", 否则输出"NO"。

做最好的自己



- 给出  $n$  个整数  $a_1, a_2, \dots, a_n$ , 现在需要对其进行分组, 使其满足以下条件:
  - 每个数都必须恰好分入一组中
  - 每一组中必须至少包含  $k$  个数
  - 在每一组中, 整数的权值之差的绝对值不能超过  $d$ 。即当  $a_i, a_j$  在同一组时, 需要满足  $|a_i - a_j| \leq d$

请判断是否存在满足条件的分组方案, 若有请输出"YES", 否则输出"NO"。

做最好的自己



- 给出  $n$  个整数  $a_1, a_2, \dots, a_n$ , 现在需要对其进行分组, 使其满足以下条件:
  - 每个数都必须恰好分入一组中
  - 每一组中必须至少包含  $k$  个数
  - 在每一组中, 整数的权值之差的绝对值不能超过  $d$ 。即当  $a_i, a_j$  在同一组时, 需要满足  $|a_i - a_j| \leq d$

请判断是否存在满足条件的分组方案, 若有请输出"YES", 否则输出"NO"。

- $1 \leq k \leq n \leq 5 \times 10^5$

做最好的自己



- 给出  $n$  个整数  $a_1, a_2, \dots, a_n$ , 现在需要对其进行分组, 使其满足以下条件:
  - 每个数都必须恰好分入一组中
  - 每一组中必须至少包含  $k$  个数
  - 在每一组中, 整数的权值之差的绝对值不能超过  $d$ 。即当  $a_i, a_j$  在同一组时, 需要满足  $|a_i - a_j| \leq d$

请判断是否存在满足条件的分组方案, 若有请输出"YES", 否则输出"NO"。

- $1 \leq k \leq n \leq 5 \times 10^5$
- $0 \leq d \leq 10^9$

做最好的自己



- 给出  $n$  个整数  $a_1, a_2, \dots, a_n$ , 现在需要对其进行分组, 使其满足以下条件:
  - 每个数都必须恰好分入一组中
  - 每一组中必须至少包含  $k$  个数
  - 在每一组中, 整数的权值之差的绝对值不能超过  $d$ 。即当  $a_i, a_j$  在同一组时, 需要满足  $|a_i - a_j| \leq d$

请判断是否存在满足条件的分组方案, 若有请输出"YES", 否则输出"NO"。

- $1 \leq k \leq n \leq 5 \times 10^5$
- $0 \leq d \leq 10^9$
- 题目来源: CF985E

做最好的自己



- 考虑划分阶段，显然，若存在分组方案，则一定可以写成若干个值域区间的形式

做最好的自己



- 考虑划分阶段，显然，若存在分组方案，则一定可以写成若干个值域区间的形式
- 如若不然，假设分组方案中存在两组  $S, T$ ，满足 $\min\{S\} \leq \min\{T\} \leq \max\{S\} \leq \max\{T\}$ ，则将  $\min\{T\}$  与  $\max\{S\}$  交换，仍是一种合法的分组方案

做最好的自己



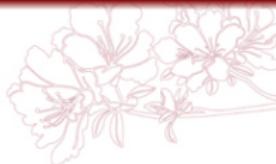
- 那么，我们分的每一组一定是一个极值域区间吗？

做最好的自己



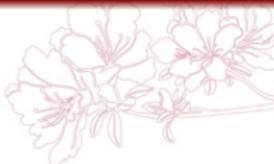
- 那么，我们分的每一组一定是一个极值域区间吗？
- 其实不是，因为他要求每一组至少有  $k$  个元素，所以我们可以在一个较小的区间分到一组，然后剩余的元素补充到相邻的两个区间里，使原本不合法的分组方案合法

做最好的自己



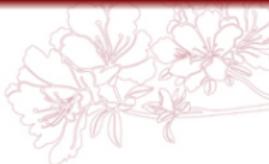
- 由于分组方案在值域上的连续性，考虑按值域划分状态，设  $f_i$  为前  $i$  个数分为一组是否合法，则  $f_i$  可更新的范围为  $[i+k, r_i]$

做最好的自己



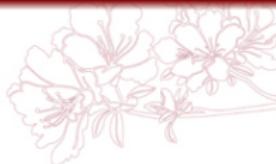
- 由于分组方案在值域上的连续性，考虑按值域划分状态，设  $f_i$  为前  $i$  个数分为一组是否合法，则  $f_i$  可更新的范围为  $[i+k, r_i]$
- 若  $i < j$ , 且  $f_j$  可更新的范围为  $[j+k, r_j]$ , 对于  $[j+k, r_i]$ , 我们在  $i$  时已标记合法, 故只需标记  $r_i$  右侧的合法点即可

做最好的自己



- 由于分组方案在值域上的连续性，考虑按值域划分状态，设  $f_i$  为前  $i$  个数分为一组是否合法，则  $f_i$  可更新的范围为  $[i+k, r_i]$
- 若  $i < j$ , 且  $f_j$  可更新的范围为  $[j+k, r_j]$ , 对于  $[j+k, r_i]$ , 我们在  $i$  时已标记合法, 故只需标记  $r_i$  右侧的合法点即可
- 故从  $i$  到  $j$ , 需要标记合法的点也单调递增, 故可以使用双指针, 时间复杂度  $O(n \log n)$ (排序)

做最好的自己



- 考虑形如  $f_i = \max / \min \{g(f_j)\}$  的 dp，若有类似于：

.....的性质，则可使用单调队列优化转移，从  $O(n)$  优化到  $O(1)$

做最好的自己



- 考虑形如  $f_i = \max/\min\{g(f_j)\}$  的 dp，若有类似于：
  - $j < k$  且  $f_j \leq f_k$ ，则  $g(f_j)$  优于  $g(f_k)$

.....的性质，则可使用单调队列优化转移，从  $O(n)$  优化到  $O(1)$

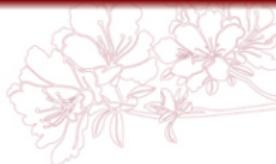
做最好的自己



- 考虑形如  $f_i = \max / \min \{g(f_j)\}$  的 dp，若有类似于：
  - $j < k$  且  $f_j \leq f_k$ , 则  $g(f_j)$  优于  $g(f_k)$
  - $j < k$  且  $f_j \geq f_k$ , 则  $g(f_j)$  优于  $g(f_k)$

.....的性质，则可使用单调队列优化转移，从  $O(n)$  优化到  $O(1)$

做最好的自己



- 考虑形如  $f_i = \max/\min\{g(f_j)\}$  的 dp，若有类似于：
  - $j < k$  且  $f_j \leq f_k$ , 则  $g(f_j)$  优于  $g(f_k)$
  - $j < k$  且  $f_j \geq f_k$ , 则  $g(f_j)$  优于  $g(f_k)$
  - $a_j \leq a_k$  且  $f_j \leq f_k$ , 则  $g(f_j)$  优于  $g(f_k)$

.....的性质，则可使用单调队列优化转移，从  $O(n)$  优化到  $O(1)$

做最好的自己



- 考虑形如  $f_i = \max/\min\{g(f_j)\}$  的 dp，若有类似于：
  - $j < k$  且  $f_j \leq f_k$ , 则  $g(f_j)$  优于  $g(f_k)$
  - $j < k$  且  $f_j \geq f_k$ , 则  $g(f_j)$  优于  $g(f_k)$
  - $a_j \leq a_k$  且  $f_j \leq f_k$ , 则  $g(f_j)$  优于  $g(f_k)$
  - $a_j \geq a_k$  且  $f_j \geq f_k$ , 则  $g(f_j)$  优于  $g(f_k)$.....的性质，则可使用单调队列优化转移，从  $O(n)$  优化到  $O(1)$

做最好的自己



- 关键是  $g(f_j)$  是  $j$  的函数，和  $i$  到  $j$  这一段其他元素的信息无关

做最好的自己



- 有  $n$  棵树排成一排，第  $i$  棵树的高度是  $d_i$ 。  
有  $q$  只鸟要从第 1 棵树到第  $n$  棵树。  
当第  $i$  只鸟在第  $j$  棵树时，它可以飞到第  
 $j + 1, j + 2, \dots, j + k_i$  棵树。  
如果一只鸟飞到一颗高度大于等于当前树的树，那么它的劳  
累值会增加 1，否则不会。  
由于这些鸟已经体力不支，所以它们想要最小化劳累值。

做最好的自己



- 有  $n$  棵树排成一排，第  $i$  棵树的高度是  $d_i$ 。  
有  $q$  只鸟要从第 1 棵树到第  $n$  棵树。  
当第  $i$  只鸟在第  $j$  棵树时，它可以飞到第  
 $j+1, j+2, \dots, j+k_i$  棵树。  
如果一只鸟飞到一颗高度大于等于当前树的树，那么它的劳  
累值会增加 1，否则不会。  
由于这些鸟已经体力不支，所以它们想要最小化劳累值。
  - $1 \leq n \leq 10^6$ ,  $1 \leq d_i \leq 10^9$ ,  $1 \leq q \leq 25$ ,  $1 \leq k_i \leq n - 1$ .

做最好的自己



- 有  $n$  棵树排成一排，第  $i$  棵树的高度是  $d_i$ 。  
有  $q$  只鸟要从第 1 棵树到第  $n$  棵树。  
当第  $i$  只鸟在第  $j$  棵树时，它可以飞到第  
 $j+1, j+2, \dots, j+k_i$  棵树。  
如果一只鸟飞到一颗高度大于等于当前树的树，那么它的劳  
累值会增加 1，否则不会。  
由于这些鸟已经体力不支，所以它们想要最小化劳累值。
  - $1 \leq n \leq 10^6$ ,  $1 \leq d_i \leq 10^9$ ,  $1 \leq q \leq 25$ ,  $1 \leq k_i \leq n - 1$ .
  - 题目来源：洛谷 P3572

做最好的自己

- 根据题意能比较直接的得到  $O(n^2)$  dp，设  $f_i$  为到第  $i$  棵树的最小劳累值，则  $f_i = \min\{f_j + (a_j \leq a_i)\} (i - k \leq j < i)$

做最好的自己



- 根据题意能比较直接的得到  $O(n^2)$  dp, 设  $f_i$  为到第  $i$  棵树的最小劳累值, 则  $f_i = \min\{f_j + (a_j \leq a_i)\} (i - k \leq j < i)$
- 如何优化? 考虑若  $j < k, f_j > f_k$  则  $k$  必定优于  $j$ , 若  $j < k, f_j = f_k, a_j \leq a_k$  则  $k$  也必定优于  $j$ , 故设  $g_i = \{f_i, -a_i\}$ , 则若  $j < k, g_j \geq g_k$ ,  $k$  必定优于  $j$

做最好的自己



- 根据题意能比较直接的得到  $O(n^2)$  dp, 设  $f_i$  为到第  $i$  棵树的最小劳累值, 则  $f_i = \min\{f_j + (a_j \leq a_i)\} (i - k \leq j < i)$
- 如何优化? 考虑若  $j < k, f_j > f_k$  则  $k$  必定优于  $j$ , 若  $j < k, f_j = f_k, a_j \leq a_k$  则  $k$  也必定优于  $j$ , 故设  $g_i = \{f_i, -a_i\}$ , 则若  $j < k, g_j \geq g_k$ ,  $k$  必定优于  $j$
- 这显然是单调队列的一个条件, 我们用单调队列维护所有可能是未来某个点的最优解的  $j$

做最好的自己



- 根据题意能比较直接的得到  $O(n^2)$  dp，设  $f_i$  为到第  $i$  棵树的最小劳累值，则  $f_i = \min\{f_j + (a_j \leq a_i)\} (i - k \leq j < i)$
- 如何优化？考虑若  $j < k, f_j > f_k$  则  $k$  必定优于  $j$ ，若  $j < k, f_j = f_k, a_j \leq a_k$  则  $k$  也必定优于  $j$ ，故设  $g_i = \{f_i, -a_i\}$ ，则若  $j < k, g_j \geq g_k$ ， $k$  必定优于  $j$
- 这显然是单调队列的一个条件，我们用单调队列维护所有可能是未来某个点的最优解的  $j$
- 那么对于队列中两个元素  $j, k$ ，若  $j < k$ ，则必有  $g_j < g_k$ ，计算  $f_i$  时，我们直接取队头进行转移，因为队头即为最优解

做最好的自己



- 根据题意能比较直接的得到  $O(n^2)$  dp，设  $f_i$  为到第  $i$  棵树的最小劳累值，则  $f_i = \min\{f_j + (a_j \leq a_i)\} (i - k \leq j < i)$
- 如何优化？考虑若  $j < k, f_j > f_k$  则  $k$  必定优于  $j$ ，若  $j < k, f_j = f_k, a_j \leq a_k$  则  $k$  也必定优于  $j$ ，故设  $g_i = \{f_i, -a_i\}$ ，则若  $j < k, g_j \geq g_k$ ， $k$  必定优于  $j$
- 这显然是单调队列的一个条件，我们用单调队列维护所有可能是未来某个点的最优解的  $j$
- 那么对于队列中两个元素  $j, k$ ，若  $j < k$ ，则必有  $g_j < g_k$ ，计算  $f_i$  时，我们直接取队头进行转移，因为队头即为最优解
- 然后我们需要将  $(i, g_i)$  加入队列，我们只需弹出队尾所有满足  $g_j \geq g_i$  的  $j$ ，因为  $i$  必定优于  $j$ ，然后再将  $i$  加入队尾，再注意一下队头是否等于  $i - k$ ，若是，需要弹出

做最好的自己



- 根据题意能比较直接的得到  $O(n^2)$  dp，设  $f_i$  为到第  $i$  棵树的最小劳累值，则  $f_i = \min\{f_j + (a_j \leq a_i)\} (i - k \leq j < i)$
- 如何优化？考虑若  $j < k, f_j > f_k$  则  $k$  必定优于  $j$ ，若  $j < k, f_j = f_k, a_j \leq a_k$  则  $k$  也必定优于  $j$ ，故设  $g_i = \{f_i, -a_i\}$ ，则若  $j < k, g_j \geq g_k$ ， $k$  必定优于  $j$
- 这显然是单调队列的一个条件，我们用单调队列维护所有可能是未来某个点的最优解的  $j$
- 那么对于队列中两个元素  $j, k$ ，若  $j < k$ ，则必有  $g_j < g_k$ ，计算  $f_i$  时，我们直接取队头进行转移，因为队头即为最优解
- 然后我们需要将  $(i, g_i)$  加入队列，我们只需弹出队尾所有满足  $g_j \geq g_i$  的  $j$ ，因为  $i$  必定优于  $j$ ，然后再将  $i$  加入队尾，再注意一下队头是否等于  $i - k$ ，若是，需要弹出
- 总时间复杂度为  $O(qn)$

做最好的自己



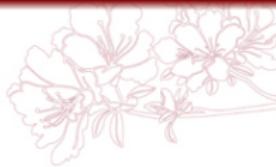
- 如  $f_i = \max / \min \{g(f_j)\} (b_j \leq b_i)$  的 dp

做最好的自己



- 如  $f_i = \max / \min \{g(f_j)\} (b_j \leq b_i)$  的 dp
- j 对 i 的贡献依然是只和 j 有关

做最好的自己



- 如  $f_i = \max / \min \{g(f_j)\} (b_j \leq b_i)$  的 dp
- j 对 i 的贡献依然是只和 j 有关
- 所有可转移到 i 的 j 满足统一的某种限制条件，如偏序关系，即  $b_j \leq b_i$

做最好的自己



- 如  $f_i = \max / \min \{g(f_j)\} (b_j \leq b_i)$  的 dp
- j 对 i 的贡献依然是只和 j 有关
- 所有可转移到 i 的 j 满足统一的某种限制条件，如偏序关系，即  $b_j \leq b_i$
- 数据结构优化转移， $O(n) \rightarrow O(\log n)$

做最好的自己



- Bessie 在一个仅允许沿平行于坐标轴方向移动的二维方阵中。她从点  $(0, 0)$  出发，想要到达  $(N, N)$  ( $1 \leq N \leq 10^9$ )。为了帮助她达到目的，在方阵中有  $P$  ( $1 \leq P \leq 10^5$ ) 个跳板。每个跳板都有其固定的位置  $(x_1, y_1)$ ，如果 Bessie 使用它，会落到点  $(x_2, y_2)$ 。  
Bessie 是一个过程导向的奶牛，所以她仅允许她自己向上或向右行走，从不向左或向下。类似地，每个跳板也设置为不向左或向下。Bessie 需要行走的距离至少是多少？

做最好的自己



- Bessie 在一个仅允许沿平行于坐标轴方向移动的二维方阵中。她从点  $(0, 0)$  出发，想要到达  $(N, N)$  ( $1 \leq N \leq 10^9$ )。为了帮助她达到目的，在方阵中有  $P$  ( $1 \leq P \leq 10^5$ ) 个跳板。每个跳板都有其固定的位置  $(x_1, y_1)$ ，如果 Bessie 使用它，会落到点  $(x_2, y_2)$ 。

Bessie 是一个过程导向的奶牛，所以她仅允许她自己向上或向右行走，从不向左或向下。类似地，每个跳板也设置为不向左或向下。Bessie 需要行走的距离至少是多少？

- 题目来源：洛谷 P6007

做最好的自己



- 把起终点也当做跳板，容易得到  $O(n^2)dp$ ，即设  $f_i$  为走到某个跳板的最少步数，则

$$f_i = \min\{f_j + stx_i - edx_j + sty_i - edy_j\} (edx_j \leq stx_i, edy_j \leq sty_i),$$

分离参数易知贡献只和 j 有关

做最好的自己



- 把起终点也当做跳板，容易得到  $O(n^2)dp$ ，即设  $f_i$  为走到某个跳板的最少步数，则
$$f_i = \min\{f_j + stx_j - edx_j + sty_j - edy_j\} (edx_j \leq stx_i, edy_j \leq sty_i),$$
分离参数易知贡献只和  $j$  有关
- 考虑优化转移，容易发现能转移到  $i$  的  $j$  满足二维偏序关系，故可对  $edx_j, stx_i$  放在一起排序，扫到  $edx_j$  时，将  $f_j - edx_j - edy_j$  放入  $edy_j$  对应的权值树状数组中，对于  $stx_i$  只需统计  $sty_i$  对应前缀  $\min$  即可

做最好的自己



- 把起终点也当做跳板，容易得到  $O(n^2)dp$ ，即设  $f_i$  为走到某个跳板的最少步数，则
$$f_i = \min\{f_j + stx_j - edx_j + sty_j - edy_j\} (edx_j \leq stx_i, edy_j \leq sty_i),$$
分离参数易知贡献只和  $j$  有关
- 考虑优化转移，容易发现能转移到  $i$  的  $j$  满足二维偏序关系，故可对  $edx_j, stx_i$  放在一起排序，扫到  $edx_j$  时，将  $f_j - edx_j - edy_j$  放入  $edy_j$  对应的权值树状数组中，对于  $stx_i$  只需统计  $sty_i$  对应前缀  $\min$  即可
- 时间复杂度  $O(n \log n)$

做最好的自己



- 形如  $f_{i,j} = \sum f_{i-1,k} c_{k,j}$ , 即  $F_i = F_{i-1} * C$ ,  $F_n = F_0 * C^n$

做最好的自己



- 形如  $f_{i,j} = \sum f_{i-1,k} c_{k,j}$ , 即  $F_i = F_{i-1} * C$ ,  $F_n = F_0 * C^n$
- $f_i$  只与  $f_{i-1}$  或前面的某几项有关

做最好的自己



- 形如  $f_{i,j} = \sum f_{i-1,k} c_{k,j}$ , 即  $F_i = F_{i-1} * C$ ,  $F_n = F_0 * C^n$
- $f_i$  只与  $f_{i-1}$  或前面的某几项有关
- $k$  这一维较小,  $O(k^3)$  复杂度可以接受

做最好的自己



- 形如  $f_{i,j} = \sum f_{i-1,k} c_{k,j}$ , 即  $F_i = F_{i-1} * C$ ,  $F_n = F_0 * C^n$
- $f_i$  只与  $f_{i-1}$  或前面的某几项有关
- $k$  这一维较小,  $O(k^3)$  复杂度可以接受
- $n$  这一维极大, 常为  $10^{18}$

做最好的自己



- 形如  $f_{i,j} = \sum f_{i-1,k} c_{k,j}$ , 即  $F_i = F_{i-1} * C$ ,  $F_n = F_0 * C^n$
- $f_i$  只与  $f_{i-1}$  或前面的某几项有关
- $k$  这一维较小,  $O(k^3)$  复杂度可以接受
- $n$  这一维极大, 常为  $10^{18}$
- 有时  $(\max, +)$  会替代  $(+, *)$  出现, 但方法依然可行

做最好的自己

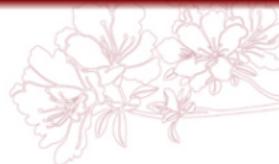


- 克露丝卡尔酱想要计数  $n+1$  个点的  $k$  阶轮的生成树个数。  
 $n+1$  个点的  $k$  阶轮的定义为：
  - 0 为中心， $1 \sim n$  构成一个环（对于  $1 \leq i < n$ ,  $i$  和  $i+1$  之间有连边，1 和  $n$  之间有连边）。
  - 对于  $1 \leq i \leq \frac{n}{k}$ , 0 和  $ik$  之间有额外连边。

保证  $n \bmod k = 0$ , 答案对  $10^9 + 7$  取模。

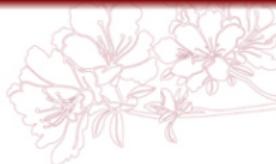
- $1 \leq k \leq n \leq 10^{18}$ ,  $n \geq 3$
- 题目来源：洛谷 P12046

做最好的自己



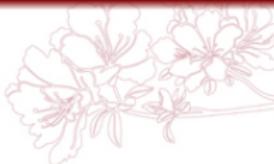
- 由于保证了  $n \bmod k = 0$ , 我们可以将每  $k$  个点划分为一个单元, 一个单元一个单元地考虑问题

做最好的自己



- 由于保证了  $n \bmod k = 0$ , 我们可以将每  $k$  个点划分为一个单元, 一个单元一个单元地考虑问题
- 形式化地, 我们称  $1 \sim n$  构成的环上的边为环边,  $0$  和  $ik$  之间的连边为轴边

做最好的自己



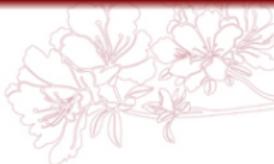
- 由于保证了  $n \bmod k = 0$ , 我们可以将每  $k$  个点划分为一个单元, 一个单元一个单元地考虑问题
- 形式化地, 我们称  $1 \sim n$  构成的环上的边为环边,  $0$  和  $ik$  之间的连边为轴边
- 对于一个单元, 考虑删边, 显然不能删除两个及以上环边, 否则必不联通, 故共有四种情况

做最好的自己



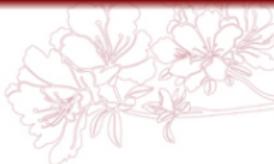
- 由于保证了  $n \bmod k = 0$ , 我们可以将每  $k$  个点划分为一个单元, 一个单元一个单元地考虑问题
- 形式化地, 我们称  $1 \sim n$  构成的环上的边为环边,  $0$  和  $ik$  之间的连边为轴边
- 对于一个单元, 考虑删边, 显然不能删除两个及以上环边, 否则必不联通, 故共有四种情况
  - 不删除

做最好的自己



- 由于保证了  $n \bmod k = 0$ , 我们可以将每  $k$  个点划分为一个单元, 一个单元一个单元地考虑问题
- 形式化地, 我们称  $1 \sim n$  构成的环上的边为环边,  $0$  和  $ik$  之间的连边为轴边
- 对于一个单元, 考虑删边, 显然不能删除两个及以上环边, 否则必不联通, 故共有四种情况
  - 不删除
  - 删除一条环边

做最好的自己



- 由于保证了  $n \bmod k = 0$ , 我们可以将每  $k$  个点划分为一个单元, 一个单元一个单元地考虑问题
- 形式化地, 我们称  $1 \sim n$  构成的环上的边为环边,  $0$  和  $ik$  之间的连边为轴边
- 对于一个单元, 考虑删边, 显然不能删除两个及以上环边, 否则必不联通, 故共有四种情况
  - 不删除
  - 删除一条环边
  - 只删除轴边

做最好的自己



- 由于保证了  $n \bmod k = 0$ , 我们可以将每  $k$  个点划分为一个单元, 一个单元一个单元地考虑问题
- 形式化地, 我们称  $1 \sim n$  构成的环上的边为环边,  $0$  和  $ik$  之间的连边为轴边
- 对于一个单元, 考虑删边, 显然不能删除两个及以上环边, 否则必不联通, 故共有四种情况
  - 不删除
  - 删除一条环边
  - 只删除轴边
  - 删除轴边和一条环边

做最好的自己



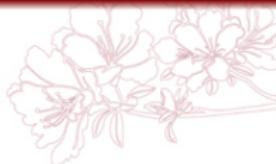
- 若不删除，考虑上一根未被删除的轴边位置，若到这根轴边之间没有环边被删除，则会构成环，不是合法的生成树，故需上一根轴边后有被删除的环边

做最好的自己



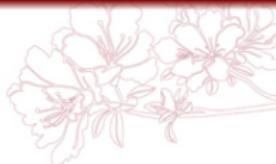
- 若不删除，考虑上一根未被删除的轴边位置，若到这根轴边之间没有环边被删除，则会构成环，不是合法的生成树，故需上一根轴边后有被删除的环边
- 若删除一条环边，同样是考虑上一根未被删除的轴边位置，若其后有被删除的环边，再删除掉这根环边，则会构成一个子联通块，不合法

做最好的自己



- 若只删除轴边，没有产生新的轴边或断点，故仍然合法，轴边和断点的相对位置关系也可继承之前的状态

做最好的自己



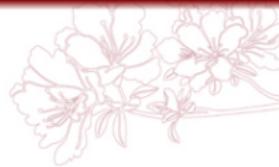
- 若只删除轴边，没有产生新的轴边或断点，故仍然合法，轴边和断点的相对位置关系也可继承之前的状态
- 若删除轴边和一条环边，若上一条轴边在断点前，则同样会产生新的联通块，不合法，否则虽然合法，但产生了新的断点

做最好的自己



- 若只删除轴边，没有产生新的轴边或断点，故仍然合法，轴边和断点的相对位置关系也可继承之前的状态
- 若删除轴边和一条环边，若上一条轴边在断点前，则同样会产生新的联通块，不合法，否则虽然合法，但产生了新的断点
- 综上，我们发现只需维护上一根轴边和上一个断点的相对位置关系，即可发生转移

做最好的自己



- 然而由于整个图是一个环，我们还需检验  $n$  和 1 之间会不会构成环或独立联通块

做最好的自己



- 然而由于整个图是一个环，我们还需检验  $n$  和 1 之间会不会构成环或独立联通块
- 我们已经知道了  $n$  端先出现的是断点还是轴边，如果我们知道了 1 端的这个信息，显然我们即可完成判断

做最好的自己



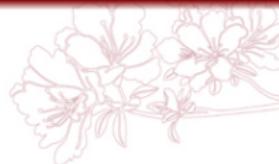
- 然而由于整个图是一个环，我们还需检验  $n$  和 1 之间会不会构成环或独立联通块
- 我们已经知道了  $n$  端先出现的是断点还是轴边，如果我们知道了 1 端的这个信息，显然我们即可完成判断
- 故我们还需记录最早出现的是断点还是轴边，用一个 0/1/2 状态（0 表示都未出现过，1 表示断点，2 表示轴边）记录即可

做最好的自己



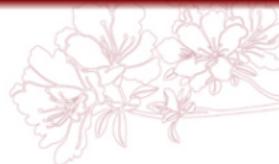
- 故我们最终的状态为  $f_{i,0/1,0/1/2}$  表示当前考虑到第  $i$  个单元，最后出现的是断点还是轴边，最先出现的是断点还是轴边；显然  $f_i$  的状态只由  $f_{i-1}$  决定

做最好的自己



- 故我们最终的状态为  $f_{i,0/1,0/1/2}$  表示当前考虑到第  $i$  个单元，最后出现的是断点还是轴边，最先出现的是断点还是轴边；显然  $f_i$  的状态只由  $f_{i-1}$  决定
- 故可以矩阵加速， $F_i = F_{i-1} * C$ ,  $F_n = F_0 * C^n$ , 快速幂计算  $C^n$  即可

做最好的自己



- 故我们最终的状态为  $f_{i,0/1,0/1/2}$  表示当前考虑到第  $i$  个单元，最后出现的是断点还是轴边，最先出现的是断点还是轴边；显然  $f_i$  的状态只由  $f_{i-1}$  决定
- 故可以矩阵加速， $F_i = F_{i-1} * C$ ,  $F_n = F_0 * C^n$ , 快速幂计算  $C^n$  即可
- 时间复杂度  $O(6^3 \log n)$

做最好的自己



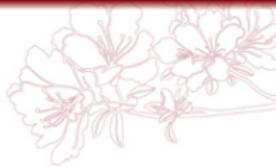
- 同矩阵乘，形如  $f_{i,j} = \sum f_{i-1,k} c_{i_k,j}$ ，或是树上形如  $f_{u,j} = \sum f_{v,k} c_{u_k,j}$ ，带修改

做最好的自己



- 同矩阵乘，形如  $f_{i,j} = \sum f_{i-1,k} c_{i_k,j}$ ，或是树上形如  $f_{u,j} = \sum f_{v,k} c_{u_k,j}$ ，带修改
- 由于矩阵乘法满足结合律，故可用线段树维护区间  $C_i$  积，修改为单点修改，查询  $F_n$  即为查询  $1 \sim n$  区间  $C_i$  积，可以支持

做最好的自己



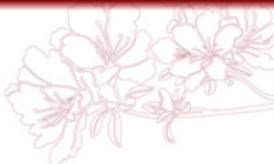
- 同矩阵乘，形如  $f_{i,j} = \sum f_{i-1,k} c_{i_k,j}$ ，或是树上形如  $f_{u,j} = \sum f_{v,k} c_{u_k,j}$ ，带修改
- 由于矩阵乘法满足结合律，故可用线段树维护区间  $C_i$  积，修改为单点修改，查询  $F_n$  即为查询  $1 \sim n$  区间  $C_i$  积，可以支持
- 若在树上，则可用树剖，线段树维护重链  $C_i$  积，轻链暴力转移即可

做最好的自己



- 同矩阵乘，形如  $f_{i,j} = \sum f_{i-1,k} c_{i_k,j}$ ，或是树上形如  $f_{u,j} = \sum f_{v,k} c_{u_k,j}$ ，带修改
- 由于矩阵乘法满足结合律，故可用线段树维护区间  $C_i$  积，修改为单点修改，查询  $F_n$  即为查询  $1 \sim n$  区间  $C_i$  积，可以支持
- 若在树上，则可用树剖，线段树维护重链  $C_i$  积，轻链暴力转移即可
- 核心仍是  $f_i$  只与  $f_{i-1}$  有关，或是  $f_u$  只与  $f_v$  有关，且其他维度极小

做最好的自己



- 本题中下标均为 1-indexed。

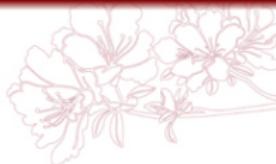
给定长度为  $n$  的字符串  $s$ , 字符集  $\Sigma = \{a, b, \dots, f\}$ 。

有  $q$  次操作, 每次操作对  $s$  进行单点修改。

对于  $i = 1, 2, \dots, q + 1$ , 求出: 进行前  $(i - 1)$  次操作后,  $s$  中满足以下条件的非空子序列  $t$  的数量:

由于答案可能很大, 只需要求出答案对 998 244 353 取模后的结果。

做最好的自己



- 本题中下标均为 1-indexed。

给定长度为  $n$  的字符串  $s$ , 字符集  $\Sigma = \{a, b, \dots, f\}$ 。

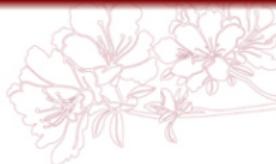
有  $q$  次操作, 每次操作对  $s$  进行单点修改。

对于  $i = 1, 2, \dots, q + 1$ , 求出: 进行前  $(i - 1)$  次操作后,  $s$  中满足以下条件的非空子序列  $t$  的数量:

- $t$  在  $s$  中出现至少两次。

由于答案可能很大, 只需要求出答案对 998 244 353 取模后的结果。

做最好的自己



- 本题中下标均为 1-indexed。

给定长度为  $n$  的字符串  $s$ , 字符集  $\Sigma = \{a, b, \dots, f\}$ 。

有  $q$  次操作, 每次操作对  $s$  进行单点修改。

对于  $i = 1, 2, \dots, q + 1$ , 求出: 进行前  $(i - 1)$  次操作后,  $s$  中满足以下条件的非空子序列  $t$  的数量:

- $t$  在  $s$  中出现至少两次。

由于答案可能很大, 只需要求出答案对  $998\,244\,353$  取模后的结果。

- $3 \leq n \leq 5 \times 10^4$ ;

做最好的自己



- 本题中下标均为 1-indexed。

给定长度为  $n$  的字符串  $s$ , 字符集  $\Sigma = \{a, b, \dots, f\}$ 。

有  $q$  次操作, 每次操作对  $s$  进行单点修改。

对于  $i = 1, 2, \dots, q + 1$ , 求出: 进行前  $(i - 1)$  次操作后,  $s$  中满足以下条件的非空子序列  $t$  的数量:

- $t$  在  $s$  中出现至少两次。

由于答案可能很大, 只需要求出答案对  $998\,244\,353$  取模后的结果。

- $3 \leq n \leq 5 \times 10^4$ ;
- $0 \leq q \leq 5 \times 10^4$ ;

做最好的自己



- 本题中下标均为 1-indexed。

给定长度为  $n$  的字符串  $s$ , 字符集  $\Sigma = \{a, b, \dots, f\}$ 。

有  $q$  次操作, 每次操作对  $s$  进行单点修改。

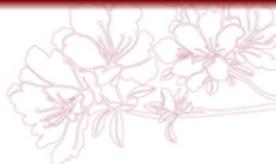
对于  $i = 1, 2, \dots, q + 1$ , 求出: 进行前  $(i - 1)$  次操作后,  $s$  中满足以下条件的非空子序列  $t$  的数量:

- $t$  在  $s$  中出现至少两次。

由于答案可能很大, 只需要求出答案对  $998\,244\,353$  取模后的结果。

- $3 \leq n \leq 5 \times 10^4$ ;
- $0 \leq q \leq 5 \times 10^4$ ;
- $s_i, c_i \in \Sigma = \{a, b, \dots, f\}$ ;

做最好的自己



- 本题中下标均为 1-indexed。

给定长度为  $n$  的字符串  $s$ , 字符集  $\Sigma = \{a, b, \dots, f\}$ 。

有  $q$  次操作, 每次操作对  $s$  进行单点修改。

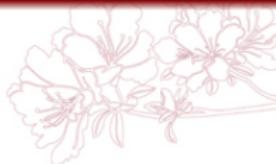
对于  $i = 1, 2, \dots, q + 1$ , 求出: 进行前  $(i - 1)$  次操作后,  $s$  中满足以下条件的非空子序列  $t$  的数量:

- $t$  在  $s$  中出现至少两次。

由于答案可能很大, 只需要求出答案对  $998\,244\,353$  取模后的结果。

- $3 \leq n \leq 5 \times 10^4$ ;
- $0 \leq q \leq 5 \times 10^4$ ;
- $s_i, c_i \in \Sigma = \{a, b, \dots, f\}$ ;
- $1 \leq p_i \leq n$ 。

做最好的自己



- 本题中下标均为 1-indexed。

给定长度为  $n$  的字符串  $s$ , 字符集  $\Sigma = \{a, b, \dots, f\}$ 。

有  $q$  次操作, 每次操作对  $s$  进行单点修改。

对于  $i = 1, 2, \dots, q + 1$ , 求出: 进行前  $(i - 1)$  次操作后,  $s$  中满足以下条件的非空子序列  $t$  的数量:

- $t$  在  $s$  中出现至少两次。

由于答案可能很大, 只需要求出答案对  $998\,244\,353$  取模后的结果。

- $3 \leq n \leq 5 \times 10^4$ ;
- $0 \leq q \leq 5 \times 10^4$ ;
- $s_i, c_i \in \Sigma = \{a, b, \dots, f\}$ ;
- $1 \leq p_i \leq n$ 。
- 题目来源: 洛谷 P11928

做最好的自己

- 出现至少两次不好处理，考虑容斥为出现过-只出现一次



做最好的自己

- 出现至少两次不好处理，考虑容斥为出现过-只出现一次
- 先求出现过的子序列数量，可以设  $f_{i, ch}$  为前  $i$  个字符，最后一个字符为  $ch$  的本质不同子序列个数

做最好的自己



- 出现至少两次不好处理，考虑容斥为出现过-只出现一次
- 先求出现过的子序列数量，可以设  $f_{i, ch}$  为前  $i$  个字符，最后一个字符为  $ch$  的本质不同子序列个数
- 对于  $ch \neq c_i$  的情况，显然结尾为  $ch$  的子序列个数不会变，直接转移即可

做最好的自己



- 出现至少两次不好处理，考虑容斥为出现过-只出现一次
- 先求出现过的子序列数量，可以设  $f_{i, ch}$  为前  $i$  个字符，最后一个字符为  $ch$  的本质不同子序列个数
- 对于  $ch \neq c_i$  的情况，显然结尾为  $ch$  的子序列个数不会变，直接转移即可
- 对于  $ch = c_i$  的情况，考虑所有以  $ch$  结尾的子序列，我们钦定他们最后的  $ch$  一定来自  $c_i$ ，否则将其替换成  $c_i$  没有任何影响

做最好的自己

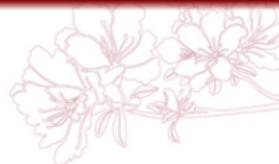
- 出现至少两次不好处理，考虑容斥为出现过-只出现一次
- 先求出现过的子序列数量，可以设  $f_{i, ch}$  为前  $i$  个字符，最后一个字符为  $ch$  的本质不同子序列个数
- 对于  $ch \neq c_i$  的情况，显然结尾为  $ch$  的子序列个数不会变，直接转移即可
- 对于  $ch = c_i$  的情况，考虑所有以  $ch$  结尾的子序列，我们钦定他们最后的  $ch$  一定来自  $c_i$ ，否则将其替换成  $c_i$  没有任何影响
- 而以  $c_i$  结尾的本质不同子序列数量，显然为前面的总子序列数量 +1，即  $f_{i, c_i} = \sum f_{i-1, k} + 1$

做最好的自己



- 出现至少两次不好处理，考虑容斥为出现过-只出现一次
- 先求出现过的子序列数量，可以设  $f_{i, ch}$  为前  $i$  个字符，最后一个字符为  $ch$  的本质不同子序列个数
- 对于  $ch \neq c_i$  的情况，显然结尾为  $ch$  的子序列个数不会变，直接转移即可
- 对于  $ch = c_i$  的情况，考虑所有以  $ch$  结尾的子序列，我们钦定他们最后的  $ch$  一定来自  $c_i$ ，否则将其替换成  $c_i$  没有任何影响
- 而以  $c_i$  结尾的本质不同子序列数量，显然为前面的总子序列数量 +1，即  $f_{i, c_i} = \sum f_{i-1, k} + 1$
- 你也许会考虑直接设  $f_i$  为前  $i$  个字符的本质不同子序列数量，转移时找到上一个和  $i$  字符相同的位置  $j$ ，由  $f_j$  转移得到  $f_i$ ；但注意我们还需要解决修改问题，所以我们应该尽量往动态  $dp$  的形式上去靠，故采用以上方法

做最好的自己



- 我们还需要求只出现一次的子序列数量，这个也很好求，仿照上面的 dp 即可

做最好的自己



- 我们还需要求只出现一次的子序列数量，这个也很好求，仿照上面的 dp 即可
- 设  $g_{i,ch}$  为前  $i$  个字符，最后一个字符为  $ch$  且只出现一次的子序列个数

做最好的自己



- 我们还需要求只出现一次的子序列数量，这个也很好求，仿照上面的 dp 即可
- 设  $g_{i,ch}$  为前  $i$  个字符，最后一个字符为  $ch$  且只出现一次的子序列个数
- 同样的，对于  $ch \neq c_i$ ，直接转移

做最好的自己



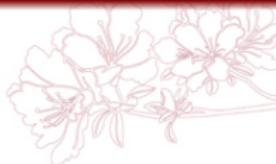
- 我们还需要求只出现一次的子序列数量，这个也很好求，仿照上面的 dp 即可
- 设  $g_{i, ch}$  为前  $i$  个字符，最后一个字符为  $ch$  且只出现一次的子序列个数
- 同样的，对于  $ch \neq c_i$ ，直接转移
- 对于  $ch = c_i$  也一样，以  $ch$  结尾的且在之前只出现一次的子序列数为  $\sum g_{i-1, k} + [c_i \text{ 是前两个 } ch]$ ，但要减去由于  $i$  这个字符导致出现不止一次的数量，也就是前面出现过的以  $ch$  结尾的子序列数量，即  $g_{i-1, ch}$

做最好的自己



- 以上两种 dp 均可以写成  $F_i = F_{i-1} * A_i$  的形式，故可用线段树维护区间转移矩阵  $A_i$  的积，即可在支持修改的同时快速求出  $F_n$

做最好的自己



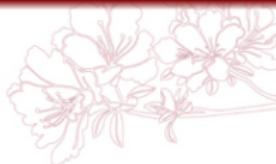
- 以上两种 dp 均可以写成  $F_i = F_{i-1} * A_i$  的形式，故可用线段树维护区间转移矩阵  $A_i$  的积，即可在支持修改的同时快速求出  $F_n$
- 时间复杂度  $O(6^3 n \log n)$

做最好的自己



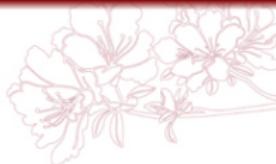
- 一个  $n$  个点的树，点  $i$  只能向  $[i - k, i - 1]$  内的点连边，求有标号生成树的个数

做最好的自己



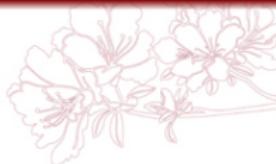
- 一个  $n$  个点的树，点  $i$  只能向  $[i - k, i - 1]$  内的点连边，求有标号生成树的个数
  - $n \leq 10^{15}, k \leq 5$

做最好的自己



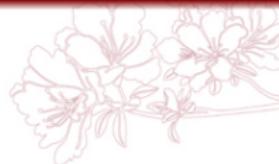
- 一个  $n$  个点的树，点  $i$  只能向  $[i - k, i - 1]$  内的点连边，求有标号生成树的个数
  - $n \leq 10^{15}, k \leq 5$
  - 题目来源：洛谷 P2109

做最好的自己



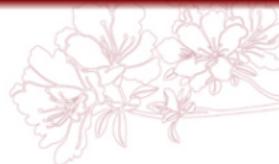
- 设  $f_{i,mask}$  表示 dp 到  $i$ ,  $[i-k, i-1]$  这些点的连通性状态为  $mask$  的方案数，通过搜索可以发现若采用最小表示法， $mask$  只有 52 种可能

做最好的自己



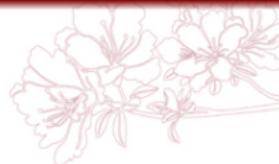
- 设  $f_{i,mask}$  表示 dp 到  $i$ ,  $[i-k, i-1]$  这些点的连通性状态为  $mask$  的方案数, 通过搜索可以发现若采用最小表示法,  $mask$  只有 52 种可能
- 枚举  $i$  连向  $[i-k, i-1]$  的这些边的选择方案即可从  $f_{i-1,mask'}$  转移到  $f_{i,mask}$

做最好的自己



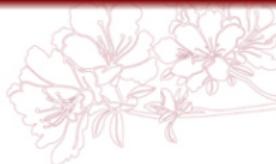
- 设  $f_{i,mask}$  表示 dp 到  $i$ ,  $[i-k, i-1]$  这些点的连通性状态为  $mask$  的方案数, 通过搜索可以发现若采用最小表示法,  $mask$  只有 52 种可能
- 枚举  $i$  连向  $[i-k, i-1]$  的这些边的选择方案即可从  $f_{i-1,mask'}$  转移到  $f_{i,mask}$
- 处理出  $i-1$  转移到  $i$  的转移矩阵, 矩阵快速幂加速即可

做最好的自己



- 设  $f_{i,mask}$  表示 dp 到  $i$ ,  $[i-k, i-1]$  这些点的连通性状态为  $mask$  的方案数, 通过搜索可以发现若采用最小表示法,  $mask$  只有 52 种可能
- 枚举  $i$  连向  $[i-k, i-1]$  的这些边的选择方案即可从  $f_{i-1,mask'}$  转移到  $f_{i,mask}$
- 处理出  $i-1$  转移到  $i$  的转移矩阵, 矩阵快速幂加速即可
- 时间复杂度  $O(52^3 \log n)$

做最好的自己



- 一个序列  $a_1, a_2, \dots, a_n$  是合法的，当且仅当：

一个序列的值定义为它里面所有数的乘积，即

$$a_1 \times a_2 \times \cdots \times a_n。$$

求所有不同合法序列的值的和对  $p$  取模后的结果。两个序列不同当且仅当他们任意一位不同。

做最好的自己



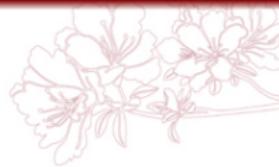
- 一个序列  $a_1, a_2, \dots, a_n$  是合法的，当且仅当：
  - $a_1, a_2, \dots, a_n$  都是  $[1, k]$  中的整数。

一个序列的值定义为它里面所有数的乘积，即

$$a_1 \times a_2 \times \cdots \times a_n.$$

求所有不同合法序列的值的和对  $p$  取模后的结果。两个序列不同当且仅当他们任意一位不同。

做最好的自己



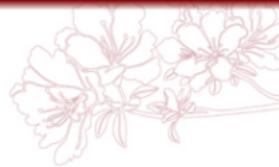
- 一个序列  $a_1, a_2, \dots, a_n$  是合法的，当且仅当：
  - $a_1, a_2, \dots, a_n$  都是  $[1, k]$  中的整数。
  - $a_1, a_2, \dots, a_n$  互不相等。

一个序列的值定义为它里面所有数的乘积，即

$$a_1 \times a_2 \times \cdots \times a_n.$$

求所有不同合法序列的值的和对  $p$  取模后的结果。两个序列不同当且仅当他们任意一位不同。

做最好的自己



- 一个序列  $a_1, a_2, \dots, a_n$  是合法的，当且仅当：
  - $a_1, a_2, \dots, a_n$  都是  $[1, k]$  中的整数。
  - $a_1, a_2, \dots, a_n$  互不相等。

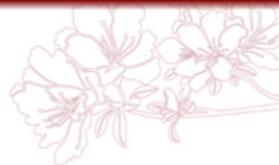
一个序列的值定义为它里面所有数的乘积，即

$$a_1 \times a_2 \times \cdots \times a_n。$$

求所有不同合法序列的值的和对  $p$  取模后的结果。两个序列不同当且仅当他们任意一位不同。

- $k \leq 10^9, n \leq 500, p \leq 10^9$ , 保证  $p$  为素数，保证  $n + 1 < k < p$

做最好的自己



- 一个序列  $a_1, a_2, \dots, a_n$  是合法的，当且仅当：
  - $a_1, a_2, \dots, a_n$  都是  $[1, k]$  中的整数。
  - $a_1, a_2, \dots, a_n$  互不相等。

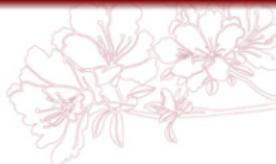
一个序列的值定义为它里面所有数的乘积，即

$$a_1 \times a_2 \times \cdots \times a_n。$$

求所有不同合法序列的值的和对  $p$  取模后的结果。两个序列不同当且仅当他们任意一位不同。

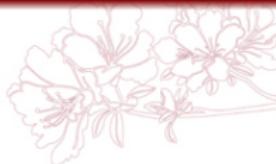
- $k \leq 10^9, n \leq 500, p \leq 10^9$ , 保证  $p$  为素数，保证  $n + 1 < k < p$
- 题目来源：洛谷 P4463

做最好的自己



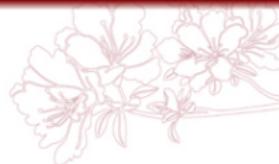
- 注意到  $a_1, a_2, \dots, a_n$  互不相等，且最终代价只和  $a_1, a_2, \dots, a_n$  是哪些数有关，和它们的排列顺序无关，故可强行规定  $a_1 < a_2 < \dots < a_n$ ，最终结果乘上  $n!$  即可

做最好的自己



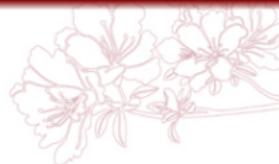
- 注意到  $a_1, a_2, \dots, a_n$  互不相等，且最终代价只和  $a_1, a_2, \dots, a_n$  是哪些数有关，和它们的排列顺序无关，故可强行规定  $a_1 < a_2 < \dots < a_n$ ，最终结果乘上  $n!$  即可
- 这样即可按下标和值域划分阶段了，设  $f_{i,j}$  为选了  $j$  个数，最大的数为  $i$  的总代价，则有  $f_{i,j} = (\sum_{l=1}^{i-1} f_{l,j-1}) * i$

做最好的自己



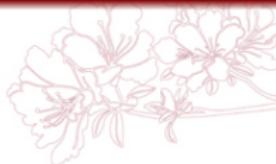
- 注意到  $a_1, a_2, \dots, a_n$  互不相等，且最终代价只和  $a_1, a_2, \dots, a_n$  是哪些数有关，和它们的排列顺序无关，故可强行规定  $a_1 < a_2 < \dots < a_n$ ，最终结果乘上  $n!$  即可
- 这样即可按下标和值域划分阶段了，设  $f_{i,j}$  为选了  $j$  个数，最大的数为  $i$  的总代价，则有  $f_{i,j} = (\sum_{l=1}^{i-1} f_{l,j-1}) * i$
- 或是设  $f_{i,j}$  为前  $i$  个数里选了  $j$  个数的总代价，则有  $f_{i,j} = i * f_{i-1,j-1} + f_{i-1,j}$

做最好的自己



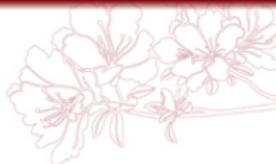
- 有一个结论， $\sum_{i=1}^n i^k$  是  $k+1$  次多项式

做最好的自己



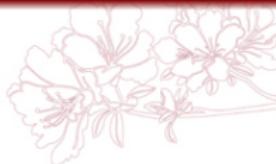
- 有一个结论， $\sum_{i=1}^n i^k$  是  $k+1$  次多项式
- 那么如果是  $f_{i,j} = (\sum_{l=1}^{j-1} f_{l,j-1}) * i$ , 从  $f_{j-1}$  到  $f_j$ , 经过了一次求和和一次乘  $i$ , 次数提高了两级, 故可用归纳法证明  $f_{i,j}$  是关于  $i$  的  $2j+1$  次多项式, 可用拉格朗日插值公式得出

做最好的自己



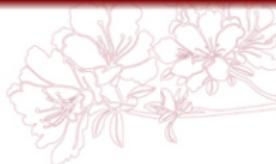
- 有一个结论， $\sum_{i=1}^n i^k$  是  $k+1$  次多项式
- 那么如果是  $f_{i,j} = (\sum_{l=1}^{j-1} f_{l,j-1}) * i$ , 从  $f_{j-1}$  到  $f_j$ , 经过了一次求和和一次乘  $i$ , 次数提高了两级, 故可用归纳法证明  $f_{i,j}$  是关于  $i$  的  $2j+1$  次多项式, 可用拉格朗日插值公式得出
- 时间复杂度  $O(n^2)$

做最好的自己



- 与此同时，若一个函数差分后是  $k$  次多项式，则其为  $k+1$  次多项式

做最好的自己



- 与此同时，若一个函数差分后是  $k$  次多项式，则其为  $k+1$  次多项式
- 那么如果是  $f_{i,j} = i * f_{i-1,j-1} + f_{i-1,j}$ ，则有  
 $f_{i,j} - f_{i-1,j} = i * f_{i-1,j-1}$ ，同样的，从  $f_{j-1}$  到  $f_j$ ，经过了一次差分和一次乘  $i$ ，故  $f_{i,j}$  同样是关于  $i$  的  $2j+1$  次多项式，同样使用拉格朗日插值公式即可

做最好的自己



- 庞博士有 3 项技能：喝汽水、猎狐和炒股，编号分别为 1, 2, 3。初始时，每项技能的熟练度为 0。

接下来有  $n$  天。在第  $i$  天，庞博士可以选择一项技能（假设是第  $j$  项）进行练习，然后在这天结束时让这项技能的熟练度增加  $a_{i,j}$  ( $0 \leq a_{i,j} \leq 10000$ )。同时，如果某一项技能（假设是第  $k$  项）已经有  $x$  天没有练习，那么在这天结束时，这项技能的熟练度会减少  $x$ 。当然，任何一项技能的熟练度都不可能小于 0。

现在，庞博士想知道：在第  $n$  天结束后，这 3 项技能的熟练度之和最大为多少。由于他非常忙，而且他的日程和对习惯的适应程度可能有变，所以庞博士把这  $T$  个问题交给了你。

做最好的自己



- 庞博士有 3 项技能：喝汽水、猎狐和炒股，编号分别为 1, 2, 3。初始时，每项技能的熟练度为 0。

接下来有  $n$  天。在第  $i$  天，庞博士可以选择一项技能（假设是第  $j$  项）进行练习，然后在这天结束时让这项技能的熟练度增加  $a_{i,j}$  ( $0 \leq a_{i,j} \leq 10000$ )。同时，如果某一项技能（假设是第  $k$  项）已经有  $x$  天没有练习，那么在这天结束时，这项技能的熟练度会减少  $x$ 。当然，任何一项技能的熟练度都不可能小于 0。

现在，庞博士想知道：在第  $n$  天结束后，这 3 项技能的熟练度之和最大为多少。由于他非常忙，而且他的日程和对习惯的适应程度可能有变，所以庞博士把这  $T$  个问题交给了你。

- $1 \leq T \leq 1000$ ,  $1 \leq n \leq 1000$ , 数据保证单个测试点内的  $\sum n$  不超过 1000。

做最好的自己



- 庞博士有 3 项技能：喝汽水、猎狐和炒股，编号分别为 1, 2, 3。初始时，每项技能的熟练度为 0。

接下来有  $n$  天。在第  $i$  天，庞博士可以选择一项技能（假设是第  $j$  项）进行练习，然后在这天结束时让这项技能的熟练度增加  $a_{i,j}$  ( $0 \leq a_{i,j} \leq 10000$ )。同时，如果某一项技能（假设是第  $k$  项）已经有  $x$  天没有练习，那么在这天结束时，这项技能的熟练度会减少  $x$ 。当然，任何一项技能的熟练度都不可能小于 0。

现在，庞博士想知道：在第  $n$  天结束后，这 3 项技能的熟练度之和最大为多少。由于他非常忙，而且他的日程和对习惯的适应程度可能有变，所以庞博士把这  $T$  个问题交给了你。

- $1 \leq T \leq 1000$ ,  $1 \leq n \leq 1000$ , 数据保证单个测试点内的  $\sum n$  不超过 1000。
- 题目来源：洛谷 P9676

做最好的自己



- 鲶鱼塘是由  $N \times N$  个网格单元构成的池塘。池塘里总共有  $M$  条鲶鱼，鲶鱼  $i$  在单元  $(X_i, Y_i)$  中，其重量为  $W_i$  克。  
Bu Dengklek 想造些长堤来抓鲶鱼。在第  $c$  列中长度为  $k$  的长堤（对于所有  $0 \leq c \leq N - 1$  和  $1 \leq k \leq N$ ），是一个从第 0 行跨到第  $k - 1$  行的矩形，盖住单元  $(c, 0), (c, 1), \dots, (c, k - 1)$ 。对于每一列，Bu Dengklek 可以按照她自己选择的某个长度造长堤，也可以不造。  
如果有某个长堤紧邻它的西侧或东侧，而且没有长堤盖住它所在的单元，鲶鱼  $i$ （对所有满足  $0 \leq i \leq M - 1$  的  $i$ ）就能被抓住；也就是说，如果

鲶鱼  $i$ （对所有满足  $0 \leq i \leq M - 1$  的  $i$ ）就能被抓住

做最好的自己



- 鲶鱼塘是由  $N \times N$  个网格单元构成的池塘。池塘里总共有  $M$  条鲶鱼，鲶鱼  $i$  在单元  $(X_i, Y_i)$  中，其重量为  $W_i$  克。  
Bu Dengklek 想造些长堤来抓鲶鱼。在第  $c$  列中长度为  $k$  的长堤（对于所有  $0 \leq c \leq N - 1$  和  $1 \leq k \leq N$ ），是一个从第 0 行跨到第  $k - 1$  行的矩形，盖住单元  $(c, 0), (c, 1), \dots, (c, k - 1)$ 。对于每一列，Bu Dengklek 可以按照她自己选择的某个长度造长堤，也可以不造。  
如果有某个长堤紧邻它的西侧或东侧，而且没有长堤盖住它所在的单元，鲶鱼  $i$ （对所有满足  $0 \leq i \leq M - 1$  的  $i$ ）就能被抓住；也就是说，如果
  - 单元  $(X_i - 1, Y_i)$  或  $(X_i + 1, Y_i)$  中 至少有一个被某个长堤盖住

鲶鱼  $i$ （对所有满足  $0 \leq i \leq M - 1$  的  $i$ ）就能被抓住

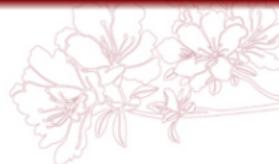
做最好的自己



- 鲶鱼塘是由  $N \times N$  个网格单元构成的池塘。池塘里总共有  $M$  条鲶鱼，鲶鱼  $i$  在单元  $(X_i, Y_i)$  中，其重量为  $W_i$  克。  
Bu Dengklek 想造些长堤来抓鲶鱼。在第  $c$  列中长度为  $k$  的长堤（对于所有  $0 \leq c \leq N - 1$  和  $1 \leq k \leq N$ ），是一个从第 0 行跨到第  $k - 1$  行的矩形，盖住单元  $(c, 0), (c, 1), \dots, (c, k - 1)$ 。对于每一列，Bu Dengklek 可以按照她自己选择的某个长度造长堤，也可以不造。  
如果有某个长堤紧邻它的西侧或东侧，而且没有长堤盖住它所在的单元，鲶鱼  $i$ （对所有满足  $0 \leq i \leq M - 1$  的  $i$ ）就能被抓住；也就是说，如果
  - 单元  $(X_i - 1, Y_i)$  或  $(X_i + 1, Y_i)$  中 至少有一个被某个长堤盖住
  - 没有长堤盖住单元  $(X_i, Y_i)$ 。

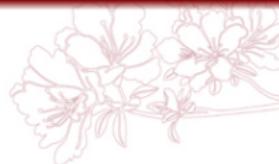
鲶鱼  $i$ （对所有满足  $0 \leq i \leq M - 1$  的  $i$ ）就能被抓住

做最好的自己



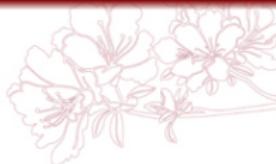
- Bu Dengklek 希望造出来的长堤能让被抓住的鲶鱼的总重量尽量大。你的任务是求出 Bu Dengklek 通过造长堤能抓住的鲶鱼的最大总重量。

做最好的自己



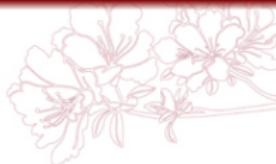
- Bu Dengklek 希望造出来的长堤能让被抓住的鲶鱼的总重量尽量大。你的任务是求出 Bu Dengklek 通过造长堤能抓住的鲶鱼的最大总重量。
  - $2 \leq N \leq 100\,000$

做最好的自己



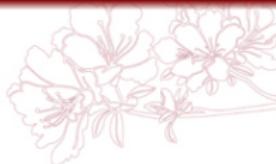
- Bu Dengklek 希望造出来的长堤能让被抓住的鲶鱼的总重量尽量大。你的任务是求出 Bu Dengklek 通过造长堤能抓住的鲶鱼的最大总重量。
  - $2 \leq N \leq 100\,000$
  - $1 \leq M \leq 300\,000$

做最好的自己



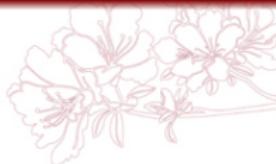
- Bu Dengklek 希望造出来的长堤能让被抓住的鲶鱼的总重量尽量大。你的任务是求出 Bu Dengklek 通过造长堤能抓住的鲶鱼的最大总重量。
  - $2 \leq N \leq 100\,000$
  - $1 \leq M \leq 300\,000$
  - $0 \leq X_i \leq N - 1, 0 \leq Y_i \leq N - 1$  (对所有满足  $0 \leq i \leq M - 1$  的  $i$ )

做最好的自己



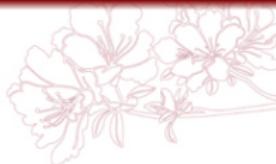
- Bu Dengklek 希望造出来的长堤能让被抓住的鲶鱼的总重量尽量大。你的任务是求出 Bu Dengklek 通过造长堤能抓住的鲶鱼的最大总重量。
  - $2 \leq N \leq 100\,000$
  - $1 \leq M \leq 300\,000$
  - $0 \leq X_i \leq N - 1, 0 \leq Y_i \leq N - 1$  (对所有满足  $0 \leq i \leq M - 1$  的  $i$ )
  - $1 \leq W_i \leq 10^9$  (对所有满足  $0 \leq i \leq M - 1$  的  $i$ )

做最好的自己



- Bu Dengklek 希望造出来的长堤能让被抓住的鲶鱼的总重量尽量大。你的任务是求出 Bu Dengklek 通过造长堤能抓住的鲶鱼的最大总重量。
  - $2 \leq N \leq 100\,000$
  - $1 \leq M \leq 300\,000$
  - $0 \leq X_i \leq N - 1, 0 \leq Y_i \leq N - 1$  (对所有满足  $0 \leq i \leq M - 1$  的  $i$ )
  - $1 \leq W_i \leq 10^9$  (对所有满足  $0 \leq i \leq M - 1$  的  $i$ )
  - 任意两条鲶鱼都不会在同一单元中。换句话说,  $X_i \neq X[j]$  或  $Y_i \neq Y[j]$  (对于所有满足  $0 \leq i < j \leq M - 1$  的  $i$  和  $j$ )。

做最好的自己



- Bu Dengklek 希望造出来的长堤能让被抓住的鲶鱼的总重量尽量大。你的任务是求出 Bu Dengklek 通过造长堤能抓住的鲶鱼的最大总重量。
  - $2 \leq N \leq 100\,000$
  - $1 \leq M \leq 300\,000$
  - $0 \leq X_i \leq N - 1, 0 \leq Y_i \leq N - 1$  (对所有满足  $0 \leq i \leq M - 1$  的  $i$ )
  - $1 \leq W_i \leq 10^9$  (对所有满足  $0 \leq i \leq M - 1$  的  $i$ )
  - 任意两条鲶鱼都不会在同一单元中。换句话说,  $X_i \neq X[j]$  或  $Y_i \neq Y[j]$  (对于所有满足  $0 \leq i < j \leq M - 1$  的  $i$  和  $j$ )。
  - 题目来源: 洛谷 P8490

做最好的自己



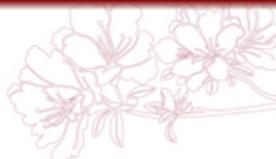
- 如果直接 dp，dp 到第  $i$  列时需记录第  $i$  列墙高  $j$  以及第  $i-1$  列墙高  $k$ ，这样枚举第  $i+1$  列墙高  $l$  后才能知道第  $i$  列鱼的贡献，进行转移，这样做时间复杂度是  $O(n^4)$  的

做最好的自己



- 如果直接 dp，dp 到第  $i$  列时需记录第  $i$  列墙高  $j$  以及第  $i-1$  列墙高  $k$ ，这样枚举第  $i+1$  列墙高  $l$  后才能知道第  $i$  列鱼的贡献，进行转移，这样做时间复杂度是  $O(n^4)$  的
- 对于一列鱼，如果有一个更高的鱼  $j$  是被左边的长堤抓住的，对于一个更矮的鱼  $j'$ ，我们也可以钦定它是被左边的长堤抓住的；也就是说，我们可以钦定一列鱼一定都是由某一侧的长堤抓住的，不会影响答案

做最好的自己



- 如果直接 dp，dp 到第  $i$  列时需记录第  $i$  列墙高  $j$  以及第  $i-1$  列墙高  $k$ ，这样枚举第  $i+1$  列墙高  $l$  后才能知道第  $i$  列鱼的贡献，进行转移，这样做时间复杂度是  $O(n^4)$  的
- 对于一列鱼，如果有一个更高的鱼  $j$  是被左边的长堤抓住的，对于一个更矮的鱼  $j'$ ，我们也可以钦定它是被左边的长堤抓住的；也就是说，我们可以钦定一列鱼一定都是由某一侧的长堤抓住的，不会影响答案
- 这样我们就可以直接记录上一列墙高以及一个 0/1 表示上一列依赖于左右哪一列，枚举这一列状态即可转移，时间复杂度  $O(n^3)$

做最好的自己



- 由于池塘大小是  $O(n^2)$  的，而鱼只有  $m$  条，这就启示我们有效状态数可能很少

做最好的自己



- 由于池塘大小是  $O(n^2)$  的，而鱼只有  $m$  条，这就启示我们有效状态数可能很少
- 更进一步我们发现，墙高只能是  $n$  或是在某条鱼下方，故状态数可优化到  $O(n)$  级别

做最好的自己



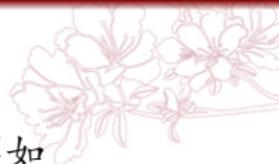
- 由于池塘大小是  $O(n^2)$  的，而鱼只有  $m$  条，这就启示我们有效状态数可能很少
- 更进一步我们发现，墙高只能是  $n$  或是在某条鱼下方，故状态数可优化到  $O(n)$  级别
- 对于转移，由于转移贡献不同，我们将高到低和低到高两种转移分开考虑

做最好的自己



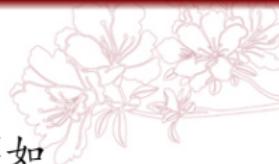
- 由于池塘大小是  $O(n^2)$  的，而鱼只有  $m$  条，这就启示我们有效状态数可能很少
- 更进一步我们发现，墙高只能是  $n$  或是在某条鱼下方，故状态数可优化到  $O(n)$  级别
- 对于转移，由于转移贡献不同，我们将高到低和低到高两种转移分开考虑
- 若低到高且上一列不依赖于当前列，则中间不会产生新的贡献，双指针得到上一列最后一个高度低于当前高度的位置，并记录前缀  $\max$  即可转移；高到低同理

做最好的自己



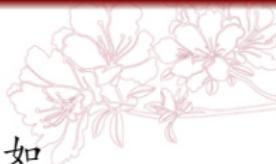
- 若低到高转移，且上一列依赖于当前列，则转移形如 $f_i \leftarrow f_j + sw_k - sw_j$ ，其中 k 是上一列最后一个高度低于当前高度的位置，sw 为 w 的前缀和

做最好的自己



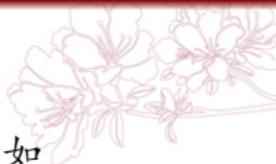
- 若低到高转移，且上一列依赖于当前列，则转移形如 $f_i \leftarrow f_j + sw_k - sw_j$ ，其中  $k$  是上一列最后一个高度低于当前高度的位置， $sw$  为  $w$  的前缀和
- 故我们仍只需双指针得到  $k$ ，并维护前缀  $f_j - sw_j$  的最大值即可

做最好的自己



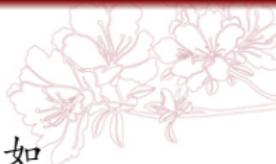
- 若低到高转移，且上一列依赖于当前列，则转移形如 $f_i \leftarrow f_j + sw_k - sw_j$ ，其中 k 是上一列最后一个高度低于当前高度的位置，sw 为 w 的前缀和
- 故我们仍只需双指针得到 k，并维护前缀  $f_j - sw_j$  的最大值即可
- 若高到低转移，且这一列依赖于上一列，则转移形如 $f_i \leftarrow f_j + sw_{i,k} - sw_{i,i}$ ，其中 k 是这一列最后一个高度低于 j 的位置

做最好的自己



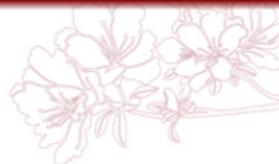
- 若低到高转移，且上一列依赖于当前列，则转移形如 $f_i \leftarrow f_j + sw_k - sw_j$ ，其中 k 是上一列最后一个高度低于当前高度的位置，sw 为 w 的前缀和
- 故我们仍只需双指针得到 k，并维护前缀  $f_j - sw_j$  的最大值即可
- 若高到低转移，且这一列依赖于上一列，则转移形如 $f_i \leftarrow f_j + sw_{i,k} - sw_{i,i}$ ，其中 k 是这一列最后一个高度低于 j 的位置
- 我们仍使用双指针，从高到低扫，维护比当前位置高的所有 j 的  $f_j + sw_{i,k}$  最大值，遇到 i 时再更新  $f_i$

做最好的自己



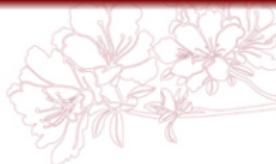
- 若低到高转移，且上一列依赖于当前列，则转移形如 $f_i \leftarrow f_j + sw_k - sw_j$ ，其中 k 是上一列最后一个高度低于当前高度的位置，sw 为 w 的前缀和
- 故我们仍只需双指针得到 k，并维护前缀  $f_j - sw_j$  的最大值即可
- 若高到低转移，且这一列依赖于上一列，则转移形如 $f_i \leftarrow f_j + sw_{i,k} - sw_{i,i}$ ，其中 k 是这一列最后一个高度低于 j 的位置
- 我们仍使用双指针，从高到低扫，维护比当前位置高的所有 j 的  $f_j + sw_{i,k}$  最大值，遇到 i 时再更新  $f_i$
- 这样即可在  $O(n+m)$  的总复杂度下解决此题

做最好的自己



- 注 1：当  $i$  所在列和  $i-1$  所在列不紧贴时需要特殊处理，在代码上需要注意一下

做最好的自己



- 注 1：当  $i$  所在列和  $i-1$  所在列不紧贴时需要特殊处理，在代码上需要注意一下
- 注 2：使用 `lower_bound` 以及线段树等数据结构可快速完成转移，无需双指针前缀和以及分类讨论，会好写不少，虽时间复杂度带  $\log$ ，仍可以通过本题。这里为给出最优解故未详述

做最好的自己



- 给定一个正  $n$  边形的蛋糕和  $m$  条不相交的对角线切分，这些切分将蛋糕分成  $m+1$  个切片。

计算将原始蛋糕每个顶点用  $k$  种颜色之一着色的方案数，要求最终切片中任何相邻顶点颜色不同。两个顶点被认为是相邻的，如果它们在原始蛋糕中是连续的，或者是同一条切分线的端点。不需要使用所有颜色。由于方案数可能很大，请输出其对 998 244 353 取模的结果。

做最好的自己



- 给定一个正  $n$  边形的蛋糕和  $m$  条不相交的对角线切分，这些切分将蛋糕分成  $m + 1$  个切片。

计算将原始蛋糕每个顶点用  $k$  种颜色之一着色的方案数，要求最终切片中任何相邻顶点颜色不同。两个顶点被认为是相邻的，如果它们在原始蛋糕中是连续的，或者是同一条切分线的端点。不需要使用所有颜色。由于方案数可能很大，请输出其对 998 244 353 取模的结果。

- $3 \leq n \leq 10^9$ ;  $0 \leq m \leq 2 \cdot 10^5$ ;  $2 \leq k \leq 10^6$ , 数据组数  $t$  满足  $1 \leq t \leq 10^4$ , 所有测试用例的  $m$  之和不超过  $2 \cdot 10^5$

做最好的自己



- 给定一个正  $n$  边形的蛋糕和  $m$  条不相交的对角线切分，这些切分将蛋糕分成  $m + 1$  个切片。

计算将原始蛋糕每个顶点用  $k$  种颜色之一着色的方案数，要求最终切片中任何相邻顶点颜色不同。两个顶点被认为是相邻的，如果它们在原始蛋糕中是连续的，或者是同一条切分线的端点。不需要使用所有颜色。由于方案数可能很大，请输出其对 998 244 353 取模的结果。

- $3 \leq n \leq 10^9$ ;  $0 \leq m \leq 2 \cdot 10^5$ ;  $2 \leq k \leq 10^6$ , 数据组数  $t$  满足  $1 \leq t \leq 10^4$ , 所有测试用例的  $m$  之和不超过  $2 \cdot 10^5$
- 题目来源：洛谷 P12107

做最好的自己



- 由两两弦不相交可知，它们代表的区间要么相离要么包含，故若设某个区间的染色方案数为  $f_i$ ，则  $f_i$  可尝试通过它的所有直接子区间  $j$  的 dp 值  $f_j$  转移得来

做最好的自己



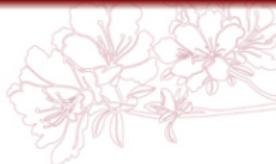
- 由两两弦不相交可知，它们代表的区间要么相离要么包含，故若设某个区间的染色方案数为  $f_i$ ，则  $f_i$  可尝试通过它的所有直接子区间  $j$  的 dp 值  $f_j$  转移得来
- 故若我们能够得到所有极小区间的 dp 值，以及从小区间转移到大区间的方式，我们即可求出答案

做最好的自己



- 首先需要得到极小区间的 dp 值，极小区间显然是个环，对于环染色的方案数有一种经典方法，即设  $f_{i,0/1}$  表示 i 个点，第一个点和最后一个点异色/同色的方案数，则有：

做最好的自己



- 首先需要得到极小区间的 dp 值，极小区间显然是个环，对于环染色的方案数有一种经典方法，即设  $f_{i,0/1}$  表示 i 个点，第一个点和最后一个点异色/同色的方案数，则有：
- $$\begin{cases} f_{i,0} = (k - 2)f_{i-1,0} + (k - 1)f_{i-1,1} \\ f_{i,1} = f_{i-1,0} \end{cases}$$

做最好的自己



- 首先需要得到极小区间的 dp 值，极小区间显然是个环，对于环染色的方案数有一种经典方法，即设  $f_{i,0/1}$  表示 i 个点，第一个点和最后一个点异色/同色的方案数，则有：
- $$\begin{cases} f_{i,0} = (k - 2)f_{i-1,0} + (k - 1)f_{i-1,1} \\ f_{i,1} = f_{i-1,0} \end{cases}$$
- 矩阵加速转移即可

做最好的自己



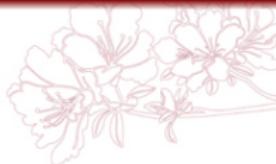
- 接下来我们考虑如何从小区间转移到大区间

做最好的自己



- 接下来我们考虑如何从小区间转移到大区间
- 我们按顺序一个一个考虑所有小区间，设  $f_{i',0/1}$  为前面所有小区间首尾两点同色/异色的方案数， $f_{j,0/1}$  表示新小区间首尾同色/异色方案数， $f_{i,0/1}$  为加入新小区间后首尾两点同色/异色的方案数，则有：

做最好的自己



- 接下来我们考虑如何从小区间转移到大区间
- 我们按顺序一个一个考虑所有小区间，设  $f_{i',0/1}$  为前面所有小区间首尾两点同色/异色的方案数， $f_{j,0/1}$  表示新小区间首尾同色/异色方案数， $f_{i,0/1}$  为加入新小区间后首尾两点同色/异色的方案数，则有：
- $$\begin{cases} f_{i,1} = \frac{f_{i',1} * f_{j,1}}{k} + \frac{f_{i',0} * f_{j,0}}{k(k-1)} \\ f_{i,0} = \frac{(f_{i',0} + f_{i',1}) * (f_{j,0} + f_{j,1})}{k} - f_{i,1} \end{cases}$$

做最好的自己



- 接下来我们考虑如何从小区间转移到大区间
- 我们按顺序一个一个考虑所有小区间，设  $f_{i',0/1}$  为前面所有小区间首尾两点同色/异色的方案数， $f_{j,0/1}$  表示新小区间首尾同色/异色方案数， $f_{i,0/1}$  为加入新小区间后首尾两点同色/异色的方案数，则有：
- $$\begin{cases} f_{i,1} = \frac{f_{i',1}*f_{j,1}}{k} + \frac{f_{i',0}*f_{j,0}}{k(k-1)} \\ f_{i,0} = \frac{(f_{i',0}+f_{i',1})*(f_{j,0}+f_{j,1})}{k} - f_{i,1} \end{cases}$$
- 对所有小区间暴力，对  $n$  边形的每条边矩阵快速幂加速即可，时间复杂度  $O(m \log n)$

做最好的自己



- 我们还需要对每个区间找到直接包含它的区间，这个也很简单，大家自己也能解决，这里给出一种做法

做最好的自己



- 我们还需要对每个区间找到直接包含它的区间，这个也很简单，大家自己也能解决，这里给出一种做法
- 我们可以用一个单调队列维护所有包含当前区间的区间，当我们扫到一个区间，我们先把尾部所有已经不包含当前区间的区间弹出，记录直接包含当前区间的区间，再插入当前区间；由于区间间只有相离或包含，故不包含当前区间的区间一定都在尾部

做最好的自己



- 我们还需要对每个区间找到直接包含它的区间，这个也很简单，大家自己也能解决，这里给出一种做法
- 我们可以用一个单调队列维护所有包含当前区间的区间，当我们扫到一个区间，我们先把尾部所有已经不包含当前区间的区间弹出，记录直接包含当前区间的区间，再插入当前区间；由于区间间只有相离或包含，故不包含当前区间的区间一定都在尾部
- 由于每个区间只会被弹出一次，这样做显然是  $O(m)$  的

做最好的自己

- 若直接 dp，可设  $f_{i,j,k,l}$  表示到第  $i$  天，当前天练习的项目是  $j$ ，另外两个项目上一次练习是  $k$  天前和  $l$  天前，时间复杂度  $O(n^3)$

做最好的自己



- 若直接 dp，可设  $f_{i,j,k,l}$  表示到第  $i$  天，当前天练习的项目是  $j$ ，另外两个项目上一次练习是  $k$  天前和  $l$  天前，时间复杂度  $O(n^3)$
- 考虑优化，我们观察到  $a$  并不大，且熟练度的下降速度随着懈怠天数是平方级别的，故猜测上一次练习必定是在  $O(\sqrt{a})$  天之内的

做最好的自己



- 若直接 dp，可设  $f_{i,j,k,l}$  表示到第  $i$  天，当前天练习的项目是  $j$ ，另外两个项目上一次练习是  $k$  天前和  $l$  天前，时间复杂度  $O(n^3)$
- 考虑优化，我们观察到  $a$  并不大，且熟练度的下降速度随着懈怠天数是平方级别的，故猜测上一次练习必定是在  $O(\sqrt{a})$  天之内的
- 事实也是如此，若你持续  $2\sqrt{a}$  天不练习某个项目，你至少会亏损  $1 + 2 + \dots + 2\sqrt{a} = 2a$  的熟练度；而若你在其中第  $\sqrt{a}$  天选择练习这个项目，你只会亏损  $2 * (1 + 2 + \dots + \sqrt{a}) = a$  的熟练度；故你只要开始练习某个项目，上一次练习一定在  $2\sqrt{a} + O(1)$  天之内

做最好的自己



- 若直接 dp，可设  $f_{i,j,k,l}$  表示到第  $i$  天，当前天练习的项目是  $j$ ，另外两个项目上一次练习是  $k$  天前和  $l$  天前，时间复杂度  $O(n^3)$
- 考虑优化，我们观察到  $a$  并不大，且熟练度的下降速度随着懈怠天数是平方级别的，故猜测上一次练习必定是在  $O(\sqrt{a})$  天之内的
- 事实也是如此，若你持续  $2\sqrt{a}$  天不练习某个项目，你至少会亏损  $1 + 2 + \dots + 2\sqrt{a} = 2a$  的熟练度；而若你在其中第  $\sqrt{a}$  天选择练习这个项目，你只会亏损  $2 * (1 + 2 + \dots + \sqrt{a}) = a$  的熟练度；故你只要开始练习某个项目，上一次练习一定在  $2\sqrt{a} + O(1)$  天之内
- 故  $k,l$  两维大小是  $O(\sqrt{a})$  的，故总复杂度是  $O(na)$  的；空间使用滚动数组即可做到  $O(a)$

做最好的自己

- 数轴上总计有  $N$  ( $1 \leq N \leq 5000$ ) 头奶牛，每一头奶牛都是荷斯坦牛 (Holstein) 或更赛牛 (Guernsey) 之一。第  $i$  头奶牛的品种为  $b_i \in \{H, G\}$ ，第  $i$  头奶牛的位置为  $x_i$  ( $0 \leq x_i \leq 10^9$ )，而第  $i$  头奶牛的重量为  $y_i$  ( $1 \leq y_i \leq 10^5$ )。根据 Farmer John 的信号，某些奶牛会组成对，使得

你需求求出未配对的奶牛的重量之和的可能的范围。具体地说，

做最好的自己

- 数轴上总计有  $N$  ( $1 \leq N \leq 5000$ ) 头奶牛，每一头奶牛都是荷斯坦牛 (Holstein) 或更赛牛 (Guernsey) 之一。第  $i$  头奶牛的品种为  $b_i \in \{H, G\}$ ，第  $i$  头奶牛的位置为  $x_i$  ( $0 \leq x_i \leq 10^9$ )，而第  $i$  头奶牛的重量为  $y_i$  ( $1 \leq y_i \leq 10^5$ )。根据 Farmer John 的信号，某些奶牛会组成对，使得
  - 每一对包含位置相差不超过  $K$  的一头荷斯坦牛  $h$  和一头更赛牛  $g$  ( $1 \leq K \leq 10^9$ )；也就是说， $|x_h - x_g| \leq K$ 。

你需求求出未配对的奶牛的重量之和的可能的范围。具体地说，

做最好的自己

- 数轴上总计有  $N$  ( $1 \leq N \leq 5000$ ) 头奶牛，每一头奶牛都是荷斯坦牛 (Holstein) 或更赛牛 (Guernsey) 之一。第  $i$  头奶牛的品种为  $b_i \in \{H, G\}$ ，第  $i$  头奶牛的位置为  $x_i$  ( $0 \leq x_i \leq 10^9$ )，而第  $i$  头奶牛的重量为  $y_i$  ( $1 \leq y_i \leq 10^5$ )。根据 Farmer John 的信号，某些奶牛会组成对，使得
  - 每一对包含位置相差不超过  $K$  的一头荷斯坦牛  $h$  和一头更赛牛  $g$  ( $1 \leq K \leq 10^9$ )；也就是说， $|x_h - x_g| \leq K$ 。
  - 每一头奶牛要么包含在恰好一对中，要么不属于任何一对。

你需求求出未配对的奶牛的重量之和的可能的范围。具体地说，

做最好的自己



- 数轴上总计有  $N$  ( $1 \leq N \leq 5000$ ) 头奶牛，每一头奶牛都是荷斯坦牛 (Holstein) 或更赛牛 (Guernsey) 之一。第  $i$  头奶牛的品种为  $b_i \in \{H, G\}$ ，第  $i$  头奶牛的位置为  $x_i$  ( $0 \leq x_i \leq 10^9$ )，而第  $i$  头奶牛的重量为  $y_i$  ( $1 \leq y_i \leq 10^5$ )。根据 Farmer John 的信号，某些奶牛会组成对，使得
  - 每一对包含位置相差不超过  $K$  的一头荷斯坦牛  $h$  和一头更赛牛  $g$  ( $1 \leq K \leq 10^9$ )；也就是说， $|x_h - x_g| \leq K$ 。
  - 每一头奶牛要么包含在恰好一对中，要么不属于任何一对。
  - 配对是极大的；也就是说，没有两头未配对的奶牛可以组成对。

你只需要求出未配对的奶牛的重量之和的可能的范围。具体地说，

做最好的自己

- 数轴上总计有  $N$  ( $1 \leq N \leq 5000$ ) 头奶牛，每一头奶牛都是荷斯坦牛 (Holstein) 或更赛牛 (Guernsey) 之一。第  $i$  头奶牛的品种为  $b_i \in \{H, G\}$ ，第  $i$  头奶牛的位置为  $x_i$  ( $0 \leq x_i \leq 10^9$ )，而第  $i$  头奶牛的重量为  $y_i$  ( $1 \leq y_i \leq 10^5$ )。根据 Farmer John 的信号，某些奶牛会组成对，使得

- 每一对包含位置相差不超过  $K$  的一头荷斯坦牛  $h$  和一头更赛牛  $g$  ( $1 \leq K \leq 10^9$ )；也就是说， $|x_h - x_g| \leq K$ 。
- 每一头奶牛要么包含在恰好一对中，要么不属于任何一对。
- 配对是极大的；也就是说，没有两头未配对的奶牛可以组成对。

你需要求出未配对的奶牛的重量之和的可能的范围。具体地说，

- 如果  $T = 1$ ，计算未配对的奶牛的最小重量和。

做最好的自己

- 数轴上总计有  $N$  ( $1 \leq N \leq 5000$ ) 头奶牛，每一头奶牛都是荷斯坦牛 (Holstein) 或更赛牛 (Guernsey) 之一。第  $i$  头奶牛的品种为  $b_i \in \{H, G\}$ ，第  $i$  头奶牛的位置为  $x_i$  ( $0 \leq x_i \leq 10^9$ )，而第  $i$  头奶牛的重量为  $y_i$  ( $1 \leq y_i \leq 10^5$ )。根据 Farmer John 的信号，某些奶牛会组成对，使得

- 每一对包含位置相差不超过  $K$  的一头荷斯坦牛  $h$  和一头更赛牛  $g$  ( $1 \leq K \leq 10^9$ )；也就是说， $|x_h - x_g| \leq K$ 。
- 每一头奶牛要么包含在恰好一对中，要么不属于任何一对。
- 配对是极大的；也就是说，没有两头未配对的奶牛可以组成对。

你需要求出未配对的奶牛的重量之和的可能的范围。具体地说，

- 如果  $T = 1$ ，计算未配对的奶牛的最小重量和。
- 如果  $T = 2$ ，计算未配对的奶牛的最大重量和。

做最好的自己

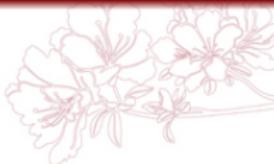
- 数轴上总计有  $N$  ( $1 \leq N \leq 5000$ ) 头奶牛，每一头奶牛都是荷斯坦牛 (Holstein) 或更赛牛 (Guernsey) 之一。第  $i$  头奶牛的品种为  $b_i \in \{H, G\}$ ，第  $i$  头奶牛的位置为  $x_i$  ( $0 \leq x_i \leq 10^9$ )，而第  $i$  头奶牛的重量为  $y_i$  ( $1 \leq y_i \leq 10^5$ )。根据 Farmer John 的信号，某些奶牛会组成对，使得

- 每一对包含位置相差不超过  $K$  的一头荷斯坦牛  $h$  和一头更赛牛  $g$  ( $1 \leq K \leq 10^9$ )；也就是说， $|x_h - x_g| \leq K$ 。
- 每一头奶牛要么包含在恰好一对中，要么不属于任何一对。
- 配对是极大的；也就是说，没有两头未配对的奶牛可以组成对。

你需要求出未配对的奶牛的重量之和的可能的范围。具体地说，

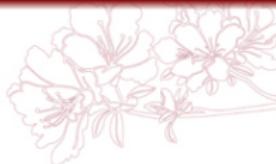
- 如果  $T = 1$ ，计算未配对的奶牛的最小重量和。
- 如果  $T = 2$ ，计算未配对的奶牛的最大重量和。
- 题目来源：洛谷 P7985

做最好的自己



- 先考虑第一问，由于使得未配对奶牛重量和最小的匹配一定是极大匹配，故无需考虑该条件

做最好的自己



- 先考虑第一问，由于使得未配对奶牛重量和最小的匹配一定是极大匹配，故无需考虑该条件
- 故我们只需最大化匹配奶牛的重量和，容易发现若两组匹配相交则可改成两组不相交的匹配，故可默认所有匹配都不相交

做最好的自己



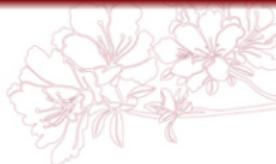
- 先考虑第一问，由于使得未配对奶牛重量和最小的匹配一定是极大匹配，故无需考虑该条件
- 故我们只需最大化匹配奶牛的重量和，容易发现若两组匹配相交则可改成两组不相交的匹配，故可默认所有匹配都不相交
- 这类似一个经典问题 (LCS)，故易想到设  $f_{i,j}$  为前 i 头 H 奶牛和前 j 头 G 奶牛匹配的最大重量和，按 i 不匹配/j 不匹配/i 和 j 匹配转移即可

做最好的自己



- 考虑第二问，我们要求极大匹配，则需在  $i/j$  不匹配时考虑是否合法，则需记录上一个未被匹配的位置  $|$

做最好的自己



- 考虑第二问，我们要求极大匹配，则需在  $i/j$  不匹配时考虑是否合法，则需记录上一个未被匹配的位置  $|$
- 若不匹配的位置和  $|$  同色，则上一个不匹配的异色位置必定和  $|$  距离超过  $k$ ，那和当前位置距离就更远了，故必定合法

做最好的自己



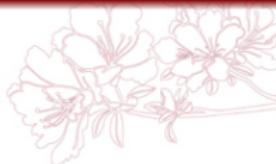
- 考虑第二问，我们要求极大匹配，则需在  $i/j$  不匹配时考虑是否合法，则需记录上一个未被匹配的位置  $|$
- 若不匹配的位置和  $|$  同色，则上一个不匹配的异色位置必定和  $|$  距离超过  $k$ ，那和当前位置距离就更远了，故必定合法
- 若不匹配的位置和  $|$  异色，则判断其距离是否超过  $k$  即可

做最好的自己



- 考虑第二问，我们要求极大匹配，则需在  $i/j$  不匹配时考虑是否合法，则需记录上一个未被匹配的位置  $|$
- 若不匹配的位置和  $|$  同色，则上一个不匹配的异色位置必定和  $|$  距离超过  $k$ ，那和当前位置距离就更远了，故必定合法
- 若不匹配的位置和  $|$  异色，则判断其距离是否超过  $k$  即可
- 故我们得到了一个  $O(n^3)$  的算法

做最好的自己



- 考虑优化，我们可以钦定上一个不匹配的位置为  $i/j$ ，分别记录为 0/1，现在不妨假设是 i

做最好的自己



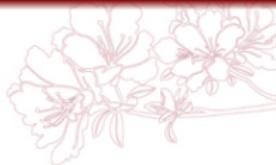
- 考虑优化，我们可以钦定上一个不匹配的位置为  $i/j$ ，分别记录为 0/1，现在不妨假设是 i
- 转移时则需枚举上上个不匹配的位置，设其为 l，若 l 和 i 同色，则可直接从  $f_{l,j-i+l+1,0}$  转移

做最好的自己



- 考虑优化，我们可以钦定上一个不匹配的位置为  $i/j$ ，分别记录为 0/1，现在不妨假设是 i
- 转移时则需枚举上上个不匹配的位置，设其为 l，若 l 和 i 同色，则可直接从  $f_{l,j-i+l+1,0}$  转移
- 若 l 和 i 异色，则 l 须小于等于某个值，并从  $f_{i-j+l-1,l,1}$  转移

做最好的自己



- 考虑优化，我们可以钦定上一个不匹配的位置为  $i/j$ ，分别记录为 0/1，现在不妨假设是 i
- 转移时则需枚举上上个不匹配的位置，设其为 l，若 l 和 i 同色，则可直接从  $f_{l,j-i+l+1,0}$  转移
- 若 l 和 i 异色，则 l 须小于等于某个值，并从  $f_{l-j+l-1,l,1}$  转移
- 容易发现，可转移到  $(i,j)$  的  $(x,y)$  满足  $x - y = c$ ，故可维护满足  $x - y = c$  的所有  $(x,y)$  的前缀 max，直接转移即可

做最好的自己



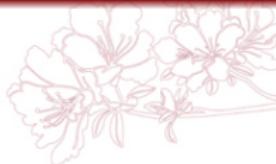
- 考虑优化，我们可以钦定上一个不匹配的位置为  $i/j$ ，分别记录为 0/1，现在不妨假设是 i
- 转移时则需枚举上上个不匹配的位置，设其为 l，若 l 和 i 同色，则可直接从  $f_{l,j-i+l+1,0}$  转移
- 若 l 和 i 异色，则 l 须小于等于某个值，并从  $f_{l-j+l-1,l,1}$  转移
- 容易发现，可转移到  $(i,j)$  的  $(x,y)$  满足  $x - y = c$ ，故可维护满足  $x - y = c$  的所有  $(x,y)$  的前缀 max，直接转移即可
- 时间复杂度  $O(n^2)$

做最好的自己



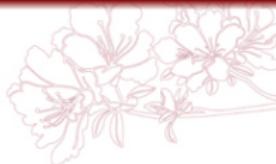
- Farmer John 的  $N$  头奶牛 ( $2 \leq N \leq 3 \cdot 10^5$ ), 编号为  $1 \dots N$ , 排列成  $1 \dots N$  的一个排列  $p_1, p_2, \dots, p_N$ 。另外给定一个长为  $N - 1$  的字符串, 由字母 U 和 D 组成。请求出最大的  $K \leq N - 1$ , 使得存在  $p$  的一个子序列  $a_0, a_1, \dots, a_K$ , 满足对于所有  $1 \leq j \leq K$ , 当字符串中第  $j$  个字母是 U 时  $a_{j-1} < a_j$ , 当字符串中的第  $j$  个字母是 D 时  $a_{j-1} > a_j$ 。

做最好的自己



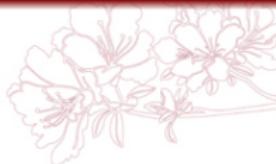
- Farmer John 的  $N$  头奶牛 ( $2 \leq N \leq 3 \cdot 10^5$ ), 编号为  $1 \dots N$ , 排列成  $1 \dots N$  的一个排列  $p_1, p_2, \dots, p_N$ 。另外给定一个长为  $N - 1$  的字符串, 由字母 U 和 D 组成。请求出最大的  $K \leq N - 1$ , 使得存在  $p$  的一个子序列  $a_0, a_1, \dots, a_K$ , 满足对于所有  $1 \leq j \leq K$ , 当字符串中第  $j$  个字母是 U 时  $a_{j-1} < a_j$ , 当字符串中的第  $j$  个字母是 D 时  $a_{j-1} > a_j$ 。
  - 题目来源: 洛谷 P8277

做最好的自己



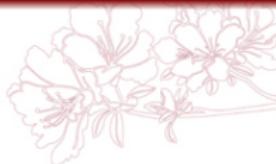
- 先设计一个肯定正确的 dp，设  $f_{i,j}$  表示必须选 i 时，p 中前 i 个和 s 中前 j 个能否匹配，转移时需要枚举上一个选的数，可树状数组优化做到  $O(n^2 \log n)$

做最好的自己



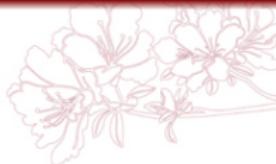
- 先设计一个肯定正确的 dp，设  $f_{i,j}$  表示必须选 i 时，p 中前 i 个和 s 中前 j 个能否匹配，转移时需要枚举上一个选的数，可树状数组优化做到  $O(n^2 \log n)$
- 由于我们的状态是一个 bool 值，故猜测某一维具有单调性，以把这一维写进 dp 值里

做最好的自己



- 先设计一个肯定正确的 dp，设  $f_{i,j}$  表示必须选 i 时，p 中前 i 个和 s 中前 j 个能否匹配，转移时需要枚举上一个选的数，可树状数组优化做到  $O(n^2 \log n)$
- 由于我们的状态是一个 bool 值，故猜测某一维具有单调性，以把这一维写进 dp 值里
- 由于 U 和 D 很不一样，故猜测须按 U 和 D 分类讨论

做最好的自己



- 先设计一个肯定正确的 dp，设  $f_{i,j}$  表示必须选 i 时，p 中前 i 个和 s 中前 j 个能否匹配，转移时需要枚举上一个选的数，可树状数组优化做到  $O(n^2 \log n)$
- 由于我们的状态是一个 bool 值，故猜测某一维具有单调性，以把这一维写进 dp 值里
- 由于 U 和 D 很不一样，故猜测须按 U 和 D 分类讨论
- 然而我们考虑  $p = [5, 6, 7, 1, 2, 3, 4]$ ,  $s = UUUU$  这个案例，即便 s 全为 U,  $f_{6,3}$  比  $f_{3,3}$  更优,  $f_{3,0}$  比  $f_{3,3}$  更优，故并不存在朴素的单调性

做最好的自己

- 考虑 U 和 D 的切换点，不妨设  $s_j$  为 D，最短的能与  $s_1, \dots, s_j$  匹配的前缀结尾为位置 k，后续有 x 个 U，我们猜测这个切换点具有单调性，即越前越好

做最好的自己

- 考虑 U 和 D 的切换点，不妨设  $s_j$  为 D，最短的能与  $s_1, \dots, s_j$  匹配的前缀结尾为位置 k，后续有 x 个 U，我们猜测这个切换点具有单调性，即越前越好
- 假设  $p_k, \dots, p_n$  中，第一个长度为  $x+1$  的上升子序列为  $b_0, b_1, \dots, b_x$ ，结尾为  $k'$ ，那么显然，设最短的能与  $s_1, \dots, s_{j+x}$  匹配的前缀结尾为 t，则  $t \geq k'$

做最好的自己

- 考虑 U 和 D 的切换点，不妨设  $s_j$  为 D，最短的能与  $s_1, \dots, s_j$  匹配的前缀结尾为位置 k，后续有 x 个 U，我们猜测这个切换点具有单调性，即越前越好
- 假设  $p_k, \dots, p_n$  中，第一个长度为  $x+1$  的上升子序列为  $b_0, b_1, \dots, b_x$ ，结尾为  $k'$ ，那么显然，设最短的能与  $s_1, \dots, s_{j+x}$  匹配的前缀结尾为 t，则  $t \geq k'$
- 考虑若  $p_k = b_0$ ，即前面匹配序列的结尾刚好是上升序列的开头，则直接将  $b_1, \dots, b_x$  接到前面序列后即可得到一个与  $s_1, \dots, s_{j+x}$  匹配的序列

做最好的自己

- 考虑 U 和 D 的切换点，不妨设  $s_j$  为 D，最短的能与  $s_1, \dots, s_j$  匹配的前缀结尾为位置 k，后续有 x 个 U，我们猜测这个切换点具有单调性，即越前越好
- 假设  $p_k, \dots, p_n$  中，第一个长度为  $x+1$  的上升子序列为  $b_0, b_1, \dots, b_x$ ，结尾为  $k'$ ，那么显然，设最短的能与  $s_1, \dots, s_{j+x}$  匹配的前缀结尾为 t，则  $t \geq k'$
- 考虑若  $p_k = b_0$ ，即前面匹配序列的结尾刚好是上升序列的开头，则直接将  $b_1, \dots, b_x$  接到前面序列后即可得到一个与  $s_1, \dots, s_{j+x}$  匹配的序列
- 否则必然有  $p_k > b_0$ ，把前面序列的  $p_k$  替换成  $b_0$ ，并接上后面序列，即可得到一个与  $s_1, \dots, s_{j+x}$  匹配的序列

做最好的自己

- 考虑 U 和 D 的切换点，不妨设  $s_j$  为 D，最短的能与  $s_1, \dots, s_j$  匹配的前缀结尾为位置 k，后续有 x 个 U，我们猜测这个切换点具有单调性，即越前越好
- 假设  $p_k, \dots, p_n$  中，第一个长度为  $x+1$  的上升子序列为  $b_0, b_1, \dots, b_x$ ，结尾为  $k'$ ，那么显然，设最短的能与  $s_1, \dots, s_{j+x}$  匹配的前缀结尾为 t，则  $t \geq k'$
- 考虑若  $p_k = b_0$ ，即前面匹配序列的结尾刚好是上升序列的开头，则直接将  $b_1, \dots, b_x$  接到前面序列后即可得到一个与  $s_1, \dots, s_{j+x}$  匹配的序列
- 否则必然有  $p_k > b_0$ ，把前面序列的  $p_k$  替换成  $b_0$ ，并接上后面序列，即可得到一个与  $s_1, \dots, s_{j+x}$  匹配的序列
- 故有  $t=k'$ ，故我们 U 和 D 的切换点具有单调性，越前越好，故可按我们证明的方式将寻找下一个切换点变成寻找从 1 开始的第一个长度为  $x+1$  的上升子序列，这类似 LIS，使用树状数组优化 dp 即可，时间复杂度  $O(n \log n)$

做最好的自己



- Bessie 找到了一份行车调度的新工作。现在有两座火车站 A 和 B，由于预算限制，只有一条单线铁道连接起车站 A 和 B。如果一列列车在  $t$  时刻离开其中一座火车站，它将在  $t + T$  ( $1 \leq T \leq 10^{12}$ ) 时刻到达另一座火车站。

现在有  $N$  ( $1 \leq N \leq 5000$ ) 列火车的出发时间需要安排。第  $i$  列火车必须在  $t_i$  时刻后从车站  $s_i$  出发 ( $s_i \in \{A, B\}$ ,  $0 \leq t_i \leq 10^{12}$ )。在同一时刻不允许铁道上有相反方向的列车，否则它们会相撞。但是，假设火车有可以忽略的尺寸，在同一时刻，铁道上可以有许多相同方向的列车。

帮助 Bessie 安排每辆列车的出发时间，在不会相撞的前提下最小化总延误时间。假设第  $i$  辆列车被安排在  $a_i$  时刻出发，总延误为  $\sum_{i=1}^n a_i - t_i$ 。

做最好的自己



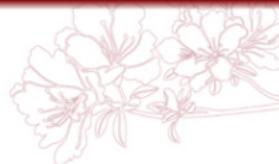
- Bessie 找到了一份行车调度的新工作。现在有两座火车站 A 和 B，由于预算限制，只有一条单线铁道连接起车站 A 和 B。如果一列列车在  $t$  时刻离开其中一座火车站，它将在  $t + T$  ( $1 \leq T \leq 10^{12}$ ) 时刻到达另一座火车站。

现在有  $N$  ( $1 \leq N \leq 5000$ ) 列火车的出发时间需要安排。第  $i$  列火车必须在  $t_i$  时刻后从车站  $s_i$  出发 ( $s_i \in \{A, B\}$ ),  $0 \leq t_i \leq 10^{12}$ )。在同一时刻不允许铁道上有相反方向的列车，否则它们会相撞。但是，假设火车有可以忽略的尺寸，在同一时刻，铁道上可以有许多相同方向的列车。

帮助 Bessie 安排每辆列车的出发时间，在不会相撞的前提下最小化总延误时间。假设第  $i$  辆列车被安排在  $a_i$  时刻出发，总延误为  $\sum_{i=1}^n a_i - t_i$ 。

- 题目来源：洛谷 P9985

做最好的自己



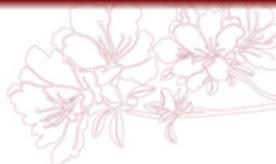
- 考虑最暴力的 dp，设  $f_{i,j,t,1/0}$  为 A 车站走了 i 班列车，B 车站走了 j 班列车，当前时刻为 t，在 A/B 车站的最小延误时间

做最好的自己



- 考虑最暴力的 dp，设  $f_{i,j,t,1/0}$  为 A 车站走了 i 班列车，B 车站走了 j 班列车，当前时刻为 t，在 A/B 车站的最小延误时间
- 由于 t 非常大，故考虑先把 t 优化掉

做最好的自己



- 考虑最暴力的 dp，设  $f_{i,j,t,1/0}$  为 A 车站走了 i 班列车，B 车站走了 j 班列车，当前时刻为 t，在 A/B 车站的最小延误时间
- 由于 t 非常大，故考虑先把 t 优化掉
- 观察到 t 一定是  $s_x + yT$  的形式，故可用  $(x, y)$  表示 t，以做到  $O(n^4)$

做最好的自己



- 考虑最暴力的 dp，设  $f_{i,j,t,1/0}$  为 A 车站走了 i 班列车，B 车站走了 j 班列车，当前时刻为 t，在 A/B 车站的最小延误时间
- 由于 t 非常大，故考虑先把 t 优化掉
- 观察到 t 一定是  $s_x + yT$  的形式，故可用  $(x, y)$  表示 t，以做到  $O(n^4)$
- 实际上，根据某些说法，最小的 dp 值对应的 y 亦最小，故可把 y 丢到 dp 值里记录以达到  $O(n^3)$

做最好的自己

- 考虑继续优化，则需去掉  $i/j/x$  某一维，比如把  $x$  这一维去掉，设  $f_{i,j,0/1}$  表示 A 车站走了  $i$  辆车，B 走了  $j$  辆，当前在 A/B 且最后一趟车准点的最小代价

做最好的自己

- 考虑继续优化，则需去掉  $i/j/x$  某一维，比如把  $x$  这一维去掉，设  $f_{i,j,0/1}$  表示 A 车站走了  $i$  辆车，B 走了  $j$  辆，当前在 A/B 且最后一趟车准点的最小代价
- 则若下一趟车在同侧，其必然准点出发，可直接转移到  $f_{i+1,j,0/1}$ （以  $i$  为例）

做最好的自己

- 考虑继续优化，则需去掉  $i/j/x$  某一维，比如把  $x$  这一维去掉，设  $f_{i,j,0/1}$  表示 A 车站走了  $i$  辆车，B 走了  $j$  辆，当前在 A/B 且最后一趟车准点的最小代价
- 则若下一趟车在同侧，其必然准点出发，可直接转移到  $f_{i+1,j,0/1}$ （以  $i$  为例）
- 否则下一趟车必然在异侧，那么异侧所有早于  $s_i + T$  的车必然都会在  $s_i + T$  时刻出发，接下来又有两种可能：

做最好的自己

- 考虑继续优化，则需去掉  $i/j/x$  某一维，比如把  $x$  这一维去掉，设  $f_{i,j,0/1}$  表示 A 车站走了  $i$  辆车，B 走了  $j$  辆，当前在 A/B 且最后一趟车准点的最小代价
- 则若下一趟车在同侧，其必然准点出发，可直接转移到  $f_{i+1,j,0/1}$ （以  $i$  为例）
- 否则下一趟车必然在异侧，那么异侧所有早于  $s_i + T$  的车必然都会在  $s_i + T$  时刻出发，接下来又有两种可能：
  - 接下来下一趟出发的车仍在异侧

做最好的自己

- 考虑继续优化，则需去掉  $i/j/x$  某一维，比如把  $x$  这一维去掉，设  $f_{i,j,0/1}$  表示 A 车站走了  $i$  辆车，B 走了  $j$  辆，当前在 A/B 且最后一趟车准点的最小代价
- 则若下一趟车在同侧，其必然准点出发，可直接转移到  $f_{i+1,j,0/1}$ （以  $i$  为例）
- 否则下一趟车必然在异侧，那么异侧所有早于  $s_i + T$  的车必然都会在  $s_i + T$  时刻出发，接下来又有两种可能：
  - 接下来下一趟出发的车仍在异侧
  - 接下来下一趟出发的车又在同侧

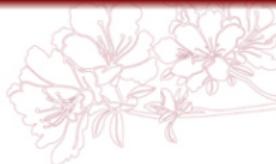
做最好的自己

- 考虑继续优化，则需去掉  $i/j/x$  某一维，比如把  $x$  这一维去掉，设  $f_{i,j,0/1}$  表示 A 车站走了  $i$  辆车，B 走了  $j$  辆，当前在 A/B 且最后一趟车准点的最小代价
- 则若下一趟车在同侧，其必然准点出发，可直接转移到  $f_{i+1,j,0/1}$ （以  $i$  为例）
- 否则下一趟车必然在异侧，那么异侧所有早于  $s_i + T$  的车必然都会在  $s_i + T$  时刻出发，接下来又有两种可能：
  - 接下来下一趟出发的车仍在异侧
  - 接下来下一趟出发的车又在同侧
- 对于后者，须继续分类讨论，故可枚举经过几个  $T$  之后下一趟出发的车准点出发，这样状态是  $O(n^2)$  的，转移是  $O(n)$  的

做最好的自己

- 考虑继续优化，则需去掉  $i/j/x$  某一维，比如把  $x$  这一维去掉，设  $f_{i,j,0/1}$  表示 A 车站走了  $i$  辆车，B 走了  $j$  辆，当前在 A/B 且最后一趟车准点的最小代价
- 则若下一趟车在同侧，其必然准点出发，可直接转移到  $f_{i+1,j,0/1}$ （以  $i$  为例）
- 否则下一趟车必然在异侧，那么异侧所有早于  $s_i + T$  的车必然都会在  $s_i + T$  时刻出发，接下来又有两种可能：
  - 接下来下一趟出发的车仍在异侧
  - 接下来下一趟出发的车又在同侧
- 对于后者，须继续分类讨论，故可枚举经过几个  $T$  之后下一趟出发的车准点出发，这样状态是  $O(n^2)$  的，转移是  $O(n)$  的
- 把  $i/j$  这一维去掉也殊途同归

做最好的自己



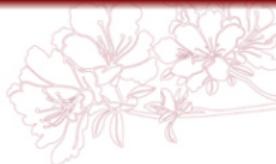
- 考虑优化转移，考虑在车站间反复横跳的过程，跳第一下后记最后一个出发时间  $\leq s_i + T$  的列车编号为  $j_1$ ，则  $(j, j_1]$  均会在时刻  $s_i + T$  出发

做最好的自己



- 考虑优化转移，考虑在车站间反复横跳的过程，跳第一下后记最后一个出发时间  $\leq s_i + T$  的列车编号为  $j_1$ ，则  $(j, j_1]$  均会在时刻  $s_i + T$  出发
- 同理，记  $i$  这一侧最后一个出发时间  $\leq s_i + 2T$  的列车编号为  $i_1$ ，则  $(i, i_1]$  在  $\leq s_i + 2T$  出发

做最好的自己



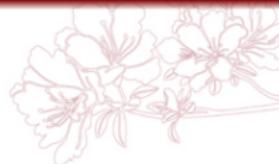
- 考虑优化转移，考虑在车站间反复横跳的过程，跳第一下后记最后一个出发时间  $\leq s_i + T$  的列车编号为  $j_1$ ，则  $(j, j_1]$  均会在时刻  $s_i + T$  出发
- 同理，记这一侧最后一个出发时间  $\leq s_i + 2T$  的列车编号为  $i_1$ ，则  $(i, i_1]$  在  $\leq s_i + 2T$  出发
- 继续往下想， $(j_1, j_2]$  会在  $\leq s_i + 3T$  出发

做最好的自己



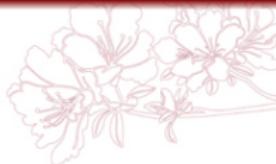
- 考虑优化转移，考虑在车站间反复横跳的过程，跳第一下后记最后一个出发时间  $\leq s_i + T$  的列车编号为  $j_1$ ，则  $(j, j_1]$  均会在时刻  $s_i + T$  出发
- 同理，记  $i$  这一侧最后一个出发时间  $\leq s_i + 2T$  的列车编号为  $i_1$ ，则  $(i, i_1]$  在  $\leq s_i + 2T$  出发
- 继续往下想， $(j_1, j_2]$  会在  $\leq s_i + 3T$  出发
- 以此类推，我们发现了什么，我们发现了从第二次开始产生的代价和  $j$  无关

做最好的自己



- 故可设  $dp2_{i,x}$  表示从  $i$  开始，反复横跳了  $x$  次后的最小代价，当  $x \geq 2$ ,  $f_{i,x}$  可由  $f_{i,x-1}$  转移

做最好的自己



- 故可设  $dp2_{i,x}$  表示从  $i$  开始，反复横跳了  $x$  次后的最小代价，当  $x \geq 2$ ,  $f_{i,x}$  可由  $f_{i,x-1}$  转移
- 转移时须知道  $\leq s_i + xT$  的最后一趟列车，可二分 ( $O(\log n)$ ) 或双指针预处理 (均摊  $O(1)$ )

做最好的自己



- 故可设  $dp2_{i,x}$  表示从  $i$  开始，反复横跳了  $x$  次后的最小代价，当  $x \geq 2$ ,  $f_{i,x}$  可由  $f_{i,x-1}$  转移
- 转移时须知道  $\leq s_i + xT$  的最后一趟列车，可二分 ( $O(\log n)$ ) 或双指针预处理 (均摊  $O(1)$ )
- 若一次跳跃后没有新列车出发，则后续两侧列车均可准点出发，可直接转移到  $dp$  数组

做最好的自己



- 故可设  $dp2_{i,x}$  表示从  $i$  开始，反复横跳了  $x$  次后的最小代价，当  $x \geq 2$ ,  $f_{i,x}$  可由  $f_{i,x-1}$  转移
- 转移时须知道  $\leq s_i + xT$  的最后一趟列车，可二分 ( $O(\log n)$ ) 或双指针预处理 (均摊  $O(1)$ )
- 若一次跳跃后没有新列车出发，则后续两侧列车均可准点出发，可直接转移到  $dp$  数组
- 故跳跃后必有列车出发，故跳跃次数不超过  $O(n)$ ，总时间复杂度  $O(n^2)$

做最好的自己



- tips: 该题 dp 顺序也较为奇特,  $dp2_{i,x}$  显然须所有满足  $j \leq pos_i$  的  $j$  均转移完才可转移; 同理,  $dp2_{j,x}$  须所有  $i \leq pos_j$  的  $i$  转移完才行, 故还需维护  $pos_i, pos_j$  指针

做最好的自己