題解

战争

题意

战争

题意

- 给定两个序列，q次求出它们卷积的$l \ldots r$项求和。

战争

题意

- 给定两个序列，q次求出它们卷积的$l \ldots r$项求和。
- 答案对1145141923取模。

战争

题意

- 给定两个序列，q次求出它们卷积的$l \ldots r$项求和。
- 答案对1145141923取模。
- (下面假设n, m同阶)

碎碎念

碎碎念

- 模数是固定11451419前缀之后凑的。

碎碎念

- 模数是固定11451419前缀之后凑的。
- $1145141923 \times 2 \geq 2^{31}$，所以有可能会爆int。

碎碎念

- 模数是固定11451419前缀之后凑的。
- $1145141923 \times 2 \geq 2^{31}$，所以有可能会爆int。
- ~~不会吧不会吧不会真的有人看到提示还爆int吧~~

算法1

算法1

- 我会枚举!

算法1

- 我会枚举！
- $O(n^2)$求出卷积，每次$O(n)$求和或者先求一遍前缀和。

算法1

- 我会枚举!
- $O(n^2)$求出卷积，每次$O(n)$求和或者先求一遍前缀和。
- 期望得分30。

算法2

算法2

- 我会NTT或FFT!

算法2

- 我会NTT或FFT!
- 虽然模的不是NTT模数，但在$a_i, b_i < 20$的情况下容易发现每一项系数$< 2 \times 10^8 < 998244353$。

算法2

- 我会NTT或FFT!
- 虽然模的不是NTT模数，但在$a_i, b_i < 20$的情况下容易发现每一项系数$< 2 \times 10^8 < 998244353$。
- 所以直接FFT或者模998244353意义下NTT即可。

算法2

- 我会NTT或FFT!
- 虽然模的不是NTT模数，但在$a_i, b_i < 20$的情况下容易发现每一项系数$< 2 \times 10^8 < 998244353$。
- 所以直接FFT或者模998244353意义下NTT即可。
- 时间复杂度$O(n \log n)$

算法2

- 我会NTT或FFT!
- 虽然模的不是NTT模数，但在$a_i, b_i < 20$的情况下容易发现每一项系数$< 2 \times 10^8 < 998244353$。
- 所以直接FFT或者模998244353意义下NTT即可。
- 时间复杂度$O(n \log n)$
- 结合算法1期望得分45。

算法3

算法3

- 我会任意模数卷积!

算法3

- 我会任意模数卷积!
- 时间复杂度$O(n \log n)$

算法3

- 我会任意模数卷积!
- 时间复杂度$O(n \log n)$
- 期望得分$45(60)$，可能需要结合算法2 期望得分60。

算法4

算法4

- 没错前面在逗你玩。

算法4

- 没错前面在逗你玩。
- 考虑将问题转化为求$\sum_{i+j\leq c} a_i b_j$，这显然可以对b求前缀和然后直接枚举i计算。

算法4

- 没错前面在逗你玩。
- 考虑将问题转化为求$\sum_{i+j\leq c} a_i b_j$，这显然可以对b求前缀和然后直接枚举i计算。
- $l \ldots r$项就可以直接拆成$\leq r$的答案$-\leq (l-1)$的答案。

算法4

- 没错前面在逗你玩。
- 考虑将问题转化为求$\sum_{i+j\leq c} a_i b_j$，这显然可以对b求前缀和然后直接枚举i计算。
- $l \ldots r$项就可以直接拆成$\leq r$的答案$- \leq (l-1)$的答案。
- 时间复杂度$O(nq)$，期望得分100。

旅游

旅游

- 题意：有一张平面图，1向无穷远处连一条边，x和$2x, 2x+1$之间连一条边。

旅游

- 题意：有一张平面图，1向无穷远处连一条边，x和$2x, 2x+1$之间连一条边。
- 求x号点位置走到y号点位置，除了开始和结束位置最少碰到几条边。

算法1

算法1

- 我会讨论！手动讨论一下$\leq$ 7的情况。

算法1

- 我会讨论！手动讨论一下≤ 7的情况。
- 期望得分20。

算法2

算法2

- 我会bfs！

算法2

- 我会bfs！
- 考虑枚举开始和结束出发的方向，也就是计算对偶图两个点之间最短路。

算法2

- 我会bfs！
- 考虑枚举开始和结束出发的方向，也就是计算对偶图两个点之间最短路。
- 注意到走到编号太大的点一定不优秀，所以对着前面一些点跑bfs即可。

算法2

- 我会bfs！
- 考虑枚举开始和结束出发的方向，也就是计算对偶图两个点之间最短路。
- 注意到走到编号太大的点一定不优秀，所以对着前面一些点跑bfs即可。
- 时间复杂度$O(w^2)$，其中w为a, b的范围，期望得分50。

算法3

算法3

- 注意到向下再向上走，要么走回这个点，要么走到这个点向下一步能走到的，要么走到这个点向上一步能走到的，一定都不优。

算法3

- 注意到向下再向上走，要么走回这个点，要么走到这个点向下一步能走到的，要么走到这个点向上一步能走到的，一定都不优。
- 那么只需要考虑先向上再向下，也就是求两个点分别向上，停在同一个点的最小代价和。

算法3

- 注意到向下再向上走，要么走回这个点，要么走到这个点向下一步能走到的，要么走到这个点向上一步能走到的，一定都不优。
- 那么只需要考虑先向上再向下，也就是求两个点分别向上，停在同一个点的最小代价和。
- 对每个点到根的链DP即可，时间复杂度$O(q \log w)$，期望得分100。

战成平手

- 题意：取石子游戏，相同的堆会同时消失，无法行动者获胜。

题解

题解

- 可以通过打表或类似方法 猜到结论：

题解

- 可以通过打表或类似方法 猜到结论：
- $\text{xor} = 0$且可以拆分成一些$x, x \text{ xor } 1$的元组的序列是必胜态(有1的情况下可以认为多一个0)。

题解

- 可以通过打表或类似方法 猜到结论：
- $\text{xor} = 0$且可以拆分成一些$x, x \text{ xor } 1$的元组的序列是必胜态(有1的情况下可以认为多一个0)。
- $\text{xor} \neq 0$且不能拆分的是必胜态。

题解

- 可以通过打表或类似方法 猜到结论：
- $\text{xor} = 0$且可以拆分成一些$x, x \text{ xor } 1$的元组的序列是必胜态(有1的情况下可以认为多一个0)。
- $\text{xor} \neq 0$且不能拆分的是必胜态。
- 否则是必败态。

题解

- 可以通过打表或类似方法 猜到结论：
- $\text{xor} = 0$且可以拆分成一些$x, x \text{ xor } 1$的元组的序列是必胜态(有1的情况下可以认为多一个0)。
- $\text{xor} \neq 0$且不能拆分的是必胜态。
- 否则是必败态。
- 可以通过简单讨论证明。

礼物

礼物

- 题意：给定一个序列，对于每个区间求出排序离散化后 $\sum i \times a_i$ 然后求和。

算法1

算法1

- 我会输入输出！

算法1

- 我会输入输出!
- 对于$a_i = 1$的情况，任何区间离散化后都只剩一个1，则答案就是区间个数$= \frac{n(n+1)}{2}$。

算法1

- 我会输入输出!
- 对于$a_i = 1$的情况，任何区间离散化后都只剩一个1，则答案就是区间个数$= \frac{n(n+1)}{2}$。
- 当然这有可能$\geq 10^9 + 7$，所以要取模。

算法1

- 我会输入输出!
- 对于$a_i = 1$的情况，任何区间离散化后都只剩一个1，则答案就是区间个数$= \frac{n(n+1)}{2}$。
- 当然这有可能$\geq 10^9 + 7$，所以要取模。
- 期望得分1。

算法2

算法2

- 我会枚举!

算法2

- 我会枚举！
- 枚举l,r然后std::sort,std::unique。

算法2

- 我会枚举!
- 枚举l,r然后std::sort,std::unique。
- 时间复杂度$O(n^3 \log n)$，期望得分25，结合算法1期望得分26。

算法3

算法3

- 枚举l，移动r，每次加一个数。

算法3

- 枚举l，移动r，每次加一个数。
- 如果它出现过那么不管，否则它的rank是比它小的数个数，且比它大的数rank都要$+1$，用两个树状数组维护一下即可。

算法3

- 枚举l，移动r，每次加一个数。
- 如果它出现过那么不管，否则它的rank是比它小的数个数，且比它大的数rank都要$+1$，用两个树状数组维护一下即可。
- 时间复杂度$O(n^2 \log n)$，期望得分45，结合算法1期望得分46。

算法4

算法4

- 考虑是一个排列的情况。

算法4

- 考虑是一个排列的情况。
- 每次末尾加一个数a_y，考虑维护所有后缀的贡献和，也就是从y开始每次往前加一个数得到的答案求和，

算法4

- 考虑是一个排列的情况。
- 每次末尾加一个数a_y，考虑维护所有后缀的贡献和，也就是从y开始每次往前加一个数得到的答案求和，
- 首先每个后缀a_y贡献至少为1。

算法4

- 考虑是一个排列的情况。
- 每次末尾加一个数a_y，考虑维护所有后缀的贡献和，也就是从y开始每次往前加一个数得到的答案求和，
- 首先每个后缀a_y贡献至少为1。
- 对于前面的每一个位置x，数是a_x。

算法4

- 考虑是一个排列的情况。
- 每次末尾加一个数a_y，考虑维护所有后缀的贡献和，也就是从y开始每次往前加一个数得到的答案求和，
- 首先每个后缀a_y贡献至少为1。
- 对于前面的每一个位置x，数是a_x。
- 如果$a_x < a_y$，则在$1..x$的后缀中y rank都$+1$，

算法4

- 考虑是一个排列的情况。
- 每次末尾加一个数a_y，考虑维护所有后缀的贡献和，也就是从y开始每次往前加一个数得到的答案求和，
- 首先每个后缀a_y贡献至少为1。
- 对于前面的每一个位置x，数是a_x。
- 如果$a_x < a_y$，则在$1..x$的后缀中y rank都$+1$，
- 否则$1..x$的后缀中x rank都$+1$。

算法4

- 考虑是一个排列的情况。
- 每次末尾加一个数a_y，考虑维护所有后缀的贡献和，也就是从y开始每次往前加一个数得到的答案求和，
- 首先每个后缀a_y贡献至少为1。
- 对于前面的每一个位置x，数是a_x。
- 如果$a_x < a_y$，则在$1..x$的后缀中y rank都$+1$，
- 否则$1..x$的后缀中x rank都$+1$。
- 用两个树状数组维护即可。

算法4

- 考虑是一个排列的情况。
- 每次末尾加一个数a_y，考虑维护所有后缀的贡献和，也就是从y开始每次往前加一个数得到的答案求和，
- 首先每个后缀a_y贡献至少为1。
- 对于前面的每一个位置x，数是a_x。
- 如果$a_x < a_y$，则在$1..x$的后缀中y rank都$+1$，
- 否则$1..x$的后缀中x rank都$+1$。
- 用两个树状数组维护即可。
- 时间复杂度$O(n \log n)$，期望得分14，结合算法1和算法3，期望得分60。

算法5

算法5

- 对于有重复的情况，我们需要对其进行一点改动：

算法5

- 对于有重复的情况，我们需要对其进行一点改动：
- 首先每个a_x只考虑当前最后一次出现的x，因为往前走的时候第二次碰到x显然不会对rank产生任何影响。

算法5

- 对于有重复的情况，我们需要对其进行一点改动：
- 首先每个a_x只考虑当前最后一次出现的x，因为往前走的时候第二次碰到x显然不会对rank产生任何影响。
- 插入y的时候，设上一次a_y出现的位置为y'，则$1..y'$的后缀答案都不会变，因为加到y'的时候把y删了没有影响。

算法5

- 对于有重复的情况，我们需要对其进行一点改动：
- 首先每个a_x只考虑当前最后一次出现的x，因为往前走的时候第二次碰到x显然不会对rank产生任何影响。
- 插入y的时候，设上一次a_y出现的位置为y'，则$1..y'$的后缀答案都不会变，因为加到y'的时候把y删了没有影响。
- 那么只需要考虑$y'+1\dots y-1$中出现的所有x，然后rank增加的区间也从$1\dots x$变成$y'+1\dots x$。

算法5

- 对于有重复的情况，我们需要对其进行一点改动：
- 首先每个a_x只考虑当前最后一次出现的x，因为往前走的时候第二次碰到x显然不会对rank产生任何影响。
- 插入y的时候，设上一次a_y出现的位置为y'，则$1..y'$的后缀答案都不会变，因为加到y'的时候把y删了没有影响。
- 那么只需要考虑$y'+1\dots y-1$中出现的所有x，然后rank增加的区间也从$1\dots x$变成$y'+1\dots x$。
- 用树套树维护即可。时间复杂度$O(n\log^2 n)$，期望得分100。

算法5

- 对于有重复的情况，我们需要对其进行一点改动：
- 首先每个a_x只考虑当前最后一次出现的x，因为往前走的时候第二次碰到x显然不会对rank产生任何影响。
- 插入y的时候，设上一次a_y出现的位置为y'，则$1..y'$的后缀答案都不会变，因为加到y'的时候把y删了没有影响。
- 那么只需要考虑$y'+1\dots y-1$中出现的所有x，然后rank增加的区间也从$1\dots x$变成$y'+1\dots x$。
- 用树套树维护即可。时间复杂度$O(n\log^2 n)$，期望得分100。
- 不过这个做法常数很大，所以std 0.7s开了3s，如果被卡常的话至少应该有85分。

THANKS!