NNSZCP-2023 赛后题解

南宁三中 01 社

Problem A. 欢迎光临

题意

- 给定字符串，查询子串有没有 nnsz。
- 字符串长度 ≤ 100。

题解

题解

- 签到题。

题解

- 签到题。
- 枚举相邻的字符判断是否为 nnsz 即可。

题解

- 签到题。
- 枚举相邻的字符判断是否为 nnsz 即可。
- 当然也可以用 KMP 做。

题解

- 签到题。
- 枚举相邻的字符判断是否为 nnsz 即可。
- 当然也可以用 KMP 做。
- 时间复杂度 $O(n)$。

题解

- 签到题。
- 枚举相邻的字符判断是否为 nnsz 即可。
- 当然也可以用 KMP 做。
- 时间复杂度 $O(n)$。
- Subtask 0 的 45 分是致敬 galaxy。

代码实现

```python
print("yes" if "nnsz" in input() else "no")
```

Problem B. 反应原理

题意

- 给定 n 和长为 $n+1$ 的序列 a。
- 求相邻两项的最大差值、所有项的最大值。
- 保证 $1 \le n \le 3 \times 10^5$。

题解

题解

- 按题意模拟即可。

题解

- 按题意模拟即可。
- 注意序列的长为 $n+1$。

代码实现

```python
n = int(input())
a = [ int(i) for i in input().split() ]
print(max(a))
print(max(a[i + 1] - a[i] for i in range(n)))
```

Problem C. 暮光闪闪

题意

- n 栋建筑物，每一栋建筑物的高度为 h_i。
- m 匹天马中，对于第 i 匹天马，其飞行的高度为 s_i。
- 对两座建筑 i,j，第 k 匹天马能够在这两座建筑之间飞行，当且仅当 $|h_i - h_j| \le s_k$。
- 对于每一匹天马，求其最多能够在多少对建筑之间穿梭。
- 保证 $1 \le m \le 10^5$，$1 \le n \le 2 \times 10^3$。

题解

题解

- 对于 Subtask 0：
  - 对于每匹天马，枚举每一对建筑，并判断它是否能在该对建筑之间穿梭。时间复杂度为 $O(n^2m)$。

题解

- 对于 Subtask 0：
  - 对于每匹天马，枚举每一对建筑，并判断它是否能在该对建筑之间穿梭。时间复杂度为 $O(n^2m)$。
- 对于 Subtask 1：
  - 由 $s_i > 0$ 可知，每一匹天马均可在建筑中任意穿梭，故答案为 $\frac{n(n-1)}{2}$。

题解

题解

- 对于 Subtask 2：
  - 预处理每对建筑之间的高度差，并将其升序排序。对于每匹天马，在高度差数组中通过二分查找得到答案。
  - 预处理高度差并排序的时间复杂度为 $O(n^2\log n^2)$，进行 m 次二分查找的时间复杂度为 $O(m\log n^2)$。故总时间复杂度为$O((n^2+m)\log n^2)$，可以通过。

题解

- 对于 Subtask 2：
  - 预处理每对建筑之间的高度差，并将其升序排序。对于每匹天马，在高度差数组中通过二分查找得到答案。
  - 预处理高度差并排序的时间复杂度为 $O(n^2 \log n^2)$，进行 m 次二分查找的时间复杂度为 $O(m \log n^2)$。故总时间复杂度为$O((n^2+m)\log n^2)$，可以通过。

- Bonus: $1 \le n, m \le 10^5, 1 \le h_i, s_i \le \sum h_i \le 10^5$ 怎么做？

代码实现

```python
from bisect import bisect_right

n, m = [ int(i) for i in input().split() ]
h = [ int(i) for i in input().split() ]
s = [ int(i) for i in input().split() ]
dh = sorted([ abs(h[i] - h[j]) for i in range(n) for j in range(i) ])

for i in s:
    print(bisect_right(dh, i))
```

Problem D. 中考录取

题意

- 按照一定规则对进行考生进行排名。
- 详细内容见题面。保证 $1 \leq n \leq 10^5$。

题解

题解

- 小清新模拟题。

题解

- 小清新模拟题。
- 本题 C++ 标程仅约 700 Byte，Python 标程仅约 400 Byte，它们都用到了以下优化技巧来减小码量。
  - 使用 tuple 而非 struct 或 class 表示考生。
  - 运用位运算技巧，仅用一个整数即可表示考生各科 A+ 情况。

题解

- 小清新模拟题。
- 本题 C++ 标程仅约 700 Byte，Python 标程仅约 400 Byte，它们都用到了以下优化技巧来减小码量。
  - 使用 tuple 而非 struct 或 class 表示考生。
  - 运用位运算技巧，仅用一个整数即可表示考生各科 A+ 情况。
- 运用位运算技巧将考生各科 A+ 情况压缩成一个整数，即可以方便地在 tuple 中存储考生的数据，又可以通过直接比较整数的大小来分出成绩的优劣。

代码实现

```python
l = [ int(i) for i in input().split() ]
n, m = [ int(i) for i in input().split() ]

a = []
for i in range(n):
    s = [ int(i) for i in input().split() ]
    t = [ sum(s) >= l[6], 0, 0 ]

    for j in range(6):
        if s[j] < l[j]:
            continue
        t[1] += 1
        t[2] |= 1 << (5 - j)
    a.append(tuple(t))

a.sort(reverse = True)
while m < n and a[m] == a[m - 1]:
    m += 1

print(m)
```

Problem E. 填数游戏

题意

构造满足以下 3 个条件的矩阵：

1. 矩阵中的元素均为自然数，且在区间 $[0, k]$ 内。
2. 矩阵中的元素各不相同。
3. 从矩阵中选出 n 个元素，使得每一行有且仅有一个元素被选出，且每一列有且仅有一个元素被选出；对于每一种符合上述规则的选择元素的方案，选出的元素总和均为 k。

题解

题解

- 可以先考虑什么时候无解。

题解

- 可以先考虑什么时候无解。
- 构造一个元素取遍 $[0, n^2-1]$ 的矩阵，并令其元素从左到右，从上到下升序排列，这显然是同时满足条件 1 和条件 2 的元素最小的矩阵。

题解

- 可以先考虑什么时候无解。
- 构造一个元素取遍 $[0, n^2 - 1]$ 的矩阵，并令其元素从左到右，从上到下升序排列，这显然是同时满足条件 1 和条件 2 的元素最小的矩阵。
- 对于该矩阵，使得其符合条件 3 的 $k = \frac{n(n^2-1)}{2} = \frac{n(n-1)(n+1)}{2}$。

题解

- 可以先考虑什么时候无解。
- 构造一个元素取遍 $[0, n^2-1]$ 的矩阵，并令其元素从左到右，从上到下升序排列，这显然是同时满足条件 1 和条件 2 的元素最小的矩阵。
- 对于该矩阵，使得其符合条件 3 的 $k = \frac{n(n^2-1)}{2} = \frac{n(n-1)(n+1)}{2}$。
- 设 $k_{\min} = \frac{n(n-1)(n+1)}{2}$，如果所给的 $k < k_{\min}$，即为无解，因为我们无法构造出元素更小的矩阵。

题解

题解

- 在同时满足条件 1 和条件 2 的基础上，同时满足以下三点的矩阵即为符合题意的矩阵。
  1. 至少存在一个符合题意的选择元素的方案。
  2. 对于矩阵中的任意两行，每一列上的两个元素的差都相等。
  3. 对于矩阵中的任意两列，每一行上的两个元素的差都相等。

题解

- 在同时满足条件 1 和条件 2 的基础上，同时满足以下三点的矩阵即为符合题意的矩阵。
  1. 至少存在一个符合题意的选择元素的方案。
  2. 对于矩阵中的任意两行，每一列上的两个元素的差都相等。
  3. 对于矩阵中的任意两列，每一行上的两个元素的差都相等。
- 前文提到的元素取遍 $[0,\ n^2-1]$ 的矩阵，已经满足了上述的第 2 点和第 3 点。我们只需在其基础上做一定改动，把第 n 行的所有元素都加上 $k-k_{\min}$，使其满足第 1 点即可。

代码实现

```python
for test_case in range(int(input())):
    n, k = [ int(i) for i in input().split() ]

    k_min = (n - 1) * n * (n + 1) // 2
    if k < k_min:
        print(-1)
        continue

    for i in range(n):
        for j in range(n):
            print(i * n + j + (k - k_min if i == n - 1 else 0), end = " ")
        print()
```

Problem F. 初生几何

题意

- 在平面直角坐标系中，抛物线 $y = x(k - x)$ 与直线 $y = -1$ 相交。抛物线与 x 轴的另一个交点为 A。
- 设线段 OA 上存在一动点 P，过点 P 作 y 轴的平行线交抛物线于点 B，交直线 $y = -1$ 于点 C。
- 试求 $OB^2 + AC^2$ 的最大值。

题解

题解

- 设 $OP = a$，$AP = b = k - a$，则 $BP = ab$，$CP = 1$。

题解

- 设 $OP = a$，$AP = b = k - a$，则 $BP = ab$，$CP = 1$。
- 注意到 $OB^2 + \mathrm{AC}^2 = a^2 + b^2 + a^2b^2 + 1 = k^2 + (ab - 1)^2$。

题解

- 设 $OP = a$，$AP = b = k - a$，则 $BP = ab$，$CP = 1$。
- 注意到 $OB^2 + \mathrm{AC}^2 = a^2 + b^2 + a^2b^2 + 1 = k^2 + (ab-1)^2$。
- 即 $ab - 1$ 取最大或最小时得到原式的最大值。

题解

- 设 $OP = a$，$AP = b = k - a$，则 $BP = ab$，$CP = 1$。
- 注意到 $OB^2 + \mathrm{AC}^2 = a^2 + b^2 + a^2b^2 + 1 = k^2 + (ab - 1)^2$。
- 即 $ab - 1$ 取最大或最小时得到原式的最大值。
- 当 P 点与 O 或 A 重合时有 $ab - 1$ 最小。

题解

- 设 $OP = a$，$AP = b = k - a$，则 $BP = ab$，$CP = 1$。
- 注意到 $OB^2 + \mathrm{AC}^2 = a^2 + b^2 + a^2b^2 + 1 = k^2 + (ab - 1)^2$。
- 即 $ab - 1$ 取最大或最小时得到原式的最大值。
- 当 P 点与 O 或 A 重合时有 $ab - 1$ 最小。
- 当 P 在抛物线对称轴上时有 $ab - 1$ 最大。

题解

- 设 $OP = a$，$AP = b = k - a$，则 $BP = ab$，$CP = 1$。
- 注意到 $OB^2 + \mathrm{AC}^2 = a^2 + b^2 + a^2b^2 + 1 = k^2 + (ab-1)^2$。
- 即 $ab - 1$ 取最大或最小时得到原式的最大值。
- 当 P 点与 O 或 A 重合时有 $ab - 1$ 最小。
- 当 P 在抛物线对称轴上时有 $ab - 1$ 最大。
- 所以答案为 $\max\left\{k^2 + 1, \left(\frac{k^2}{4} + 1\right)^2\right\}$。

代码实现

```python
for test_case in range(int(input())):
    a, b = [ int(i) for i in input().split() ]
    k = a / b
    ans = max((k * k / 4 + 1) ** 2, k * k + 1)
    print(ans)
```

Problem G. 排序算法

题意

- 给定长为 n 的序列 a_i。
- 求下面这个排序算法的正确性。
- 如果正确，求出语句 `std::swap(a[i], a[j]);` 的执行次数。
- 保证 $1 \le n \le 2 \times 10^5$，$1 \le a_i \le 10^9$。

```
for (int i = 0; i < n; ++i)
    for (int j = 0; j < n; ++j)
        if (a[i] < a[j])
            std::swap(a[i], a[j]);
```

算法 0

算法 0

- 我会模拟！

算法 0

- 我会模拟！
- 模拟这个过程并判断序列是否有序。

算法 0

- 我会模拟！
- 模拟这个过程并判断序列是否有序。
- 时间复杂度 $O(n^2)$。

算法 0

- 我会模拟！
- 模拟这个过程并判断序列是否有序。
- 时间复杂度 $O(n^2)$。
- 期望通过 Subtask 0，得到 20 分。

观察 0

观察 0

- 我会思考性质！

观察 0

- 我会思考性质！
- **注意到算法是正确的。**

观察 0

- 我会思考性质！
- **注意到算法是正确的。**
- $i = 1$ 时，循环即找到最大值。

观察 0

- 我会思考性质！
- **注意到算法是正确的。**
- $i = 1$ 时，循环即找到最大值。
- $i \geq 2$，外层循环到 i 时，$a_1, a_2, \cdots, a_{i-1}$ 必然不小于 a_i。
  - 对 $j < i$ 的部分，最终得到的 a_i 必然是 i 前缀的最大值，即序列最大值。
  - 对 $j \geq i$ 的部分，由于 a_i 已经是最大值，不会发生交换。

观察 0

- 我会思考性质！
- **注意到算法是正确的。**
- $i = 1$ 时，循环即找到最大值。
- $i \geq 2$，外层循环到 i 时，$a_1, a_2, \cdots, a_{i-1}$ 必然不小于 a_i。
  - 对 $j < i$ 的部分，最终得到的 a_i 必然是 i 前缀的最大值，即序列最大值。
  - 对 $j \geq i$ 的部分，由于 a_i 已经是最大值，不会发生交换。
- 所以算法正确，输出 NO 并没有分。

算法 1

算法 1

- 我还会思考性质！

算法 1

- 我还会思考性质！
- 性质：$\boldsymbol{i \geq 2}$，每轮外层循环对答案的贡献为 i 前缀中比 a_i 大的不同元素的个数。

算法 1

- 我还会思考性质！
- 性质：$\boldsymbol{i \geq 2}$，**每轮外层循环对答案的贡献为 i 前缀中比 a_i 大的不同元素的个数**。
- 朴素处理第 1 轮外层循环，再维护一个数据结构支持插入元素、查询大于某个元素的不同元素个数。

算法 1

- 我还会思考性质！
- 性质：$\boldsymbol{i \geq 2}$，**每轮外层循环对答案的贡献为 i 前缀中比 a_i 大的不同元素的个数**。
- 朴素处理第 1 轮外层循环，再维护一个数据结构支持插入元素、查询大于某个元素的不同元素个数。
- 平衡树、线段树和树状数组即可。

算法 1

- 我还会思考性质！
- **性质：$\boldsymbol{i \geq 2}$，每轮外层循环对答案的贡献为 i 前缀中比 a_i 大的不同元素的个数。**
- 朴素处理第 1 轮外层循环，再维护一个数据结构支持插入元素、查询大于某个元素的不同元素个数。
- 平衡树、线段树和树状数组即可。
- 时间复杂度 $O(n\log n)$。python 常数大只有树状数组能过。

算法 1

- 我还会思考性质！
- 性质：**$\boldsymbol{i \geq 2}$，每轮外层循环对答案的贡献为 i 前缀中比 a_i 大的不同元素的个数**。
- 朴素处理第 1 轮外层循环，再维护一个数据结构支持插入元素、查询大于某个元素的不同元素个数。
- 平衡树、线段树和树状数组即可。
- 时间复杂度 $O(n \log n)$。python 常数大只有树状数组能过。
- 期望通过所有子任务。

彩蛋

- 论文题。https://arxiv.org/pdf/2110.01111.pdf

Problem H. 购买车券

题意

- 给定一棵 n 个点的无根树 T，每次删去一个叶子结点直至删空。
- 对合法的操作序列计数，对 $998\,244\,353$ 取模。
- 叶子结点的定义是度数不大于 1 的结点。
- 操作序列不同，当且仅当某一次删去的叶子不同。
- 保证 $1 \le n \le 2 \times 10^5$。

算法 0

算法 0

- 我会枚举！

算法 0

- 我会枚举！
- 考虑在树上暴搜方案。

算法 0

- 我会枚举！
- 考虑在树上暴搜方案。
- 时间复杂度 $O(n!)$。

算法 0

- 我会枚举！
- 考虑在树上暴搜方案。
- 时间复杂度 $O(n!)$。
- 期望通过 Subtask 0。

算法 1

算法 1

- 我会性质！

算法 1

- 我会性质！
- 由于整棵树是一条链，在删除前 $n-1$ 个结点时，树上都恰好有 2 个叶子结点，故答案为 2^{n-1}。

算法 1

- 我会性质！
- 由于整棵树是一条链，在删除前 $n-1$ 个结点时，树上都恰好有 2 个叶子结点，故答案为 2^{n-1}。
- 结合算法 0 期望通过 Subtask 0, 2。

算法 2

算法 2

- 我还会性质！

算法 2

- 我还会性质！
- 对一个菊花树，在删除第 i（$i \le n-2$）个结点时，树上有 $n-i$ 个叶子结点；在删除第 $n-1$ 个结点时，树上有 2 个叶子结点。

算法 2

- 我还会性质！
- 对一个菊花树，在删除第 i（$i \le n-2$）个结点时，树上有 $n-i$ 个叶子结点；在删除第 $n-1$ 个结点时，树上有 2 个叶子结点。
- 故答案为：

算法 2

- 我还会性质！
- 对一个菊花树，在删除第 i（$i \le n-2$）个结点时，树上有 $n-i$ 个叶子结点；在删除第 $n-1$ 个结点时，树上有 2 个叶子结点。
- 故答案为：

$$(n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 2 \times 1 = (n-1)! \times 2$$

算法 2

- 我还会性质！
- 对一个菊花树，在删除第 i（$i \leq n-2$）个结点时，树上有 $n-i$ 个叶子结点；在删除第 $n-1$ 个结点时，树上有 2 个叶子结点。
- 故答案为：
$$(n-1)\times(n-2)\times\cdots\times 3\times 2\times 2\times 1=(n-1)!\times 2$$
- 结合算法 0, 1 期望通过 Subtask 0, 2, 3。

算法 3

算法 3

- 我会“動的計画法”！

算法 3

- 我会“動的計画法”！
- 思考对于一棵一般的树怎么做。

算法 3

- 我会“動的計画法”！
- 思考对于一棵一般的树怎么做。
- 以结点 x 为根，设 f_u 代表将结点 u 所在的子树删空的方案数。

算法 3

- 我会“動的計画法”！
- 思考对于一棵一般的树怎么做。
- 以结点 x 为根，设 f_u 代表将结点 u 所在的子树删空的方案数。
- 不好转移，我们钦定 u 是 u 所在的子树最后删掉的点。

算法 3

- 我会“動的計画法”！
- 思考对于一棵一般的树怎么做。
- 以结点 x 为根，设 f_u 代表将结点 u 所在的子树删空的方案数。
- 不好转移，我们钦定 u 是 u 所在的子树最后删掉的点。
- 这样就比较好转移了。转移考虑 u 的若干个儿子，第 i 个儿子为 v_i：

算法 3

- 我会“動的計画法”！
- 思考对于一棵一般的树怎么做。
- 以结点 x 为根，设 f_u 代表将结点 u 所在的子树删空的方案数。
- 不好转移，我们钦定 u 是 u 所在的子树最后删掉的点。
- 这样就比较好转移了。转移考虑 u 的若干个儿子，第 i 个儿子为 v_i：

$$\boxed{f_u = \prod_i \left[f_{v_i} \binom{\sum_j^i s_j}{s_i}\right] = \left(\prod_i f_{v_i}\right) \frac{(s_u - 1)!}{\prod_i (s_{v_i}!)}}$$

算法 3

- 我会“動的計画法”！
- 思考对于一棵一般的树怎么做。
- 以结点 x 为根，设 f_u 代表将结点 u 所在的子树删空的方案数。
- 不好转移，我们钦定 u 是 u 所在的子树最后删掉的点。
- 这样就比较好转移了。转移考虑 u 的若干个儿子，第 i 个儿子为 v_i：

$$\boxed{f_u = \prod_i \left[f_{v_i} \binom{\sum_j^i s_j}{s_i} \right] = \left(\prod_i f_{v_i} \right) \frac{(s_u - 1)!}{\prod_i (s_{v_i}!)}}$$

- 其中 s_u 代表以 u 为根所在的子树大小。

算法 3

- 我会“動的計画法”！
- 思考对于一棵一般的树怎么做。
- 以结点 x 为根，设 f_u 代表将结点 u 所在的子树删空的方案数。
- 不好转移，我们钦定 u 是 u 所在的子树最后删掉的点。
- 这样就比较好转移了。转移考虑 u 的若干个儿子，第 i 个儿子为 v_i：

$$\boxed{f_u = \prod_i \left[f_{v_i} \binom{\sum_j^i s_j}{s_i} \right] = \left(\prod_i f_{v_i} \right) \frac{(s_u - 1)!}{\prod_i (s_{v_i}!)}}$$

- 其中 s_u 代表以 u 为根所在的子树大小。
- 令 $x = 1, 2, \cdots, n$ 跑这个 DP，对所有 f_x 求和即为答案。

算法 3

- 我会“動的計画法”！
- 思考对于一棵一般的树怎么做。
- 以结点 x 为根，设 f_u 代表将结点 u 所在的子树删空的方案数。
- 不好转移，我们钦定 u 是 u 所在的子树最后删掉的点。
- 这样就比较好转移了。转移考虑 u 的若干个儿子，第 i 个儿子为 v_i：

$$\boxed{f_u = \prod_i \left[f_{v_i} \binom{\sum_j^i s_j}{s_i} \right] = \left(\prod_i f_{v_i} \right) \frac{(s_u - 1)!}{\prod_i (s_{v_i}!)}}$$

- 其中 s_u 代表以 u 为根所在的子树大小。
- 令 $x = 1, 2, \cdots, n$ 跑这个 DP，对所有 f_x 求和即为答案。
- 时间复杂度为 $O(n^2)$。结合算法 1, 2 期望通过 Subtask 0, 1, 2, 3。

算法 4

算法 4

- 我会二次扫描！

算法 4

- 我会二次扫描！
- 考虑优化。

算法 4

- 我会二次扫描！
- 考虑优化。
- 设 g_u 代表将结点 u 视为树根且最后删，删空整棵树的方案数。

算法 4

- 我会二次扫描！
- 考虑优化。
- 设 g_u 代表将结点 u 视为树根且最后删，删空整棵树的方案数。
- 转移依然考虑 u 的儿子 v：

算法 4

- 我会二次扫描！
- 考虑优化。
- 设 g_u 代表将结点 u 视为树根且最后删，删空整棵树的方案数。
- 转移依然考虑 u 的儿子 v：

$$\boxed{g_v = f_v \times \frac{g_u}{f_v \times \binom{n-1}{n-s_v-1}} \times \binom{n-1}{s_v-1} = g_u \times \frac{\binom{n-1}{s_v-1}}{\binom{n-1}{n-s_v-1}} = g_u \times \frac{s_v}{n-s_v}}$$

算法 4

- 我会二次扫描！
- 考虑优化。
- 设 g_u 代表将结点 u 视为树根且最后删，删空整棵树的方案数。
- 转移依然考虑 u 的儿子 v：

$$\boxed{g_v = f_v \times \frac{g_u}{f_v \times \binom{n-1}{n-s_v-1}} \times \binom{n-1}{s_v-1} = g_u \times \frac{\binom{n-1}{s_v-1}}{\binom{n-1}{n-s_v-1}} = g_u \times \frac{s_v}{n-s_v}}$$

- 答案即为 $\sum g_i$。

算法 4

- 我会二次扫描！
- 考虑优化。
- 设 g_u 代表将结点 u 视为树根且最后删，删空整棵树的方案数。
- 转移依然考虑 u 的儿子 v：

$$\boxed{g_v = f_v \times \frac{g_u}{f_v \times \binom{n-1}{n-s_v-1}} \times \binom{n-1}{s_v-1} = g_u \times \frac{\binom{n-1}{s_v-1}}{\binom{n-1}{n-s_v-1}} = g_u \times \frac{s_v}{n-s_v}}$$

- 答案即为 $\sum g_i$。
- 时间复杂度为 $O(n)$。期望通过所有子任务。

算法 4

- 我会二次扫描！
- 考虑优化。
- 设 g_u 代表将结点 u 视为树根且最后删，删空整棵树的方案数。
- 转移依然考虑 u 的儿子 v：

$$\boxed{g_v = f_v \times \frac{g_u}{f_v \times \binom{n-1}{n-s_v-1}} \times \binom{n-1}{s_v-1} = g_u \times \frac{\binom{n-1}{s_v-1}}{\binom{n-1}{n-s_v-1}} = g_u \times \frac{s_v}{n-s_v}}$$

- 答案即为 $\sum g_i$。
- 时间复杂度为 $O(n)$。期望通过所有子任务。
- *题目来源：2023 年全国高中数学联赛一试第 8 题。*

Problem I. 花卉培育

题意

构造序列 a_i 使得其满足条件，或判断无解：

- 对每个 $1 \le j \le q$，$\left(\prod_{i=l_j}^{r_j} a_i\right) \bmod 3 = \mathrm{v_j}$。

$1 \le n, q \le 3 \times 10^5$。

观察 1

观察 1

- $0 \le a_i \le 10^9$ 等价于 $0 \le a_i \le 2$。

算法 0

算法 0

- 我会看表！

算法 0

- 我会看表！
- 对 Subtask 0 发现 $q = 0$，输出任意一个长为 n 的序列即可。

算法 0

- 我会看表！
- 对 Subtask 0 发现 $q = 0$，输出任意一个长为 n 的序列即可。
- 期望得分 2 分。

算法 1

算法 1

- 我会枚举！

算法 1

- 我会枚举！
- 对 Subtask 1 发现 $1 \leq n, q \leq 10$。

算法 1

- 我会枚举！
- 对 Subtask 1 发现 $1 \le n, q \le 10$。
- 枚举每一个位置放数，判断是否满足条件即可。

算法 1

- 我会枚举！
- 对 Subtask 1 发现 $1 \le n, q \le 10$。
- 枚举每一个位置放数，判断是否满足条件即可。
- 时间复杂度 $O(n \cdot 3^n)$。

观察 2

观察 2

- 我会观察性质！

观察 2

- 我会观察性质！
- 注意到：

观察 2

- 我会观察性质！
- 注意到：

$$2^i \bmod 3 = \begin{cases} 2, & i\text{ 是奇数；} \\ 1, & i\text{ 是偶数；} \end{cases}$$

观察 2

- 我会观察性质！
- 注意到：

$$2^i \bmod 3 = \begin{cases} 2, & i\text{ 是奇数；} \\ 1, & i\text{ 是偶数；} \end{cases}$$

- 并且对 $v_i = 0$ 的询问，只需要让这之间有 0 即可。

算法 3

算法 3

- 考虑设 x_i 代表第 i 个位置填 1 或 2，分别为 0 或 1。

算法 3

- 考虑设 x_i 代表第 i 个位置填 1 或 2，分别为 0 或 1。
- $v_i = 1$ 代表有偶数个 x_i 为 1，而 $v_i = 2$ 代表有奇数个 x_i 为 0。

算法 3

- 考虑设 x_i 代表第 i 个位置填 1 或 2，分别为 0 或 1。
- $v_i = 1$ 代表有偶数个 x_i 为 1，而 $v_i = 2$ 代表有奇数个 x_i 为 0。
- 考虑 q 个异或方程：

算法 3

- 考虑设 x_i 代表第 i 个位置填 1 或 2，分别为 0 或 1。
- $v_i\ =\ 1$ 代表有偶数个 x_i 为 1，而 $v_i = 2$ 代表有奇数个 x_i 为 0。
- 考虑 q 个异或方程：

$$x_l \oplus x_{l+1} \oplus \cdots \oplus x_r\ =\ 0 \text{ or } 1$$

算法 3

- 考虑设 x_i 代表第 i 个位置填 1 或 2，分别为 0 或 1。
- $v_i = 1$ 代表有偶数个 x_i 为 1，而 $v_i = 2$ 代表有奇数个 x_i 为 0。
- 考虑 q 个异或方程：

$$x_l \oplus x_{l+1} \oplus \cdots \oplus x_r = 0 \text{ or } 1$$

- Bitset 优化高斯消元解异或方程组即可。

算法 3

- 考虑设 x_i 代表第 i 个位置填 1 或 2，分别为 0 或 1。
- $v_i = 1$ 代表有偶数个 x_i 为 1，而 $v_i = 2$ 代表有奇数个 x_i 为 0。
- 考虑 q 个异或方程：

$$x_l \oplus x_{l+1} \oplus \cdots \oplus x_r = 0 \text{ or } 1$$

- Bitset 优化高斯消元解异或方程组即可。
- 维护某个点有没有被覆盖，用前缀和 + 差分可以解决 $v_i = 0$ 的询问。

算法 3

- 考虑设 x_i 代表第 i 个位置填 1 或 2，分别为 0 或 1。
- $v_i = 1$ 代表有偶数个 x_i 为 1，而 $v_i = 2$ 代表有奇数个 x_i 为 0。
- 考虑 q 个异或方程：

$$x_l \oplus x_{l+1} \oplus \cdots \oplus x_r = 0 \text{ or } 1$$

- Bitset 优化高斯消元解异或方程组即可。
- 维护某个点有没有被覆盖，用前缀和 + 差分可以解决 $v_i = 0$ 的询问。
- 时间复杂度为 $O(\frac{n^3}{w})$。

算法 4

算法 4

- 对于填 1, 3 的情况，考虑一个前缀异或 $y_i = \bigoplus x_j, j \le i$。

算法 4

- 对于填 1, 3 的情况，考虑一个前缀异或 $y_i = \bigoplus x_j, j \leq i$。
- 这些方程等价于 $y_r \oplus y_{l-1} = 0 \text{ or } 1$。

算法 4

- 对于填 1, 3 的情况，考虑一个前缀异或 $y_i = \bigoplus x_j, j \le i$。
- 这些方程等价于 $\mathrm{y}_r \oplus y_{l-1} \; = \; 0 \text{ or } 1$。
- 还要高斯消元吗？

算法 4

- 对于填 1, 3 的情况，考虑一个前缀异或 $y_i = \bigoplus x_j, j \le i$。
- 这些方程等价于 $y_r \oplus y_{l-1} = 0 \text{ or } 1$。
- 还要高斯消元吗？
- 求解的过程就是把 $y_i = 0$ 的放到一个集合，1 的放到另一个集合。

算法 4

- 对于填 1, 3 的情况，考虑一个前缀异或 $y_i = \oplus x_j, j \le i$。
- 这些方程等价于 $y_r \oplus y_{l-1} = 0$ or 1。
- 还要高斯消元吗？
- 求解的过程就是把 $y_i = 0$ 的放到一个集合，1 的放到另一个集合。
- 考虑进行 2-SAT。

算法 4

- 对于填 1, 3 的情况，考虑一个前缀异或 $y_i = \bigoplus x_j\,, j \le i$。
- 这些方程等价于 $y_r \oplus y_{l-1} = 0$ or 1。
- 还要高斯消元吗？
- 求解的过程就是把 $y_i = 0$ 的放到一个集合，1 的放到另一个集合。
- 考虑进行 2-SAT。
- 时间复杂度为 $O(n)$。期望得分 100 分。

Problem J. 繁星满天

题意

题意

- 用不超过 1500 次操作作出点 $(\frac{a}{b},\frac{c}{d})$。

题意

- 用不超过 1500 次操作作出点 $(\frac{a}{b},\frac{c}{d})$。
- 操作包括：
  - 作出一个整点；
  - 作出两条经过已知点的直线的交点。

题意

- 用不超过 1500 次操作作出点 $(\frac{a}{b},\frac{c}{d})$。
- 操作包括：
  - 作出一个整点；
  - 作出两条经过已知点的直线的交点。
- 每次作出的点都要在 $(0,0)$ 和 (p,p) 之间，即 $0 \le x,y \le p$。

题意

- 用不超过 1500 次操作作出点 $(\frac{a}{b},\frac{c}{d})$。
- 操作包括：
  - 作出一个整点；
  - 作出两条经过已知点的直线的交点。
- 每次作出的点都要在 $(0,0)$ 和 (p,p) 之间，即 $0 \leq\ x,y \leq\ p$。
- 保证 $1 \leq a \leq b \leq 10^7$，$1 \leq c \leq d \leq 10^7$，$2 \leq p \leq 10^7$。

题意

- 用不超过 1500 次操作作出点 $(\frac{a}{b},\frac{c}{d})$。
- 操作包括：
  - 作出一个整点；
  - 作出两条经过已知点的直线的交点。
- 每次作出的点都要在 $(0,0)$ 和 (p,p) 之间，即 $0\le x,y\le p$。
- 保证 $1\le a\le b\le 10^7$，$1\le c\le d\le 10^7$，$2\le p\le 10^7$。
- 下文记 $t=\max(b,d)$。

算法 0

算法 0

限制：$\boldsymbol{t \le 10^3}$，$\boldsymbol{p \ge 10^7}$。

算法 0

限制：$\boldsymbol{t \leq 10^3}$，$\boldsymbol{p \geq 10^7}$。

- 这时怎么连都是合法的。

算法 0

限制：$\boldsymbol{t \le 10^3}$，$\boldsymbol{p \ge 10^7}$。

- 这时怎么连都是合法的。
- 连接 $(0,0)$ 和 $(1,d)$。取 $x=c$ 与其交于 $(c,\frac{c}{d})$。

算法 0

限制：$\boldsymbol{t \leq 10^3}$，$\boldsymbol{p \geq 10^7}$。

- 这时怎么连都是合法的。
- 连接 $(0,0)$ 和 $(1,d)$。取 $x=c$ 与其交于 $(c,\frac{c}{d})$。
- 这样就可以作出 $y=\frac{c}{d}$。

算法 0

限制：$t \le 10^3$，$p \ge 10^7$。

- 这时怎么连都是合法的。
- 连接 $(0,0)$ 和 $(1,d)$。取 $x=c$ 与其交于 $(c,\frac{c}{d})$。
- 这样就可以作出 $y=\frac{c}{d}$。
- 同理作出 $x=\frac{a}{b}$。交点就是 $(\frac{a}{b},\frac{c}{d})$。

算法 0

限制：$t \le 10^3$，$p \ge 10^7$。

- 这时怎么连都是合法的。
- 连接 $(0,0)$ 和 $(1,d)$。取 $x=c$ 与其交于 $(c,\frac{c}{d})$。
- 这样就可以作出 $y=\frac{c}{d}$。
- 同理作出 $x=\frac{a}{b}$。交点就是 $(\frac{a}{b},\frac{c}{d})$。
- 期望通过 Subtask 0。

算法 0.5

算法 0.5

限制：$\boldsymbol{t \leq 10^6}$，$\boldsymbol{p \geq t}$。

算法 0.5

限制：$t \leq 10^6$，$p \geq t$。

- 和上一个没有什么区别。只是为了防止写挂留了一档分。

算法 1

算法 1

限制：$\boldsymbol{t \leq 10^7}$，$\boldsymbol{p = \left\lceil \frac{t}{2} \right\rceil + 1}$。

算法 1

限制：$\boldsymbol{t \le 10^7}$，$\boldsymbol{p = \left\lceil \frac{t}{2} \right\rceil + 1}$。

- 这时你不能直接连接 $(0,0)$ 和 $(1,t)$ 了。

算法 1

限制：$\boldsymbol{t \le 10^7}$，$\boldsymbol{p = \left\lceil \frac{t}{2} \right\rceil + 1}$。

- 这时你不能直接连接 $(0,0)$ 和 $(1,t)$ 了。
- 你可以先作出 $\left(\frac{1}{2},\frac{t}{2}\right)$ 或 $(1,\frac{t}{2})$。然后再进行连接。

算法 1

限制：$\boldsymbol{t \le 10^7}$，$\boldsymbol{p = \left\lceil \frac{t}{2} \right\rceil + 1}$。

- 这时你不能直接连接 $(0,0)$ 和 $(1,t)$ 了。
- 你可以先作出 $\left(\frac{1}{2},\frac{t}{2}\right)$ 或 $(1,\frac{t}{2})$。然后再进行连接。
- 假设 t 对应的分子为 s。如果 $s \ge \frac{t}{2}$ 就作第二个点，反之作第一个点。

算法 1

限制：$t \le 10^7$，$p = \left\lceil \frac{t}{2} \right\rceil + 1$。

- 这时你不能直接连接 $(0,0)$ 和 $(1,t)$ 了。
- 你可以先作出 $\left(\frac{1}{2},\frac{t}{2}\right)$ 或 $(1,\frac{t}{2})$。然后再进行连接。
- 假设 t 对应的分子为 s。如果 $s \ge \frac{t}{2}$ 就作第二个点，反之作第一个点。
- 期望通过 Subtask 2。

算法 2

算法 2

限制：$\boldsymbol{t \le 10^7}$，$\boldsymbol{p = \left\lceil \sqrt{t} \right\rceil + 2}$。

算法 2

限制：$\boldsymbol{t \le 10^7}$，$\boldsymbol{p = \left\lfloor \sqrt{t} \right\rfloor + 2}$。

- 发现这是根号，根号分治一下。

算法 2

限制：$\boldsymbol{t \le 10^7}$，$\boldsymbol{p = \lceil\sqrt{t}\rceil + 2}$。

- 发现这是根号，根号分治一下。
- 将每一个格子分成 $\lceil\sqrt{t}\rceil \times \lceil\sqrt{t}\rceil$ 后，情况转化为了 Subtask 1。

算法 2

限制：$t \le 10^7$，$p = \lceil \sqrt{t} \rceil + 2$。

- 发现这是根号，根号分治一下。
- 将每一个格子分成 $\lceil \sqrt{t} \rceil \times \lceil \sqrt{t} \rceil$ 后，情况转化为了 Subtask 1。
- 然后就能做了。

算法 3

算法 3

限制：$\boldsymbol{t \le 10^7}$，$\boldsymbol{p = 2}$。

算法 3

限制：$\boldsymbol{t \le 10^7}$，$\boldsymbol{p = 2}$。

- 我们先作出 $\frac{1}{t}$，然后将其扩大 s 倍。

算法 3

限制：$\boldsymbol{t \le 10^7}$，$\boldsymbol{p = 2}$。

- 我们先作出 $\frac{1}{t}$，然后将其扩大 s 倍。
- 可以作出 $(\frac{t}{2^k}, \frac{1}{2^k})$，其中 $2^k > t$。这样就有了 $\frac{1}{t}$。

算法 3

限制：$t \leq 10^7$，$p = 2$。

- 我们先作出 $\frac{1}{t}$，然后将其扩大 s 倍。
- 可以作出 $(\frac{t}{2^k}, \frac{1}{2^k})$，其中 $2^k > t$。这样就有了 $\frac{1}{t}$。
- 扩大 s 倍可以考虑倍增。

算法 3

限制：$\boldsymbol{t \le 10^7}$，$\boldsymbol{p = 2}$。

- 我们先作出 $\frac{1}{t}$，然后将其扩大 s 倍。
- 可以作出 $(\frac{t}{2^k},\frac{1}{2^k})$，其中 $2^k > t$。这样就有了 $\frac{1}{t}$。
- 扩大 s 倍可以考虑倍增。
- 实现的细节很多。注意封装。