

# Технокубок 2017/2018. Третий отборочный раунд.

## Условия и разбор задач.

### A. ACM ICPC

ограничение по времени на тест

2 секунды

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

В одном маленьком, но очень гордом ВУЗе было принято решение выиграть ACM ICPC. Для этого нужно составить как можно больше команд из трёх человек, но так как желающих студентов всего 6, было решено составить две команды.

Участнику с номером  $i$  по результатам тестирования была присвоена некоторая сила  $a_i$ . Силой команды называется сумма сил всех участников в этой команде. Теперь руководству ВУЗа интересно, возможно ли собрать две команды с одинаковой силой. Ответьте на этот вопрос руководства.

#### Входные данные

В единственной строке находятся шесть целых чисел  $a_1, \dots, a_6$  ( $0 \leq a_i \leq 1000$ ) — силы участников.

#### Выходные данные

Выведите «YES» (без кавычек), если из данных участников можно собрать две команды с одинаковой силой, и «NO» (без кавычек) — иначе.

Вы можете выводить каждую букву в любом регистре (строчную или заглавную).

#### Примеры

##### входные данные

1 3 2 1 2 1

##### выходные данные

YES

##### входные данные

1 1 1 1 1 99

## выходные данные

NO

### Примечание

В первом тесте можно взять в первую команду 1-го, 2-го и 6-го участника, во вторую — 3-го, 4-го и 5-го: силы команд будут  $1 + 3 + 1 = 2 + 1 + 2 = 5$ .

Во втором тесте участник номер 6 слишком сильный и его команда будет заведомо сильнее другой.

### 886A - ACM ICPC

В этой задаче достаточно перебрать все тройки чисел, проверяя сумму этой тройки на равенство сумме оставшейся тройки. Если равенство достигается, ответ YES, иначе — NO.

## В. Влад и столовые

ограничение по времени на тест

2 секунды

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Влад очень любит ходить в столовые. За свою жизнь он сделал это  $n$  раз. К сожалению, в последнее время он начал замечать, что его выбор не отличается разнообразием. Ему хочется это исправить, и поэтому он провёл следующее исследование.

Сначала Влад присвоил каждой столовой индивидуальный номер. Затем он выписал в ряд все номера столовых, в которых он был, в порядке посещения. Теперь он хочет найти такую столовую, что последний раз он в ней был раньше последних посещений всех остальных столовых. Иными словами, он хочет найти столовую, в которой не был дольше всего на текущий момент. Помогите ему в этом.

### Входные данные

В первой строке находится одно целое число  $n$  ( $1 \leq n \leq 2 \cdot 10^5$ ) — количество выписанных Владом номеров столовых.

Во второй строке находятся  $n$  целых чисел  $a_1, a_2, \dots, a_n$  ( $0 \leq a_i \leq 2 \cdot 10^5$ ) — номера столовых в порядке посещения Владом. Влад мог посетить одну и ту же столовую несколько раз. Обратите внимание, не обязательно, что столовые пронумерованы подряд.

### Выходные данные

Выведите одно число — номер столовой, в которой Влад не был как можно дольше.

### Примеры

#### входные данные

```
5  
1 3 2 1 2
```

#### выходные данные

```
3
```

#### входные данные

```
6  
2 1 2 2 4 1
```

#### выходные данные

```
2
```

## Примечание

Рассмотрим первый тест. В нём есть три столовые, причём последнее посещение столовых с номерами 1 и 2 состоялись после последнего посещения столовой с номером 3, поэтому эта столовая — искомая.

Во втором тесте также есть три столовые с номерами 1, 2 и 4. Столовые с номерами 1 и 4 были посещены после последнего посещения столовой с номером 2, поэтому ответ 2. Обратите внимание, что Влад мог пропустить некоторые номера в нумерации столовых.

## 886B - Влад и столовые

В этой задаче есть два решения.

1. Давайте в массив `last` для каждой столовой запишем момент её последнего вхождения, а затем найдём минимум в этом массиве. Номер, на котором достигается является номером искомой столовой.

2. Будем бежать по массиву с конца и для каждого номера столовой помнить, посещали мы его раньше или нет. Если очередной рассмотренный нами номер не встречался ранее, то он будет являться нашим новым кандидатом на ответ. В итоге последний кандидат и является ответом.

## С. Петя и катакомбы

ограничение по времени на тест

1 секунда

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Однажды Петя — очень храбрый исследователь — решил заняться изучением великих парижских катакомб. Так как Петя не очень опытный в этом деле, его исследование заключается в хождении по катакомбам.

Катакомбы представляют из себя набор комнат, некоторые из которых соединены переходами, переход может соединять комнату с самой собой, по переходам можно ходить в любую сторону. Переходы могут находиться на разных глубинах, что позволяет им не пересекаться. Каждую минуту Петя выбирает произвольным образом переход из комнаты, в которой он находится, и проходит его за одну минуту. Когда он заходит в комнату в минуту  $i$ , он делает в журнал запись — число  $t_i$ :

- Если Петя уже был в этой комнате, он записывает минуту, когда он был в этой комнате в предыдущий раз;
- Иначе Петя записывает в журнал произвольное неотрицательное целое число, строго меньшее текущей минуты  $i$ .

Изначально Петя находился в какой-то из комнат в минуту 0, при этом число  $t_0$  он не записывал.

В какой-то момент Петя устал, бросил журнал и пошёл домой (не обязательно в той же комнате, из которой он начинал). Вася нашёл этот журнал, и теперь ему интересно: какое наименьшее количество комнат может быть в катакомбах, согласно журналу Пети?

### Входные данные

В первой строке находится одно целое число  $n$  ( $1 \leq n \leq 2 \cdot 10^5$ ) — количество записей в журнале Пети.

Во второй строке находятся  $n$  целых чисел  $t_1, t_2, \dots, t_n$  ( $0 \leq t_i < i$ ) — записи в журнале.

### Выходные данные

В первой строке выведите единственное число — минимальное возможное количество комнат в парижских катакомбах.

## Примеры

### входные данные

2  
0 0

### выходные данные

2

### входные данные

5  
0 1 0 1 3

### выходные данные

3

## Примечание

В первом тестовом примере последовательность комнат, по которым прошёл Петя, могла быть  $1 \rightarrow 1 \rightarrow 2$  или  $1 \rightarrow 2 \rightarrow 1$ , или, например,  $1 \rightarrow 2 \rightarrow 3$ . Минимальное число комнат равно 2.

Во втором тестовом примере эта последовательность могла быть  $1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 1$ .

## 886C - Петя и катакомбы

Сначала заметим, что если в журнале есть две одинаковые записи  $t_i = t_j$ ,  $i < j$ , то хотя бы одна из них сделана в ранее не посещённой комнате, так как иначе  $t_j$  было бы хотя бы  $i$ . Из этого следует, что среди всех одинаковых записей может быть не больше одной, которая соответствует ранее посещённой комнате.

Тогда если  $cnt_i$  — количество раз, которое  $i$  встретилось в журнале, то минимальное возможное количество комнат равно хотя бы  $\sum_{i=0}^{n-1} \max(0, cnt_i - 1)$ . Также нетрудно заметить, что эта величина достигается: скажем, что момент первого вхождения каждой записи соответствует возвращению в соответствующую комнату, а все последующие — посещению новых.

Таким образом, задача решается подсчётом  $cnt_i$  для всех  $i$  от 0 до  $n$  и вычислением указанной суммы.

Итоговая асимптотика решения:  $O(n)$ .

## D. Восстановление строки

ограничение по времени на тест

2 секунды

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Назовем подстроку некоторой строки самой частой, если количество ее вхождений не меньше количества вхождений любой другой подстроки.

Дано множество строк. Назовем какую-то строку хорошей, если все элементы множества являются ее самыми частыми подстроками. Восстановите непустую минимальную по длине хорошую строку, а из таких — лексикографически минимальную. Если же таких строк не существует, то выведите «NO» (без кавычек).

Подстрокой называется непрерывная подпоследовательность букв строки. Например, «ab», «c», «abc» являются подстроками строки «abc», а «ac» — нет.

Количество вхождений подстроки в строку равно количеству таких позиций в данной строке, в которых встречается данная подстрока. Вхождения подстроки могут перекрываться.

Строка  $a$  лексикографически меньше строки  $b$ , если  $a$  является префиксом  $b$ , или в  $a$  стоит меньшая буква на первой позиции, где  $a$  и  $b$  отличаются.

### **Входные данные**

В первой строке дано целое число  $n$  ( $1 \leq n \leq 10^5$ ) — количество строк в данном множестве.

Далее следуют  $n$  строк, каждая из которых содержит непустую строку из строчных латинских букв. Гарантируется, что все строки различны.

Суммарная длина всех данных строк не превышает  $10^5$ .

### **Выходные данные**

Выведите непустую минимальную по длине хорошую строку, а из таких минимальную лексикографически, либо «NO» (без кавычек), если таких строк не существует.

### **Примеры**

#### **входные данные**

4

mail

ai

lru

cf

#### ВЫХОДНЫЕ ДАННЫЕ

cfmailru

#### ВХОДНЫЕ ДАННЫЕ

3

kek

preseq

cheburek

#### ВЫХОДНЫЕ ДАННЫЕ

NO

### Примечание

Можно показать, что в первом тесте есть два варианта хорошей строки минимальной длины: «cfmailru» и «mailrucf». Первый вариант является минимальным лексикографически.

### 886D - Восстановление строки

- Если какая-то строка самая частая, то все ее подстроки тоже самые частые.
- Если строка  $ab$  или любая аналогичная является самой частой, то после буквы  $a$  всегда стоит буква  $b$ , перед  $b$  всегда стоит буква  $a$ .
- Если в графе следования букв существует цикл, то есть какая-то буква следует после себя (возможно через несколько других букв), то ответ не существует, потому что любая конечная строка не удовлетворяет предыдущему наблюдению.

Исходя из предыдущих наблюдений, такой граф разбивается на непересекающиеся пути. Все строки, соответствующие путям должны встречаться в непустой хорошей строке. Если вывести их в лексикографическом порядке, то получим лексикографически минимальную непустую хорошую строку минимальной длины.



## Е. Максимум в массиве

ограничение по времени на тест

2 секунды

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Как-то раз Петя решал очень интересную задачу. Однако, какие бы оптимизации Петя ни применял, его решение всё равно получало вердикт Time limit exceeded. После тщательного анализа кода он обнаружил, что проблема заключается в его функции поиска максимума в массиве из  $n$  положительных чисел, которая работала слишком долго. В отчаянии Петя решил добавить весьма неожиданное отсечение с параметром  $k$ , после чего функция приняла следующий вид:

```
int fast_max(int n, int a[]) {
    int ans = 0;
    int offset = 0;
    for (int i = 0; i < n; ++i)
        if (ans < a[i]) {
            ans = a[i];
            offset = 0;
        } else {
            offset = offset + 1;
            if (offset == k)
                return ans;
        }
    return ans;
}
```

Таким образом, функция последовательно перебирает элементы массива, поддерживая текущий максимум, при этом если после  $k$  последовательных итераций этот максимум не изменился, то текущий максимум возвращается как ответ.

Теперь Пете стало интересно, как часто его функция даёт неверный ответ. Он просит вас посчитать количество перестановок чисел от 1 до  $n$  таких, что результат работы его функции на этих перестановках не равен  $n$ . Так как ответ может быть большим, выведите его по модулю  $10^9 + 7$ .

## Входные данные

В единственной строке находятся два целых числа  $n$  и  $k$  ( $1 \leq n, k \leq 10^6$ ), разделённые пробелом — размер перестановок и параметр  $k$  соответственно.

## Выходные данные

Выведите количество перестановок, на которых функция Пети выдаёт неправильный ответ, взятое по модулю  $10^9 + 7$ .

### Примеры

#### входные данные

5 2

#### выходные данные

22

#### входные данные

5 3

#### выходные данные

6

#### входные данные

6 3

#### выходные данные

84

### Примечание

Искомые перестановки из второго примера:

[4, 1, 2, 3, 5], [4, 1, 3, 2, 5], [4, 2, 1, 3, 5], [4, 2, 3, 1, 5], [4, 3, 1, 2, 5], [4, 3, 2, 1, 5].

## 886E - Максимум в массиве

По сути требуется найти число таких перестановок  $p$  длины  $n$ , что существует позиция  $i$ , что  $p_i \neq n$ ,  $p_i$  больше любого  $p_j$  при  $j$  из  $[1, i - 1]$  и больше любого  $p_j$  при  $j$  из  $[i + 1, i + k]$ . Назовём такие перестановки хорошими.

Определим  $D(n)$  как число таких перестановок, которые удовлетворяют этому условию и дополнительно имеют  $p[n] = n$ . Заметим, что если  $k \geq n$ , то  $D(n) = 0$ . Переберём позицию  $j$  элемента  $n - 1$  в перестановке  $w$ , где  $w[n] = n$ . Если  $j < n - k$ , то перестановка очевидно хорошая. Иначе так как  $n - k > = j$ , то между числами  $n - 1$  и  $n$  находится меньше  $k$  элементов, следовательно перестановка будет хорошей только если  $i < j$  для  $i$  из определения. В таком случае префикс перестановки

до  $j$  позиции включительно будет хорошей перестановкой на  $j$  элементах на некоторых числах от 1 до  $n-1$ , у которой максимальное число стоит на последней позиции.

Таким образом верна рекурсивная формула:

$$D(n) = (n - k - 1) \cdot (n - 2)! + \sum_{h=n-k}^{n-1} D(h) \cdot \frac{(n-2)!}{(n-h-1)!}$$

Что можно посчитать за  $O(n^2)$ , или за  $O(n)$ , переписав в виде

$$D(n) = (n - k - 1) \cdot (n - 2)! + (n - 2)! \cdot \sum_{h=n-k}^{n-1} \frac{D(h)}{(h-1)!}$$

и используя префиксные суммы для выражения  $\frac{D(h)}{(h-1)!}$ .

## Ф. Симметричные проекции

ограничение по времени на тест

2 секунды

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Дано множество из  $n$  точек на плоскости. Назовем прямую, проходящую через начало координат, хорошей, если проекция данного множества точек на эту прямую образует симметричное мультимножество. Найдите количество хороших прямых.

Мультимножество — это множество, в котором разрешается несколько одинаковых элементов.

Мультимножество точек называется симметричным, если существует такая точка  $P$  на плоскости, что данное мультимножество обладает [центральной симметрией](#) относительно точки  $P$ .

### Входные данные

В первой строке дано целое число  $n$  ( $1 \leq n \leq 2000$ ) — количество точек в данном множестве.

Каждая из следующих  $n$  строк содержит два целых числа  $x_i$  и  $y_i$  ( $-10^6 \leq x_i, y_i \leq 10^6$ ), описывающие координаты точек множества. Гарантируется, что все точки различны.

### Выходные данные

Если хороших прямых бесконечно много, выведите  $-1$ .

Иначе выведите одно целое число — количество хороших прямых.

### Примеры

#### входные данные

```
3
1 2
2 1
3 3
```

#### выходные данные

```
3
```

#### входные данные

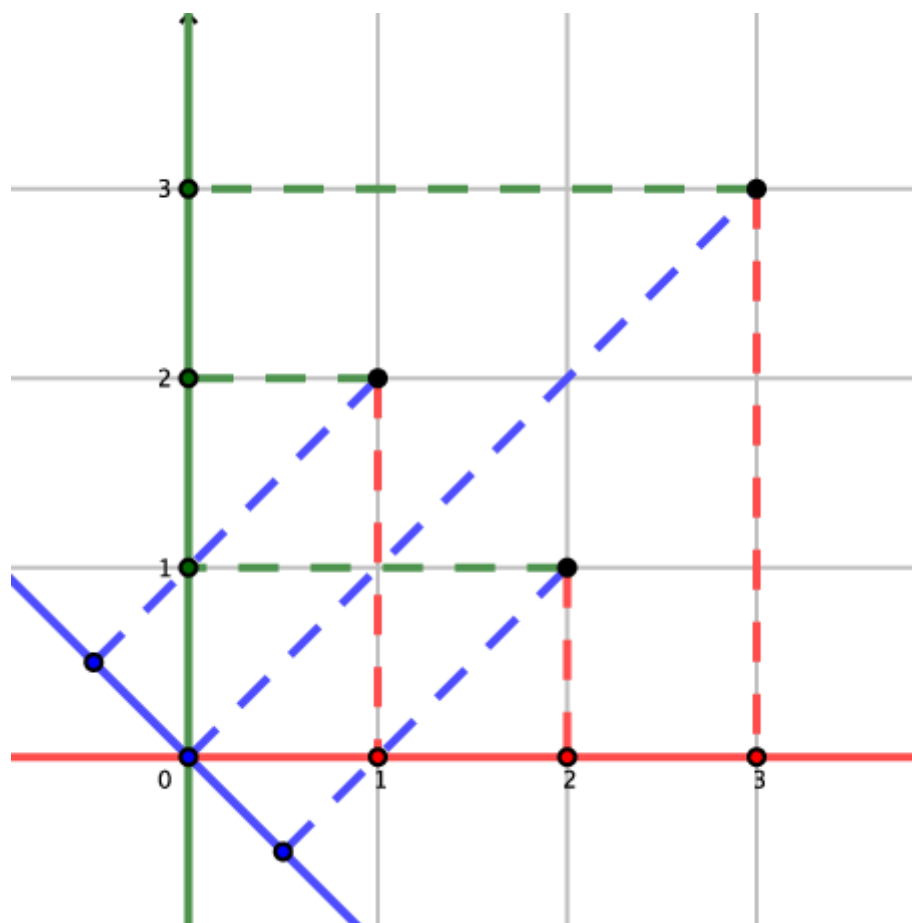
```
2
4 3
1 2
```

#### выходные данные

```
-1
```

## Примечание

Иллюстрация к первому тесту из условия:



Во втором примере хорошей является любая прямая, проходящая через начало координат.

## 886F - Симметричные проекции

Заметим, что при проекции множества точек на прямую, центр масс исходной системы точек переходит в центр масс спроецированной системы. Из этого следует, что если прямая является хорошей, то центр масс множества точек перейдет в центр симметрии. Также заметим, что если две точки симметричны относительно центра масс, то они будут симметричными при проекции на любую хорошую прямую. То есть такие точки можно выкинуть.

Возьмем любую из оставшихся точек. Переберем точку, которой будет симметрична выбранной точке при проекции. Остается заметить, что прямая, на которую нужно спроецировать, чтобы две выбранные точки перешли в себя, восстанавливается однозначно — это прямая, перпендикулярная прямой, проходящей через центр масс

и середину отрезка между выбранными точками. Поэтому есть не более  $n$  кандидатов, которые могут быть хорошими прямыми. Проверку, что прямая является хорошей, можно выполнять наивно за  $O(n \log n)$ .

Итоговая асимптотика:  $O(n^2 \log n)$