D. Поликарп и Div 3

ограничение по времени на тест: 3 секунды
ограничение по памяти на тест: 256 мегабайт

Поликарп очень любит числа, которые делятся на 3.

У него есть длинное число s. Поликарп хочет вырезать из него наибольшее количество чисел, которые делятся на 3. Для этого он проводит произвольное количество вертикальных разрезов между парами соседних цифр. В результате, после проведения m разрезов получается $m + 1$ число. Каждое из полученных чисел Поликарп анализирует и подсчитывает количество тех, которые делятся на 3.

Например, если исходное число $s = 3121$, то Поликарп может разрезать его на три части двумя разрезами $3|1|21$. В результате он получит два числа, которые делятся на 3.

Поликарп может провести произвольное количество разрезов, каждый разрез проводится между парой соседних цифр. Числа, которые получатся в итоге не могут содержать лишних лидирующих нулей (то есть число может начинаться с `0` тогда и только тогда, когда это число в точности один символ '0'). Например, `007`, `01` и `00099` не являются корректными числами, а `90`, `0` и `10001` — являются.

Какое наибольшее количество делящихся на три чисел он сможет получить?

Входные данные

В первой строке входных данных записано целое положительное число s. Длина числа s — от 1 до $2 \cdot 10^5$ цифр. Первая (самая левая) цифра отлична от `0`.

Выходные данные

Выведите наибольшее количество делящихся на 3 чисел, которые Поликарп сможет получить, проведя вертикальные разрезы в заданном числе s.

Примеры

входные данные	Скопировать
3121	
выходные данные	**Скопировать**
2	

входные данные	Скопировать
6	
выходные данные	**Скопировать**
1	

входные данные	Скопировать
1000000000000000000000000000000000	
выходные данные	**Скопировать**
33	

входные данные	Скопировать
201920181	
выходные данные	**Скопировать**
4	

Примечание

Пример оптимального разделения числа для первого теста — `3|1|21`.

Во втором тесте никаких разрезов совершать не нужно. Заданное число 6 образует одно число, делящееся на 3.

В третьем тесте надо провести разрезы между каждой парой цифр. В результате получим одну цифру `1` и 33 цифры `0`. Каждая из 33 цифр `0` образует число, которое делится на 3.

A. Гордыня

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Вам дан массив a длиной n, вы можете выполнять определенные операции над ним. Каждая операция выгладит следующим образом: выберите два **соседних** элемента из a, пусть это будут x и y, и замените один из них величиной $gcd(x, y)$, где gcd обозначает наибольший общий делитель.

Какое минимальное число операций необходимо, чтобы сделать все элементы массива равными 1?

Входные данные

Первая строка содержит одно целое число n ($1 \le n \le 2000$) — количество элементов в массиве.

Вторая строка содержит n целых чисел $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$) — элементы массива.

Выходные данные

Выведите `-1`, если невозможно сделать все элементы массива равными 1. Иначе выведите минимальное число операций, необходимых для того, чтобы сделать все числа равными 1.

Примеры

входные данные	Скопировать
5 2 2 3 4 6	
выходные данные	**Скопировать**
5	

входные данные	Скопировать
4 2 4 6 8	
выходные данные	**Скопировать**
-1	

входные данные	Скопировать
3 2 6 9	
выходные данные	**Скопировать**
4	

Примечание

В первом примере можно изменить все числа на 1, используя следующие 5 шагов:

- $[2, 2, 3, 4, 6]$.
- $[2, 1, 3, 4, 6]$
- $[2, 1, 3, 1, 6]$
- $[2, 1, 1, 1, 6]$
- $[1, 1, 1, 1, 6]$
- $[1, 1, 1, 1, 1]$

Можно доказать, что нельзя достичь того же меньше, чем за 5 операций.

C. Divan и битовые операции

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт

Однажды *Divan* проанализировал последовательность $a_1, a_2, \ldots, a_n$, состоящую из n целых неотрицательных чисел, следующим образом. Он рассмотрел все непустые *подпоследовательности* последовательности a, вычислил побитовое исключающее ИЛИ её элементов, после чего просуммировал все полученные результаты, получив *удобство* последовательности a.

Напомним, что последовательность c является *подпоследовательностью* последовательности d, если c может быть получена из d путем удаления нескольких элементов (возможно, ни одного). Например, $[1, 2, 3, 4]$, $[2, 4]$ и $[2]$ являются подпоследовательностями $[1, 2, 3, 4]$, а $[4, 3]$ и $[0]$ не являются.

Divan очень гордился проведенным анализом, но теперь потерял и последовательность a, и значение ее удобства! Однако *Divan* помнит значение побитового ИЛИ на m непрерывных подотрезках последовательности a. Оказалось, что каждый элемент последовательности входит хотя бы в один из этих m отрезков.

Divan просит вас найти удобство последовательности a, используя информацию, которую он помнит. Если возможны несколько значений удобства, выведите любое.

Так как ответ может быть большим, выведите его по модулю $10^9 + 7$.

Входные данные

Каждый тест содержит несколько наборов входных данных.

Первая строка содержит одно целое число t ($1 \le t \le 10^3$) — количество наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа n и m ($1 \le n, m \le 2 \cdot 10^5$) — количество чисел в последовательности и количество отрезков, значения побитового ИЛИ которых смог запомнить *Divan*, соответственно.

Далее идут m строк, в которых содержатся описания отрезков по одному в строке.

Каждый отрезок задаётся тремя целыми числами l, r и x ($1 \le l \le r \le n$, $0 \le x \le 2^{30} - 1$) — левая и правая граница отрезка, а также значение побитового ИЛИ элементов $a_l, a_{l+1}, \ldots, a_r$, соответственно.

Гарантируется, что каждый элемент последовательности входит хотя бы в один из данных отрезков. Гарантируется, что существует последовательность, которая удовлетворяет всем данным ограничениям.

Гарантируется, что сумма значений n, а также сумма значений m по всем наборам входных данных не превосходят $2 \cdot 10^5$.

Выходные данные

Для каждого набора входных данных выведите любое возможное удобство последовательности a по модулю $10^9 + 7$.

Пример

входные данные	Скопировать

```
3
2 1
1 2 2
3 2
1 3 5
2 3 5
5 4
1 2 7
3 3 7
4 4 0
4 5 2
```

выходные данные	Скопировать

```
4
20
112
```

Примечание

В первом примере одной из последовательностей, которая подходит под ограничения, является $[0, 2]$. Рассмотрим все её непустые подпоследовательности:

B. Хорошие последовательности

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Белка Лисска интересуется последовательностями. Также у нее есть предпочтения в целых числах. Она думает, что n целых чисел $a_1, a_2, ..., a_n$ *хорошие*.

Теперь ей интересны хорошие последовательности. Последовательность $x_1, x_2, ..., x_k$ называется *хорошей*, если она удовлетворяет следующим трем условиям:

- Последовательность строго возрастает, то есть $x_i < x_{i+1}$ для всех i $(1 \le i \le k - 1)$.
- Никакие два соседних элемента не являются взаимно простыми, то есть $gcd(x_i, x_{i+1}) > 1$ для всех i $(1 \le i \le k - 1)$ (где $gcd(p, q)$ обозначает наибольший общий делитель чисел p и q).
- Все элементы данной последовательности являются хорошими целыми числами.

Найдите длину самой длинной хорошей последовательности.

Входные данные

Входные данные состоят из двух строк. Первая строка содержит единственное целое число n $(1 \le n \le 10^5)$ — количество хороших целых чисел. Во второй строке задан список хороших целых чисел $a_1, a_2, ..., a_n$ через пробел, в порядке строгого возрастания $(1 \le a_i \le 10^5; a_i < a_{i+1})$.

Выходные данные

Выведите единственное целое число — длину самой длинной хорошей последовательности.

Примеры

входные данные	Скопировать
5 2 3 4 6 9	

выходные данные	Скопировать
4	

входные данные	Скопировать
9 1 2 3 5 6 7 8 9 10	

выходные данные	Скопировать
4	

Примечание

В первом примере следующие последовательности являются примерами хороших последовательностей: [2; 4; 6; 9], [2; 4; 6], [3; 9], [6]. Длина самой длинной хорошей последовательности — 4.

C. Аюб и утерянный массив

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт

У Аюба был массив целых чисел a длины n и у этого массива было два необычных свойства:

- Все целые числа в этом массиве были от l до r (включительно).
- Сумма всех элементов в массиве делилась на 3.

К сожалению, Аюб потерял свой массив, но он помнит размер массива n, а также числа l и r, так что он попросил вас посчитать количество способов восстановить массив по этим данным.

Так как ответ может быть очень большим, выведите его по модулю $10^9 + 7$ (то есть остаток при делении на $10^9 + 7$). В случае если не существует ни одного подходящего массива (Аюб что-то перепутал) — выведите 0.

Входные данные

Первая и единственная строка содержит три целых числа n, l и r ($1 \le n \le 2 \cdot 10^5, 1 \le l \le r \le 10^9$) — размер массива и диапазон чисел в нём.

Выходные данные

Выведите остаток при делении на $10^9 + 7$ количества способов восстановить массив.

Примеры

входные данные	Скопировать
2 1 3	
выходные данные	**Скопировать**
3	

входные данные	Скопировать
3 2 2	
выходные данные	**Скопировать**
1	

входные данные	Скопировать
9 9 99	
выходные данные	**Скопировать**
711426616	

Примечание

В первом примере возможны массивы $[1, 2], [2, 1], [3, 3]$.

Во втором примере единственный возможный массив — $[2, 2, 2]$.

A2. Нулевая сумма (сложная версия)

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Это сложная версия задачи. Разница между версиями заключается в том, что в этой версии массив содержит нули. Вы можете делать взломы только в том случае, если обе версии задачи решены.

Вам дан массив $[a_1, a_2, \ldots a_n]$, состоящий из чисел -1, 0 и 1. Требуется предъявить **разбиение** этого массива на несколько отрезков $[l_1, r_1], [l_2, r_2], \ldots, [l_k, r_k]$, обладающее следующим свойством:

- Обозначим за s_i знакопеременную сумму элементов в i-м отрезке, то есть $s_i = a_{l_i} - a_{l_i+1} + a_{l_i+2} - a_{l_i+3} + \ldots \pm a_{r_i}$. Например знакопеременная сумма на отрезке $[2, 4]$ в массиве $[1, 0, -1, 1, 1]$ равна $0 - (-1) + 1 = 2$.
- Сумма значений s_i по всем отрезкам из разбиения должна быть равна нулю.

Обратите внимание, каждое s_i **не** обязано равняться 0, условие стоит только на сумму s_i по всем отрезкам разбиения.

Набор отрезков $[l_1, r_1], [l_2, r_2], \ldots, [l_k, r_k]$ называется **разбиением** массива a длины n, если $1 = l_1 \le r_1, l_2 \le r_2, \ldots, l_k \le r_k = n$, причем для всех $i = 1, 2, \ldots k - 1$ выполняется $r_i + 1 = l_{i+1}$. Иными словами, каждый элемент массива должен принадлежать ровно одному отрезку.

Требуется предъявить разбиение массива, обладающее описанными свойствами, или сказать, что такого разбиения не существует.

Обратите внимание, минимизировать количество отрезков в разбиении **не** требуется.

Входные данные

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит одно целое число t ($1 \le t \le 10\,000$) — количество наборов входных данных. Далее следуют описания наборов:

Первая строка каждого описания содержит единственное целое число n ($1 \le n \le 200\,000$) — длину массива a.

Вторая строка каждого описания содержит n целых чисел $a_1, a_2, \ldots, a_n$ (a_i равно -1, 0 или 1) — элементы массива.

Гарантируется, что сумма n по всем наборам входных данных не превосходит $200\,000$.

Выходные данные

Для каждого набора входных данных выведите целое число k — количество отрезков в получившемся разбиении. Если требуемого разбиения не существует, выведите -1.

Далее, если разбиение существует, то в i-й из следующих k строк выведите два целых числа l_i и r_i — границы i-го отрезка. При этом должны выполняться условия:

- $l_i \le r_i$ для всех i от 1 до k.
- $l_{i+1} = r_i + 1$ для всех i от 1 до $(k - 1)$.
- $l_1 = 1, r_k = n$.

Если существует несколько корректных разбиений массива на отрезки, разрешается вывести любое из них.

Пример

входные данные Скопировать

```
5
4
0 0 0 0
7
-1 1 0 1 0 1 0
5
0 -1 1 0 1
3
1 0 1
1
1
```

выходные данные Скопировать

```
4
1 1
2 2
3 3
```

```
4 4
4
1 1
2 2
3 5
6 7
-1
2
1 1
2 3
-1
```

Примечание

В первом наборе входных данных массив можно просто разбить на 4 отрезка по одному элементу, равному 0. Тогда общая сумма будет равна $0 + 0 + 0 + 0 = 0$.

Во втором наборе входных данных массив разбивается на 4 отрезка. На первом из них знакопеременная сумма будет равна -1, на втором 1, на третьем $0 - 1 + 0 = -1$, на четвёртом $1 - 0 = 1$. Получаем общую сумму: $-1 + 1 - 1 + 1 = 0$.

В третьем наборе входных данных можно показать, что требуемого разбиения не существует.

B. Никита и строка

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Однажды Никита нашел строку, состоящую только из символов «a» и «b».

Никита считает, что строка красивая, если её можно разрезать на 3 строки (возможно, нулевой длины) так, что, не меняя порядок, 1-я и 3-я состоят только из букв «a», а 2-я только из букв «b».

Никита хочет сделать строку красивой, выкинув из нее некоторые символы (или не выкидывая их вовсе), но не меняя их порядок. Какой наибольшей длины строку он сможет получить?

Входные данные

В первой строке содержится непустая строка, длиной не более $5\,000$, состоящая только из строчных букв латинского алфавита «a» и «b».

Выходные данные

Выведете одно число — максимально возможную длину получившейся красивой строки.

Примеры

входные данные	Скопировать
abba	

выходные данные	Скопировать
4	

входные данные	Скопировать
bab	

выходные данные	Скопировать
2	

Примечание

В первом примере строка уже красивая.

Во втором примере нужно убрать одну из букв «b», чтобы строка стала красивой.

E. Гуманоид

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

На некоторой космической станции работают n космонавтов. Космонавт с номером i ($1 \le i \le n$) имеет *стойкость* a_i.

На эту космическую станцию пробрался злобный гуманоид. *Сила* этого гуманоида равна h. Также гуманоид взял с собой **две** *зелёных* сыворотки и **одну** *синюю* сыворотку.

За одну секунду гуманоид может сделать любое из трёх действий:

1. поглотить космонавта со *стойкостью* **строго меньше** *силы* гуманоида;
2. употребить *зелёную* сыворотку, если такая ещё осталась;
3. употребить *синюю* сыворотку, если такая ещё осталась.

При поглощении космонавта со *стойкостью* a_i, этот космонавт исчезает, а *сила* гуманоида увеличивается на $\lfloor \frac{a_i}{2} \rfloor$, то есть целую часть от $\frac{a_i}{2}$. Например, если гуманоид поглощает космонавта со *стойкостью* 4, его *сила* увеличивается на 2, а если гуманоид поглощает космонавта со *стойкостью* 7, его *сила* увеличивается на 3.

При употреблении *зелёной* сыворотки, эта сыворотка исчезает, а *сила* гуманоида увеличивается в 2 раза.

При употреблении *синей* сыворотки, эта сыворотка исчезает, а *сила* гуманоида увеличивается в 3 раза.

Гуманоиду интересно, какое максимальное количество космонавтов он сможет поглотить, если будет действовать оптимально.

Входные данные

В первой строке входных данных дано целое число t ($1 \le t \le 10^4$) — количество наборов входных данных.

В первой строке каждого набора входных данных даны целые числа n ($1 \le n \le 2 \cdot 10^5$) — количество космонавтов и h ($1 \le h \le 10^6$) — изначальная *сила* гуманоида.

Во второй строке каждого набора входных данных даны n целых чисел a_i ($1 \le a_i \le 10^8$) — *стойкости* космонавтов.

Гарантируется, что сумма n по всем наборам входных данных не превосходит $2 \cdot 10^5$.

Выходные данные

Для каждого набора входных данных в отдельной строке выведите максимальное количество космонавтов, которое сможет поглотить гуманоид.

Пример

входные данные Скопировать

```
8
4 1
2 1 8 9
3 3
6 2 60
4 5
5 1 100 5
3 2
38 6 3
1 1
12
4 6
12 12 36 100
4 1
2 1 1 15
3 5
15 1 13
```

выходные данные Скопировать

```
4
3
3
3
0
4
```

```
4
3
```

Примечание

В первом случае можно действовать так:

1. употребить *зелёную* сыворотку. $h = 1 \cdot 2 = 2$
2. поглотить космонавта 2. $h = 2 + \lfloor \frac{1}{2} \rfloor = 2$
3. употребить *зелёную* сыворотку. $h = 2 \cdot 2 = 4$
4. поглотить космонавта 1. $h = 4 + \lfloor \frac{2}{2} \rfloor = 5$
5. употребить *синюю* сыворотку. $h = 5 \cdot 3 = 15$
6. поглотить космонавта 3. $h = 15 + \lfloor \frac{8}{2} \rfloor = 19$
7. поглотить космонавта 4. $h = 19 + \lfloor \frac{9}{2} \rfloor = 23$

C. Упоротые четные числа

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 512 мегабайт

Алиса и Боб играют в игру на последовательности $a_1, a_2, \ldots, a_n$ длины n. Их ходы чередуются, и **Алиса ходит первой**.

В свой ход каждый игрок должен выбрать одно число и убрать его из последовательности. Игра заканчивается, когда в последовательности не остается чисел.

Алиса выигрывает, если сумма убранных ей чисел **четная**; в противном случае выигрывает Боб.

Ваша задача определить кто победит, если оба игрока играют оптимально.

Входные данные

Первая строка содержит целое число t ($1 \leq t \leq 100$) — Количество наборов входных данных. Затем следуют описания наборов входных данных.

В первой строке набора входных данных содержится целое число n ($1 \leq n \leq 100$), означающее длину последовательности.

Во второй строке набора входных данных содержится n целых чисел $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$), сама последовательность.

Выходные данные

Для каждого набор входных данных выведите «Alice» (без кавычек) если Алиса побеждает, и «Bob» (без кавычек) в противном случае.

Пример

входные данные	Скопировать

```
4
3
1 3 5
4
1 3 5 7
4
1 2 3 4
4
10 20 30 40
```

выходные данные	Скопировать

```
Alice
Alice
Bob
Alice
```

Примечание

В первом примере Алиса всегда выбирает два нечетных числа, Таким образом, сумма выбранных ей чисел всегда четна, а значит она побеждает.

В третьем примере у Боба есть выигрышная стратегия, заключающаяся в том, что он всегда выбирает число той же четности, что и Алиса в свой последний ход. Следовательно, Боб всегда выигрывает.

В четвертом примере Алиса всегда выбирает два четных числа, Таким образом, сумма выбранных ей чисел всегда четна, а значит она побеждает.

C. Случайные события

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Рон — счастливый обладатель перестановки a длины n.

Перестановкой длины n является массив, состоящий из n различных целых чисел от 1 до n в произвольном порядке. Например, $[2, 3, 1, 5, 4]$ — перестановка, но $[1, 2, 2]$ — не перестановка (2 встречается в массиве дважды) и $[1, 3, 4]$ тоже не перестановка ($n = 3$, но в массиве встречается 4).

Над перестановкой Рона ставится m экспериментов следующего вида: (r_i, p_i). Это обозначает, что элементы в диапазоне $[1, r_i]$ (другими словами, префикс длины r_i) будут отсортированы в возрастающем порядке с вероятностью p_i. Все эксперименты проводятся в том же порядке, в котором задаются во входных данных.

Для примера рассмотрим перестановку $[4, 2, 1, 5, 3]$ и эксперимент $(3, 0.6)$. После такого эксперимента с вероятностью 60% перестановка примет вид $[1, 2, 4, 5, 3]$, а с вероятностью 40% останется без изменений.

Вам требуется определить, с какой вероятностью перестановка станет полностью отсортированной по возрастанию после m экспериментов.

Входные данные

Каждый тест содержит один или несколько наборов входных данных. В первой строке записано количество наборов входных данных t ($1 \le t \le 100$).

Первая строка каждого набора входных данных содержит два целых числа n и m ($1 \le n, m \le 10^5$) — количество элементов в перестановке и количество экспериментов.

Вторая строка каждого набора входных данных содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — описание перестановки.

Каждая из следующих m входных данных содержат целое число r_i и вещественное число p_i ($1 \le r_i \le n, 0 \le p_i \le 1$) — длина префикса массива и вероятность, с которой он отсортируется. Все вероятности даны с точностью не более 6 знаков после запятой.

Гарантируется, что сумма n и сумма m не превосходят 10^5 ($\sum n, \sum m \le 10^5$).

Выходные данные

Для каждого набора входных данных выведите единственное число — вероятность, что после всех экспериментов перестановка окажется отсортированной. Ваш ответ будет считаться правильным, если его абсолютная или относительная ошибка не превосходит 10^{-6}.

Формально, пусть ваш ответ равен a, а ответ жюри равен b. Ваш ответ будет зачтен, если и только если $\frac{|a-b|}{\max(1, |b|)} \le 10^{-6}$.

Пример

входные данные Скопировать

```
4
4 3
4 3 2 1
1 0.3
3 1
4 0.6
5 3
4 2 1 3 5
```

```
3 0.8
4 0.6
5 0.3
6 5
1 3 2 4 5 6
4 0.9
5 0.3
2 0.4
6 0.7
3 0.5
4 2
1 2 3 4
2 0.5
4 0.1
```

выходные данные Скопировать

```
0.600000
0.720000
0.989500
1.000000
```

Примечание

Для первого набора входных данных можно показать, что только от того, выполнится ли сортировка с помощью правила $(4, 0.6)$, зависит, будет ли итоговая перестановка отсортированной.

B. Прогулка по Аллее

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт

На Главной Аллее в ЛКШ расположены n лавочек, пронумерованных целыми числами от 1 до n слева направо. Также на Аллее есть m торговых палаток, i-я ($1 \le i \le m$) из которых расположена около s_i-й лавочки.

Петя сейчас находится в начале Аллеи перед 1-й лавочкой и он хочет дойти до n-й лавочки. Расстояние между последовательными лавочками Петя проходит за 1 минуту. У него с собой есть рюкзак с бесконечным количеством вафель. Петя планирует есть вафли из рюкзака и покупать их в торговых палатках во время прогулки.

Петя ест вафли только находясь около лавочек, причём он съест вафлю, находясь около i-й ($1 \le i \le n$) лавочки тогда и только тогда, когда выполняется **хотя бы одно** из следующих условий:

- Около i-й лавочки есть торговая палатка. Тогда Петя купит вафлю в торговой палатке и сразу съест её.
- Петя ещё ни разу не ел вафлю. Тогда Петя возьмёт вафлю из рюкзака и сразу съест её.
- После того, как Петя съел последнюю вафлю, прошло хотя бы d минут. Иными словами, Петя не ел вафли около каждой из лавочек $i-1, i-2, \ldots, \max(i-d+1, 1)$. Тогда Петя возьмёт вафлю из рюкзака и сразу съест её.

Вы можете считать, что Петя не тратит время на то, чтобы съесть вафлю. Петя никогда не будет есть две или более вафли около одной лавочки.

Вы хотите **минимизировать** количество вафель, которые съест Петя в течении прогулки. Для этого вы попросите администрацию ЛКШ убрать **ровно одну** торговую палатку с Аллеи до того как Петя начнёт движение.

Определите минимальное возможное количество вафель, которые съест Петя в течении прогулки. Также найдите количество торговых палаток, таких что если убрать одну из них, Петя съест минимальное возможное число вафель.

Входные данные

В первой строке дано одно целое число t ($1 \le t \le 10^3$) — количество наборов входных данных. Далее следуют описания этих наборов.

В первой строке даны три целых числа n, m и d ($2 \le d \le n \le 10^9$, $2 \le m \le \min(10^5, n)$) — количество лавочек, количество торговых палаток и параметр d из условия, соответственно.

Во второй строке даны m целых чисел $s_1, s_2, \ldots, s_m$ ($1 \le s_i \le n$) — номера лавочек, около которых расположены торговые палатки. Гарантируется, что $s_i < s_{i+1}$ для всех $1 \le i \le m-1$.

Гарантируется, что сумма m по всем наборам входных данных не превосходит 10^5.

Выходные данные

Для каждого набора входных данных выведите два целых числа — минимальное количество вафель, которые съест Петя, если будет убрана ровно одна торговая палатка, и количество торговых палаток, таких что если убрать одну из них, Петя съест минимальное возможное число вафель.

Пример

входные данные Скопировать

```
8
6 2 2
2 5
8 3 2
3 5 8
10 4 9
2 8 9 10
30 5 8
6 8 15 24 29
30 5 8
6 8 12 20 27
8 8 3
1 2 3 4 5 6 7 8
2 2 2
1 2
1000000000 3 20000000
57008429 66778899 837653445
```

выходные данные Скопировать

```
3 1
4 1
4 4
6 4
5 2
7 7
1 1
51 1
```

Примечание

В первом наборе данных $n = 6$, $m = 2$, $d = 2$ и $s = [2, 5]$. При таком расположении торговых палаток Петя съест 4 вафли в течении прогулки (обратите внимание, что необходимо убрать ровно одну торговую палатку; этот случай разобран только для того, чтобы показать как Петя принимает решение о том, следует ли съесть вафлю у очередной лавочки):

- Около 1-й лавочки Петя съест вафлю из рюкзака, так как он ещё ни разу не ел вафлю.
- Около 2-й лавочки Петя съест вафлю из торговой палатки, так как около этой лавочки есть торговая палатка.
- Около 3-й лавочки Петя не будет есть вафлю, так как с момента как он съел вафлю прошла $1 < d$ минута.
- Около 4-й лавочки Петя съест вафлю, так как с момента как он съел вафлю прошло $2 \geq d$ минуты.
- Около 5-й лавочки Петя съест вафлю из торговой палатки, так как около этой лавочки есть торговая палатка.
- Около 6-й лавочки Петя не будет есть вафлю, так как с момента как он съел вафлю прошла $1 < d$ минута.

Если убрать 1-ю торговую палатку, то Петя съест 3 вафли (около лавочек 1, 3 и 5). Если же убрать 2-ю торговую палатку, то Петя съест 4 вафли (около лавочек 1, 2, 4 и 6).

Таким образом, минимальное количество вафель, которые съест Петя — 3; существует только одна палатка, убрав которую можно достигнуть такого количества вафель.

Во втором наборе входных данных

- если убрать 1-ю или 2-ю палатку, то Петя съест 5 вафель около лавочек 1, 3, 5, 7, 8;
- если убрать 3-ю палатку, то Петя съест 4 вафли около лавочек 1, 3, 5, 7.

В третьем наборе входных данных Петя съест 4 вафли вне зависимости от вашего выбора.

Обратите внимание, что Петя не заинтересован в минимизации количества вафель, которые он съест, поэтому он будет есть вафли строго в соответствии с правилом, описанным в условии.

C. Корова и сообщение

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Корова Бесси только что перехватила сообщение, которое Фермер Джон отправил Бургер Куин! Бесси уверена, что в нем скрыто секретное сообщение.

Сообщение представляет из себя строку s, состоящую из строчных букв латинского алфавита. Бесси считает, что строка t скрыта в строке s, если t является подпоследовательностью s, индексы которой формируют арифметическую прогрессию. Например, строка aab скрыта в строке aaabb, потому что она получена в индексах 1, 3 и 5, которые формируют арифметическую прогрессию с шагом 2. Бесси считает, что любая скрытая строка, которая имеет наибольшее количество вхождений и является скрытым сообщением. Два вхождения подпоследовательности S различны, если различаются их множества индексов. Помогите Бесси узнать количество вхождений скрытого сообщения!

Например, в строке aaabb, a скрыта 3 раза, b скрыта 2 раза, ab скрыта 6 раз, aa скрыта 3 раза, bb скрыта 1 раз, aab скрыта 2 раза, aaa скрыта 1 раз, abb скрыта 1 раз, aaab скрыта 1 раз, aabb скрыта 1 раз и aaabb скрыта 1 раз. Количество вхождений скрытого сообщения равно 6.

Входные данные

В первой строке задана строка s, состоящая из строчных букв латинского алфавита ($1 \leq |s| \leq 10^5$) — текст, который перехватила Бесси.

Выходные данные

Выведите единственное число — количество вхождений секретного сообщения.

Примеры

входные данные	Скопировать
aaabb	
выходные данные	**Скопировать**
6	

входные данные	Скопировать
usaco	
выходные данные	**Скопировать**
1	

входные данные	Скопировать
lol	
выходные данные	**Скопировать**
2	

Примечание

В первом примере скрыты следующие строки (с соответствующими множествами индексов):

- a встречается как (1), (2), (3);
- b встречается как (4), (5);
- ab встречается как $(1, 4)$, $(1, 5)$, $(2, 4)$, $(2, 5)$, $(3, 4)$, $(3, 5)$;
- aa встречается как $(1, 2)$, $(1, 3)$, $(2, 3)$;
- bb встречается как $(4, 5)$;
- aab встречается как $(1, 3, 5)$, $(2, 3, 4)$;
- aaa встречается как at $(1, 2, 3)$
- abb встречается как $(3, 4, 5)$
- aaab встречается как $(1, 2, 3, 4)$;
- aabb встречается как $(2, 3, 4, 5)$;
- aaabb встречается как $(1, 2, 3, 4, 5)$;

Заметим, что все множества индексов являются арифметическими прогрессиями.

E. Живая последовательность

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт

В Японии цифра 4 читается как смерть, поэтому Боб решил построить *живую последовательность*. *Живая последовательность* a содержит все натуральные числа, не содержащие цифры 4.
$a = [1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, \ldots]$.

Например, число 1235 входит в последовательность a, а числа 4321, 443 не входят в последовательность a.

Боб понял, что не умеет быстро искать конкретное число по позиции k в последовательности, поэтому просит вас о помощи.

Например, если Боб хочет узнать число на позиции $k = 4$ (индексация с 1), то вам нужно ответить $a_k = 5$.

Входные данные

В первой строке входных данных дано единственное целое число t ($1 \le t \le 10^4$) — количество наборов входных данных в тесте.

В единственной строке каждого набора входных данных задано одно целое число k ($1 \le k \le 10^{12}$) — позиция, интересующая Боба.

Выходные данные

Для каждого набора входных данных выведите в отдельной строке число a_k в индексации с 1.

Пример

входные данные	Скопировать
7 3 5 22 10 100 12345 827264634912	

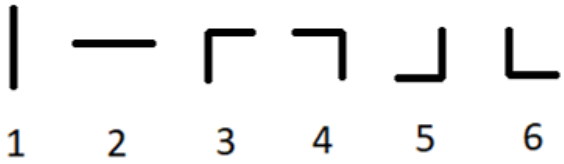
выходные данные	Скопировать
3 6 25 11 121 18937 2932285320890	

C. Трубы

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Вам задана система труб. Она состоит из двух рядов, каждый из которых состоит из n труб. Верхняя левая труба имеет координаты $(1, 1)$, а нижняя правая — $(2, n)$.

Всего существует шесть типов труб: два типа прямых труб и четыре типа изогнутых труб. Ниже приведены примеры всех шести типов:

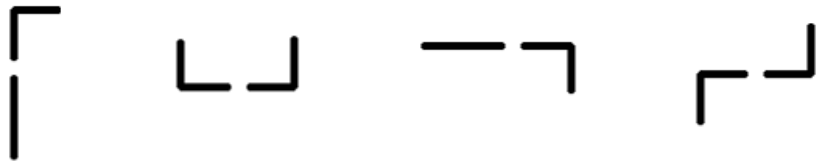


Типы труб

Вы можете поворачивать каждую из заданных труб на 90 градусов по часовой или против часовой стрелки **любое (возможно, нулевое) количество раз** (таким образом, типы 1 и 2 могут переходить друг в друга, а также $3, 4, 5, 6$ могут переходить друг в друга).

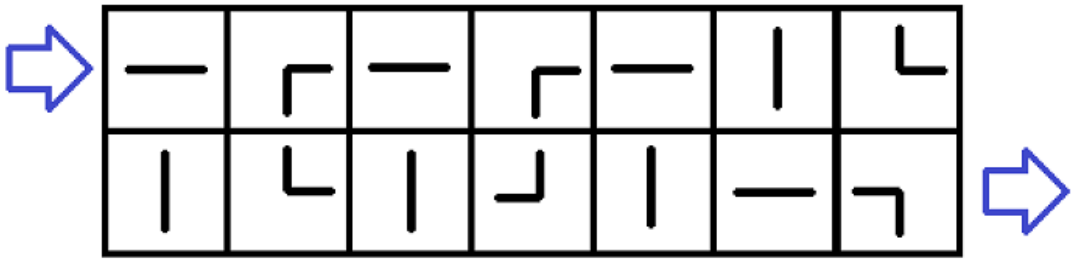
Вы хотите повернуть некоторые трубы таким образом, чтобы поток воды мог начать свое движение в $(1, 0)$ (слева от верхней левой трубы), пойти по трубе в $(1, 1)$, как-то потечь по **связным трубам** до трубы $(2, n)$ и потечь вправо в $(2, n + 1)$.

Трубы называются связными, если они являются соседними в системе и их концы соединены. Ниже несколько примеров связных труб:



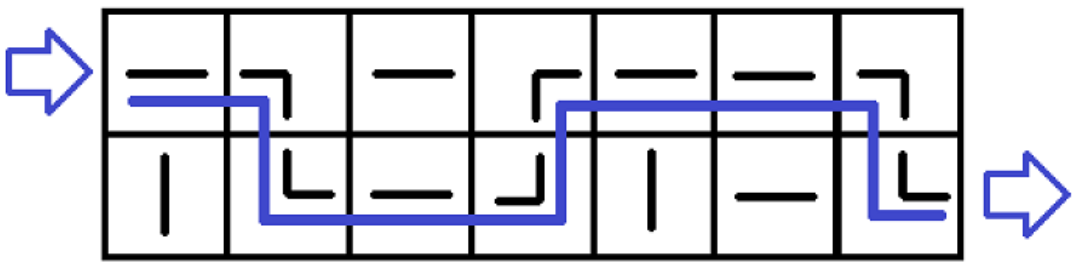
Примеры связных труб

Попробуем описать задачу при помощи примера:



Входные данные первого примера

И его решение ниже:



Решение первого примера

Как вы можете заметить, поток воды — это плохо нарисованная синяя линия. Чтобы достичь ответа, нам необходимо повернуть трубу на позиции $(1, 2)$ на 90 градусов по часовой стрелке, трубу на позиции $(2, 3)$ на 90 градусов, трубу на позиции $(1, 6)$ на 90 градусов, трубу на позиции $(1, 7)$ на 180 градусов и трубу на позиции $(2, 7)$ на 180 градусов. Тогда поток воды сможет достичь $(2, n + 1)$ из $(1, 0)$.

Вам необходимо ответить на q независимых запросов.

Входные данные

Первая строка входных данных содержит одно целое число q ($1 \le q \le 10^4$) — количество запросов. Затем следуют q запросов.

Каждый запрос состоит ровно из трех строк. Первая строка запроса содержит одно целое число n ($1 \le n \le 2 \cdot 10^5$) — количество труб в каждом ряду. Следующие две строки содержат описания первого и второго рядов соответственно. Описание каждого ряда состоит из n цифр от 1 до 6 без каких-либо пробелов между ними, каждая цифра соответствует типу трубы в соответствующей клетке. Прочтите условие задачи, чтобы понять, какие цифры соответствуют каким типам труб.

Гарантируется, что сумма n по всем запросам не пресоходит $2 \cdot 10^5$.

Выходные данные

Для i-го запроса выведите ответ на него — «YES» (без кавычек), если возможно повернуть некоторые трубы таким образом, чтобы поток воды смог достичь $(2, n + 1)$ из $(1, 0)$, и «NO» иначе.

Пример

входные данные Скопировать

```
6
7
2323216
1615124
1
3
4
2
13
24
2
12
34
3
536
345
2
46
54
```

выходные данные Скопировать

```
YES
YES
YES
NO
YES
NO
```

Примечание

Первый запрос из тестового примера описан в условии задачи.

C. Палиндромный базис

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Вам дано положительное целое число n. Назовём некоторое положительное целое число a без лидирующих нулей палиндромным, если оно остаётся таким же после переворота порядка его цифр. Найдите количество способов представить n как сумму положительных палиндромных чисел. Два способа считаются различными, если количества вхождений хотя бы одного палиндромного числа в них различаются. Например, $5 = 4 + 1$ и $5 = 3 + 1 + 1$ считаются различными, но $5 = 3 + 1 + 1$ и $5 = 1 + 3 + 1$ считаются одинаковыми.

Формально, вам нужно найти количество различных мультимножеств положительных палиндромных чисел, сумма которых равна n.

Поскольку ответ может быть очень большим, выведите его по модулю $10^9 + 7$.

Входные данные

Первая строка входных данных содержит единственное целое число t ($1 \le t \le 10^4$) — количество наборов входных данных.

Каждый набор входных данных содержит единственное целое число n ($1 \le n \le 4 \cdot 10^4$) — требуемую сумму палиндромных чисел.

Выходные данные

Для каждого набора входных данных выведите единственное целое число, обозначающее требуемый ответ по модулю $10^9 + 7$.

Пример

входные данные	Скопировать
2 5 12	

выходные данные	Скопировать
7 74	

Примечание

Для первого набора входных данных существует 7 способов представить 5 как сумму положительных палиндромных чисел:

- $5 = 1 + 1 + 1 + 1 + 1$
- $5 = 1 + 1 + 1 + 2$
- $5 = 1 + 2 + 2$
- $5 = 1 + 1 + 3$
- $5 = 2 + 3$
- $5 = 1 + 4$
- $5 = 5$

Во втором наборе входных данных существует всего 77 способов представить 12 как сумму положительных целых чисел, но среди них представления $12 = 2 + 10$, $12 = 1 + 1 + 10$ и $12 = 12$ не являются корректными представлениями 12 как сумму положительных палиндромных чисел, поскольку 10 и 12 не являются палиндромными числами. Таким образом, существует 74 способа представить 12 как сумму положительных палиндромных чисел.

C. Тенцинг и шарики

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт

Enjoy erasing Tenzing, identified as Accepted!

У Тенцинга есть n шариков, расположенных в один ряд. Цвет i-го шарика слева — a_i.

Тенцинг может проделать следующую операцию любое количество раз:

- выбрать i и j такие, что $1 \le i < j \le |a|$ и $a_i = a_j$,
- удалить из массива $a_i, a_{i+1}, \ldots, a_j$ (и уменьшить индексы всех элементов справа от a_j на $j - i + 1$).

Тенцинг хочет узнать максимальное количество шариков, которое он может удалить.

Входные данные

Каждый тест содержит несколько наборов входных данных. Первая строка входных данных содержит одно целое число t ($1 \le t \le 10^3$) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка содержит одно целое число n ($1 \le n \le 2 \cdot 10^5$) — количество шариков.

Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — цвета шариков.

Гарантируется, что сумма n по всем наборам входных данных не превосходит $2 \cdot 10^5$.

Выходные данные

Для каждого набора входных данных выведите максимальное количество шариков, которые может удалить Тенцинг.

Пример

входные данные	Скопировать
2 5 1 2 2 3 3 4 1 2 1 2	

выходные данные	Скопировать
4 3	

Примечание

В первом примере Тенцинг выберет $i = 2$ и $j = 3$ в первой операции так, что $a = [1, 3, 3]$. Затем Тенцинг снова выберет $i = 2$ и $j = 3$ во второй операции так, что $a = [1]$. Таким образом, Тенцинг может удалить в общей сложности 4 шарика.

Во втором примере Тенцинг выберет $i = 1$ и $j = 3$ в первой и единственной операции так, что $a = [2]$. Таким образом, Тенцинг может удалить 3 шарика.

D. Backspace

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Вам заданы две строки s и t, каждая из которых состоит из строчных букв латинского алфавита. Вы собираетесь посимвольно напечатать строку s, начиная с первого символа и заканчивая последним.

Когда вы собираетесь напечатать какой-то символ, вместо того, чтобы нажать на кнопку, печатающую этот символ, вы можете нажать кнопку «Backspace». Нажатие на эту кнопку удаляет последний напечатанный символ, который еще не был удален (или ничего не делает, если все напечатанные символы уже удалены или вы еще не напечатали ни одного символа). Например, если строка s — «abcbd», и вы нажимаете на кнопку Backspace вместо печати первого и четвертого символа, в результате получится строка «bd» (первое нажатие Backspace не удалит ни одного символа, а второе нажатие этой кнопки удалит символ «c»). Другой пример: если s равна «abcaa», и вы нажимаете Backspace вместо двух последних букв, получается «a».

Вы должны определить, можно ли получить строку t, если вы попробуете набрать строку s, нажимая Backspace вместо нажатия кнопок, соответствующих некоторым (возможно, ни одному) буквам строки s.

Входные данные

В первой строке задано одно целое число q ($1 \le q \le 10^5$) — количество наборов входных данных.

Первая строка каждого набора входных данных содержит строку s ($1 \le |s| \le 10^5$). Каждый символ строки s — строчная буква латинского алфавита.

Вторая строка каждого набора входных данных содержит строку t ($1 \le |t| \le 10^5$). Каждый символ строки t — строчная буква латинского алфавита.

Гарантируется, что суммарное по всем наборам входных данных количество символов во всех строках не превосходит $2 \cdot 10^5$.

Выходные данные

Для каждого набора выходных данных выведите «YES», если можно получить строку t, набирая строку s и заменяя некоторые символы нажатиями клавиши Backspace, или «NO» в противном случае.

Каждую букву можно выводить в любом регистре (YES, yes, Yes будут распознаны как положительный ответ, NO, no и nO будут распознаны как отрицательный ответ).

Пример

входные данные	Скопировать
4 ababa ba ababa bb aaa aaaa aababa ababa	

выходные данные	Скопировать
YES NO NO YES	

Примечание

Рассмотрим пример из условия.

Чтобы получить «ba» из «ababa», можно нажать Backspace вместо первого и четвертого символа.

Нет способа получить «bb» при попытке напечатать «ababa».

Нет способа получить «aaaa» при попытке напечатать «aaa».

Чтобы получить «ababa» при попытке напечатать «aababa», можно нажать Backspace вместо печати первого символа, а затем напечатать все оставшиеся символы.

C. Diluc и Kaeya

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Магнат винодельческой империи в Мондштадте, непревзойденный во всех отношениях. Мыслитель из рыцарей Фавониуса с экзотической внешностью}.

На этот раз братья имеют дело со странным куском дерева, помеченного их именами. Этот кусок дерева можно представить в виде строки из n символов. Каждый символ — это либо `'D'`, либо `'K'`. Вы хотите сделать некоторое количество разрезов (возможно, 0) в этой строке, разделив ее на несколько последовательных частей, каждая из которых имеет длину не менее 1. Оба брата ведут себя достойно, поэтому они хотят разделить дерево как можно более равномерно. Они хотят знать, на какое максимальное число кусков можно разделить дерево, чтобы отношение числа символов `'D'` и числа символов `'K'` в каждом куске было одинаковым.

Kaeya, любознательный мыслитель, хотел бы знать решение сразу для нескольких сценариев. Он хочет узнать ответ для каждого **префикса** данной строки. Помогите ему решить эту задачу!

Для строки мы определяем отношение как $a : b$, где `'D'` встречается в ней a раз, а `'K'` встречается b раз. Обратите внимание, что a или b могут быть равны 0, но не оба. Отношения $a : b$ и $c : d$ считаются равными тогда и только тогда, когда $a \cdot d = b \cdot c$.

Например, для строки `'DDD'` отношение будет $3 : 0$, для `'DKD'` — $2 : 1$, для `'DKK'` — $1 : 2$, а для `'KKKKDD'` — $2 : 4$. Обратите внимание, что отношения двух последних строк равны между собой, но не равны отношениям первых двух строк.

Входные данные

Каждый тест содержит несколько наборов входных данных. Первая строка содержит количество наборов входных данных t ($1 \le t \le 1000$). Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит целое число n ($1 \le n \le 5 \cdot 10^5$) — длина дерева.

Вторая строка каждого набора входных данных содержит строку s длиной n. Каждый символ s будет либо `'D'`, либо `'K'`.

Гарантируется, что сумма n по всем наборам входных данных не превышает $5 \cdot 10^5$.

Выходные данные

Для каждого набора входных данных выведите n целых чисел через пробел. i-е из этих чисел должно быть равно ответу для префикса $s_1, s_2, \ldots, s_i$.

Пример

входные данные Скопировать

```
5
3
DDK
6
DDDDDD
4
DKDK
1
D
9
DKDKDDDDK
```

выходные данные Скопировать

```
1 2 1
1 2 3 4 5 6
1 1 1 2
1
1 1 1 2 1 2 1 1 3
```

Примечание

Для первого набора входных данных нет способа разбить `'D'` или `'DDK'` на более чем один блок с равным отношением количеств `'D'` и `'K'`, в то время как `'DD'` можно разбить на `'D'` и `'D'`.

Для второго набора входных данных, каждый префикс длины i вы можете разделить на i блоков `'D'`.

C. Поедание конфет

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт

Tsumugi купила n вкусных конфет в Light Music Club. Они пронумерованы целыми числами от 1 до n, у i-й конфеты концентрация сахара описана целым числом a_i.

Yui любит конфеты, но она может есть не более m конфет каждый день, из соображений здоровья.

Дни нумеруются в 1-индексации (пронумерованы $1, 2, 3, \ldots$). Съесть конфету i в d-й день будет стоить $(d \cdot a_i)$ сахарного штрафа, так как со временем конфеты становятся более сладкими. Каждая конфета может быть съедена не более одного раза.

Итоговый сахарный штраф равен **сумме** сахарных штрафов всех съеденных конфет.

Предположим, что Yui выбирает ровно k конфет, и ест их в любом порядке, в каком захочет. Какой **минимальный** сахарный штраф она может получить?

Так как Yui нерешительная девушка, она просит ответить вас на этот вопрос для всех k от 1 до n.

Входные данные

В первой строке записаны два целых числа n и m ($1 \leq m \leq n \leq 200\,000$).

Во второй строке записаны n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 200\,000$).

Выходные данные

Вы должны вывести n целых чисел $x_1, x_2, \ldots, x_n$ в отдельной строке, разделяя пробелами, где x_k это минимальный сахарный штраф который Yui может получить, съев ровно k конфет.

Примеры

входные данные	Скопировать
9 2 6 19 3 4 4 2 6 7 8	
выходные данные	**Скопировать**
2 5 11 18 30 43 62 83 121	

входные данные	Скопировать
1 1 7	
выходные данные	**Скопировать**
7	

Примечание

Проанализируем ответ для $k = 5$ первого примера. Вот **один** из возможных способов съесть 5 конфет, чтобы минимизировать суммарный сахарный штраф:

- День 1: конфеты 1 и 4
- День 2: конфеты 5 и 3
- День 3 : конфета 6

Итоговый штраф равен $1 \cdot a_1 + 1 \cdot a_4 + 2 \cdot a_5 + 2 \cdot a_3 + 3 \cdot a_6 = 6 + 4 + 8 + 6 + 6 = 30$. Мы можем доказать, что это минимальный возможный сахарный штраф, который Yui может получить если она съест 5 конфет, таким образом $x_5 = 30$.

E. Блоковая последовательность

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

Дана последовательность целых чисел a длины n.

Назовём последовательность *красивой*, если она имеет вид ряда блоков, каждый из которых начинается со своей длины, то есть сначала идёт длина блока, а потом его элементы. Например, последовательности $[3, 3, 4, 5, 2, 6, 1]$ и $[1, 8, 4, 5, 2, 6, 1]$ являются *красивыми* (разные блоки покрашены разными цветами), а $[1]$, $[1, 4, 3]$, $[3, 2, 1]$ — нет.

За одну операцию вы можете удалить из последовательности любой элемент. Какое минимальное количество операций нужно сделать, чтобы данная последовательность стала *красивой*.

Входные данные

В первой строке входных данных содержится одно целое число t ($1 \le t \le 10^4$) — количество наборов входных данных в тесте. Далее следуют описания наборов.

Первая строка каждого набора содержит одно целое число n ($1 \le n \le 2 \cdot 10^5$) — длину последовательности a.

Вторая строка каждого набора содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — элементы последовательности a.

Гарантируется, что сумма значений n по всем наборам входных данных не превосходит $2 \cdot 10^5$

Выходные данные

Для каждого набора входных данных выведите одно число — минимальное количество удалений, которые нужно совершить чтобы последовательность a стала *красивой*.

Пример

входные данные Скопировать

```
7
7
3 3 4 5 2 6 1
4
5 6 3 2
6
3 4 1 6 7 7
3
1 4 3
5
1 2 3 4 5
5
1 2 3 1 2
5
4 5 5 1 5
```

выходные данные Скопировать

```
0
4
1
1
2
1
0
```

Примечание

В первом наборе входных данных примера данная последовательность уже является *красивой*, как и показано в условии.

Во втором наборе входных данных примера можно сделать последовательность красивой только удалив из неё все элементы.

В пятом наборе входных данных примера можно сделать последовательность красивой, удалив первый и последний элемент. Тогда последовательность станет $[2, 3, 4]$.