Технокубок 2020 - Отборочный Раунд 1

A. КУС

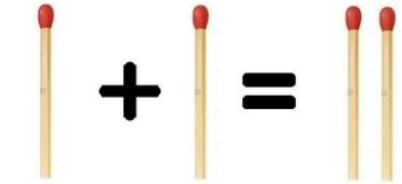
ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

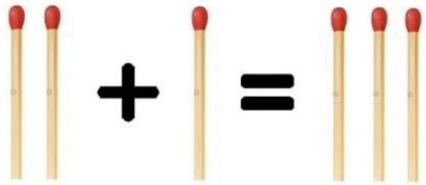
Назовем корректным уравнением из спичек (обозначим его как КУС) уравнение вида a+b=c, где все числа a, b и c целые и больше нуля.

Например, уравнения 2+2=4 (||+||=||||) и 1+2=3 (|+||=|||) являются КУС, а уравнения 1+2=4 (|+||=||||), 2+2=3 (||+||=|||) и 0+1=1 (+|=|) — нет.

У вас есть n спичек. Вы хотите составить КУС используя все ваши спички. К сожалению, возможно, что у вас не получится составить КУС, используя все ваши спички. Но вы можете докупить несколько спичек и затем собрать КУС!

Например, если n=2, вы можете купить две спички и составить |+|=||, и если n=5 вы можете купить одну и составить ||+|=|||.





Посчитайте минимальное количество спичек, которое вам нужно купить для составления КУС.

Обратите внимание, что вам нужно ответить на q независимых запросов.

Входные данные

Первая строка содержит одно число q ($1 \le q \le 100$) — количество запросов.

Единственная строка каждого запроса содержит одно число n ($2 \le n \le 10^9$) — количество спичек.

Выходные данные

На каждый запрос выведите одно число — минимальное количество спичек, которое вам нужно купить для составления КУС.

Пример

входные данные
4 2 5 8 11
выходные данные
2 1 0 1

Примечание

Первый и второй запросы объяснены в условии.

В третьем запросе вы можете составить 1+3=4 (|+|||=||||), не докупая спичек.

В четвертом запросе вам нужно купить одну спичку и составить 2+4=6 (||+||||=||||||).

B. Уравнивание строк

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам заданы две строки s и t одинаковой длины, состоящие из строчных букв латинского алфавита. Вы можете выполнять любое (возможно нулевое) количество операций над этими строками.

В течении каждой операции вы выбираете два **соседних** символа в **любой** строке и присваиваете значение первого символа значению второго или наоборот.

Например, если s равна «acbc» вы можете получить следующие строки за **одну** операцию:

- «aabc» (если выполните присвоение $s_2=s_1$);
- «ccbc» (если выполните присвоение $s_1=s_2$);
- «accc» (если выполните присвоение $s_3=s_2$ или $s_3=s_4$);
- «abbc» (если выполните присвоение $s_2=s_3$);
- «acbb» (если выполните присвоение $s_4=s_3$);

Обратите внимание, что такие же операции вы можете применять и к строке t.

Вам нужна выполнить несколько (возможно ноль) таких операций, чтобы строка s стала равна t. Определите, возможно ли это.

Обратите внимание, что вам нужно ответить на q независимых запросов.

Входные данные

Первая строка содержит целое число q ($1 \le q \le 100$) — количество запросов. Каждый запрос состоит из двух последовательных строк.

Первая строка каждого запроса содержит строку s ($1 \le |s| \le 100$), состоящую из строчных букв латинского алфавита.

Вторая строка каждого запроса содержит строку t ($1 \le |t| \le 100$, $|t|=|s|$), состоящую из строчных букв латинского алфавита.

Выходные данные

На каждый запрос выведите «YES», если возможно сделать строку s равной t, и «NO» в обратном случае.

Вы можете выводить ответ в любом регистре (например, строки «yEs», «yes», «Yes» и «YES» будут учтены как положительный ответ).

Пример

входные данные
3 xabb aabx technocup technocup a z
выходные данные
YES YES NO

Примечание

В первом запросе вы можете применить две операции $s_1=s_2$ (после неё s превратится в «aabb») и $t_4=t_3$ (после нее t превратится в «aabb»).

Во втором запросе строки равны изначально, а значит ответ «YES».

В третьем запросе вы не можете сделать строки s и t равными. Таким образом, ответ «NO».

C. Спаси природу

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Реальность жестока, поэтому, хоть в душе вы и экоактивист, но в жизни — простой кассир в кинотеатре. Но не стоить о пускать руки, ведь вы еще можете помочь природе!

По работе вам необходимо продать n билетов. Цена i-го билета равна p_i. Как продавец, вы можете выбрать порядок (то есть перестановку), в котором будут продаваться билеты. Вы узнали, что ваш кинотеатр участвует в двух экологических программах, применяя их **к порядку, который выбрали вы**:

- $x\%$ цены каждого a-го проданного билета (a-й, $2a$-й, $3a$-й и так далее билеты) *в вашем порядке* будет направлено на разработку и продвижение возобновляемых источников энергии.
- $y\%$ цены каждого b-го проданного билета (b-й, $2b$-й, $3b$-й и так далее билеты) *в вашем порядке* будет направлено на борьбу с загрязнением окружающей среды.

Если билет попал в обе программы одновременно, то в сумме $(x+y)\%$ будет направлено на сохранение природы. Также, известно, что все цены билетов кратны 100, а потому нет проблем с округлениями.

Например, если вам надо продать билеты с ценами [400,100,300,200], а кинотеатр отдает 10% от каждого 2-го проданного билета и 20% от каждого 3-го проданного билета, то, продав их в порядке [100,200,300,400] вы отправите на благое дело $100\cdot0+200\cdot0.1+300\cdot0.2+400\cdot0.1= =120$. Однако, выбрав порядок [100,300,400,200], можно набрать $100\cdot0+300\cdot0.1+400\cdot\cdot0.2 + 200\cdot0.1=130$.

Природа не может ждать, а потому вы решили изменить порядок продажи билетов таким образом, что **суммарный** взнос в обе программы достигнет значения хотя бы k за **минимальное** коли-чество проданных билетов. Либо скажите, что это невозможно. Другими словами, найдите минимальное количество билетов, которые нужно продать, чтобы заработать как минимум k.

Входные данные

В первой строке задано единственное целое число q ($1 \le q \le 100$) — количество независимых запросов. Каждый запрос состоит из 5 строк.

В первой строке каждого запроса содержится единственное целое число n ($1 \le n \le 2\cdot10^5$) — количество билетов.

Во второй строке заданы n целых чисел $p_1,p_2,\ldots,p_n$ ($100 \le p_i \le 10^9$, $p_i \bmod 100=0$) — соответствующие цены билетов.

В третьей строке заданы два целых числа x и a ($1 \le x \le 100$, $x+y \le 100$, $1 \le a \le n$) — параметры первой программы.

В четвертой строке заданы два целых числа y и b ($1 \le y \le 100$, $x+y \le 100$, $1 \le b \le n$) — параметры второй программы.

В пятой строке задано единственное целое число k ($1 \le k \le 10^{14}$) — суммарный необходимый взнос.

Гарантируется, что суммарное количество билетов в одном тесте не превосходит $2 \cdot 10^5$.

Выходные данные

Выведите q чисел — по одному на запрос.

Для каждого запроса выведите минимальное количество билетов, которое вам предстоит продать, чтобы суммарный взнос на экопрограммы достиг хотя бы k, при условии, что вы можете продавать билеты в произвольном порядке.

Если невозможно достигнуть необходимого взноса, даже продав все билеты, выведите -1.

Пример

входные данные
4 1 100 50 1 49 1 100 8 100 200 100 200 100 200 100 100 10 2 15 3 107 3 1000000000 1000000000 1000000000 50 1 50 1 3000000000 5 200 100 100 100 100 69 5 31 2 90

выходные данные
-1 6 3 4

Примечание

В первом запросе общий взнос равен $50+49=99<100$, поэтому собрать необходимую сумму невозможно.

Во втором запросе вы можете выбрать порядок следующим образом: [100,100,200,200,100,200,100,100] и общий взнос от первых 6 проданных билетов будет равен $100 \cdot 0+100 \cdot 0.1+200 \cdot 0.15+200 \cdot 0.1+100 \cdot 0+200 \cdot 0.25=10+30+20+50=110$.

В третьем запросе вся стоимость билета идет на защиту экологии.

В четвертом запросе вы можете выбрать порядок как [100,200,100,100,100] и суммарный взнос за первые 4 билета будет равен $100 \cdot 0+200 \cdot 0.31+100 \cdot 0+100 \cdot 0.31=62+31=93$.

D. Сортировка последовательности

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам задана последовательность целых чисел $a_1, a_2, \ldots, a_n$.

Вы можете применять следующую операцию к этой последовательности: выбрать число x и переместить **все** элементы

равные x либо в начало, либо в конец массива a. Обратите внимание, что за **одну** операцию вы перемещаете все элементы в **одном** направлении.

Например, если $a=[2,1,3,1,1,3,2]$, вы можете получить следующие последовательности за одну операцию (для удобства, обозначим элементы равные X как X-элементы):

- $[1,1,1,2,3,3,2]$, если вы переместите все 1-элементы в начало;
- $[2,3,3,2,1,1,1]$, если вы переместите все 1-элементы в конец;
- $[2,2,1,3,1,1,3]$, если вы переместите все 2-элементы в начало;
- $[1,3,1,1,3,2,2]$, если вы переместите все 2-элементы в конец;
- $[3,3,2,1,1,1,2]$, если вы переместите все 3-элементы в начало;
- $[2,1,1,1,2,3,3]$, если вы переместите все 3-элементы в конец;

Вам нужно посчитать минимальное количество операций, после применения которых последовательность a станет отсортирована в порядке неубывания. Последовательность отсортирована в порядке неубывания, если для любого индекса i от 2 до n, выполняется условие $a_{i-1} \le a_i$.

Обратите внимание, что вам нужно ответить на q независимых запросов.

Входные данные

Первая строка содержит целое число q ($1 \le q \le 3 \cdot 10^5$) — количество запросов. Каждый запрос состоит из двух последовательных строк.

Первая строка каждого запроса содержит целое число n ($1 \le n \le 3 \cdot 10^5$) — количество элементов массива.

Вторая строка каждого запроса содержит n целых чисел $a_1,a_2,\ldots,a_n$ ($1 \le a_i \le n$) — элементы массива.

Гарантируется, что сумма n по всем запросам не превосходит $3 \cdot 10^5$.

Выходные данные

На каждый запрос выведите одно число — минимальное количество операций для сортировки последовательности a в порядке неубывания.

Пример

входные данные
3 7 3 1 6 6 3 1 1 8 1 1 4 4 4 7 8 8 7 4 2 5 2 6 2 7
выходные данные
2 0 1

Примечание

В первом запросе вы можете переместить все 1-элементы в начало (после этого последовательность превратится в $[1,1,1,3,6,6,3]$) и затем переместить все 6-элементы в конец.

Во втором запросе последовательность отсортирована изначально, а значит ответ равен нулю.

В третьем запросе вам нужно переместить все 2-элементы в начало.

E. Раскрась дерево

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам задано взвешенное дерево, состоящее из n вершин. Напомним, что деревом называется связный граф без циклов. Вершины u_i и v_i соединены ребром веса w_i.

Определим k-раскраску дерева как присвоение **каждой** вершине ровно k цветов, таким образом, что каждый цвет используется не более двух раз. Считайте, что различных доступных цветов бесконечно много.

Назовем ребро *насыщенным* в k-раскраске, если его концы имеют хотя бы один общий цвет (т.е. найдется цвет, который присвоен обоим концам ребра).

Так же определим *счет* k-раскраски как сумму весов *насыщенных* ребер.

Вам необходимо найти максимально возможный *счет* k-раскраски заданного дерева.

Вам нужно ответить на q независимых запросов.

Входные данные

Первая строка содержит целое число q ($1 \le q \le 5 \cdot 10^5$) — количество запросов.

Первая строка каждого запроса содержит два целых числа n и k ($1 \le n,k \le 5 \cdot 10^5$) — количество вершин в дереве и количество цветов, которое нужно присвоить каждой вершине.

Каждая из следующих $n-1$ строк описывает ребра дерева. Ребро i обозначается тремя целыми числами u_i, v_i и w_i — номерами вершин, которые оно соединяет, и весом ребра ($1 \le u_i,v_i \le n$, $u_i \ne v_i$ $1 \le w_i \le 10^5$). Гарантируется, что заданный набор ребер образует дерево.

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

Выходные данные

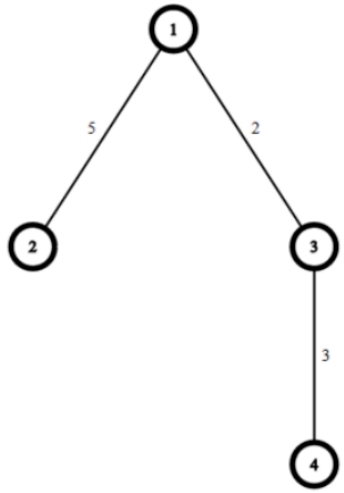
На каждый запрос выведите одно целое число — максимально возможный *счет* *k*-раскраски заданного дерева.

Пример

входные данные
2 4 1 1 2 5 3 1 2 3 4 3 7 2 1 2 5 1 3 4 1 4 2 2 5 1 2 6 2 4 7 3
выходные данные
8 14

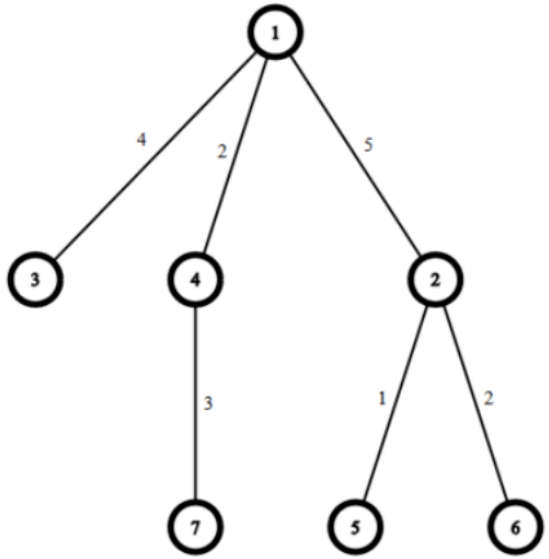
Примечание

Дерево, соответствующее первому запросу в тестовом примере:



Одна из возможных ***k***-раскрасок в первом запросе: *(1),(1),(2),(2)*, тогда ребра номер *1* и *3* являются насыщенными и сумма их весов равна ***8***.

Дерево, соответствующее второму запросу в тестовом примере:



Одна из возможных k-раскрасок во втором запросе: (1,2),(1,3),(2,4), (5,6),(7,8),(3,4),(5,6), тогда ребра номер 1, 2, 5 и 6 являются насыщенными и сумма их весов равна 14.

F. Стек-уничтожимые массивы

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Давайте рассмотрим следующий процесс: изначально у вас есть пустой стек и массив s длины l. Вы пытаетесь добавлять элементы массива в стек в порядке $s_1, s_2, s_3, \ldots s_l$. Причем если стек пустой или элемент на вершине стека не равен текущему, то вы просто добавляете текущий элемент на вершину стека. Иначе вы не добавляете элемент и более того, удаляете верхний элемент стека.

Если после данного процесса стек окажется пустым, то массив s считается *стек-уничтожимым*.

Примеры стек-уничтожимых массивов:

- [1,1];
- [2,1,1,2];
- [1,1,2,2];
- [1,3,3,1,2,2];
- [3,1,3,3,1,3];
- [3,3,3,3,3,3];
- [5,1,2,2,1,4,4,5];

Давайте рассмотрим изменения стека более подробно при s=[5,1,2,2,1,4,4,5] (элемент на вершине текста выделен жирным шрифтом).

1. после добавления s_1=5 стек будет равен [5];
2. после добавления s_2=1 стек будет равен [5,1];
3. после добавления s_3=2 стек будет равен [5,1,2];
4. после добавления s_4=2 стек будет равен [5,1];
5. после добавления s_5=1 стек будет равен [5];
6. после добавления s_6=4 стек будет равен [5,4];
7. после добавления s_7=4 стек будет равен [5];
8. после добавления s_8=5 стек станет пустым.

Вам задан массив $a_1, a_2, \ldots, a_n$. Вам нужно посчитать количество подотрезков этого массива, которые являются стек-уничтожимыми.

Обратите внимание, что вам нужно ответить на q независимых запросов.

Входные данные

Первая строка содержит целое число q ($1 \le q \le 3 \cdot 10^5$) — количество запросов.

Первая строка каждого запроса содержит единственное целое число n ($1 \le n \le 3 \cdot 10^5$) — длину массива a.

Вторая строка каждого запроса содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — элементы массива.

Гарантируется, что сумма n по всем запросам не превосходит $3 \cdot 10^5$.

Выходные данные

На каждый запрос выведите одно число — количество стек-уничтожимых подотрезков массива a.

Пример

входные данные
3 5 2 1 1 2 2 6 1 2 1 1 3 2 9 3 1 2 2 1 6 6 3 3
выходные данные
4 1 8

Примечание

В первом запросе есть четыре стек-уничтожимых подотрезка $a_{1\ldots4}=[2,1,1,2], a_{2\ldots3}=[1,1], a_{2\ldots5}=[1,1,2,2], a_{4\ldots5}=[2,2]$.

Во втором запросе только один стек-уничтожимый подотрезок — $a_{3\ldots4}$

В третьем запросе есть восемь стек-уничтожимых подотрезков $a_{1\ldots8}, a_{2\ldots5}, a_{2\ldots7}, a_{2\ldots9}, a_{3\ldots4}, a_{6\ldots7}, a_{6\ldots9}, a_{8\ldots9}$.

G. Деревянный плот

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Представьте, что вы застряли на необитаемом острове. Единственный способ спастись — построить деревянный плот и выйти на нем в море. К счастью, вы смогли смастерить пилу, а на острове оказался лес. Более того, вы уже спилили несколько деревьев и подготовили их, получив в конечном счёте n бревен, где длина i-го бревна равна a_i.

Деревянный плот, который вы хотите построить, имеет следующий вид: 2 бревна длины x и x бревен длины y. Площадь такого плота будет равна $x \cdot y$. При этом оба значения x **и** y **должны быть целыми**, так как это единственный метод измерения, который вы освоили, будучи на необитаемом острове. А также, оба значения x **и** y **должны быть не менее** 2, так как плот шириной в одно бревно неустойчив.

Вы можете разрезать бревна на части, но не можете соединять два бревна в одно. Плот какой максимальной площади вы сможете построить?

Входные данные

В первой строке задано единственное целое число n ($1 \le n \le 5 \cdot 10^5$) — количество бревен в вашем распоряжении.

Во второй строке заданы n целых чисел $a_1, a_2, \ldots, a_n$ ($2 \le a_i \le 5 \cdot 10^5$) соответствующие длины бревен.

Гарантируется, что бы всегда сможете построить плот размера хотя бы 2×2.

Выходные данные

Выведите единственное число — максимальную площадь плота, который вы сможете построить.

Примеры

входные данные
1 9

выходные данные
4

входные данные
9 9 10 9 18 9 9 9 28

выходные данные
90

Примечание

В первом примере, вы можете разрезать бревно длины 9 на 5 частей: 2+2+2+2+1. Теперь мы сможете собрать плот 2×2, используя 2 бревна длины x=2 и x=2 бревна длины y=2.

Во втором примере, вы можете разрезать a4=18 на две части 9+9 и a_8=28 на три части 10+9+9. Теперь вы можете собрать плот 10×9, используя 2 бревна длины 10 и 10 бревен длины 9.