A. В поисках Саске

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Наруто пробрался в логово Орочимару и пытается найти Саске. В логове Орочимару есть T комнат. Каждая дверь в комнату характеризуется количеством печатей n на ней и целочисленными силами печатей a_1, a_2, ... a_n. Силы печатей a_i **не равны нулю** и не превосходят 100 по модулю, а n всегда **четно**.

Для того чтобы открыть комнату, Наруто необходимо подобрать такие n печатей с целочисленными силами b_1, b_2, ..., b_n, чтобы $a_1 \cdot b_1 + a_2 \cdot b_2 + ... + a_n \cdot b_n = 0$. Силы печатей Наруто b_i также **должны быть ненулевыми** (как и данные a_i) и не превосходить 100 по модулю. Необходимо подобрать печати для всех комнат в логове.

Входные данные

В первой строке задано число T $(1 \le T \le 1000)$ — количество комнат в логове Орочимару. В следующих строках содержатся описания дверей.
В первой строке описания каждой двери содержится четное число n $(2 \le n \le 100)$ — количество печатей на двери.
В следующей строке содержится последовательность целых ненулевых чисел a_1, a_2, ..., a_n через пробел ($|a_i| \le 100$, $a_i \ne 0$) — силы печатей.

Выходные данные

Для каждой двери на отдельной строке необходимо вывести последовательность ненулевых целых чисел b_1, b_2, ..., b_n ($|b_i| \le 100$, $b_i \ne 0$) через пробел — силы печатей, необходимые для открытия этой двери. Если есть несколько искомых последовательностей, выведите любую из них. Можно доказать, что ответ всегда существует.

Пример

входные данные

```
2
2
1 100
4
1 2 3 6
```

выходные данные

```
-100 1
1 1 1 -1
```

Примечание

Для первой двери Наруто может использовать силы печатей $[-100,1]$. Необходимое условие будет выполняться: $1\cdot(-100)+100\cdot1=0$.
Для второй двери Наруто может использовать силы печатей $[1,1,1,-1]$. Необходимое условие будет выполняться: $1\cdot1+2\cdot1+3\cdot1+6\cdot(-1)=0$.

Разбор задачи A

1413A - В поисках Саске

Заметим, что подходит следующий массив: $-a_2, a_1, -a_4, a_3, ..., -a_n, a_n-1$.

B. Новая техника

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 512 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Все техники в мире ниндзя состоят из печатей. Сейчас Наруто учит новую технику, которая состоит из $n\cdot m$ различных печатей, обозначенных различными целыми числами. Все печати этой техники были записаны на табличке размером $n\times m$.

Наруто потерял таблицу. Он успел выучить элементы каждой строки из этой таблички слева направо, а также элементы всех столбцов из этой таблички сверху вниз, но он не помнит порядок самих строк и столбцов. Необходимо восстановить табличку, чтобы Наруто смог доучить новую технику.

Входные данные

Первая строка содержит единственное число t ($1 \le t \le 100000$) — количество тестовых случаев. В следующих строках содержатся описания тестовых случаев.
Первая строка каждого описания содержит два разделенных пробелом числа n и m ($1\le n, m\le 500$) — количество строк и столбцов в табличке. Все печати в табличке занумерованы натуральными натуральными числами от 1 до $n\cdot m$.
Следующие n строк содержат по m разделенных пробелом чисел — элементы произвольной строки исходной таблички слева направо.
Следующие m строк содержат по n разделенных пробелом чисел — элементы произвольного столба исходной таблички сверху вниз.
Сумма nm по всем тестовым случаям не превосходит 250000. Гарантируется, что каждая строчка таблицы встречается во входных данных ровно один раз, как и каждый столбец. Также гарантируется, что любое число от 1 до nm встречается ровно один раз в описании элементов строк, как и в описаниях элементов столбцов столбцах. Наконец, гарантируется, что существует таблица, подходящая под описание.

Выходные данные

Для каждого тестового случая необходимо вывести n строк по m чисел, разделенных пробелом — восстановленную табличку. Можно показать, что такая табличка единственная.

Пример

входные данные

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

выходные данные

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

Примечание

Рассмотрим первый тестовый случай. Матрица 2×3. Даны строки и столбцы в произвольном порядке.

Одна из строк $[6,5,4]$. Одна из строк $[1,2,3]$.

Один из столбцов $[1,6]$. Один из столбцов $[2,5]$. Один из столбцов $[3,4]$.

Вам нужно восстановить матрицу. Ответ дан в примере выходных данных.

Разбор задачи B

1413B - Новая техника

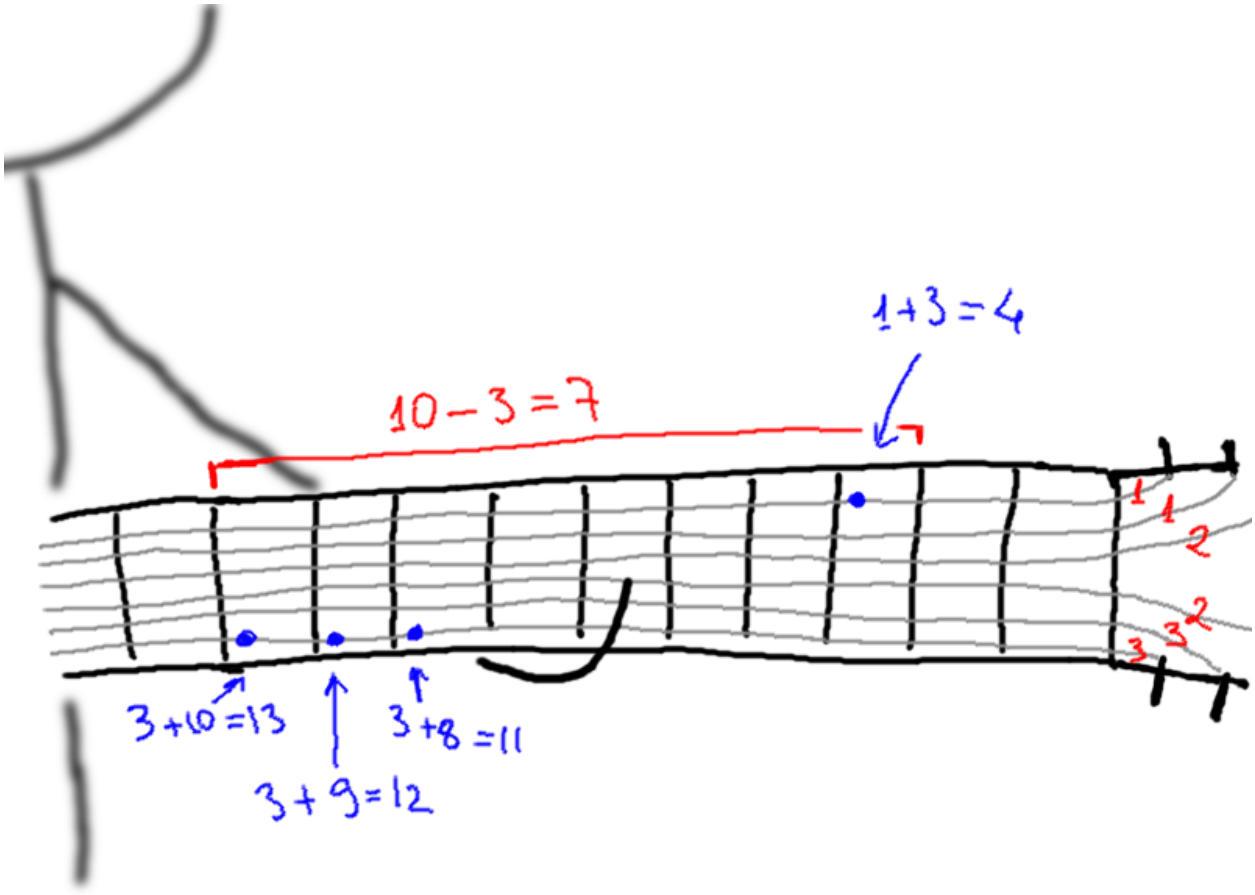
Для решения этой задачи достаточно для каждой строки определить на каком месте в исходной таблице она находится. Если мы рассмотрим первый символ в строке и найдём в каком столбце и на каком месте он находится, то это место будет как раз будет задавать положение нашей строки в исходной таблице. Так как все символы исходной таблицы различны, то положение каждой строки однозначно восстановится.

C. Простота исполнения

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

После боя с Шикамару Таюя решила, что ее флейта слишком предсказуемая, и поменяла ее на гитару. У гитары Таюи 6 струн и бесконечное количество ладов, пронумерованных с 1. Если зажать лад номер j на i-й струне, то получится нота a_i+j.

Таюя хочет сыграть мелодию из n нот. Ноты можно сыграть, используя различные комбинации струны и лада. Простота исполнения зависит от разности максимального и минимального номера используемых ладов. Чем эта величина меньше, тем проще исполнить технику. Требуется определить минимальную возможную разность.
Например, если $a=[1,1,2,2,3,3]$, а последовательность нот — $4,11,11,12,12,13,13$ (что соответствует второму примеру), то можно сыграть первую ноту на первой струне, а все остальные ноты на шестой струне. Тогда максимальный лад будет 10, минимальный 3, и разница составит $10-3=7$, как показано на рисунке.



Входные данные

В первой строке через пробел заданы 6 чисел a_1, a_2, ..., a_6 ($1 \le a_i \le 10^9$) — описания струн гитары Таюи.

Во второй строке задано число n ($1 \le n \le 100000$) — количество нот, которое хочет сыграть Таюя.

В следующей строке через пробел заданы n чисел $b_1, b_2, ..., b_n$ ($1 \le b_i \le 10^9$) — описание нот. Гарантируется, что $b_i > a_j$ для всех $1 \le i \le n$ и $1 \le j \le 6$. Иными словами, каждую ноту можно сыграть на любой струне.

Выходные данные

Необходимо вывести минимальную возможную разность номеров используемых ладов.

Примеры

входные данные

```
1 4 100 10 30 5
6
101 104 105 110 130 200
```

выходные данные

```
0
```

входные данные

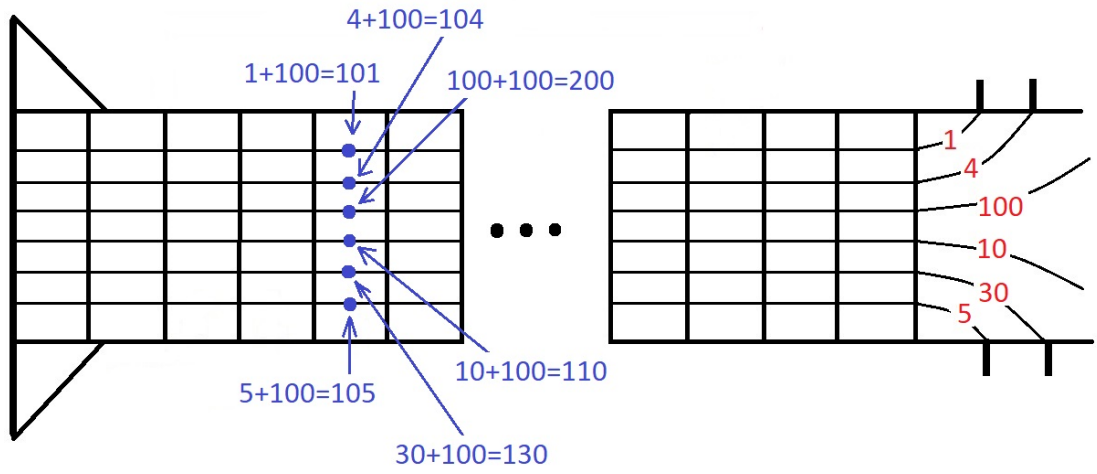
```
1 1 2 2 3 3
7
13 4 11 12 11 13 12
```

выходные данные

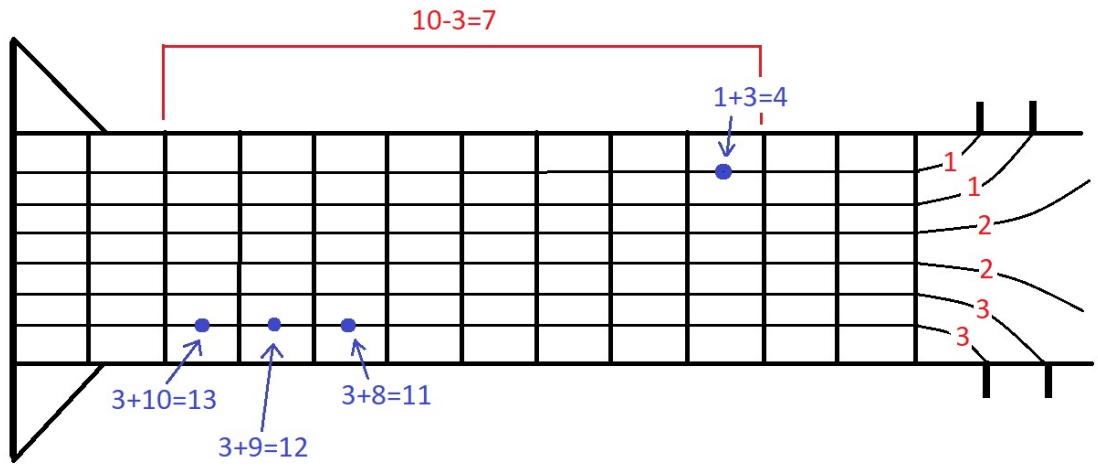
```
7
```

Примечание

В первом примере оптимальным будет сыграть первую ноту на первой струне, вторую ноту на второй струне, третью ноту на шестой струне, четвертую ноту на четвертой струне, пятую ноту на пятой струне, шестую ноту на третьей струне. Тогда для исполнения каждой ноты будет использоваться лад 100, а разница составит $100-100=0$.



Во втором примере оптимальным будет, например, сыграть вторую ноту на первой струне, а все остальные ноты на шестой струне. Тогда максимальный лад будет 10, минимальный 3, и разница составит $10-3=7$.



Разбор задачи C

1413C - Простота исполнения

Рассмотрим все возможные значения ладов, которые могут использоваться, для этого выпишем массив пар $(b_j - a_i, j)$, для всех возможных i, j, и посортируем его. Тогда в данном массиве надо найти подотрезок, с минимальной разницей первых аргументов, и при этом содержащий все возможные варианты второго аргумента (так как это будет означать, что для каждой ноты мы можем выбрать некоторую струну, чтобы лад попал в нужный отрезок). Для этого посчитаем массив $right[l]$, в котором для всех возможных левых концов будет содержаться минимальный правый край, такой что на подотрезке $[l, right(l)]$ встречаются все возможные значения второго аргумента. Нетрудно заметить, что $right[l] \leq right[l+1]$, так как если на отрезке $[l+1, right[l+1]]$ содержатся все возможные варианты второго аргумента, то и на отрезке $[l, right[l+1]]$ это тоже будет верно. Тогда чтобы искать все значения $right[l]$, можно двигать два указателя, и в каждый момент времени поддерживать множество нот, которые встречаются на текущем отрезке.
Когда мы нашли массив $right$, нам надо просто взять минимум разности первых аргументов, по всем возможным подотрезкам $[l, right[l]]$. Итоговая асимптотика $O(nm \log(nm))$.

D. Сюрикены

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

Тентен продает оружие в магазине для ниндзя. Сегодня она хочет продать n сюрикенов стоимостью $1, 2, ..., n$ рё (местная валюта). В течение дня она будет выкладывать сюрикены на витрину, которая в начале дня пуста. Работа в магазине достаточно простая: в каждый момент времени либо Тентен выкладывает на витрину очередной сюрикен из еще не выложенных, либо к ней в магазин заходит ниндзя и покупает сюрикен с витрины. Поскольку ниндзя очень экономные, то в магазине они покупают **самый дешевый** из сюрикенов, которые сейчас лежат на витрине. В течение дня Тентен ведет список событий, и в ее списке есть два типа записей:

- `+` означает, что она выкладывает на витрину очередной сюрикен;
- `- x` означает, что у нее купили сюрикен стоимостью x.

Сегодня у Тентен выдался удачный день, и у нее купили все сюрикены. В конце дня ей стало интересно, не ошиблась ли она в своих записях, и в каком порядке она могла выкладывать сюрикены, чтобы у нее получился такой список.

Входные данные

В первой строке дано целое число n ($1 \le n \le 10^5$) — количество сюрикенов.
В следующих $2n$ строках дана информация о произошедших событиях в формате, описанном выше. Гарантируется, что среди событий ровно n событий первого типа, а также, что во всех событиях второго типа присутствуют все числа от 1 до n по одному разу.

Выходные данные

Необходимо вывести «`YES`», если такой список мог получиться, и «`NO`» иначе.
Если список получиться мог, то в следующей строке необходимо вывести n различных чисел через пробел — стоимости сюрикенов в порядке, в котором Тентен должна выкладывать их на витрину. Если существует несколько решений, выведите любое из них.

Примеры

входные данные

```
4
+
+
- 2
+
- 3
+
- 1
- 4
```

выходные данные

```
YES
4 2 3 1
```

входные данные

```
1
- 1
+
```

выходные данные

```
NO
```

входные данные

```
3
+
+
+
- 2
- 1
- 3
```

выходные данные

```
NO
```

Примечание

В первом примере Тентен сначала выложила на витрину сюрикены стоимостью 4 и 2. После этого к ней пришел покупатель и забрал самый дешевый сюрикен стоимостью 2. После этого к оставшемуся сюрикену стоимостью 4 она выложила сюрикен стоимостью 3. После этого к ней пришел покупатель и забрал сюрикен стоимостью 3. После этого она выложила сюрикен стоимостью 1. После этого к ней пришел покупатель и забрал сюрикен стоимостью 1. Потом к ней пришел еще один покупатель и забрал последний сюрикен стоимостью 4. Обратите внимание, что порядок $[2,4,3,1]$ также является верным.

Во втором примере первый покупатель пришел и купил сюрикен раньше, чем Тентен выложила хотя бы один сюрикен на витрину, что, очевидно, невозможно.
В третьем примере Тентен выложила на витрину все сюрикены, после этого к ней пришел покупатель и забрал сюрикен стоимостью 2. Такого быть не могло, потому что этот сюрикен не был самым дешевым из тех, что лежали на витрине (ведь там был еще сюрикен стоимостью 1).

Разбор задачи D

1413D - Сюрикены

Заметим, что если в данный момент покупается сюрикен стоимости xx, то единственная информация, которую мы должны запомнить про оставшиеся сюрикены, что их стоимости $\geq x$. Также понятно, что если мы рассматриваем в какой-либо момент времени сюрикены, оставшиеся на витрине, то чем раньше сюрикен был положен на витрину, тем более сильное ограничение на него действует. Теперь рассмотрим какие варианты событий могут происходить при покупке сюрикена стоимости x. Если про все сюрикены на витрине

мы знаем, что их стоимость $>x$, тогда ответ на задачу отрицательный. Иначе у всех сюрикенов, у которых ограничение меньше чем xx, мы должны поднять ограничение до x, и удалить любой из них, так как никакие другие сюрикены мы удалить не можем, а все эти сюрикены неразличимы между собой. Но так как мы знаем, что на самый последний сюрикен положенный на витрину действует наиболее слабое ограничение, то можно ничего не поддерживать и просто каждый раз удалять его, а после проверить, что такая последовательность действий нам подходит. Проверку для конкретной последовательности можно делать используя любую кучу на минимум. Итоговая асимптотика будет $O(n \cdot \log n)$.

E. Oracle соло мид

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Меха-Наруто играет в компьютерную игру. У его персонажа есть следующая способность: нанести вражескому герою a единиц урона, затем восполнять ему b единиц здоровья в конце каждой секунды, начиная со следующей, на протяжении ровно c секунд. В частности, если эта способность применяется в момент времени t, то здоровье врага уменьшается на a в момент времени t, а затем увеличивается на b в моменты $t+1$, $t+2$, ..., $t+c$.

У способности есть время перезарядки, равное d секундам, то есть если Меха-Наруто применяет способность в момент времени t, то в следующий раз он может её применить не раньше момента $t+d$. По некоторым причинам Меха-Наруто может использовать способность только в целые моменты времени, поэтому все изменения здоровья врага также происходят в целочисленные моменты.
Эффекты от разных применений заклинания накладываются друг на друга. В частности, если вражеский герой находится под действием k заклинаний, применённых ранее и ещё не истёкших, то его здоровье увеличится на $k \cdot b$. Помимо этого все изменения, которые происходят в один и тот же момент времени, учитываются одновременно.
Теперь Меха-Наруто интересно, может ли он убить своего оппонента просто применяя свою способность так часто, как только можно (то есть каждые d секунд). Герой считается погибшим, если уровень его здоровья становится равным 0 или ниже. Предположим, что здоровье вражеского персонажа не изменяется никаким образом, кроме как от применения заклинания. Какое наибольшее количество здоровья может быть у врага, чтобы Меха-Наруто мог его убить?

Входные данные

Первая строка содержит единственное число t ($1 \le t \le 10^5$) — число тестов.
Каждый тест описывается четвёркой натуральных чисел a, b, c и d ($1 \le a,b,c,d \le 10^6$), записанных через пробел и означающих соответственно мгновенный урон от способности,

ежесекундно восполняемый объём здоровья, время действия каждого заклинания и время перезарядки способности.

Выходные данные

Для каждого теста выведите на отдельной строке -1, если способность может рано или поздно убить любого врага, каким бы большим ни был его уровень здоровья; в противном случае выведите наибольшее число здоровья, при котором оппонент будет убит.

Пример

входные данные

```
7
1 1 1 1
2 2 2 2
1 2 3 4
4 3 2 1
228 21 11 3
239 21 11 3
1000000 1 1000000 1
```

выходные данные

```
1
2
1
5
534
-1
500000500000
```

Примечание

В первом тесте из условия любая единица урона отменяется через секунду, поэтому Меха-Наруто не может нанести больше, чем 1 единицу урона.

В четвёртом тесте из условия герой оппонента получает:

- 4 урона (1-е применение способности) в момент 0;
- 4 урона (2-е применение способности), и 3 единицы здоровья восполняются (1-е применение) в момент 1 (всего 5 урона к начальному уровню здоровья);
- 4 урона (3-е применение способности), и 6 здоровья восполняется (1-е и 2-е применения) в момент 2 (всего 3 урона к изначальному здоровью);
- и так далее.

Можно доказать, что ни к какому моменту времени враг не получит суммарно 6 или больше урона, поэтому ответ на этот тест есть 5. Обратите внимание, как производится пересчёт здоровья: например, если бы у врага было 8 здоровья, он бы **не** умер в момент времени 1,

как если бы мы сначала вычли из его здоровья 4 единицы, а затем сочли бы его мёртвым, не успев добавить 3 единицы от лечения.

В шестом тесте из условия герой со сколько угодно большим количеством здоровья рано или поздно умрёт.

В седьмом тесте из условия ответ не вмещается в 32-битный целочисленный тип.

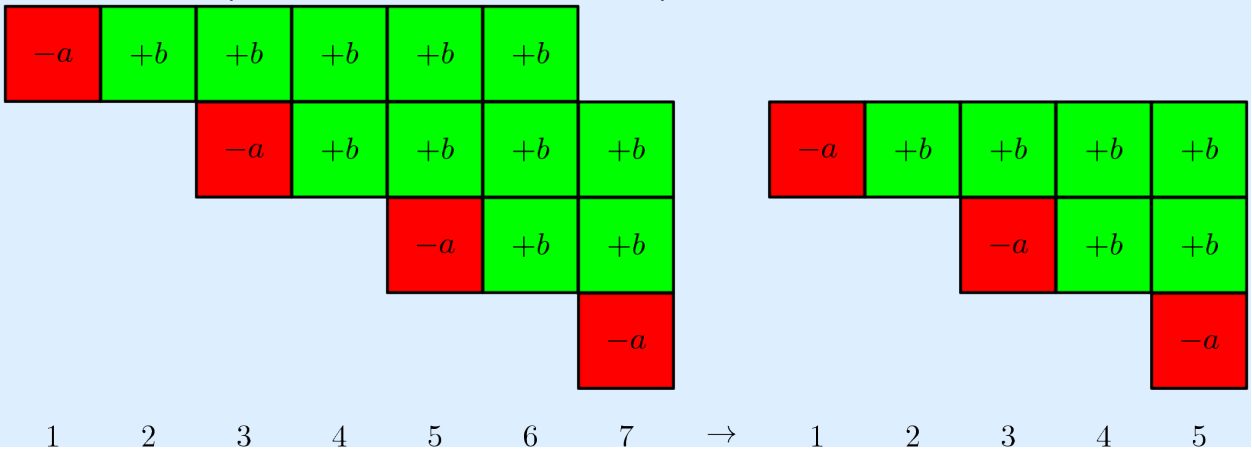
Разбор задачи E

1413E - Oracle соло мид

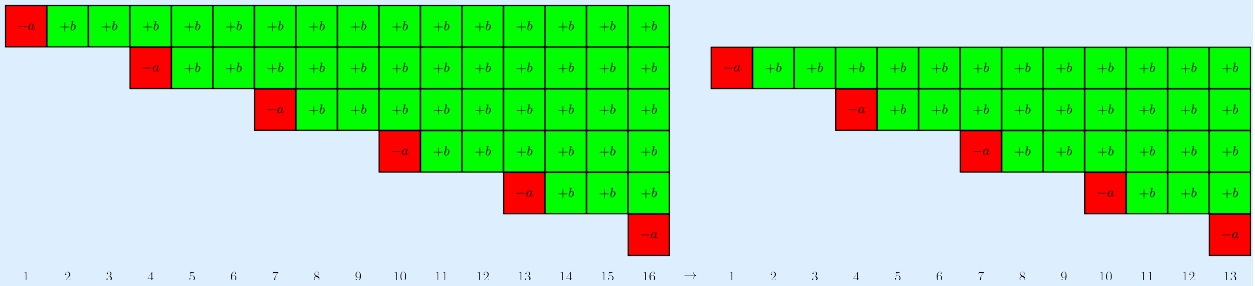
Будем использовать схематичные изображения всего происходящего в таком виде, что каждое изменение здоровья будет отображено в виде клетки, каждая строка будет обозначать одно применение заклинания, и каждый столбец будет обозначать секунду, в которую что-то происходит.

Во-первых, если $a>b\cdot c$, то ответ -1. В самом деле, после t секунд суммарный нанесённый урон равен $(a-bc)$ для всех заклинаний, действие которых уже прошло, а также ещё какой-то урон от заклинаний, которые ещё действуют. Первое слагаемое может быть сколь угодно большим, а второе ограничено снизу, например, числом $-bc^2$, поскольку максимум c заклинаний ещё не истекли, и каждое из них вылечило врага максимум на b единиц здоровья в секунду на протяжении максимум c секунд. Таким образом, нанесённый урон может быть сколь угодно велик.

С другой стороны, если $a \le bc$, то ответ всегда существует. В самом деле, имеет смысл смотреть только на моменты времени, делящиеся на d, т. е. в точности на моменты, когда мы наносили урон. Очевидно, что для любого другого t в момент времени t у врага было не меньше здоровья, чем за секунду до этого. Также если $t \ge c$, то у врага не меньше здоровья, чем в момент времени $t-d$. В самом деле, разница в этих уровнях здоровья лишь в одном заклинании, которое целиком истекло, а это неотрицательное число:



Теперь, когда мы знаем, что нужно рассматривать лишь $t<c$, мы точно знаем, что ответ существует. Поймём теперь, когда в общем случае следует вычитать d из t, чтобы уменьшить здоровье врага. Из таких же соображений следует, что если $t<c$, то при уменьшении t на d здоровье врага увеличивается на $a-tb$, а поскольку $t=dk$ для некоторого целого k, мы увеличиваем здоровье на $a-bdk$. Очевидно, это следует делать, пока $a-bdk<0$:



Иными словами, задача звучит теперь следующим образом: найти наибольшее такое k, что $a \geq bdk$, и применить заклинание $(k+1)$ раз. У врага будет наименьший уровень здоровья как раз как только мы применим заклинание в $(k+1)$-й раз. Ответ, таким образом, равен $a(k+1)-k(k+1)2bd$. Время работы составляет $O(1)$ на тест.

Также можно было заметить, что здоровье врага — выпуклая функция от времени, и найти ответ тернарным поиском. Это занимает $O(\log\ maxanswer)$ времени на тест, что всё ещё достаточно быстро.

F. Дороги и рамен

ограничение по времени на тест: 5 секунд
ограничение по памяти на тест: 512 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

В стране Огня есть n деревень и $n-1$ двусторонняя дорога, причем из каждой деревни можно добраться в каждую по дорогам. В этой стране есть только два типа дорог: каменные и песчаные. Поскольку страна Огня очень прогрессивная и постоянно обновляется, то каждый день рано утром строители выбирают одну дорогу и меняют ее тип (с каменной на песчаную или наоборот). А еще в стране Огня все любят рамен, поэтому каждый день на каждой **каменной** дороге устанавливается одна палатка с раменом, а в конце дня палатку убирают.

В течение каждого из ближайших m дней, после того как очередную дорогу переделают, Наруто и Джирайя выбирают себе простой маршрут по стране Огня. Их маршрут может начинаться с любой деревни и заканчиваться тоже в любой (возможно, той же), но по каждой дороге они могут проходить не более одного раза. Поскольку Наруто и Джирайя очень любят рамен, то на каждой каменной дороге они обязательно покупают ровно одну тарелку рамена и кто-нибудь один из них ее ест. Поскольку у ниндзя все должно быть честно, то они выбирают только те пути, проходя по которым они могут съесть равное количество тарелок с раменом. Наруто и Джирайя любят много путешествовать, поэтому

каждый день они выбирают самый длинный путь из подходящих. Для каждого дня необходимо определить максимальную длину пути (то есть количество дорог в нём), которым могут пойти Наруто и Джирайя.

Входные данные

В первой строке дано натуральное число n ($2 \le n \le 500000$) — число деревень в стране Огня. Каждая из следующей $(n-1)$ строки содержит описание дороги: три натуральных числа u, v и t ($1 \le u,v \le n$, $t \in \{0,1\}$). Первые два числа определяют номера деревень, между которыми проложена дорога, а третье число определяет начальный тип дороги: 0 — песчаная, 1 — каменная. Дороги нумеруются числами от 1 до $(n-1)$ в порядке, заданном во входных данных.

В следующей строке задано натуральное число m ($1 \le m \le 500000$) — число дней, которые путешествуют Наруто и Джирайя.

Каждая из последующих m строк содержит одно число id ($1 \le id \le n-1$) — номер дороги, которую переделывают утром соответствующего дня.

Гарантируется, что между любыми двумя деревнями есть путь по дорогам.

Выходные данные

Необходимо вывести mm строк, i-я из которых содержит максимальную длину подходящего пути в i-й день.

Пример

входные данные

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

выходные данные

```
3
2
3
3
2
```

Примечание

После изменения дороги под номером 3 самый длинный путь состоит из дорог 1, 2 и 4.

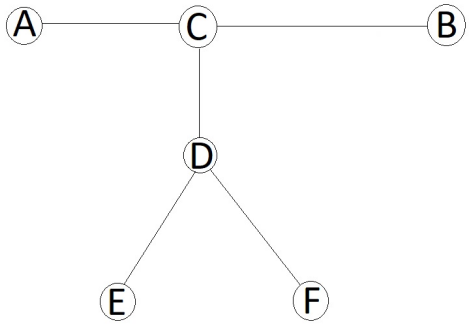
После изменения дороги под номером 4 самый длинный путь может состоять из дорог 1 и 2.
После изменения дороги под номером 1 самый длинный путь может состоять из дорог 1, 2 и 3.
После изменения дороги под номером 3 самый длинный путь состоит из дорог 1, 2 и 4.
После изменения дороги под номером 4 самый длинный путь может состоять из дорог 2 и 4.

Разбор задачи F

1413F - Дороги и рамен

Один из оптимальных путей всегда начинается в одном из концов диаметра дерева.

Доказательство:



Пусть на рисунке выше, AB — это диаметр дерева, а оптимальный ответ — это EF. Тогда чётность количества каменных дорог на DE и DF одинаковая, также отсюда чётность количества каменных дорог на CE и CF тоже одинаковая. Так как диаметр — это самый длинный путь в дереве, то чётность количества каменных дорог на AC и BC различна (иначе диаметр был бы оптимальным ответом). Значит чётность количества каменных дорог на CE совпадает с чётностью на AC или на BC. Пусть, не теряя общности, это будет путь AC. Тогда на пути AE чётное количество каменных дорог. Заметим, что так как AB — это диаметр, то AC не короче, чем CF, отсюда AD не короче, чем DF, и наконец отсюда AE не короче, чем EF. А следовательно существует оптимальный путь, начинающийся в одном из концов диаметра.
Осталось научиться решать задачу, когда один из концов пути зафиксирован. Если подвесить дерево за эту вершину, и для каждой вершины выписать чётность количества дорог до корня, то тогда изменение веса одного ребра меняет все чётности в некотором поддереве. Если занумеровать все вершины дерева в порядке обхода dfs, то тогда любое

поддерево является непрерывным отрезком в такой нумерации. А значит наша задача свелась к следующей: «Изначально есть массив из нулей и единиц, приходят запросы заменить все значения на подотрезке на противоположные, и после каждого запроса надо найти позицию, в которой записан ноль, и при этом у неё самый большой параметр глубины». Эту задачу можно решать с помощью дерева отрезков, поддерживая в вершине ДО самую глубокую вершину, в которой написан ноль, и самую глубокую вершину, в которой написана единица. Итоговая асимптотика $O(n \log n)$.