

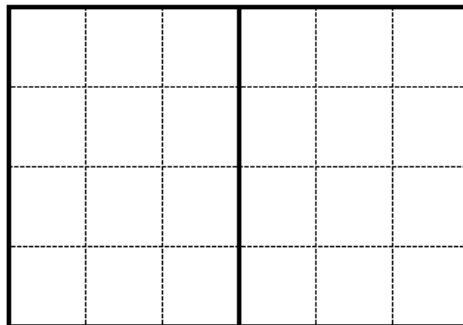
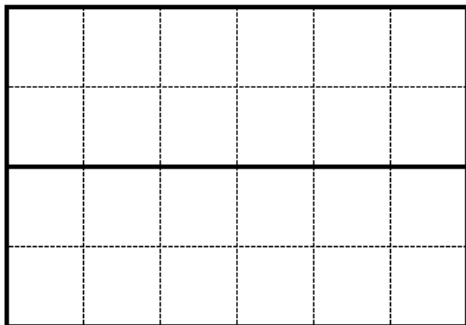
## Задача 1. Задача

### Подзадача 1

При  $k = 1$  всегда ответ равен 1.

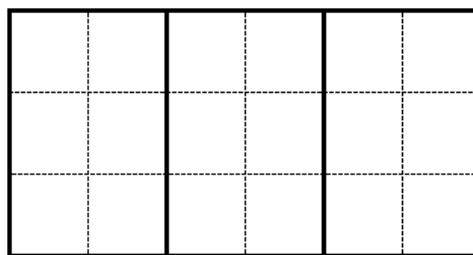
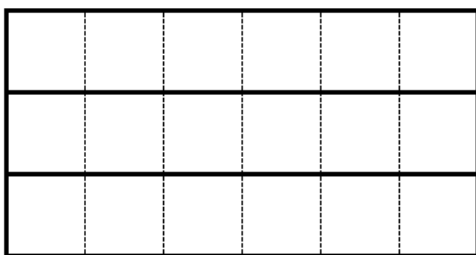
### Подзадача 2

Разрезать торт на две части можно вдоль одной из сторон, если длина этой стороны чётная, соответственно, нужно проверить чётность чисел  $w$  и  $h$ :

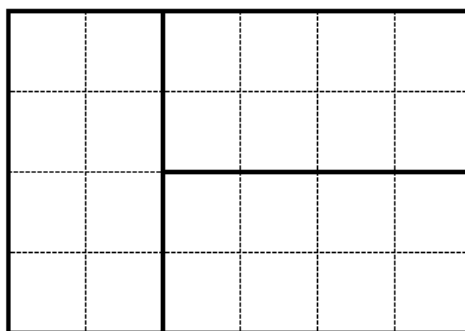
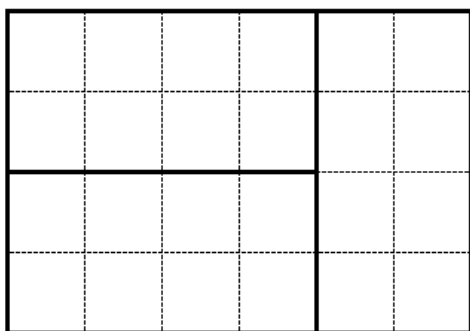


### Подзадача 3

Аналогично можно разрезать торт двумя параллельными разрезами вдоль одной из сторон, если её длина делится на 3:

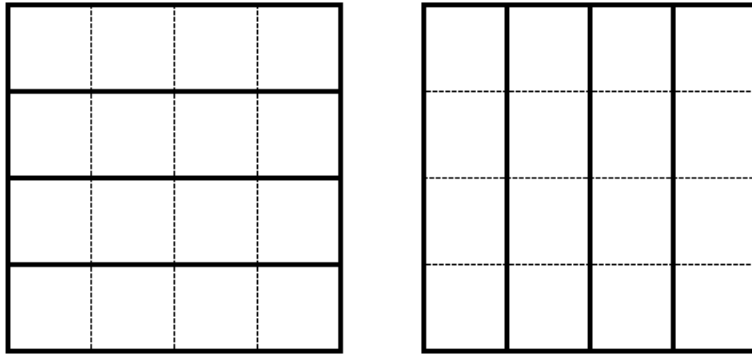


Кроме того, при соотношении сторон 3:2 (или 2:3) существует ещё 2 варианта:

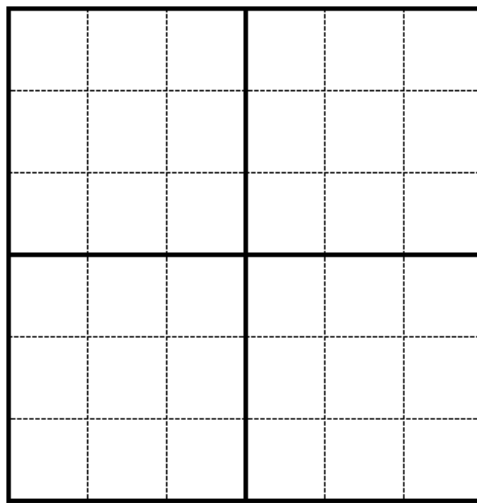


### Подзадача 4

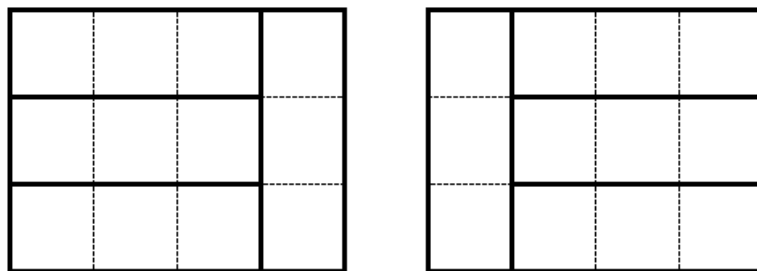
Можно сделать три параллельных разреза, если длина стороны делится на 4:



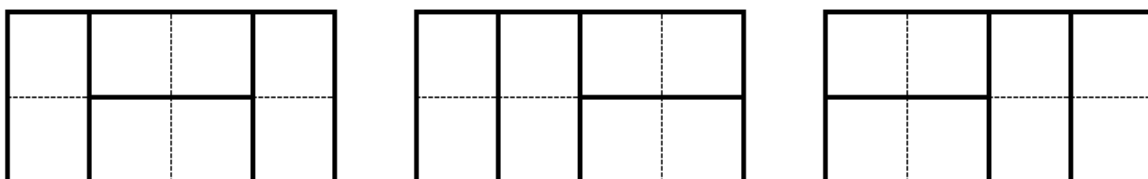
Также можно разрезать торт на 4 части, сделав разрезы через середины сторон, если длины сторон делятся на 2:



**Подзадача 5** Помимо вариантов, указанных в предыдущей подзадаче, могут существовать ещё следующие конструкции. Если длины сторон относятся как 4:3 (или 3:4), то существует дополнительно 2 варианта:



Наконец, если длины сторон относятся как 2:1 (или 1:2) и длина большей стороны делится на 4, то есть ещё 3 варианта:



## Задача 2. Баланс

### Подгруппа 1

Ответом будет сумма всех неотрицательных и максимального отрицательного числа, так как они все будут находиться на одном отрезке.

### Подгруппа 2

Здесь перебор всех подотрезков с явным пересчетом суммы и проверкой на наличие отрицательного и неотрицательного значения.

Решение  $O(n^3)$

### Подгруппа 3

Немного оптимизированный вариант подгруппы 2. Здесь поддерживаем сумму и наличие отрицательного и неотрицательного значения сразу, когда меняем правую границу подотрезка.

Решение  $O(n^2)$

### Подгруппа 4

Сумма неотрицательных не может быть больше 0, значит, нужно ответом будет отрезок из 2 элементов — из 0 и отрицательного числа. Найдем среди всех таких подотрезков максимальный.

### Полное решение

Посчитаем при помощи префиксных сумм три массива — сумму элементов, количество отрицательных и количество неотрицательных. Теперь задача сводится к обычной задаче о нахождении максимальной суммы при помощи префиксных сумм, только с дополнительным условием на проверку наличия отрицательного и неотрицательного значения на отрезке.

## Задача 3. Незадача

### Подзадача 1

В первой подзадаче необходимо было придумать решение для однозначного числа. Если  $a_i \neq x$ , то выводим  $a_i$ , так как введённое число и так не содержит запрещённой цифры. В случае, если  $a_i = x$ , необходимо вывести сумму двух цифр, цифры  $a_i - 1$  и цифры 1.

### Подзадача 2

Во второй подзадаче гарантируется, что на калькуляторе не работает кнопка с цифрой 9,  $a_i \leq 99$ . Для всех  $a_i \leq 89$ , содержащих в своей записи цифру 9, будет работать стратегия из первой подзадачи: выпишем сумму  $a_i - 1$  и 1. Например, если  $a_i = 79$ , то ответом будет  $78 + 1$ .

Числа от 90 до 98 можно разложить в сумму  $a_i - 10$  и 10. Например, если  $a_i = 93$ , то ответ  $83 + 10$ .

Наконец, для числа 99 ответ, например,  $88 + 11$  или  $3 * 33$  и т.д.

### Подзадача 3

При решении третьей подзадачи может возникнуть ошибочное предположение, что любое число раскладывается на сумму двух, не содержащих цифры  $x$ , чисел. Однако это утверждение неверно, в качестве контрпримера можно рассмотреть, например,  $a_i = 19$  или  $a_i = 21$  при  $x = 1$ .

Можно доказать, что те значения, которые нельзя разложить в сумму двух чисел, всё ещё можно разложить в сумму не более четырёх слагаемых. Выполним предподсчет ответа для всех чисел от 2 до 999. Каждое из чисел попытаемся разложить в сумму двух слагаемых. Те числа, которые разложить не получилось, представим в виде суммы любых двух значений, для которых разложение уже вычислено. В худшем случае мы получим 4 слагаемых по 3 цифры в каждом, а также 3 знака «+», итого 15 знаков в записи.

### Подзадача 4

В четвертой подзадаче отсутствуют ограничения на  $a_i$ , а значит решение, описанное в предыдущей подзадаче, может работать слишком долго.

Для решения рассмотрим два случая:

1).  $x = 0$

Разобьём данное целое число  $a_i$  на сумму двух неотрицательных чисел. Прделаем это поразрядно, имитируя деление числа на 2 столбиком, двигаясь от младших разрядов к старшим. Выполняя деление, будем распределять цифры в разрядах так, чтобы они не содержали 0 в своей записи.

Обозначим цифру в текущем разряде как  $cur$ . Если  $cur < 2$ , то необходимо занять единицу из следующего по старшинству разряда и записать  $cur = cur + 10$ .

Тогда  $a = cur/2$ ,  $b = cur - a$  — цифры, которые мы запишем в тот же разряд в двух слагаемых. В последнем разряде весь остаток можно переложить в разряд первого числа.

Удалив ведущие нули в записи результирующих слагаемых, можно вывести сумму в качестве результата.

2).  $x \neq 0$

(\*) Заранее отметим случай, когда  $x = 1$  и последняя цифра числа  $a_i$  также равна 1. Вынесем в отдельное слагаемое цифру 2 и продолжим обрабатывать уже значение  $a_i - 2$ . Позже мы вернемся к этому действию и обоснуем его необходимость.

Так же, как и в первом случае, пройдем по разрядам числа от младших к старшим. Если цифра в текущем разряде  $cur = x$ , то разложим её на сумму  $a = 1$  и  $b = x - 1$ , иначе на сумму  $a = 0$  и  $b = cur$ .

Тогда получим разложение исходного числа в сумму двух: цифры первого числа всегда равны либо 0, либо 1. Все цифры второго числа гарантированно не равны  $x$ .

Особый случай представляет  $x = 1$ . Первое число при таком алгоритме формирования гарантированно будет чётным с учётом (\*), поэтому давайте поделим его на 2. В записи полученных слагаемых в таком случае будут только цифры 5 и 0.

### Подзадача 5

Для решения пятой и шестой подзадач необходимо использовать идею динамического программирования. Дополнительные ограничения на  $a_i$  в пятой подзадаче позволяют писать как двумерную динамику, так и одномерную, описываемую в шестой подзадаче. Ограничения упрощают реализацию: например, нет необходимости предподсчитывать заранее делители, достаточно проверять все числа до корня стандартным алгоритмом, допускается альтернативная реализация с большим числом состояний.

### Подзадача 6

Для начала вычислим оптимальные представления числа только с учётом операции умножения. Введём  $dpmult[n]$  — длина минимального по длине арифметического выражения по умножению. Для ускорения перебора делителей числа можно выполнить предподсчёт, например, используя решето Эратосфена. Рассматривая очередной делитель числа  $n$ , проверим, что он не содержит цифры  $x$  и пересчитаем результат по формуле  $len(y) + 1 + dpmult[n/y]$ , а также для дальнейшего восстановления ответа запомним величину делителя, который позволил достичь минимальной длины выражения ( $mult[n] = y$ ).

Теперь перейдем к подсчету основного ответа с учетом уже и сложения, и умножения. Введём массив для такой динамики:  $dpsum[n]$ . Будем перебирать все  $y$ , не превышающие половины  $n$  и значения 200. Пересчитывать значения динамики будем через формулу  $dpmult[y] + 1 + dpsum[n - y]$ , также запоминая, какое значение для данного арифметического выражения было наилучшим  $sum[n] = y$ .

## Задача 4. Атака фениксов

### Подгруппа 1

В этой подгруппе гоблины никак не могут убить ни одного феникса, значит каждый феникс совершит по 2 атаки. Нужно проверить, хватит ли фениксов, чтобы уничтожить всех гоблинов.

### Подгруппа 2

В этой подгруппе один феникс за первую атаку уничтожает любой отряд гоблинов.

### Подгруппа 3

При  $D \geq 2$  феникс уничтожает любой отряд, рассмотрим вариант, когда  $D = 1$ . Если  $g_i = 1$ , то на  $i$ -й отряд гоблинов достаточно одного феникса. Если  $g_i = 2$ , то при  $H \geq 2$  также достаточно одного феникса, если же  $H = 1$ , то на отряд гоблинов нужно 2 феникса.

### Подгруппа 4

Давайте заметим следующий важный факт: Если мы сформируем отряд из  $x$  фениксов и отправим его атаковать отряд гоблинов и при этом погибнет как минимум 2 феникса, то нам лучше послать на этот отряд гоблинов два отряда фениксов — первый из 1 феникса и второй из  $x - 1$  фениксов, таким образом у нас погибнет только феникс из первого отряда. Это означает, что если при нападении на отряд гоблинов может погибнуть не более одного феникса.

На каждый отряд гоблинов можно отправить как минимум  $\lfloor \frac{m}{n} \rfloor$  фениксов. Если все фениксы погибнут или все выживут, то понятно какой ответ. Рассмотрим вариант, где у нас при нападении на отряд гоблинов формируется по два отряда фениксов, первый из 1 феникса и второй из  $\lfloor \frac{g_i-1}{2} \rfloor$ , тогда погибает  $n$  фениксов, но у нас еще остается  $m - n \cdot \lfloor \frac{g_i-1}{2} \rfloor$  фениксов, нужно посчитать сколько фениксов необходимо, чтобы при нападении на отряд гоблинов ни один феникс не погиб и минимизировать потери фениксов.

#### Подгруппа 5, 6

Подгруппы с динамическим программированием. Будем считать  $dp[i][j]$  — сколько погибнет фениксов, мы побили первые  $i$  отрядов гоблинов и всего потратили  $j$  фениксов. Тогда  $dp[i][j] = \min(dp[i][j], dp[i-1][k] + cnt(j-k))$ , где  $k$  от 1 до  $j-1$ , а  $cnt(j-k)$  — количество погибших фениксов, если на  $i$ -м отряде гоблинов мы используем  $j-k$  фениксов.

Решение  $O(n \cdot m^2)$

#### Подгруппа 7 и полное решение

Подгруппа 7 приводит к полному решению, но с упрощенными формулами.

Давайте посчитаем для каждого отряда гоблинов два значения: количество фениксов, необходимое для уничтожения гоблинов без потерь  $cnt = \max(\lfloor \frac{g_i}{2 \cdot D} \rfloor, \lfloor \frac{g_i - H}{D} \rfloor + 1, 1)$  и количество фениксов, необходимое для уничтожения отряда гоблинов при потере 1 феникса (нужно вычесть один удар одного феникса, а затем посчитать сколько фениксов с двойными атаками нужно для уничтожения оставшихся гоблинов в отряде). После этого еще посчитаем сколько фениксов можно сэкономить, если позволить одному фениксу погибнуть (здесь мы не минимизируем кол-во погибших фениксов, а минимизируем количество использованных фениксов) и запишем в массив  $add$ .

Если можно победить без потерь, то побеждаем.

Отсортируем  $add$  по невозрастанию, таким образом будем высвобождать максимальное количество фениксов для возможного использования в следующих отрядах, зная что для текущего отряда гоблинов один феникс точно погибнет. Проходимся по массиву  $add$  и поддерживаем количество погибших фениксов, в момент, когда суммы использованных  $add$  будет достаточно, чтобы победить, то прекращаем цикл и выводим количество погибших фениксов.