Разбор задач

Задача 1. Вишнево-черешневый сад

Пусть всего есть n черешен и m вишен. Тогда, если $m - 1 \geqslant n$, то можно посадить все деревья (для этого необходимо сначала высадить по одной черешне между двумя вишнями, а затем все оставшиеся черешни посадить правее). Иначе возможно посадить только $2 \cdot m + 1$ деревьев — высаживать по одной черешне между каждыми двумя вишнями.

За правильный ответ на первый тест участник получает 25 баллов.

Аналогично за правильный ответ на второй тест.

За третий тест начисляется 25 баллов.

За четвертый — 25.

Задача 2. Влад и дрон

Оптимальный ответ достигается, если начать в точке (7, B), а также задать программу «RURLUURURRLUDLLDLLUDL». Длина такой программы равна 21, что является минимальным ответом.

Если нет точки, для которой программа участника является корректной, то он получает 0 баллов.

Иначе участник получает $\max(0, 100 - 15 \cdot diff)$, где $diff$ — разность длины ответа участника и правильного ответа.

Задача 3. Андрей и аквариум

Будем перебирать первое число (назовем его A) так, чтобы $A \cdot A \cdot A \leqslant 24$, а также 24 делилось на A. Тогда нам подходят числа $1, 2$ (3 не подходит, так как $3 \cdot 3 \cdot 3 = 27 > 24$). Второе число (назовем его B) будем перебирать так, чтобы $A \leqslant B \cdot B \leqslant \frac{24}{A}$. Тогда получаются следующие случаи:

1. 1 1 24
2. 1 2 12
3. 1 3 8
4. 1 4 6
5. 2 2 6
6. 2 3 4

Таким образом, всего существует шесть подходящих аквариумов.

Если в какой-то строке введено не три числа, либо введено не число, либо произведение в какой-либо строке не равно 24, то участник получает 0 баллов.

Иначе участник получается $\lfloor \frac{100 \cdot cnt}{6} \rfloor$, где cnt — количество различных троек в ответе участника.

Тройки, различающиеся порядком, считаются одинаковыми.

Задача 4. Марта и треугольник Серпинского

Очевидно, что с каждой стороны будет одинаковое количество треугольников, поэтому посчитаем их количество с одной из них, а потом умножим на 3.

Если на какой-то итерации было n треугольников, граничащих с центральным, то на следующей каждый из них разобьется на 2 поменьше. Следовательно, на следующей итерации будет $2 \cdot n$ треугольников. Таким образом, на итерации под номером i будет $3 \cdot 2^{i-1}$ треугольников, граничащих с центральным.

Если в ответе не 4 числа, либо введено не число, то участник получает 0 баллов.

Иначе за каждый тест участник получает по 25 баллов.

Задача 5. Егор и шифр

В этой задаче нужно было просто повторить процесс перекатывания кубика. Это можно было сделать, представляя перекатывания кубика, а можно было сделать небольшую копию из бумаги и повращать ее. Правильный ответ: «ETHERNET»

Если длина ответа не равна 7, то участник получает 0 баллов.

Если ответ участника совпадает с ответом жюри, то он получает 100 баллов.

Иначе участник получает $\lfloor \frac{100}{7} \rfloor \cdot (f - 1)$, где f — позиция первого символа в ответе участника, который не совпадает с соответствующим ему в ответе жюри.

Разбор задач

Задача 1. Ракета на старт

В задаче было 4 подзадачи, в каждой из которых требовалось найти наибольшую возрастающую подпоследовательность данной последовательности (НВП).

Для решения задачи можно было воспользоваться так называемой жадной стратегией (например, каждый раз брать очередной подходящий элемент последовательности) вместе с перебором вариантов.

С помощью жадного алгоритма и перебора первого элемента в подпоследовательности можно было получить следующие ответы на подзадачи $1-3$ (возможны и другие варианты правильных ответов):

1. 3 4
2. 1 3 5 8
3. 4 5 6 7 8

В последней подзадаче перебирать только первые элементы в ответе, действуя дальше жадно, недостаточно, а перебрать все варианты для каждого из следующих элементов было бы слишком долго. Чтобы получить ответ максимальной длины, можно было рассуждать так:

- Возьмем 2 первым элементом в нашу НВП (этот элемент первый и минимальный, с него можно начать подпоследовательность с любыми другими элементами).
- Вместо того, чтобы дальше брать 5, возьмем следующую после пятерки 3. Этот ход мотивирован тем, что любой элемент, потенциально следующий после 5 в нашей НВП, также может следовать и после 3. Однако, не каждый элемент после 3 может встать и после 5. Таким образом, выбирая между соседними тройкой и пятеркой, взять тройку выгоднее.
- Далее можно было бы взять 4, потом 7, затем 8 и 9.
- Получаем желаемую подпоследовательность длины шесть: 2 3 4 7 8 9.

Задача 2. Межпланетные грузовые перевозки

В этой задаче необходимо найти максимальную массу перевозимого груза. За одну операцию можно было увеличить грузоподъемность всех уже заказанных кораблей, либо заказать еще один корабль, грузоподъемность которого будет равна y. Можно было сделать не более k операций, причем изначально было заказано x кораблей с начальной грузоподъемностью y.

Решение задачи сильно облегчает следующее наблюдение: поскольку операция увеличения грузоподъемности затрагивает только корабли, заказанные до её выполнения, выгодно сначала заказать определенное количество кораблей, и только после этого увеличивать их грузоподъемность. Таким образом, заказывая k_1 кораблей, можно будет увеличить грузоподъемность каждого из них на $k_2 = k - k_1$.

Тогда максимальная масса перевозимого груза будет равна $(x+k_1)\cdot(y+k_2)$. Остается выбрать k_1, при котором данное произведение будет максимально. Заметим, что сумма $s = x+k_1+y+k_2 = x+y+k$ не зависит от выбора k_1. Обозначим $a = x + k_1$, $b = y + k_2$. Покажем, что произведение a и b максимально, когда значения a и b отличаются друг от друга как можно меньше, а их сумма не меняется. Если s четная, тогда можно считать, что $a = \frac{s}{2} + z$, $b = \frac{s}{2} - z$. Их произведение тем больше, чем меньше величина z. В этом несложно убедиться при перемножении указанных величин. Аналогичный вывод можно сделать и для нечетной суммы (в этом случае оптимально, чтобы перемножаемые величины отличались только на 1).

Тогда мы получаем следующие ответы для каждой из предложенных ситуаций:

Номер ситуации	x	y	k	k_1	k_2	**Грузоподъёмность**
1	1	1	2	1	1	4
2	3	4	4	2	2	30
3	6	6	7	3	4	90
4	2	8	8	7	1	81

Задача 3. Яблочный пирог

Для того, чтобы составить формулу подсчета количества заготовок яблок, необходимо количество заготовок, расположенных вдоль одной стороны n, умножить на количество заготовок, расположенных вдоль другой стороны m. А чтобы знать, сколько заготовок помещается вдоль каждой стороны, необходимо длину этой стороны поделить на диаметр заготовки. Таким образом формула примет вид $(n/k) \cdot (m/k)$.

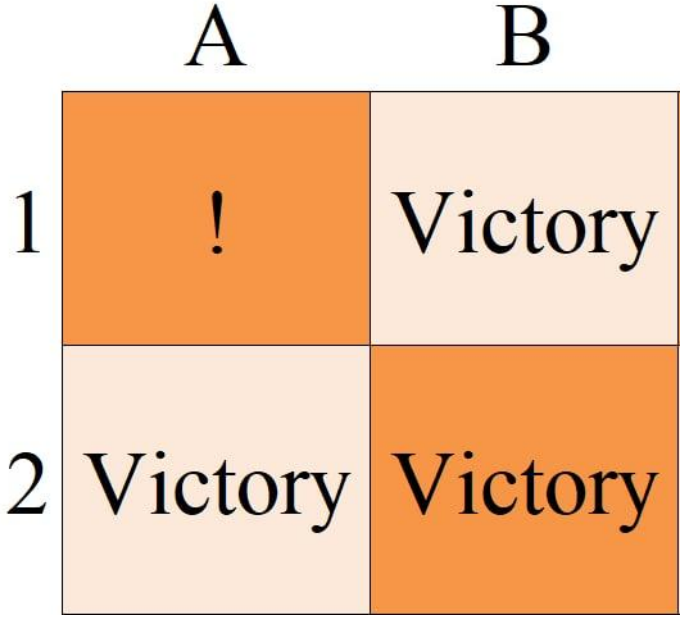
Для расчета количества слив, необходимых для пирога, можно заметить, что слив в каждом ряду и в каждом столбце всегда на 1 меньше, чем заготовок яблок. Ведь сливы помещаются строго между яблоками. Следовательно, формула примет вид $(n/k - 1) \cdot (m/k - 1)$

Значит, один из возможных ответов на задачу следующий:

- $(n/k) * (m/k)$
- $(n/k - 1) * (m/k - 1)$

Задача 4. Петя, Ваня и Леон

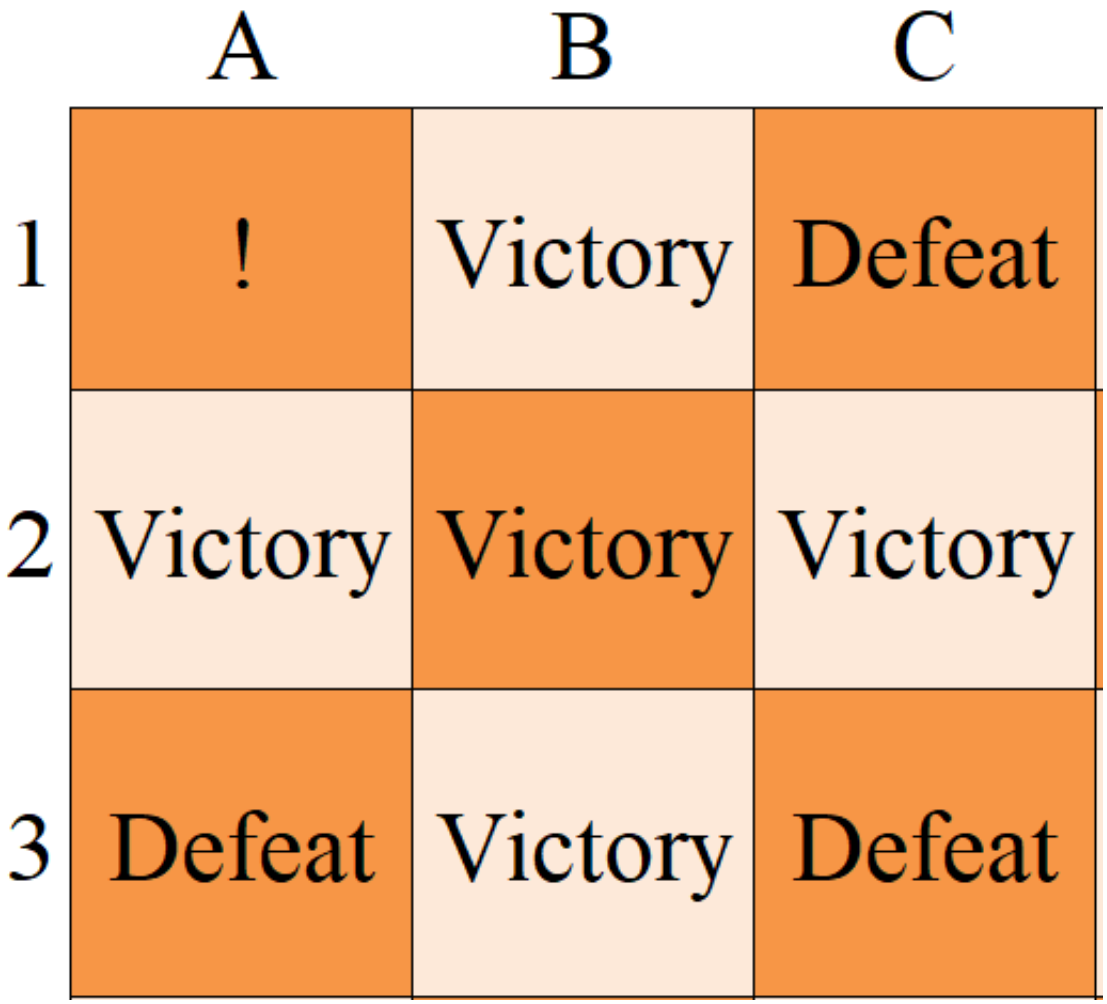
Для того, чтобы понять, какие же клетки нижней строки могут привести Петю к победе независимо от игры Вани, необходимо понять, из каких клеток Петя может всегда выиграть, а из каких способен проиграть. Если мы оставим ту же финишную клетку $A1$ и представим, что игра происходит на поле размером 2×2, то увидим, что Петя способен выиграть своим первым же ходом. Посмотрим для этого на рисунок



Отсюда делаем вывод, что клетки $A2$, $B1$, $B2$ – выигрышные для ходящего игрока.

Теперь представим ситуацию с полем размером 3×3. На поле такого размера отметим уже известные нам заранее выигрышные для Пети клетки $A2$, $B1$, $B2$. И, посмотрев на поле, заметим, что есть клетки, из которых можно сходить только в отмеченные выигрышные клетки. Это клетки $C1$ и $A3$. Отмечаем их как проигрышные клетки для ходящего игрока. Соответственно клетки $C2$

и $B3$ становятся выигрышными, так как из них есть ход в проигрышные клетки. А клетка $C3$ становится проигрышной по тому же алгоритму (из неё возможны ходы только в выигрышные клетки). Это можно представить в виде рисунка.



Аналогичным образом разбираем решение и для доски 7×7. Если из клетки ход возможен только в выигрышные клетки, то считаем её проигрышной. Если же из клетки есть шанс совершить ход в проигрышную клетку, то считаем её выигрышной. Изобразим наше решение в виде рисунка.

	A	B	C	D	E	F	G
1	!	Victory	Defeat	Victory	Defeat	Victory	Defeat
2	Victory	Victory	Victory	Victory	Victory	Victory	Victory
3	Defeat	Victory	Defeat	Victory	Defeat	Victory	Defeat
4	Victory	Victory	Victory	Victory	Victory	Victory	Victory
5	Defeat	Victory	Defeat	Victory	Defeat	Victory	Defeat
6	Victory	Victory	Victory	Victory	Victory	Victory	Victory
7	Defeat	Victory	Defeat	Victory	Defeat	Victory	Defeat

Из рисунка становится понятно, что в последнем ряду выигрышным клеткам соответствуют столбцы B, D, и F. Это и является правильным ответом.

Задача 5. Поезд

В задаче требовалось найти, сколько раз Василию придется перейти из одного вагона в соседний для того, чтобы встретиться с Петром. Заметим, что это число равно разности между номером вагона Петра и номером вагона Василия.

Как найти номер вагона, в котором находится место с номером p, зная, что в каждом вагоне K мест, и места имеют сквозную нумерацию?

Для этого необходимо понимать, что места с номерами $1 \dots K$ находятся в первом вагоне, места с номерами $K+1 \dots 2K$ — во втором, и так далее. Если же нумеровать вагоны с нуля, то номер вагона, в котором находится место p, равно $\lfloor \frac{p-1}{K} \rfloor$.

Тогда ответ на задачу равен $\lfloor \frac{Y-1}{K} \rfloor - \lfloor \frac{X-1}{K} \rfloor$.

Код на языке программирования Python3, соответсвующий решению на 100 баллов:

```
k = int(input())
x = int(input())
y = int(input())
a = (x - 1) // k
b = (y - 1) // k
print(b - a)
```

Стоит отметить, что в задаче присутствует содержательная группа номер 3, в которой все числа не превосходят 100. Она имеет разве что только косвенное отношение к полному решению, и может быть решена с помощью моделирования действий Василия.

Задача 6. Странное устройство

Для начала разберем решение для подгруппы, где $N \leqslant 20$. В этой подзадаче можно было перебрать все возможные варианты почти любым способом. Для прохождения третьей группы нужно было каким-то образом ускорить перебор (например, используя запоминание ответов для уже найденных меньших значений).

Рассуждая над решением подзадачи номер 1, где $K = 2$, можно было заметить следующее: посмотрим на чётность числа N. Если оно нечетно, то последнее действие, которое было сделано

с устройством — это нажатие первой кнопки, увеличивающей число на дисплее на 1. В противном случае, если N чётно, несложно понять, что в оптимальном решении последним действием была нажата вторая кнопка. Таким образом, следующее решение проходит первую подгруппу:

- Если N равно 0, то нажимать на кнопки не требуется.
- Если N нечетно, уменьшаем N на единицу, потому что предыдущим действием могла быть нажата только первая кнопка.
- Если N четно, делим N на 2, поскольку в оптимальном решение предыдущим действием была нажата вторая кнопка.

Ниже приведение код, реализующий идею выше:

```
n = int(input())
k = int(input())
ans = 0
while n > 0:
    if n % 2 == 0:
        n //= 2
    else:
        n -= 1
    ans += 1
print(ans)
```

Поскольку каждое второе действие делит N на 2 (в худшем случае), то будет сделано не более 60 операций.

Для полного решения требовалось обобщить идею, работающую только при $K = 2$. По аналогии, если N не делится на K, то предыдущие $N \ mod \ K$ действий были прибавлением 1 (где $a \ mod \ b$ — остаток при делении a на b). Если же N кратно K, то в оптимальном решении последним действием было умножение на K. Таким образом, алгоритм, решающий задачу на 100 баллов следующий:

- Если N равно 0, то ничего делать не требуется.
- Если N не кратно K, то нужно прибавить к ответу $N \ mod \ K$, а из N вычесть $N \ mod \ K$.
- Если N кратно K, то нужно к ответу прибавить 1, а N поделить на K.

Ниже приведен код, реализующий данный алгоритм.

```
n = int(input())
k = int(input())
ans = 0
while n > 0:
    if n % k == 0:
        ans += 1
        n //= k
    else:
        ans += n % k
        n -= n % k
print(ans)
```

Задача 7. Удаление данных

Для решения первой подзадачи можно было написать простейший перебор с возвратом. Такое решение набирало не менее 20 баллов.

Для решения второй подзадачи требовалось заметить следующий факт: **выгодно удалять элемент, соседний с одним из минимальных элементов**.

Таким образом, решение второй подзадачи выглядит следующий образом:

- Если в массиве остался один элемент, завершаем алгоритм.
- Если в массиве хотя бы два элемента, находим минимальный. Пусть он равен a_{min} (если их несколько, выбирем любой), и удаляем одного из его соседей. При этом к ответу добавляется a_{min}.

Чтобы решить задачу на 100%, нужно было заметить следующий факт: **При удалении элемента, соседнего с минимальным, минимум массива не меняется.** Это значит, что на каждой итерации цикла из предыдущего листинга к ответу будет прибавляться одно и то же число, равное минимальному элементу изначального массива a. Поскольку всего будет ровно $n - 1$ интерация, то ответ равен $a_{min} \cdot (n - 1)$.

Таким образом, решение на 100 баллов имеет следующий вид:

```
n = int(input())

mn = 10 ** 9
for i in range(n):
        cur = int(input())
        mn = min(mn, cur)

print(mn * (n - 1))
```

Разбор задач

Задача 1. Починка блюдца

Так как изначальное блюдце имело размер $N \times N$, а размер частей блюдца равен $K \times K$, то всего частей получилось $\frac{N}{K} \times \frac{N}{K}$.

Рассмотрим два соседних горизонтальных ряда частей блюдца. Между ними нужно проклеить границу длины N, значит на это действие потребуется N банок клея. Всего таких соседних рядов будет $\frac{N}{K} - 1$. Значит всего потребуется $N \cdot \left(\frac{N}{K} - 1\right)$ банок клея.

Аналогично нужно рассмотреть соседние вертикальные ряды частей блюдца. Для склеивания их также потребуется $N \cdot \left(\frac{N}{K} - 1\right)$ банок клея.

Значит, всего потребуется $2N \cdot \left(\frac{N}{K} - 1\right)$ банок.

Ниже приведен пример решения задачи на языке Python.

```
n = int(input())
k = int(input())
print(2 * n * (n // k - 1))
```

Задача 2. Ну все, я попрыгал!

Так как по условию задачи нельзя делать два длинных прыжка подряд, будем чередовать прыжки: сначала сделаем длинный прыжок величиной Y, затем короткий величиной X, затем снова длинный, и так далее. Последний прыжок при этом может оказаться и меньше максимальной величины соответствующего прыжка. Нетрудно понять, что данный способ минимизирует количество прыжков.

Для получения частичных баллов по данной задаче можно было реализовать данный алгоритм при помощи простого цикла. Пример подобного решения на языке Python приведен ниже.

```
x = int(input())
y = int(input())
n = int(input())

ans = 0
while n > 0:
    if ans % 2 == 0:
        n -= y
    else:
        n -= x
    ans += 1

print(ans)
```

Однако, так как количество прыжков может быть достаточно большим, данное решение может работать достаточно долго.

Для решения задачи на полный балл нужно научиться быстро вычислять необходимое количество прыжков. Заметим, что до тех пор, пока оставшееся количество ступеней больше, чем $X + Y$, мы сможем сделать сначала длинный прыжок, а затем короткий. Таким образом, мы можем сразу добавить к ответу $2 \cdot \left\lfloor \frac{N}{X+Y} \right\rfloor$ прыжков. После этого останется пройти $N \mod (X + Y)$ ступеней. Здесь $\left\lfloor \frac{A}{B} \right\rfloor$ означает деление A на B с округлением вниз, а $A \mod B$ — взятия остатка от деления числа A на число B.

Теперь необходимо определить, сколько еще прыжков нужно сделать, чтобы пройти оставшиеся ступени. Возможны три варианта:

1. Осталось пройти 0 ступеней. В этом случае нет необходимости в дополнительных прыжках.
2. Осталось пройти не более Y ступеней. В этом случае к ответу нужно добавить один длинный прыжок.
3. Осталось пройти больше, чем Y ступеней. В этом случае нужно сначала сделать длинный прыжок, а затем короткий, то есть добавятся два прыжка.

Ниже приведен пример решения задачи на языке Python.

```
x = int(input())
y = int(input())
n = int(input())

ans = 2 * (n // (x + y))
n %= x + y

if n > 0:
    n -= y
    ans += 1
if n > 0:
    n -= x
    ans += 1

print(ans)
```

Задача 3. Андрей и порталы

Для начала заметим, что всегда можно дойти от изначальной позиции Андрея до места проведения олимпиады, не пользуясь порталами. Данное перемещение можно осуществить за $|s - e|$ секунд.

Теперь следует рассмотреть случай, когда выгодно воспользоваться порталами. Нетрудно понять, что в этом случае оптимальный алгоритм выглядит так:

1. Дойти от изначальной позиции до ближайшего к ней портала.
2. Телепортироваться в ближайший к месту проведения олимпиады портал.
3. Дойти от этого портала до места проведения олимпиады.

Таким образом, единственное, что нужно сделать — это найти ближайшие к изначальной позиции и месту проведения олимпиады порталы. Это можно сделать во время считывания данных. Будем поддерживать две переменные — расстояния до ближайших порталов к двум требуемым точкам. Во время считывания позиции очередного портала нужно проверить, находится он ближе к какой-либо из двух точек, чем ранее найденный портал, или нет.

Ниже приведен пример решения задачи на языке Python.

```
xs = int(input())
xe = int(input())
n = int(input())

INF = 10 ** 9
```

```
x1 = INF
x2 = INF
ans = abs(xs - xe)

for i in range(n):
    x = int(input())

    if abs(x - xs) < x1:
        x1 = abs(x - xs)
    if abs(x - xe) < x2:
        x2 = abs(x - xe)

ans = min(ans, 1 + x1 + x2)
print(ans)
```

Задача 4. Путешествие по джунглям

Для начала поймем, что если Коко может добраться до лианы с номером i, то она может добраться и до всех предыдущих лиан. Будем рассматривать лианы слева направо и поддерживать номер самой правой лианы, до которой сможет добраться горилла.

Обозначим за X максимальный номер лианы, до которой сможет добраться Коко. Изначально скажем, что $X = 1$, так как в начале Коко находится на лиане с номером 1. Пусть мы рассмотрели первые $i - 1$ лиан, и на данный момент известно, что Коко сможет добраться до лианы с номером X. Теперь рассмотрим лиану с номером i и поймем, насколько далеко горилла сможет прыгнуть, используя ее.

Если $X < i$, то Коко никак не сможет попасть на лиану с номером i, а значит она не сможет воспользоваться данной лианой. В этом случае ответ на задачу равен X.

Если же $X \geqslant i$, то можно попробовать воспользоваться текущей лианой. Известно, что с нее Коко может прыгнуть не далее, чем на a_i метров вправо. Так как расстояние между соседними лианами равно D, Коко сможет допрыгнуть с текущей лианы на лиану с номером $i + \left\lfloor \frac{a_i}{D} \right\rfloor$. Если данное число больше, чем X, то обновим значение X, ведь теперь горилла смогла добраться до лианы с бОльшим номером.

Ниже приведен пример решения задачи на языке Python.

```
n = int(input())
d = int(input())

X = 1
i = 1
while i <= n and i <= X:
    jump = int(input())
    X = max(X, i + jump // d)
    i += 1

X = min(X, n)
print(X)
```

Задача 5. Финансовая реформа

Используя ровно одну банкноту, можно набрать суммы $x, x + 1, \ldots, y$ бурлей. Теперь поймем, какие суммы можно набрать, используя ровно две банкноты. Минимальная такая сумма составит

$2 \cdot x$ рублей, а максимальная — $2 \cdot y$ бурлей. Также можно понять, что все суммы в отрезке $[2 \cdot x, 2 \cdot y]$ также можно набрать.

Пользуясь данным фактом можно понять, что, используя ровно k банкнот, можно набрать суммы в отрезке $[k \cdot x, k \cdot y]$ бурлей. Таким образом, множество всевозможных сумм, которые можно набрать, выглядит следующим образом: $[x, y] \cup [2 \cdot x, 2 \cdot y] \cup \ldots \cup [k \cdot x, k \cdot y] \cup \ldots$

Теперь нужно найти такое минимальное число N, начиная с которого можно набрать все суммы. Заметим, что если существуют два отрезка сумм $[k \cdot x, k \cdot y]$ и $[(k+1) \cdot x, (k+1) \cdot y]$, такие, что $k \cdot y + 1 \geqslant (k+1) \cdot x$, то начиная с суммы $k \cdot x$ можно набрать любую сумму. Таким образом, чтобы найти ответ, необходимо найти такое минимальное k, что $k \cdot y + 1 \geqslant (k+1) \cdot x$. Если такого k не существует, то в качестве ответа нужно вывести -1.

Для получения частичных баллов по данной задаче можно было искать необходимое k, перебрав всевозможные варианты циклом. Однако, необходимое k может быть достаточно большим, либо может не существовать вовсе.

Для того, чтобы найти требуемое k быстрее, нужно решить неравенство $k \cdot y + 1 \geqslant (k+1) \cdot x$. Для начала раскроем скобки и сгруппируем слагаемые: $k \cdot (y - x) \geqslant x - 1$. Таким образом, $k \geqslant \frac{x-1}{y-x}$. Так как нас интересуют только целые значения k, минимальным подходящим значением будет $\left\lceil \frac{x-1}{y-x} \right\rceil$. Здесь $\lceil \frac{A}{B} \rceil$ означает деление с округлением вверх.

Обратим внимание, что если $x = y$, то знаменатель дроби становится равным нулю. Данные случаи нужно обработать отдельно. Если $x = y = 1$, то, очевидно, можно набрать любые суммы, начиная с $N = 1$. Если же $x = y$ и $x > 1$, то требуемого N не существует: действительно, используя банкноты номинала x можно набрать только суммы, кратные x.

Для того, чтобы выполнить округление вверх, можно воспользоваться формулой: $\lceil \frac{A}{B} \rceil = \lfloor \frac{A+B-1}{B} \rfloor$.

Ниже приведен пример решения задачи на языке Python.

```
x = int(input())
y = int(input())

if x == y:
    if x == 1:
        print(1)
    else:
        print(-1)
else:
    A = x - 1
    B = y - x
    k = (A + B - 1) // B
    n = max(1, k * x)
    print(n)
```