Задача 1. Лука в кинотеатре

Для начала рассмотрим решение задачи при помощи перебора. Нам известно, что от начала показа фильмов прошло t минут. Показ первого фильма начинается в моменты времени $0, a, 2a, 3a, \ldots$, а показ второго фильма — в моменты времени $0, b, 2b, 3b, \ldots$. То есть, от нас требуется найти минимальный момент времени, не меньший t, который делится на a или b без остатка. Будем последовательно увеличивать t на 1 до тех пор, пока данное условие не будет выполнено.

Пример решения на языке Python:

```
a = int(input())
b = int(input())
t = int(input())

start = t
while start % a != 0 and start % b != 0:
    start += 1

print(start - t)
```

Такое решение имеет вычислительную сложность $\mathcal{O}(\min(a, b))$, что слишком много при больших значениях a и b.

Научимся быстро искать минимальное число, не меньшее t, которое делится на a. Для этого вычислим остаток от деления числа t на a. Пусть этот остаток равен r_a. Тогда минимальное число, не меньшее t и делящееся на a, равно $t + (a - r_a)$, за исключением случая, когда $r_a = 0$: тогда число t уже делится на a. Аналогично найдем ближайшее число, не меньшее t, которое делится на b. Теперь остается выбрать из этих двух чисел наименьшее и вычесть из него t — это и будет ответ на задачу.

Пример решения на языке Python:

```
a = int(input())
b = int(input())
t = int(input())

da = (a - t % a) % a
db = (b - t % b) % b

print(min(da, db))
```

Такое решение имеет вычислительную сложность $\mathcal{O}(1)$.

Задача 2. Лука покупает динозавров

Для начала вычислим, какое количество баллов придется накопить, чтобы набрать необходимые для приобретения всей коллекции $r \cdot l$ баллов, из которых у Луки уже имеется t. Таким образом, нужно получить еще $s = \max(0, t - r \cdot l)$ баллов.

Теперь заметим, что за каждые три последовательно купленных товара Лука получит фиксированное количество баллов, равное $p = p_1 + p_2 + p_3$. Следовательно, Луке сначала придется некоторое количество раз получить p баллов, покупая по три товара, а в конце, возможно, получить еще некоторое количество баллов, купив еще не более двух товаров в магазине. Количество операций начисления p баллов равно $\left\lfloor \frac{s}{p} \right\rfloor$, где запись $\left\lfloor \frac{x}{y} \right\rfloor$ обозначает частное от деления x на y, округленное вниз.

После этого количество баллов, которые останется набрать, окажется равно остатку от деления s на p. Рассмотрим следующие случаи:

1. Остаток от деления s на p равен нулю. В этом случае Луке больше не требуется совершать покупки. Если $p_3 = 0$ или $p_2 = p_3 = 0$, то последние одна или две покупки не являются необходимыми, поэтому нужно не забыть вычесть их количество из ответа.

2. Остаток от деления s на p не превосходит p_1. В этом случае Луке придется совершить еще одну покупку.

3. Остаток от деления s на p не превосходит $p_1 + p_2$. В этом случае Луке придется совершить еще две покупки.

4. Остаток от деления s на p больше, чем $p_1 + p_2$. В этом случае Луке придется совершить еще три покупки.

Пример решения на языке Python:

```
t = int(input())
l = int(input())
r = int(input())
p1 = int(input())
p2 = int(input())
p3 = int(input())

s = max(0, r * l - t)
p = p1 + p2 + p3

ans = s // p * 3
s %= p

if s == 0:
    if p3 == 0:
        ans -= 1
        if p2 == 0:
            ans -= 1
elif s <= p1:
    ans += 1
elif s <= p1 + p2:
    ans += 2
else:
    ans += 3

print(ans)
```

Вычислительная сложность данного решения: $\mathcal{O}(1)$.

Задача 3. Лука наблюдает за кузнечиками

Давайте сразу заметим, что кузнечик после некоторого числа прыжков окажется в точке старта тогда и только тогда, когда суммарное расстояние, которое он пропрыгал, кратно n.

В таком случае вопрос стоит переформулировать следующим образом: за какое минимальное ненулевое количество прыжков, каждый из который длины k или $k+1$ (в дальнейшем будем называть их короткими и длинными прыжками соответственно), можно пропрыгать суммарное расстояние, кратное n?

Полезным для решения задачи фактом, является то, что ответ не превышает n. Действительно, выберем один из двух видов прыжков, например короткий, и используем его n раз. Суммарное расстояние будет в точности kn, что кратно n, то есть кузнечик вернётся на то место, с которого начал. То есть n - это точно подходящее количество прыжков, следовательно ответ не превысит n.

Также вполне очевидно, что порядок прыжков не важен: если поменять местами какие-то прыжки, то суммарное расстояние от этого не изменится. Значит нам важно только количество коротких и длинных прыжков.

В решении на 60 баллов мы можем просто перебрать количество коротких и длинных прыжков, определить какие из них дают подходящее суммарное расстояние, и среди всего этого найти минимальное общее количество прыжков.

```
n = int(input())
k = int(input())

ans = n
for x in range(n + 1):
    for y in range(n + 1 - x):
        if x + y == 0:
            continue
        dist = x * k + y * (k + 1)
        if dist % n == 0 and x + y < ans:
            ans = x + y

print(ans)
```

Такое решение работает за $O(n^2)$

В качестве альтернативного решения на неполный балл переберем общее количество прыжков, а потом определим сколько из этих прыжков будут короткими; при этом оставшиеся будут длинными. Для каждого разбиения будем проверять, даёт ли оно подходящее суммарное расстояние, и если хотя бы одно разбиение подходит, то мы можем в этот же момент вывести ответ. Перебирая общее количество прыжков по возрастанию, мы знаем, что все меньшие ответы не подходят, а для текущего количества разбиение существует. Значит, это количество и есть минимальный ответ.

Такое решение подводит нас ближе к полному. Давайте тоже перебирать общее количество прыжков, но вместо того чтобы считать разбиения на короткие и длинные прыжки, просто посмотрим, какие суммарные расстояния могут получиться с таким количеством прыжков.

Пусть r – общее количество прыжков. Минимальным суммарным расстоянием будет rk – для этого нужно r раз использовать короткий прыжок. Выберем из этого разбиения любой короткий прыжок и заменим его на длинный. Суммарное расстояние станет равно $rk + 1$, при этом у нас останется еще $r - 1$ коротких прыжков. Проделав такую операцию ещё $r - 1$ раз мы получим максимальное по суммарному расстоянию разбиение, состоящее из r длинных прыжков. В процессе замен у нас получались все возможные расстояния от rk до $r(k+1)$. Отсюда сделаем вывод, что, используя r прыжков, можно получить любое расстояние на этом отрезке.

Осталось научиться определять есть ли на отрезке от rk до $r(k + 1)$ расстояния, кратные n. Заметим, что если какое-то расстояние s кратно n, то $\lfloor \frac{s-1}{n} \rfloor \neq \lfloor \frac{s}{n} \rfloor$. Значит, если на нашем отрезке нет расстояний, кратных n, то для всех s на нашем отрезке выполняется $\lfloor \frac{s-1}{n} \rfloor = \lfloor \frac{s}{n} \rfloor$. Так как числа такого вида возрастают, достаточно проверить, что $\lfloor \frac{rk-1}{n} \rfloor = \lfloor \frac{r(k+1)}{n} \rfloor$. Если это верно, то r не подходит в качестве ответа, и нужно перебирать его дальше; в ином случае мы выводим r и завершаем программу.

В коде это выглядит следующим образом:

```
n = int(input())
k = int(input())

for r in range(1, n + 1):
    if (r * k - 1) // n == (r * (k + 1)) // n:
        continue
    else:
        print(r)
        break
```

Задача 4. Лука и массив

Для начала рассмотрим случай для $k = 1$, тогда две операции стоят одинаковое количество единиц, но первая операция всегда как минимум уменьшает число на 1, значит используя только первую операцию можно свести число к 1.

При $1 \leqslant a_i \leqslant 10^5$ можно воспользоваться методом динамического программирования: база $dp[1] = 0$, переход $dp[i] = min(dp[i-1] + 1, dp[\lfloor \frac{a_i}{2} \rfloor] + \text{k})$.

Для решения на полный балл надо понять, что для получения $\lfloor \frac{a_i}{2} \rfloor$ из a_i есть два способа:

1. Воспользуемся первой операцией, тогда нужно использовать k энергии.
2. Воспользуемся второй операцией, тогда нужно использовать $a_i - \lfloor \frac{a_i}{2} \rfloor$ энергии.

Давайте на каждой итерации смотреть что выгоднее и уменьшать число a_i.

Пример решения на языке Python.

```
n = int(input())
k = int(input())

ans = 0
for i in range(n):
    cur = int(input())
    while cur >= k * 2:
        cur //= 2
        ans += k
    ans += cur - 1

print(ans)
```

Задача 5. Лука и локальная сеть динозавров

Решение заключается в том, чтобы отсортировать фигурки по координате x, в случае равенства сортируем по координате y. Теперь надо последовательно соединять динозавров, соседствующих в упорядоченном списке. Так как всего в списке n элементов, то проводов понадобится $n - 1$.

Пример решения на языке Python.

```
n = int(input())

points = []

for i in range(n):
    x = int(input())
    y = int(input())
    points.append((x, y, i + 1))

points.sort()
print(n - 1)
for i in range(n - 1):
    print(points[i][2], points[i + 1][2])
```