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)
```