Разбор задач

Задача 1. Биатлон

Подзадача 1 (30 баллов, $s \leqslant 10^5$)

Участник может перебирать x от 1 до s и проверять равенство $n * x + k * p == s$. Это даёт решение без использования формулы, но работает только для маленьких s.

Количество основных кругов $n = r + 1$ (традиционная схема: после каждого рубежа идёт круг, а после последнего рубежа — финишный круг).

```
r = int(input())
k = int(input())
p = int(input())
s = int(input())
n = r + 1
for x in range(1, s + 1):
    if n * x + k * p == s:
        print(x)
        break
```

Подзадача 2 (30 баллов, k = 0)

При отсутствии промахов формула упрощается: $x = s/n$. Достаточно вывести частное от деления.

```
n = r + 1
print(s // n)
```

Подзадача 3 (40 баллов, полные ограничения)

Используется прямая формула с целочисленным делением. Условие гарантирует, что $s - k * p$ положительно и кратно n.

Общая дистанция $n * x + k * p == s$, откуда

```
x = (s - k * p) // n
print(x)
```

Задача 2. Морская прогулка

Если прямоугольники пересекаются, то их пересечение также является прямоугольником. Найдём его стороны.

Для этого для каждого прямоугольника найдём координату x его левой стороны, а среди этих двух левых сторон найдём самую правую, обозначим её через $maxL$. Иными словами, $maxL = max(min(x_1, x_2), min(x_3, x_4))$.

Аналогично среди двух правых сторон прямоугольников найдём самую левую, обозначим её $minR$. $minR = min(max(x_1, x_2), max(x_3, x_4))$.

Если $maxR \leqslant minL$, то прямоугольники не пересекаются и ответ равен 0. Иначе ширина пересечения прямоугольников равна $maxR - minL$.

Аналогично следует поступить со второй стороной пересечения, найдя самую высокую из двух нижних сторон $maxD$ и самую низкую из двух самых высоких сторон этих прямоугольников $minU$. Снова проверим, что $minU - maxD > 0$, и если это так, то выведем произведение найденных сторон пересечения; иначе – 0.

Задача 3. Спонтанная покупка

Без ограничения общности $k_1 \leqslant k_2 \leqslant \cdots \leqslant k_n$, то есть отсортируем все значения k_i по возрастанию.

Давайте перебирать значения стоимости первого товара. Пусть первый элемент имеет индекс d (перебираем d от 1 до n). Будем поддерживать два счётчика, i и j, причём i — максимальный индекс в массиве, такой, что $k_d - k_i \leqslant t$, j — минимальный индекс в массиве, такой, что $k_j - k_d \leqslant t$. Тогда $-t \leqslant k_i - k_d \leqslant k_{i+1} - k_d \leqslant \cdots \leqslant k_{j-1} - k_d \leqslant k_j - k_d \leqslant t$, то есть для всех индексов u от i до j

выполняется равенство $|k_u - k_d| \leqslant t$, а это значит, что любой индекс, лежащий в промежутке от i до j, может являться вторым элементом (таких элементов всего $j-i+1-1 = j-i$), тогда все остальные элементы подходят в качестве третьих (таких элементов всего $n - (j - i + 1) = n - j + i - 1$). Тогда для каждого первого элемента найдём индексы i и j и в ответ добавим значение $(j-i)\cdot(n-j+i-1)$.

Как быстро искать i и j? С помощью метода двух указателей! Действительно, всегда выполняется условие $i \leqslant d \leqslant j$, а при увеличении d увеличиваются i и j, но никогда не уменьшаются.

Итоговая асимптотика: $O(n \log n)$ на сортировку массива, $O(n)$ — проход по массиву и запуск метода двух указателей, итого — $O(n \log n)$.

Задача 4. Бабушкин огород

В качестве наивного решения можно завести двумерный массив $n \times m$, и на каждом шаге явно отмечать в нём, какие клетки поливаются. Такой подход, конечно же, не проходит по времени для полного решения. Заметим, что после каждой операции в каждом столбце неполитые клетки идут подряд, начиная с верхней строки. Тогда можно завести массив $a[i]$, в котором число неполитых клеток находится в i-м столбце, считая слева (изначально $a[i] = n$ для всех $1 \leqslant i \leqslant m$).

Пусть на очередном шаге выбрали клетку (x, y). Это значит, что в столбце номер y либо стало $x - 1$ неполитых клеток, если выбранная клетка ещё не была полита (то есть если $a[y] \geqslant x$), либо их количество не изменилось. Таким образом, $a[y]$ меняется на минимум из $a[y]$ и $x - 1$. Кроме того, так же изменится и количество неполитых клеток во всех столбцах, лежащих правее: $a[j] = \min(a[j], x - 1)$ для всех $y \leqslant j \leqslant m$.

Осталось заметить, что изменения всех столбцов, лежащих правее выбранного столбца номер y, можно проводить не сразу же, а в самом конце. Действительно, обработаем все операции описанным образом (для каждой операции в клетку (x, y) поменяем значение $a[y]$), а затем пройдём по всем столбцам слева направо. Если мы знаем, сколько неполитых клеток есть в столбце номер j, то в столбце $j + 1$ их точно не больше. А значит, если мы знаем значение $a[j + 1]$ в текущий момент, то на самом деле неполитых клеток в этом столбце ровно $\min(a[j+1], a[j])$. Следовательно, достаточно пройти по массиву a, пересчитать все значения, а затем в качестве ответа вывести сумму всех его чисел.

Задача 5. Внешний долг

Посчитаем для каждого разряда количество уникальных цифр. Эта величина может принимать значения от 0 до 10, и для получения ответа достаточно просто перемножить их все (если на первом разряде можно различить a разных цифр, а на втором — b разных цифр, то ясно, что доступно ab последовательностей из двух знаков).

Заведём массив D, в котором будем хранить характер отображения цифр. Например, $D[0] = 1110111$, что означает, что для отображения нуля должны гореть все сегменты, кроме четвёртого; $D[1] = 0010010$ и так далее.

Итак, пусть есть разряд, работоспособность которого описывает строка s. Поймём, как в этом разряде будет отображаться цифра 0. Для этого достаточно поэлементно перемножить строки s и $D[0]$, ведь гореть будут только те сегменты, которые и работают, и используются в цифре 0. Обозначим строку, получившуюся в результате перемножения, через $a[0]$. Аналогично построим строки $a[1]$, $a[2]$, …, $a[9]$. Осталось только заметить, что уникально отображаться будут в точности те цифры, для которых среди $a[i]$ есть только один результат, ведь если $a[i] = a[j]$, то цифры i и j неотличимы. Таким образом, количество уникальных цифр равно количеству строк, которые встречаются среди $a[0], \ldots, a[9]$ ровно один раз. Посчитать это можно, например, используя словарь, то есть *dict* в питоне или *map* в C++.