Разбор задач

Задача 1. Прямые и окружности

Первая подзадача: $n = 0$ (нет прямых, только окружности). Каждая построенная окружность увеличивает количество разбиений плоскости на 1. Ответ – $m + 1$.

Вторая подзадача: $m = 0$ (нет окружностей, только прямые). Каждая построенная прямая увеличивает количество разбиений плоскости на 2. Ответ – $2 \times n$.

Полное решение: каждая новая прямая увеличивает количество разделений плоскости на 2 (кроме самой первой, которая увеличивает на 1), а каждая окружность добавляет ещё столько же областей, на сколько все прямые разделили плоскость. В общем случае будет $2 \times n \times (m + 1)$ областей, а при $n = 0$ – только $m + 1$.

```
n = int(input())
m = int(input())
if n == 0:
    ans = m + 1
else:
    ans = 2 * n * (m + 1)
print(ans)
```

Задача 2. Братья и сёстры

Первая подзадача: неполное решение (полный перебор по мальчикам и девочкам). При указанных ограничениях ($a, b \leqslant 100$) количество девочек y не может быть больше 101, а количество мальчиков x не превысит 10000. Переберём вложенным циклом все возможные значения пары x и y и проверим, выполняются ли условия задачи.

```
a = int(input())
b = int(input())
for x in range(1, 10001):
    for y in range(2, 102):
        if x % (y - 1) == 0 and x // (y - 1) == a:
            if (x - 1) % y == 0 and (x - 1) // y == b:
                print(x, y)
```

Улучшение решения: перебор только по мальчикам. Из первого утверждения следует, что у Ани братьев в a раз больше, чем сестёр, значит, если x – количество мальчиков, а y – количество девочек в семье, то $a = \frac{x}{y-1}$, $y - 1 = \frac{x}{a}$, что означает, что x кратно a. Будем перебирать количество мальчиков, для каждого такого количества вычислять количество девочек и определять, является ли оно корректным.

```
a = int(input())
b = int(input())
x = a
y = x // a + 1
while (x - 1) % y > 0 or (x - 1) // y != b:
    x += a
    y = x // a + 1
print(x)
print(y)
```

Полное решение:

Пусть x – количество мальчиков, а y – количество девочек. Тогда имеем систему уравнений:

$$\begin{cases} a = \frac{x}{y-1} \\ b = \frac{x-1}{y} \end{cases}$$

Тогда
$\begin{cases} a \times (y-1) = x \\ b \times y = x - 1 \end{cases}$
Подставим x во второе уравнение:
$b \times y = a \times y - a - 1$
Отсюда
$y \times (b-a) = -a - 1$
Окончательно:
$\begin{cases} y = \frac{a+1}{a-b} \\ x = a \times (y-1) \end{cases}$

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

Задача 3. Шестёрки

Частичное решение сводится к вычислению квадрата числа и поиску нужной позиции:

```
n = int(input())
k = int(input())
ans = str(int('6' * n) ** 2)[k - 1]
print(ans)
```

При больших значениях n вычисление результата будет производиться очень долго и может не уложиться в отведённую память.

Полное решение: давайте попробуем перемножить числа традиционным образом (в столбик) и найти закономерность.

При умножении числа, состоящего из n шестёрок, на одну число 6 получится число $39\ldots96$, длина которого равна $n+1$.

Теперь нам нужно n раз сложить эти числа со сдвигом (например, при $n = 4$):
...39996
..39996
.39996
39996

При этом возникает перенос единиц в следующий разряд, причём их переносится столько, сколько девяток складывается в очередном разряде. В результате получится число, у которого первые $n-1$ цифр – четвёрки, потом идёт одна цифра 3, ещё $n-1$ цифр 5, и на последней позиции стоит цифра 6. Это позволяет нам упростить алгоритм:

```
n = int(input())
k = int(input())
if k < n:
    print('4')
elif k == n:
    print('3')
elif k < 2 * n:
    print('5')
else:
    print('6')
```

Задача 4. Награждение участников олимпиады

Рассмотрим последовательность сумм, которые нужно потратить при последовательных различных разницах между стоимостью соседних призов.

Если разница равна 0, то нужно потратить:
$S_0 = 1 * a_n + 1 * a_{n-1} + \ldots + 1 * a_2 + 1 * a_1$ денежных единиц.
Если разница равна 1, то нужно потратить:
$S_1 = 1 * a_n + 2 * a_{n-1} + 3 * a_{n-2} + \ldots + (n-1) * a_2 + n * a_1$ денежных единиц.
Если разница равна 2, то нужно потратить:
$S_2 = 1 * a_n + 3 * a_{n-1} + 5 * a_{n-2} + \ldots + (2 * (n-1) - 1) * a_2 + (2 * n - 1) * a_1$ денежных единиц.
Если разница равна 3, то нужно потратить:
$S_3 = 1 * a_n + 4 * a_{n-1} + 7 * a_{n-2} + \ldots + (3 * (n-1) - 2) * a_2 + (3 * n - 2) * a_1$ денежных единиц.
Если разница равна 4, то нужно потратить:
$S_4 = 1 * a_n + 5 * a_{n-1} + 9 * a_{n-2} + \ldots + (4 * (n-1) - 3) * a_2 + (4 * n - 3) * a_1$ денежных единиц.
И так далее.

Первая группа тестов позволяет решить задачу просто подбором нужной разницы. При помощи двойного цикла построим набор получающихся сумм и найдём самую большую, не превосходящую заложенной сметы P.

Для полного решения нужно заметить, что последовательности этих сумм образуют арифметическую прогрессию. Действительно, если рассмотреть разницу между последовательными рядом стоящими суммами S_{i+1} и S_i, то видно, что она всегда равна $0 * a_n + 1 * a_{n-1} + 2 * a_{n-2} + \ldots + (n-2) * a_2 + (n-1) * a_1$, то есть разница постоянная. А значит, набор чисел $S_0, S_1, S_2 \ldots$ действительно образует арифметическую прогрессию. Тогда нужно найти разницу этой прогрессии, вычислив $d = S_1 - S_0$. Далее осталось найти самый большой номер элемента прогрессии, не превосходящего P. Для этого вычтем из P число S_0 и поделим нацело на d. Для этого решения даже нет необходимости сохранять исходные числа a_i в массив, достаточно вычислить S_0 и S_1 непосредственно при считывании чисел a_i.

Полное решение на языке Python:

```
n = int(input())
s0 = 0
s1 = 0
for i in range(n):
    x = int(input())
    s0 += x
    s1 += (n - i) * x
p = int(input())
d = s1 - s0
ans = (p - s0) // d
print(ans)
```

Задача 5. Фермер Джон и древний камень

1. При небольших ограничениях задачу можно решить полным перебором всех прямоугольников внутри поля и проверкой каждого на предмет наличия внутри него камня. Сложность такого решения $O(n^2 * m^2)$, если проверка вхождения камня происходит при помощи сравнения четырёх границ с координатами камня.

Далее любой прямоугольник будем обозначать парой его углов $(x_1, y_1) - (x_2, y_2)$, где (x_1, y_1) – координаты левого верхнего угла, а (x_2, y_2) – координаты правого нижнего угла.

2. Подсчитаем, сколько всего прямоугольников содержится внутри прямоугольника со сторонами $a \times b$. Для начала сделаем это со сложностью $O(a * b)$. Прямоугольников, у которых правый нижний угол находится в клетке (x, y), ровно $x * y$ штук. Перебирая все правые нижние углы и складывая эти слагаемые, получим общее число прямоугольников в любом большем прямоугольнике размера $a \times b$. Далее можно заметить, что любой прямоугольник, содержащий внутри себя камень, имеет левый верхний угол в прямоугольнике с углами $(1, 1) - (x, y)$, а правый нижний – в прямоугольнике

$(x, y) - (n, m)$. Тогда таких прямоугольников будет $x * y * (n - x + 1) * (m - y + 1)$. Используя этот факт, можно подсчитать число всех прямоугольников, и потом вычесть из него запрещённые. Если все прямоугольники внутри прямоугольника $n \times m$ считать по методу, указанному выше в этом пункте, получим сложность решения $O(n * m)$, что ещё не даёт полное решение.

3. Если идея с подсчётом и вычитанием запрещённых прямоугольников не пришла в голову, можно получить ответ включением-исключением, используя метод подсчёта всех прямоугольников внутри прямоугольника $a \times b$ получить ответ включением-исключением. Для этого выделим внутри большого прямоугольника $a \times b$ четыре максимальных по размеру и не содержащих клетку с камнем, подсчитаем число прямоугольников внутри каждого и вычтем те, что находятся в попарных пересечениях этих прямоугольников. Это решение тоже имеет сложность $O(n * m)$.

4. Теперь перейдём к полным решениям. Подсчитаем количество всех прямоугольников внутри прямоугольника $a \times b$ за $O(a)$. Сгруппируем их по правым нижним углам, расположенным в одной строке. Тогда количества прямоугольников, у которых правый нижний угол находится на первой строке, образуют прогрессию $1 + 2 + \ldots + b$; количества тех, у которых правый нижний угол во второй строке, образуют прогрессию $2 + 4 + \ldots + 2 * b$; ... количества тех, у которых правый нижний угол в строке номер a, образуют прогрессию $a + a * 2 + \ldots + a * b$. Используем для каждой такой прогрессии формулу суммы и переберём их по всем строкам. Получим ответ на подзадачу о числе всех прямоугольников. Теперь, если воспользоваться идеей из пункта 2 (вычтя запрещённые $(n - x + 1) * (m - y + 1)$ из общего числа в исходном прямоугольнике $n \times m$), либо идеей из пункта 3 (подсчёт не содержащих запрещённую клетку методом включения-исключения), то получим решение за $O(n)$.

5. Наконец, покажем как решить за $O(1)$ при помощи формулы. Если просуммировать каждую из прогрессий в предыдущем пункте, получим следующий набор: $(1 + b) * b/2, (2 + 2 * b) * b/2, \ldots (a + a * b) * b/2$. Их можно просуммировать и без цикла. Вынесем $(1 + b) * b/2$ за скобку, внутри получим $(1 + 2 + \ldots + a)$, то есть снова прогрессию. Тогда формула для подсчёта всех прямоугольников внутри прямоугольника $a \times b$ имеет вид $(b + 1) * b/2 * (a + 1) * a/2$.

Используя эту формулу и идею из пункта 2, получим формулу $(m + 1) * m/2 * (n + 1) * n/2 - x * (n - x + 1) * y * (m - y + 1)$.

Используя эту формулу и идею из пункта 3, получим ответ при помощи включения-исключения.

Полное решение на языке Python:

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

ans = (m + 1) * m // 2 * (n + 1) * n // 2
ans -= x * y * (n - x + 1) * (m - y + 1)

print(ans)
```