Оглавление

Сумма произведений

По данному натуральному числу n вычислите сумму: $1 \cdot 2 + 2 \cdot 3 + \ldots + (n-1) \cdot n$.

Проходим циклом от $i = 1$ до $n - 1$, накапливаем сумму произведений $i \cdot (i + 1)$. Выводим выражение в нужном формате: для каждого i печатаем $i \times (i + 1)$ с плюсами между слагаемыми, кроме последнего. В конце выводим знак равенства и сумму.

Шахматы

Даны координаты белого коня и черной пешки. Определить, может ли конь побить пешку за один ход.

Конь ходит буквой "Г": на две клетки по одной координате и на одну по другой. Проверяем все 8 возможных ходов коня: $(x_1 \pm 1, y_1 \pm 2)$ и $(x_1 \pm 2, y_1 \pm 1)$. Если какая-то из этих клеток совпадает с (x_2, y_2), то ответ YES, иначе NO.

Кинотеатр

В нечётных рядах N мест, в чётных — $N+1$ место. Билеты продаются подряд по рядам. По номеру билета K найти ряд и место.

Один цикл из двух рядов (нечётный и чётный) содержит $N + (N + 1) = 2N + 1$ мест. Определяем, сколько полных циклов прошло: $p = \lfloor K/(2N+1) \rfloor$. Остаток $c = K\%(2N+1)$ показывает положение внутри текущей пары рядов.

Если $c = 0$, значит билет последний в паре — это последнее место чётного ряда: ряд $2p$, место $N + 1$.

Если $1 \le c \le N$, то это нечётный ряд: ряд $2p + 1$, место c.

Если $N + 1 \le c \le 2N$, то это чётный ряд: ряд $2p + 2$, место $c - N$.

Подарки

Есть A упаковок по 1 конфете и B упаковок по 3 конфеты. В каждый подарок нужно положить ровно N конфет. Упаковки по 3 нельзя вскрывать. Найти максимальное число подарков.

Сначала посчитаем, сколько подарков можно было бы сделать, если бы не было ограничения на вскрытие троек: $total = \lfloor (A + 3B)/N \rfloor$. Но реальное число может быть меньше, если для заполнения подарков не хватает единичных конфет.

Рассмотрим остаток $r = N\%3$. Если $r = 0$, то каждый подарок можно собрать только из троек, и ограничение не мешает — ответ $total$.

Если $r > 0$, то для каждого подарка нужно r единичных конфет (остальное из троек). Всего на $total$ подарков нужно $total \cdot r$ единичных конфет. Если $A \ge total \cdot r$, то ответ $total$, иначе ответ $\lfloor A/r \rfloor$ (столько подарков можно собрать, используя все единичные).

Пробежка - легкая версия

Каждый день Рита получает A минут заряда (неиспользованный заряд копится). В i-й день она бежит $B + i - 1$ минут. Найти первый день, когда накопленного заряда не хватит на пробежку.

Моделируем процесс: в день i имеем накопленный заряд Z (изначально 0). Сначала добавляем A (зарядка), затем тратим $B + i - 1$ на пробежку. Если после пробежки заряд становится отрицательным, то этот день и есть ответ.

Так как $A, B \le 100$, можно просто симулировать.

Пробежка - сложная версия

Каждый день добавляется A минут заряда, в i-й день тратится $B + i - 1$ минут. Накопленный заряд неограничен. Найти первый день, когда заряда не хватит.

Пусть $d = A - B$. Если $d < 0$, то в первый же день заряда меньше потребности, ответ 1.

Если $d \geq 0$, рассмотрим накопленный заряд к началу дня k. Он равен сумме добавок за предыдущие дни минус сумма трат:

$$S_{k-1} = (k-1)A - \sum_{i=1}^{k-1}(B + i - 1) = (k-1)d - \frac{(k-2)(k-1)}{2}.$$

Заряда хватит в день k, если $S_{k-1} \geq 0$. Условие $S_{k-1} \geq 0$ даёт

$$(k-1)d \geq \frac{(k-2)(k-1)}{2} \quad \Rightarrow \quad d \geq \frac{k-2}{2} \quad \Rightarrow \quad k \leq 2d + 2.$$

Значит, при $k = 2d + 2$ заряд в начале дня равен 0, а потребление $B + 2d + 1 > 0$, поэтому в этот день уже не хватит. Таким образом, первый день, когда не хватит — это $k = 2d + 2 = 2(A - B) + 2$.

Лес - легкая версия

Миша ходит по циклу: A на север, B на восток, C на юг, D на запад. Старт в центре квадрата $[-K, K] \times [-K, K]$. Выход при достижении границы ($|x| = K$ или $|y| = K$). Найти число шагов до первого выхода.

Моделируем движение по шагам. На каждом шаге проверяем, не вышли ли за границы. Как только вышли, определяем, на каком именно шаге это произошло, и вычитаем лишние шаги, сделанные после пересечения границы в этом направлении.

Так как $A, B, C, D, K \leq 100$, полная симуляция допустима.

Лес - сложная версия

Миша ходит по циклу: A на север, B на восток, C на юг, D на запад. Старт в центре квадрата $[-K, K] \times [-K, K]$. Выход при достижении границы ($|x| = K$ или $|y| = K$). Найти число шагов до первого выхода.

Рассмотрим функцию $f(x, y)$, которая моделирует один цикл из точки (x, y) и определяет, произойдёт ли выход на каком-либо этапе, и если да, то через сколько шагов. Если за цикл выход не случился, возвращаем -1. В цикле последовательно выполняются:

- Север: y увеличивается на A. Выход на северной границе при $y + t = K$, номер шага t.
- Восток: x увеличивается на B. Выход на восточной границе при $x + t = K$, номер шага $A + t$.
- Юг: y уменьшается на C. Выход на южной границе при $y - t = -K$, номер шага $A + B + t$.
- Запад: x уменьшается на D. Выход на западной границе при $x - t = -K$, номер шага $A + B + C + t$.

Условия проверяются именно в таком порядке, при этом координаты обновляются после каждого этапа для последующих проверок.

Если после полного цикла выход не произошёл, то новое положение — $(x+B-D,\ y+A-C)$. Обозначим $dx = B - D$, $dy = A - C$.

Если $f(0, 0) \neq -1$, ответ сразу $f(0, 0)$. Иначе ищем минимальное целое $c \geq 0$, при котором $f(c \cdot dx,\ c \cdot dy) \neq -1$. Так как гарантируется, что выход когда-нибудь произойдёт, такое c существует. Находим его бинарным поиском. Тогда ответ:

$$c \cdot (A + B + C + D) + f(c \cdot dx,\ c \cdot dy).$$

```python
a, b, c, d, k = map(int, [input() for _ in range(5)])
def f(x, y):
    if abs(x) > k or abs(y) > k:
        return 0
    y += a
    if y >= k:
        return a - (y - k)
    x += b
    if x >= k:
        return a + b - (x - k)
    y -= c
    if y <= -k:
        return a + b + c + (y + k)
    x -= d
    if x <= -k:
        return a + b + c + d + (x + k)
    return -1
if f(0, 0) != -1:
    print(f(0, 0))
else:
    dx, dy = b - d, a - c
    L, R = 0, 10**9
    while L + 1 != R:
        mid = (L + R) // 2
        if f(dx * mid, dy * mid) == -1:
            L = mid
        else:
            R = mid
    print(R * (a + b + c + d) + f(dx * R, dy * R))
```

Не был предателем... - легкая версия

Есть ключ T, строка S и число N. Нужно найти количество вхождений T как подстроки в строку, полученную N-кратным повторением S.

Строка S повторяется N раз. Вхождения T могут полностью лежать внутри одного повторения S или пересекать границы между соседними повторениями.

Можно просто сконструировать строку $R = S$ повторённую N раз и стандартным алгоритмом (например, через ‘string::find‘ в цикле) подсчитать все вхождения, включая пересекающиеся.

Так как $|S| \leq 100$, $|T| \leq 100$, $N \leq 500$, длина итоговой строки не превышает 50000, что вполне допустимо для простого поиска.

Не был предателем... - сложная версия

Есть ключ T, строка S и число N. Нужно найти количество вхождений T как подстроки в строку, полученную N-кратным повторением S.

Все возможные вхождения T в повторяющуюся строку можно разбить на классы по остатку от деления начальной позиции на $|S|$. Для каждого остатка p (от 0 до $|S|-1$) проверяем, совпадает ли T с символами S, начиная с позиции p и далее по циклу (с переходом на следующий повтор S). Если совпадает, то такие вхождения будут повторяться каждые $|S|$ символов.

Пусть $L = |S| \cdot N$ — длина всей строки, $len = |T|$. Последняя возможная стартовая позиция вхождения это $L - len$. Для данного p первое вхождение начинается в позиции p, следующие — $p + |S|$, $p + 2|S|$, ... пока не превысим $L - len$. Количество таких вхождений равно $\left\lfloor \frac{L-len-p}{|S|} \right\rfloor + 1$, если $p \leq L - len$, иначе 0.

Суммируем по всем p, для которых T совпадает с циклическим сдвигом S.