I. Краткие рекомендации по решению задач, примеры решений

В состав пакета для каждой задачи входят решения (в электронном виде) на языках программирования Python и Си++.

Ниже представлены краткие рекомендации к решению задач.

Задача 1. Параллельный перенос квадрата
Задача на вывод формулы.

Частичное решение (***x*** = 0):

При данном ограничении получаем след в виде прямоугольника с шириной ***n*** и высотой ***n*** + ***y***. Площадь вычисляется по формуле ***n*** * (***n*** + ***y***).

Полное решение: достроим получившийся шестиугольник до прямоугольника. Тогда площадь нужной фигуры определится по формуле:

(***n*** + ***x***) * (***n*** + ***y***) - $\frac{x \cdot y}{2}$ - $\frac{x \cdot y}{2}$ = n^2 + ***n*** * ***x*** + ***n*** * ***y*** = ***n*** * (***n*** + ***x*** + ***y***).

Числа ***x*** и ***y*** необходимо предварительно взять по модулю каждое.

Пример программы (Python):

```
n = int(input())
x = abs(int(input()))
y = abs(int(input()))
print(n * (n + x + y))
```

Задача 2. Компьютерный класс.
Задача на целочисленное деление, вывод формулы.

Частичное решение (для небольших ***n***): полный перебор. Пусть первое значение ***b*** = 1, тогда, пока на длинной стороне ***a*** = ***n*** - 2 * ***b*** компьютеров больше, чем на короткой, увеличиваем число подходящих расстановок и переходим к следующему ***b***.

```
n = int(input())
ans = 0
b = 1
while b < n - 2 * b:
  ans += 1
  b += 1
print(ans)
```

Полное решение: пусть наибольшее количество компьютеров, которые можно расставить на участок ***b***, равно ***k***. Тогда ***a*** <= ***k*** + 1 и 3 * ***k*** + 1 <= ***n***. Отсюда ***k*** = $\frac{n-1}{3}$ (с округлением вниз).

Пример программы:

```
n = int(input())
ans = (n - 1) // 3
print(ans)
```

Задача 3. Юбилейные числа.
Задача на сокращение перебора.

Частичное решение (для небольших ***n***): полный перебор. У первого юбилейного числа значение длины числа ***length*** = 1 и количество пятёрок ***a*** в ней тоже 1. Устанавливаем счетчик ***count*** на 1. Пока счетчик меньше ***n***, переходим к следующему юбилейному числу.

```
n = int(input())
count = 0
length = 1
a = 0
while count < n:
```

```
        count += 1
        a += 1
        if a == length:
            length += 1
            a = 1
    print(a, length - a)
```

Полное решение: нетрудно заметить, что существует ровно p - 1 юбилейных чисел длины p ($p > 1$).

Тогда имеем:

1 двузначное число (50);

2 трехзначных чисел (500 и 550);

3 четырёхзначных чисел (5000, 5500 и 5550) и так далее.

Попробуем определить длину n-го юбилейного числа. Для этого придется определить такое x, что сумма $1 + 2 + 3 + ... + x < n$.

Слева арифметическая прогрессия с суммой членов $\frac{x \cdot (x+1)}{2}$. Требуемый x можно определить бинарным поиском по ответу или через корни квадратного уравнения.

Тогда определена длина n -го юбилейного числа, она равна $x + 1$. Причем нужное нам число будет $a = n - \frac{x \cdot (x+1)}{2}$ по счету среди всех чисел этой длины, что соответствует количеству пятёрок. Нулей же останется $b = x + 2 - a$.

Пример программы (Python):

```
n = int(input())
left = 0
right = 10 ** 10
while left < right - 1:
    middle = (left + right) // 2
    if middle * (middle + 1) // 2 < n:
        left = middle
    else:
        right = middle
a = n - left * (left + 1) // 2
b = left + 2 - a
print(a, b)
```

Задача 4. Сумма квадратов

Задача на массивы. Два указателя.

Разберем частный случай для небольших n. Создадим массив квадратов чисел L = [1, 4, 9, 16, 25, ...], не превышающих заданное число. Переберём в нём все возможные различные пары чисел, найдём их сумму и, если она равна n, увеличим счетчик ответов на 1.

Пример программы (Python):

```
n = int(input())
L = [1]
k = 2
while k ** 2 <= n:
    L.append(k ** 2)
    k += 1
ans = 0
for i in range(len(L) - 1):
    for j in range(i + 1, len(L)):
        if L[i] + L[j] == n:
            ans += 1
print(ans)
```

Полное решение. Для такого же массива квадратов чисел применим поиск по методу двух указателей. Понятно, что для каждого числа в массиве существует не более одного другого подходящего элемента массива, поэтому для упорядоченного списка чисел нет необходимости перебирать все вторые слагаемые большие того, для которых сумма оказалась заведомо больше требуемой. Установим левый указатель на первый элемент массива, правый – на последний и будем действовать по следующему плану, пока левый указатель левее правого:

1) сумма чисел, на которые указывают указатели равна ***n***. Увеличиваем счетчик ответов на 1 и сдвигаем левый указатель на 1 позицию вправо, а правый – на одну позицию влево;
2) сумма чисел, на которые указывают указатели больше ***n***. Для числа, на который указывает левый указатель в нашем массиве точно нет нужного второго слагаемого, больших, чем тот, на который указывает правый указатель (но, возможно, он есть среди меньших). Сдвигаем правый указатель на 1 позицию влево;
3) сумма чисел, на которые указывают указатели меньше ***n***. Для числа, на который указывает правый указатель в нашем массиве точно нет нужного первого слагаемого, меньших, чем тот, на который указывает левый указатель (но, возможно, он есть среди больших). Сдвигаем левый указатель на 1 позицию вправо.

Решение получим за один проход по массиву.

Пример программы (Python):

```
n = int(input())
L = [1]
k = 2
while k ** 2 <= n:
    L.append(k ** 2)
    k += 1
ans = 0
count_1 = 0
count_2 = len(L) - 1
ans = 0
while count_1 < count_2:
    if L[count_1] + L[count_2] == n:
        ans += 1
        count_2 -= 1
        count_1 += 1
    elif L[count_1] + L[count_2] > n:
        count_2 -= 1
    else:
        count_1 += 1
print(ans)
```

Ещё одним способом решения является использование структуры данных «множество». В этом случае мы для каждого элемента множества квадратов чисел можем просто проверить, имеется ли в этом же множестве недостающая до ***n*** сумма.

Пример программы (Python):

```
n = int(input())
S = set()
k = 1
while k ** 2 <= n:
    S.add(k ** 2)
    k += 1
ans = 0
for i in S:
    if n - i != i and n - i in S:
```

```
        ans += 1
print(ans // 2)
```

Задача 5. Идеальная пара.
Задача на теорию чисел, факторизация.

Частичное решение: перебор всех чисел, начиная с единицы с последующей проверкой на «идеальность».

Пример программы (Python):

```
def ideal(n, m):
    if n != m:
        if n % 2 == m % 2:
            p = (n * m) ** 0.5
            if int(p) ** 2 == n * m or int(p + 1) ** 2 == n *
m:
                return True
    return False

n = int(input())
m = 1
while not(ideal(n, m)):
    m += 1
print(m)
```

Полное решение. Для того, чтобы число ***m*** образовывало с числом ***n*** идеальную пару, необходимо, чтобы в разложении их произведения на простые множители получались только чётные степени (только тогда среднее геометрическое $\sqrt{n \cdot m}$ будет целым). Значит, ***m*** равно произведению всех простых множителей в разложении ***n***, показатель степени у которых нечетный.

Например, для ***n*** = 360 = $2^3 * 3^2 * 5$ число ***m*** должно быть равно 2 * 5 = 10.

Число ***m*** можно найти из факторизации (разложения на простые множители) ***n*** при помощи перебора до корня.

Если при этом числа ***m*** и ***n*** одной четности, то наша работа завершена~-- их среднее арифметическое тоже будет целым. Но так бывает не всегда, например, при ***n*** = 16 получаем m = 1.

- Если ***m*** нечётно, то придется его домножить на минимальное чётное число-квадрат, то есть, на 4 (при этом нужно проверить, чтобы ***m*** и ***n*** оставались различными). Если ***m*** и ***n*** совпали, придётся ***m*** увеличить ещё раз (например, при ***n*** = 4).
- Если же ***m*** чётно, то придется его домножить на минимальное нечётное число-квадрат, отличное от 1, то есть, на 9.

Пример программы (Python):

```
n = int(input())
n1 = n
L = []
i = 2
while i <= n1:
    while n1 % i == 0:
        if L == [] or i != L[-1]:
            L.append(i)
        else:
            L.pop()
        n1 //= i
    i += 1
    if i ** 2 > n or i > n1:
```

```
        L.append(n1)
        break
ans = 1
for i in L:
    ans *= i
if n % 2 == 1 and n == ans:
    ans *= 9
elif n % 2 == 0 and ans % 2 == 1:
    ans *= 4
    if ans == n:
        ans *= 4
elif n % 2 == 0 and n == ans:
    ans *= 4
print(ans)
```