

Задача А. Стенания Сипчика

1 Решение на 40 баллов

Для решения задачи на 40 баллов можно написать два цикла и честно «построить стену».

2 Полное решение

Для полного решения заметим, что первый и четвертый, седьмой и так далее ряды (если они есть) будут повторяться, так как каждый ряд смещается на треть кирпича, поэтому достаточно посчитать их количество только для первых трех рядов.

Для начала посчитаем количество кирпичей, которые мы вообще не будем ломать при строительстве стены. В первом ряду их будет $\lfloor \frac{w}{s} \rfloor$, во втором ряду — $\lfloor \frac{w-\frac{s}{3}}{s} \rfloor$, а в третьем — $\lfloor \frac{w-2\frac{s}{3}}{s} \rfloor$. Четвертый и последующие ряды будут повторяться, поэтому достаточно считать только для первых трех. Аналогично можно посчитать остаток, который необходимо заполнить кусками.

Чтобы каждый раз не делить на три, можно ширину стены и кирпича изначально домножить на три, на ответ это никак не повлияет.

Таким образом считаем количество целых кирпичей в стене, также считаем сколько нужно заполнить кусками, делим на ширину кирпича и получаем ответ на задачу.

Так как кирпичи можно ломать сколько угодно, то в целом не важно сколько кирпичей мы вообще ломать не будем, можно посчитать суммарную длину рядов и поделить её на ширину кирпича, то есть $\lfloor \frac{n*W}{s} \rfloor$.

Задача В. Игры с матрицей

Можно заметить, что по условию размеры матриц небольшие, как и количество возможных операций, поэтому переберем все возможные варианты. Для простоты лучше реализовать каждую из операций в виде функции, которая принимает матрицу и возвращает новую матрицу. Также заметим, что вращать матрицу можно только в одну сторону, например против часовой стрелки, так как поворот на 90 градусов против часовой стрелки — тоже самое, что поворот по часовой на 270 градусов. Кроме того, поворот на 180 градусов против часовой стрелки — это два поворота на 90 градусов. Соответственно, нам нужно уметь поворачивать матрицу на 90 градусов против часовой стрелки, этим мы покроем все возможные вращения.

```
def rotate90(a, n):
    b = [[0] * n for i in range(n)]
    for i in range(n):
        for j in range(n):
            b[n - j - 1][i] = a[i][j]
    return b
```

При зеркальном отражении матрицы относительно главной диагонали (транспонировании) индексы элементов меняются местами:

```
def T(a, n):
    b = [[0] * n for i in range(n)]
    for i in range(n):
        for j in range(n):
            b[j][i] = a[i][j]
    return b
```

А отражение относительно побочной диагонали можно выразить через повороты и транспонирование: повернем матрицу на 90 градусов против часовой стрелки, транспонируем ее и повернем еще на 270 градусов.

Для полного решения задачи переберем все возможные пары операций и применим их к первой матрице, если результат в какой-то момент будет равен второй матрице — ответ на задачу «YES», если после полного перебора условие равенства не выполнилось — ответ «NO».

Задача С. Сипчик тусует буквы

3 Решение на 30 баллов

Для решения задачи в данных ограничениях достаточно отсортировать список букв любимого слова Сипчика `tlist.sort()`, и сравнить данный список с сортированным списком букв среза соответствующей длины, с началом в позиции $i - list(s[i : i + tlen])$ для всех позиций i .

4 Полное решение

Посчитаем cnt — «паспорт» строки t , т.е. количество вхождений каждой буквы алфавита. Чтобы избавиться от лишних вычитаний будем в качестве индекса в списке cnt использовать непосредственно ASCII код символа — $ord(t[i])$. Список будет чуть большего размера 128 вместо 26, но код упростится.

Попутно посчитаем количество различных символов в строке t — cnt_dif_t .

Будем двигаться по строке s . Очередной символ $s[i]$ влияет только на одно значение в списке cnt , т.е. лишь одно значение может совпасть с частотой в t при добавлении $s[i]$ либо наоборот перестать совпадать. Это позволяет пересчитывать количество совпадающих счетчиков в окне ширины $|t|$ в строке s за $O(1)$ операций.

Более подробно алгоритм следующий:

- Изначально $cur_cnt_dif = cnt_dif_t$.
- Двигаемся по строке.
- Считаем код очередного символа $c = ord(s[i])$.
- Если до этого $cnt[c]$ был 0, то баланс нарушится, $cur_cnt_dif + = 1$.
- Изменяем баланс счетчика $cnt[c] - = 1$
- Если сейчас $cnt[c]$ стал 0, то баланс восстановился, $cur_cnt_dif - = 1$.
- Первые $|t|$ символов только добавляются, как только будут набраны $|t|$ символов проверяем cur_cnt_dif на равенство 0 и обновляем ответ.
- Как только будут набраны $|t| + 1$ символ нужно будет по описанной выше схеме убрать из окна начальный символ, т.е. $s[i - |t|]$, пересчитав cur_cnt_dif .

Задача D. Формульное безумие

5 Решение для $n < 10^2$

Решение за $O(N^3)$, где мы перебираем 3-мя форами все переменные получало от 4 до 8 баллов

6 Решение для $n < 10^3$

Решение за $O(N^2)$, где мы перебираем 2-мя форами переменные y и z , а переменную x проверяем подходит ли она получало до 20 баллов.

7 Решение для $n < 10^4$

Предыдущее решение можно было оптимизировать: найти максимальные значения y и z , а так же найти нужные границы по четности, то такое решение получало до 30 баллов.

8 Решение для $n < 10^8$

Предположим что мы набрали какое-то число мест по 5, тогда нам требуется посчитать количество разбиений остатка на места по 2 и 4 места. Допустим этот остаток четный (иначе бы мы не смогли его разбить), тогда количество мест по 4 варьируется от 0 до $\lfloor \frac{ost}{4} \rfloor$, где ost это сколько мест осталось распределить, а значит количество вариантов будет $\lfloor \frac{ost}{4} \rfloor + 1$.

Тогда мы получаем решение за $O(N)$, перебираем коэффициент при z и считаем кол-во вариантов x и y с помощью формулы. Если аккуратно определить границы и шаг в цикле, то такое решение набирает 40 баллов.

9 Полное решение

Из предыдущего решения заметим, что при увеличении коэффициента при z на 4, количество способов разбить остаток изменяется на 5, но если z изменить на 2, то количество способов меняется периодически, сначала на 2, потом на 3, снова на 2 и т.д. Такое можно свести к двум арифметическим прогрессиям с шагом 4, посчитать их сумму и получить итоговый ответ.

Получаем решение за $O(1)$

Задача Е. Игра на сплочение

10 Решение для $r < 10^6$

За линию просчитаем сколько каких действий было выполнено. Сумма цифр кратна 3 это признак делимости на 3, что упрощает решение.

11 Решение для $l = 1, r = 10^x$

Количество хлопков и приседаний от 1 до r это $r/31$ и $r/3$ соответственно. Осталось посчитать количество чисел с 3 в записи.

Заметим, что количество чисел в записи которых есть хотя бы одна 3 это общее количество чисел минус количество чисел в записи которых нет 3. Это сильно упрощает нашу задачу, тогда для $r = 10$ ответ будет $10 - 9^1$, для $r = 100$ ответ $100 - 9^2$ и т.д.

12 Полное решение

Допустим у нас есть функция f которая умеет считать ответ от 1 до x , тогда ответ от l до r будет равен $f(r) - f(l - 1)$. Так же заметим, что ответ для числа без 3 в записи, например 456, это $f(456) = f(400) + f(50) + f(6)$, но если есть 3 в записи, то ответ будет равен такому: $f(1332) = f(1000) + f(300) + 32$, так как после первой 3 в числе нам подходит любой суффикс меньший нашего.

Осталось научиться считать f для чисел вида $y \cdot 10^x$, что на самом деле довольно просто, так как для 10^x мы уже знаем ответ из предыдущей подгруппы. Если $y < 3$, то к ответу надо прибавить $y \cdot f(10^x)$, если $y = 3$, то к ответу надо прибавить $y \cdot f(10^x) + 1$, иначе ответ увеличится на $(y - 1) \cdot f(10^x) + 10^x$.

Задача Ф. Сипчик под лавандовым рафом

13 Решение на 10 баллов

Для решения задачи на 10 баллов нужно было реализовать логику обработки покупателя на кассе, когда ее номер уже известен.

Для проверки товара на обработку по одной единице или с помощью количества единицведем список *oneonly* на $10^5 + 1$ элементов, чтобы отметить идентификаторы товаров, которые надо пробивать по одному. Тогда проверка товара на способ обработки будет осуществляться за $O(1)$, достаточно посмотреть на значение *oneonly[id]*.

По заданному списку товаров покупателя проверим очередной товар *good_id, good_amount* на способ обработки. Вслучае обработки по одному обработка займет *good_amount* времени(трудозатрат), иначе *len(str(good_amount))* времени(трудозатрат).

Обработав весь список товаров покупателя вычислим время(трудозатраты), которое кассир потратит на обработку его товаров — *customersgoodsduration[i]*.

Для случая одной кассы трудозатраты это сумма трудозатрат на каждого покупателя, а времена выхода вычисляются последовательным проходом по списку покупателей.

14 Решение на 20 баллов

Воспользуемся алгоритмом вычисления трудозатрат(времени) кассира на обслуживание каждого покупателя — *customersgoodsduration[i]* из решения на 10 баллов.

Обрабатываем покупателей, стоящих в очереди, кассы всего две. Будем поддерживать *timefree[0], timefree[1]*. Время, когда освободится касса первая и вторая касса, соответственно.

Очередной покупатель номер *i* займет кассу на *customersgoodsduration[i] + 1* времени(1 на оплату). То есть если покупатель отправился на кассу *j*, то она освободится в *timefree[j] + customersgoodsduration[i] + 1*, а кассир кассы *j* увеличит трудозатраты на *customersgoodsduration[i]*.

Выбор кассы *j* это индекс минимального из двух значений *timefree[0]* и *timefree[1]*, в случае равенства выбираем *j = 0*.

15 Полное решение

В полном решении номер кассы выбирается с помощью *min* по массиву времен освобождения касс *timefree*. Все остальные элементы решения уже рассмотрены в предыдущих подзадачах.