# Задача A. Хорошие-хорошие подотрезки

*Автор и разработчик задачи: Илья Хлюстов*

Для начала научимся быстро определять, что отрезок является хорошим. Отрезок $[l; r]$ является хорошим, если $pref_r - pref_{l-1} = 0$, где pref – массив префиксных сумм, а $l - 1$ в случае, если $l$ — первый элемент, указывает на нулевой элемент.

Теперь сформируем критерий хорошего-хорошего отрезка. Отрезок $[l; r]$ не является хорошим-хорошим, если на отрезке $[l - 1; r]$ все префиксные суммы уникальны, то есть не существует двух индексов $i \neq j$, что $pref_i = pref_j$. В таком случае у нас не найдётся ни одного подотрезка с нулевой суммой.

После того, как мы сформулировали критерий хорошего-хорошего отрезка, необходимо определиться с алгоритмом. Для каждого индекса $r$ будем искать наименьший индекс $l$ такой, что $[l; r]$ – не хороший-хороший отрезок. Очевидно, что для любого $l \leqslant l' \leqslant r$ отрезок $[l'; r]$ так же не является хорошим-хорошим, а для любого $l'' \leqslant l$, $[l''; r]$ – хороший-хороший отрезок, соответственно мы можем прибавить к ответу количество таких $l''$. Осталось разобраться с тем, как искать $l$, ключевым замечанием тут является то, что для любых двух искомых отрезков $[l_1; r_1], [l_2; r_2]$ будет выполняться следующее: если $r_1 \leqslant r_2$, то $l_1 \leqslant l_2$. Это замечание позволяет нам использовать идею двух указателей, будем поддерживать наидлиннейший отрезок уникальных значений, двигая правую границу. Для этого нам поможет структура данных set/map.

# Задача B. Время на марсе

*Автор и разработчик задачи: Егор Юлин*

Будем решать задачу следующим образом — смоделируем работу часов, будем смотреть необходимое количество дисплеев для отображения каждого времени в промежуток с $H_1 : M_1$ по $H_2 : M_2$.

Пусть у нас сейчас время $h : m$. Количество дисплеев будем считать следующим образом: преобразуем $h$ и $m$ в строку и сложим их, получится строка $h + m$. После этого нужно алгоритмом из условия посчитать количество дисплеев.

Тогда ответом на задачу будет максимальное количество дисплеев, которые использовались в отрезок времени $[H_1 : M_1; H_2 : M_2]$.

# Задача C. Очередная задача про запросы на перестановках

*Автор и разработчик задачи: Илья Хлюстов*

Первое замечание: всего пар делитель и делимое будет $O(n \log n)$, так как для каждого $i$ делимых на него будет $\lfloor \frac{n}{i} \rfloor$. Соответственно их количество — $\sum_{i=1}^{n} \frac{n}{i} = n \sum_{i=1}^{n} \frac{1}{i}$, здесь второй множитель в правом выражении является суммой гармонического ряда, которая асимптотически равна $O(\log n)$.

Второе замечание: данную задачу будет проще решать в оффлайн.

В рамках одного запроса есть 3 разных типа пар делителей и делимого:

1. Оба индекса находятся вне отрезка запроса.
2. Один индекс находится в отрезке запроса, а другой – вне.
3. Оба индекса внутри отрезка запроса, а значит должен быть учтён.

Далее давайте научимся считать только третий тип запросов, для этого заведём вспомогательный массив `count` на $n$ элементов, изначально инициализированный нулями. Будем идти от 1 до $n$ и в $i$-й позиции мы прибавим единицу к $\texttt{count}_j$ таким, что $p_j \mid p_i$. Так же для всех запросов, левая граница которых равна $i$ будем запоминать для них сумму элементов массива `count` на отрезке этого запроса, назовём эту величину $left_k$ для $k$-го запроса. Та же логика и для правой границы, назовём величину $right_k$.

Оказывается, что ответ на $k$-й запрос $right_k - left_k$.

Доказательство:

- *Рассмотрим первый тип пар.* Они никак не повлияют на $left_k$ или $right_k$.

- *Рассмотрим второй тип пар.* Тут есть две ситуации. Первая: когда $i$ было внутри отрезка, тогда $j$ вне отрезка(исходя из типа запроса) и никак не повлияет ни на $left_k$, ни на $right_k$. Вторая: $i$ было вне отрезка, а $j$ внутри. Тут тоже две ситуации. Первая: $i$ было слева от отрезка запроса, в таком случае $left_k$ и $right_k$ Оба увеличатся на 1, что не изменит ответ. Вторая: $i$ было справа от отрезка запроса, в таком случае $left_k$ и $right_k$ не изменятся.

- *Рассмотрим третий тип пар.* Оба индекса находятся внутри отрезка, соответственно $i$ увеличит значение $\texttt{count}_j$ уже после того, как мы вычислим $left_k$, но до того, как мы вычислим $right_k$, соответственно, ответ увеличится на 1. Что и требовалось доказать.

Теперь разберёмся с тем, как подсчитать это всё эффективно: нам необходимо сделать $O(n \log n)$ операций прибавления единицы в конкретной позиции, $O(m)$ запросов суммы на отрезке, эти операции поддерживает дерево отрезков или дерево Фенвика. Итоговая асимптотика: $O(n \log^2 n + m \log n)$.

# Задача D. Детали и ресурсы

*Автор задачи: Илья Хлюстов, разработчик: Даниил Конов*

Всего для $N$ рабочих, каждый из которых должен изготовить $M$ деталей понадобится $N * M * K$ килограммов металла, ведь на каждую деталь тратится $K$ килограммов метала. Тогда понадобится хотя бы $\left\lceil \frac{N*M*K}{L} \right\rceil$ контейнеров, так как каждый может вмещать в себя $L$ килограммов металла, тогда нужно потратить хотя бы $\left\lceil \frac{N*M*K}{L} \right\rceil * S$ рублей на закупки - это и есть ответ на задачу.

# Задача E. Очередная задача на конструктив

*Автор и разработчик задачи: Егор Юлин*

Для решения этой задачи заметим следующее, если у нас есть два целых числа, сумма которых фиксирована, то их произведение будет максимально тогда и только тогда, когда они отличаются не более чем на 1. Будем использовать это при решении нашей задачи.

Так же заметим, что ответ будет максимален в том случае, если у нас в обоих массивах будет максимальное количество цифр 9. Тогда будет строить массивы следующим образом:

- Если наша оставшаяся сумма хотя бы 18, то в конец обоих массивов добавим по 9 (и из суммы вычтем 18)

- Если сумма меньше 18, то в оба массива запишем максимально близкие числа (т.е. числа, которые отличаются не более чем на 1)

После данного построения можно посчитать значение получившегося выражения, оно и будет ответом.

# Задача F. Посчитай

*Автор задачи: Илья Хлюстов, разработчик: Михаил Величковский*

Легко заметить, что номера кабинетов представляют из себя прогрессию вида $k+k^2+k^3+\ldots+k^n$. Если вы не знаете или не помните, как считается сумма этой прогрессии - можно её посчитать в цикле, каждый раз прибавляя степень числа k в диапазоне от 1 до $n$. Иначе пишите формулу: $\frac{k*(k^n-1)}{k-1}$.