Задача A. Электронное табло

Если в некоторый момент на табло записано число X, то для того, чтобы отобразить на нем число $X+1$, нужно произвести столько операций замен цифр, в каком количестве разрядов числа X и $X+1$ отличаются. Например, чтобы перейти от числа 2413 к числу 2414, нужно произвести одну операцию замены, а чтобы перейти от числа 7999 к 8000, нужно заменить все 4 цифры. Ответ на задачу — это суммарное количество операций замен для всех требуемых переходов, то есть для всех значений X от A до $B-1$.

С помощью цикла `for` переберем все возможные значения X от A до $B-1$. Один из вариантов, что делать дальше — это честно выписать цифры чисел X и $X+1$ и проверить все 4 разряда на равенство. В другом варианте можно заметить, что количество различных цифр соответствует количеству нулей в конце записи числа $X+1$:

- если оно не делится на 10, то заменить нужно только одну цифру ($1234 \to 1235$),
- если делится на 10, но не на 100, то две ($1239 \to 1240$),
- если делится на 100, но не на 1000, то три ($1299 \to 1300$),
- если делится на 1000, то четыре ($1999 \to 2000$).

Первый вариант решения: `https://ideone.com/ShhEVT`
Второй вариант решения: `https://ideone.com/OX05fy`

Задача B. Остатки сумм

Насчитаем вспомогательный массив $s[0..n]$ следующим образом:

- $s_0 = 0$
- $s_i = a_1 + a_2 + \ldots + a_i = s_{i-1} + a_i$, $i \geqslant 1$.

Массив s называется массивом префиксных сумм массива a. С помощью этого массива можно эффективно находить сумму элементов любого подотрезка массива $a[l..r]$:

$$a_l + a_{l+1} + \ldots + a_r = (a_1 + a_2 + \ldots + a_r) - (a_1 + a_2 + \ldots + a_{l-1}) = s_r - s_{l-1}$$

Таким образом, мы можем перейти к новой задаче: по заданному массиву s найти количество пар индексов l, r таких, что $1 \leqslant l \leqslant r \leqslant n$ и $(s_r - s_{l-1}) \mod x = y$, что на самом деле можно упростить и записать как

$$0 \leqslant l < r \leqslant n,\ (s_r - s_l) \mod x = y.$$

Здесь $A \mod B$ означает остаток от деления A на B.

Заметим, что если $(s_r - s_l) \mod x = y$, то $s_r \mod x = (s_l + y) \mod x$. Здесь можно провести аналогию с обычным вычитанием чисел – если для чисел A, B, C верно, что $A - B = C$, то верно и $A = B + C$, только в нашем случае всех операции впоследствии заменяются взятием остатка от деления. Например, раз $(17 - 7) \mod 6 = 4$, то и $17 \mod 6 = (7 + 4) \mod 6 = 5$.

Итак, теперь наша задача выглядит следующим образом. Требуется найти количество пар индексов l, r, что

$$0 \leqslant l < r \leqslant n,\ s_r \mod x = (y + s_l) \mod x.$$

Заведем массив $t[0..n]$, для каждого элемента которого верно $t_i = (s_i + y) \mod x$, а все элементы s заменим остатком от деления на x. А теперь требуется найти количество пар индексов l, r, что

$$0 \leqslant l < r \leqslant n,\ s_r = t_l.$$

Эту задачу будем решать следующим образом. Заведем новый массив-счетчик f, где f_j будет означать, сколько элементов массива t на текущий момент равны j. Изначально все значения массива f равны нулю. Будем перебирать индекс r слева направо от 1 до n. Для каждого r в массиве f будут находится только те значения элементов t_l, для которых $l < r$.

Пусть имеется некоторое значение r. Сперва нужно добавить в массив f элемент массива t, который раньше нельзя было использовать как t_l из условия выше — это элемент t_{r-1}. Количество подходящих значений элементов массива t для текущего значения s_r равно f_{s_r}; сумма этих значений и есть ответ на задачу.

Реализация: https://ideone.com/flTBPD.

Простое, но медленное решение, получающее 60 баллов: https://ideone.com/cVDKG5. Нужно не забыть, что ответ на задачу и сумма элементов подотрезка могут превысить 32-битные типы данных (`longint` для `FPC` и `int` для `C++`), в таком случае это решение набирает на 10 баллов меньше.

Задача C. Числа на листочке

Для удобства обозначим через $f(a)$ сумму чисел, которую выпишет Вася на листочек, если начнет с числа a. Например, если $b = 2$, $f(10) = 10 + \lfloor \frac{10}{2} \rfloor + \lfloor \frac{10}{4} \rfloor + \lfloor \frac{10}{8} \rfloor = 10 + 5 + 2 + 1 = 18$.

В задаче нужно найти такое минимальное значение a, что $f(a) \geqslant n$.

Нетрудно заметить, что $f(0) \leqslant f(1) \leqslant f(2) \leqslant f(3) \leqslant \ldots$, или, другими словами, для любого a выполняется $f(a) \leqslant f(a+1)$. То есть, чем больше a, тем больше значение $f(a)$. Это так, потому что для каждого k, $\lfloor \frac{a}{b^k} \rfloor \leqslant \lfloor \frac{a+1}{b^k} \rfloor$, а значит и сумма чисел на листике по всем k тоже больше для $a + 1$, чем для a.

Это дает возможность воспользоваться так называемой техникой бинарного поиска (в данном случае, бинарным поиском по ответу).

Вкратце, бинарный поиск выглядит следующим образом. Заведем два значения, l и r, которые будут означать, что ответ на задачу находится между l и r. В нашем случае изначально можно положить $l = 0$, $r = n$ (т.к. $f(0) = 0 < n$, $f(n) \geqslant n$, а значения f не убывают). Будем сужать область поиска ответа, причем будем поддерживать так называемый инвариант, что всегда выполняется $f(l) < n$, $f(r) \geqslant n$. Пусть $m = \lfloor \frac{l+r}{2} \rfloor$. Теперь есть два случая:

- $f(m) < n$. В таком случае можно сузить поиск, положив $l = m$ (и сохранив инвариант),
- $f(m) \geqslant n$. В таком случае можно сузить поиск, положив $r = m$ (и снова сохранив инвариант)

Таким образом, вычислив один раз значение функции f, можно в два раза уменьшить длину отрезка поиска ответа. Таких уменьшений будет не больше 30, потому что $2^{30} > 10^9$— максимального значения n. Однажды настанет момент, когда $r - l = 1$, то есть отрезок состоит из двух значений, но мы знаем, что $f(l) < n$ и $f(r) \geqslant n$, следовательно ответ на задачу равен текущему значению r.

Решение: https://ideone.com/FDYS4c.

Решение без бинарного поиска (68 баллов): https://ideone.com/pPH5u7

Задача D. Тройки

Немного про битовые операции https://bit.ly/2QMlTFc.

Решение на 38 баллов, перебирающее все возможные тройки чисел: https://ideone.com/m1fF6K

Для решения на 60 баллов нужно воспользоваться фактом, что если $c = a \oplus b$, то $b = c \oplus a$ и $a = b \oplus c$, то есть операция «хог» («исключающее или») обратна сама себе (для операции «ог» («или»), кстати, это свойство не выполняется). Таким образом, значение z всегда однозначно восстанавливается из x, y: $z = (x|y) \oplus y$. https://ideone.com/8CFUr5.

Пусть $F(A, B, C)$ означает количество троек x, y, z, таких, что $0 \leqslant x < A$, $0 \leqslant y < B$, $0 \leqslant z < C$ и $((x|y) == (y \oplus z))$. Тогда, пользуясь принципом включений-исключений (https://bit.ly/33dg4Dk), ответ на задачу можно вычислить как $F(R+1, R+1, R+1)$ - $F(L, R+1, R+1)$ - $F(R+1, L, R+1)$ - $F(R+1, R+1, L)$ + $F(L, L, R+1)$ + $F(R+1, L, L)$ + $F(L, R+1, L)$ - $F(L, L, L)$.

Для вычисления $F(A, B, C)$нужно хорошо понимать идею динамического программирования. В интернете существует множество статей про динамическое программирование, поэтому в рамках данного разбора мы не будем останавливаться на том, что оно из себя представляет. В данном

случае будет использоваться техника динамического программирования «по числу» — по битовому представлению данных чисел.

Пусть $dp_{i,fa,fb,fc}$ означает количество троек префиксов чисел x, y, z длины i (от старших к младшим), для каждого бита j которого выполнялось $(x_j|y_j) == (y_j \oplus z_j)$ и при том:

- $fa = 1$, если данный префикс числа x полностью совпадает с префиксом длины i бит числа A, и $fa = 0$ если префиксы не совпадают, но при этом префикс числа x меньше префикса числа A как двоичное число,
- $fb = 1$, если данный префикс числа y полностью совпадает с префиксом длины i бит числа B, и $fb = 0$ если префиксы не совпадают, но при этом префикс числа y меньше префикса числа B как двоичное число,
- $fc = 1$, если данный префикс числа z полностью совпадает с префиксом длины i бит числа C, и $fc = 0$ если префиксы не совпадают, но при этом префикс числа z меньше префикса числа C как двоичное число.

База: $dp_{0,1,1,1} = 1$ — для длины 0 существует только одна тройка префиксов — пустая, и так как пустые множества что для x, y, z, что для A, B, C, совпадают, $fa = fb = fc = 1$.

Переход из состояния динамического программирования делается следующим образом. Пусть мы находимся в состоянии $dp_{i,fa,fb,fc}$. Переберем 8 вариантов значений $i+1$-го по старшинству бита чисел x, y, z. Тогда переход можно совершить, если $(x_{i+1}|y_{i+1}) == (y_{i+1} \oplus z_{i+1})$ и при том, никакой префикс не стал больше своей соотвествующей границы. Такое могло произойти, если текущие префиксы совпадают, а новый бит числа больше этого же бита в числе-границе, то есть когда $fa = 1$ и $x_{i+1} = 1$, а $A_{i+1} = 0$ и аналогично для y, z. В таком случае можно совершить переход в $dp_{i+1,\_,\_,\_}$ с новыми пересчитанными значениями fa, fb, fc.

Ответ для $F(A, B, C)$ будет лежать в $dp_{n,0,0,0}$, где n - длина битовой записи чисел (достаточно взять $n = 30$).

Вариант реализации: `https://ideone.com/uzuo57`