Тройка, девятка, ноль ...

Решение.

Я, конечно, понимаю, что светлая мысль, перебрать все числа от A до B и подсчитать нужные, в той или иной форме проскочила у многих где-то на задворках сознания. Но я очень надеюсь, что там, на задворках, она и осталась, припечатанная тяжёлым ботинком. Понятно, что никакой эффективности от такого решения ждать нельзя.

Будем думать. Начнём с чего попроще. Посчитаем числа, в которых есть хотя бы одна тройка, например. С лёту ничего не придумывается? Ну, выпишем тогда все интересующие нас числа и посмотрим, что получится:

3, 13, 23, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 43, … (1)

Что-то ничего интересного... А посмотрим, что же мы выкинули. А выкинули мы вот что:

1,2,4,5,6,7,8,9,10,11,12,14,15,16,17,18,19,20,21,22,24,25,26,27,28,29,40,41,42,44,45,… (2)

Хороший ряд, только троек в нём нет. Неприятно как-то. Да ведь дело не в отсутствии тройки, а в том, что цифр всего 9. Где-то это уже было... Были всякие там системы счисления, причём не только двоичная, но и другие. Девять цифр – девятеричная система. Заменим в выписанном ряду чисел 4 на 3, 5 на 4, 6 на 5 и т.д. и получим

1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,17,18,20,21,22,23,24,25,26,27,28,30,31,32,33,34,… (3)

Действительно, чудесный ряд, если посмотреть на него непредвзятым взглядом, не зная предистории, то сразу видно – подряд выписаны все натуральные числа, причём записаны они в девятеричной системе счисления. И подсчитать их не представляет никакого труда. Понятно, что количество чисел равно последнему выписанному числу – начинали-то с 1. Переводить числа из девятеричной системы в десятичную мы умеем.

А если не умеем? Или вовсе не знаем, что это такое — системы счисления. Тогда самое время узнать... В учебных материалах лежат статья

Яглом И. Системы счисления. // Квант. – 1991. – № 12. – С. 15-22.

и две небольших по объёму книги

Фомин С.В. Системы счисления. – 5-е изд. - М.: Наука, гл. ред. физ.-мат. лит., 1987. - 48 с. - (Популярные лекции по математике).

Гашков С.Б. Системы счисления и их применение. – М.: МЦНМО, 2004. – 52 с. - (Библиотека «Математическое просвещение») .

Литература на эту тему огромна, и я с трудом выбрал именно эти три источника. Первые два написаны, если так можно выразится, в классическом стиле, последняя — в более «модерном» (думаю, что в ней и опытный читатель найдёт что-нибудь новое), - я имею в виду круг рассматриваемых вопросов. Но все они очень интересны, каждый по-своему, что и обусловило выбор.

Последнее записанное число – 34, $34_9 = 3 \cdot 9^1 + 4 = 31_{10}$. Действительно, в ряду (3) выписано ровно 31 число. А начинали мы с чего? С чисел от 1 до 45, из которых выбросили все числа, содержащие хотя бы одну тройку, - это ряд (2). И осталось у нас 31 число. Значит, хотя бы одна тройка присутствовала в 14 (= 45 – 31) числах. Посмотрим на ряд (1) – и вправду, в нём ровно 14 чисел.

Обозначим для краткости количество чисел, содержащих хотя бы одну тройку и не превосходящих X, через F(X).

Всё хорошо? Проверим. А чему равно F(25)? Уменьшаем в числе 25 обе цифры на 1,

полученную запись 14 переводим из девятеричной системы в десятичную: $14_9=13_{10}$, и получаем ответ: 25 – 13 = 12.

Что-то не то... Ожидали-то мы ответа 3 (числа 3, 13 и 23). Понятно. Двойку в записи 25 не надо было заменять на единицу – ведь 2<3. Что получается? Запись 25 переходит в 24, $24_9=22_{10}$, 25 – 22 = 3.

Последняя проверка. А найдём-ка мы F(33) – тройка ведь в нашем случае играет особую роль. Запись 33 переходит в 22. $22_9=20_{10}$, 33 – 20 = 13. А должно быть 7 (3, 13, 23, 30, 31, 32, 33). Почему? Понятно, дело в следующем: посмотрим на ряд (2) – понятно, что в нём нет ни 33, ни 32, ни 31, ни 30; наибольшее число, не превосходящее 33 в этом ряду – это 29, в ряду (3) ему соответствует запись 28. $28_9=26_{10}$, т.е. число 28 стоит в ряду (3) на 26-ом месте, и, конечно же, на 26-ом месте в ряду (2) стоит число 29. Значит, среди первых 33 чисел 26 не содержат ни одной тройки, а содержат хотя бы одну тройку 33 – 26 = 7 чисел. То, что надо!

Итак, чтобы вычислить F(X), следует поступить следующим образом:

1.Пусть Y – наибольшее число, не превосходящее X и не содержащее в своей записи ни одной тройки.

2.В записи числа Y уменьшим все цифры, большие или равные 3, на 1. Назовём результат Z.

3.Рассматривая Z как девятеричную запись некоторого числа T, вычислим X-T. Полученное число и есть F(X).

Ну, а если мы умеем вычислять F(X), то количество чисел из интервала от A до B, содержащих хотя бы одну тройку в своей записи, подсчитать совсем легко: это просто F(B)-F(A-1). Идея очень простая, но весьма эффективная и применима на удивление часто.

Осталось найти Y– наибольшее число, не превосходящее X и не содержащее в своей записи ни одной тройки. Это несложно:

1. находим самую левую тройку в записи числа X (понятно, что если троек в записи X нет, то Y=X),
2. заменяем её на двойку (поскольку Y<=X),
3. а затем все цифры правее этой самой двойки заменяем на девятки (поскольку, какие цифры не ставь правее двойки, условие Y<X всё равно выполняется, а мы хотим найти наибольшее Y).

Проверяем. Пусть X=123123123 – подчёркнута самая левая тройка. Тогда получаем Y=122999999. Пусть X=333333. Тогда Y=2999999. Всё верно.

Конечно, точно также можно найти и количество чисел, на превосходящих X и содержащих хотя бы одну девятку. Единственное отличие: когда мы ищем наибольшее число, не превосходящее X и не содержащее в своей записи ни одной девятки (аналог числа Y), цифры, справа от самой левой девятки, надо заменять на восьмёрки (девятка запрещена!).

Проверим. Пусть X=5793975. Получаем Y=5788888. Пусть X=9999. Тогда Y=8888. Да, всё хорошо.

А как быть с нулём? Да почти точно также. Находим самый левый ноль и заменяем его... Да, а на что мы его заменяем? Не на –1 же (кстати, можно и на –1, но не будем залезать в такие дебри). Сформулируем то же самое действие иначе: находим в числе X самый левый 0; отбрасываем все цифры правее этого нуля – и получаем число X^*; уменьшаем X^* (ведь новое число должно быть меньше X, раз уж в X нашёлся 0) на 1 и дописываем к тому, что получилось столько девяток, сколько цифр мы отбросили (ведь мы хотим найти наибольшее число без нулей, меньшее X).

Проверяем. Пусть X=55050505. Тогда X^*=550, X^*-1=549, и Y=54999999. Пусть X=3210123. Тогда получаем X^*=3210, X^*-1=3209, и Y=3209999. Оп-паньки! Не получилось. В Y торчит невесть откуда затесавшийся 0. То есть как раз понятно, откуда затесавшийся

– из-за единички перед нулём. Что ж теперь делать, отслеживать такие единички? А если перед ней обнаружится ещё одна единичка, или даже не одна... Скучно как-то. Решим проблему проще. Есть в X^*-1 нолик? Тогда двигаемся влево, пока не задавим его. А если появился ещё один? Продолжаем движение налево. Ясно, что скоро это кончится в конце-то концов – ведь длина X^* всё время уменьшается.

Ну, а раз уж всё равно нам придётся двигаться справа налево, то почему бы не делать этого с самого начала. На скорость это в худшем случае не повлияет, а код попроще получится.

Кстати, даже если запрещённая цифра и не 0, то всё равно как-то ловчее получается двигаться справа налево. Я рискну в этот раз включить в текст куски кода – с кодом в данном случае как-то попонятнее получается.

```
Function Y (X: LongInt; Oblige: Byte): LongInt;
//Если X>0, то возвращает максимальное число без цифры Oblige, не превосходящее X
//Если X=0, то возвращает 0 (даже если Oblige=0)
Var w: LongInt;
    N, k: Byte;
Begin
  w:= X;
  k:= 0;
  N:= 0;
  While w>0 Do Begin
    Inc(k);
    If (w mod 10 = Oblige) Then Begin
      Dec(w);
      X:= w;
      N:= k;
    End;
    w:= w div 10;
  End;

  If N>1 Then Begin
    If Oblige = 9 Then w:= 8
                  Else w:= 9;
    For k:= N-1 DownTo 1 Do Begin
      X:= X*10 + w;
    End;
  End;
  Y:= X;
End;
```

Итак, Y вычислять мы умеем, цифры уменьшать умеем, переводить из девятеричной системы в десятичную тоже умеем. С одной цифрой всё. Только проверим ещё, как работает наш алгоритм с 0. Пусть X=45. Нулей в этой записи нет, уменьшим, как и выше, обе цифры на 1 (обе они, разумеется, не меньше 0), а $34_9 = 3 \cdot 9^1 + 4 = 31_{10}$ – это мы уже делали. Получается, что хотя бы один 0 присутствует в записи 14 (=45-31) чисел. Это каких же? 10, 20, 30, 40. И всё! Откуда же взялись ещё 10 чисел. Ответ простой – из-за лидирующих нулей, из-за нулей-невидимок в начале числа, из-за тех нулей, которые мы обычно не записываем. Когда мы считали числа без нулей, мы не учли 9 однозначных чисел 01, 02, 03, 04, 05, 06, 07, 08 и 09, да ещё и одно "0-значное" число 00. Они-то и дали разницу.

В этом месте, мне кажется, будет эффективней приостановиться и призадуматься, чем читать чьи-то объяснения. Всё-таки не удержусь и коротенькое объяснение дам: мы ведь предполагали изначально все цифры равными в правах, т.е. каждая цифра может с

одинаковым успехом присутствовать в числе на любом месте. А это не так в случае с 0.

Повторим и уточним определение величины F(X):

Обозначим для краткости количество натуральных чисел, содержащих хотя бы одну тройку (девятку, ноль...) и не превосходящих X, через F(X).

И теперь, если мы считаем числа, содержащие цифру 0, то осталось внести в алгоритм вычисления F(X) два уточнения:

1. F(X) надо уменьшить на 1 (чтобы учесть число 0)
2. Если X – P-значное натуральное число, то F(X) надо уменьшить на $9^1+9^2+...9^{P-1}$.

Откуда взялась такая величина ($9^1+9^2+...9^{P-1}$) ? Дело в том, что мы включили все числа с лидирующими нулями, т.е. все одно-, дву-, ..., (P-1)-значные числа. А мы должны учитывать только те из них, которые содержат в своей записи и не лидирующие нули тоже (расположенные в середине или в конце числа, в общем, правее первой ненулевой цифры). Сколько же чисел мы учли напрасно? Столько, сколько среди все одно-, дву-, ..., (P-1)-значных чисел имеется чисел без нулей в записи (имеется в виду, конечно, "обычная" запись чисел, без лидирующих нулей). Сколько таких чисел? Понятно, что имеется ровно 9 таких однозначных чисел. А двузначных чисел без нулей в записи (если не считать лидирующих) имеется ровно $9 \cdot 9 = 9^2$. В самом деле, на первом из двух мест может стоять любая цифра от 1 до 9, а на втором месте, независимо от выбора первой цифры, имеется 9 вариантов – можно точно также выбрать любую из цифр от 1 до 9. Аналогично, имеется ровно $9 \cdot 9 \cdot 9 = 9^3$ трёхзначных чисел без нулей в записи (не считая лидирующих) и т.д.

Здесь использовано (совершенно очевидное) *комбинаторное правило произведения*: если некоторый объект A может быть выбран из совокупности объектов m способами и после каждого такого выбора объект B можно выбрать n способами, то пара объектов (A, B) в указанном порядке может быть выбрана $m \cdot n$ способами.

Правило это очевидно (по крайней мере, в данном случае), да и написано о нём везде, где только возможно. Но, несмотря на очевидность комбинаторных выкладок в нашей задаче, я очень советую почитать что-нибудь о комбинаторике — комбинаторные сюжеты встречаются в программировании (и совсем не только спортивном) сплошь и рядом. И, второе: эта задача начинает линию задач с комбинаторыми сюжетами в RunSite, так что хотя бы для решения следующих задач обретённые знания пригодятся.

В частности, в учебных материалах я выкладываю (в (файле **vilenkin76.pd)** один параграф из книги

Н.Я. Виленкин. Индукция. Комбинаторика. Пособие для учителей. М.: Просвещение, 1976. 48 с.

Изложение там довольно краткое. Но немного ниже приводится ссылка на другую книгу этого же автора, в которой и изложение подробнее, и круг затрагиваемых вопросов много шире, и, что особенно приятно, имеется масса задач.

В результате получили вот такую функцию F1 (один – это потому, что она учитывает только одну цифру, а нам ещё понадобятся и две, и три):

```
Function F1 (X: LongInt; Oblige: Byte): LongInt;
Var Res: LongInt; // Res – количество натуральных чисел
                  // без цифры Oblige, не превосходящих X
    d9: LongInt;  // d9 – содержит последовательные степени девятки
    X0: LongInt;
    w: Byte;
Begin
  X0:= X;
  X:= Y(X, Oblige); // эта функция приведена выше
  Res:= 0;
```

```
  If Oblige=0 Then Res:= 1;
  d9:= 1;
  While X>=10 Do Begin
    w:= X mod 10;
    X:= X div 10;
    If (w>=Oblige) Then Dec(w);
    Res:= Res + w*d9;
    d9:= d9*9;
    If Oblige=0 Then Inc(Res,d9);
  End;
  If X>=Oblige Then Dec(X);
  Res:= Res + X*d9;
  F1:= X0 - Res;
End;
```

Результат, конечно, замечательный, но намёки про две и про три цифры вызывают какую-то отчуждённость ☺.

На самом деле, всё самое страшное уже позади. Осталось совсем чуть-чуть - просто повторить те же рассуждения в слегка изменившейся обстановке.

Давайте выпишем, например, числа без троек и семёрок, хотя такого задания в задаче и нет:

1,2,4,5,6,8,9,10,11,12,14,15,16,18,19,20,21,22,24,25,26,28,29,40,41,42,44,45,46,48,49,50,51,52,54,55,… (4)

Заполним теперь "дырки" в ряду цифр – уменьшим цифы 4, 5 и 6 на 1, а 8 и 9 – на 2 (поскольку до них уже две дыры). Получим в результате вот такой ряд чисел

1,2,3,4,5,6,7,10,11,12,13,14,15,16,17,20,21,22,23,24,25,26,27,30,31,32,33,34,35,36,37,40,41,42,43,44,… (5)

или, просто говоря, последовательно выписанные натуральные числа, записанные в восьмеричной системе счисления. Всего в ряду (5) выписаны $44_8 = 4 \cdot 8 + 4 = 36$ чисел. Это означает, что среди чисел от 1 до 55 имеется 36 чисел без троек и семёрок, и 55-36=19 чисел, в которых имеется хотя бы одна тройка или хотя бы одна семёрка.

А что это нам даёт? Ведь мы хотим узнать количество чисел, в которых есть хотя бы одна тройка **И** хотя бы одна семёрка, а вовсе не "или". Ничего страшного. Применим ещё одно правило, которое совершенно очевидно в данном случае:

,

где A и B – это произвольные конечные множества, а прямые скобки обозначают количество элементов в соотвествующем множестве.

Это правило красиво называется формулой включений и исключений и применимо для любого количества множеств, не только для двух, хотя, конечно, для большего количества множеств оно выглядит и доказывается посложнее. А для двух множеств – совершенно очевидное соотношение.

Нам эта формула понадобится ниже и для трёх множеств, да и вообще, формула полезная, а соображения, применяемые при её доказательстве (а имеется немало различных доказательств) ещё полезнее. В учебных материалах имеется файл **`onoff.pdf`** – небольшой фрагмент, посвящённый этой формуле, прекрасной, особенно для начинающих, книги

Н.Я. Виленкин. Комбинаторика. М.: Наука, 1969. 328 с.

Электронную версию этой книги можно свободно скачать для некоммерческого использования в образовательных целях по этому адресу:

http://www.math.ru/lib/book/djvu/kombinatorika.djvu

Впрочем, уже упоминавшийся и выложенный в учебных материалах (файл **`vilenkin76.pdf`**) фрагмент другой книги Н.Я.Виленкина тоже рассказывает о формуле включений и исключений (причём о наиболее актуальных в нашей задаче случаях двух и трёх множеств — особенно подробно).

В нашем случае A – это множество натуральных чисел, содержащих в своей записи хотя

бы одну тройку (и не превосходящих 55, конечно), B – содержаших хотя бы одну семёрку. Тогда, понятно, что A∪B - это множество чисел, содержащих в своей записи хотя бы одну тройку или хотя бы одну семёрку, а A∩B - то же самое, только **И** вместо "или". Мы уже вычислили |A∪B|=19. Вычислять |A| и |B| мы уже умеем, - у нас ведь есть функция F1: |A|= 15, |B|= 5. Отсюда сразу получаем |A∩B|= |A| + |B| - |A∪B| = 15 + 5 – 19 = 1.

Осталось всё-таки посчитать |A∪B|. Обозначим через F2(X) количество натуральных чисел, не превосходящих X, которые содержат в своей записи хотя бы одну из двух цифр (в нашем примере – тройку или семёрку). Итак, чтобы вычислить F2(X), следует поступить следующим образом:

1. Пусть Y – наибольшее число, не превосходящее X и не содержащее в своей записи ни одной из заданных (обязательно присутствующих) цифр.
2. В записи числа Y уменьшим все цифры, большие или равные меньшей из заданных цифр, на 1, а затем все цифры, большие или равные большей из заданных цифр, ещё на 1 . Назовём результат Z.
3. Рассматривая Z как восьмеричную запись некоторого числа T, вычислим X-T.
4. Если среди заданных цифр присутствует 0, то уменьшим результат на 1 (чтобы учесть число 0), и, если X – P-значное натуральное число, то уменьшим результат ещё на $9^1+9^2+...9^{P-1}$.

Обоснований здесь не требуется – всё полностью аналогично вычислению F1. Проще посмотреть на текст функции F2 и сравнить с текстом функции F1:

```
Function F2 (X: LongInt; Oblige1, Oblige2: Byte): LongInt;
Var Res: LongInt; // Res – количество натуральных чисел, не превосходящих X и
                  // не содержащих в своей записи цифр Oblige1 и Oblige2
    d8: LongInt;  // d8 – содержит последовательные степени восьмёрки
    X0, X1: LongInt;
    w: Byte;
Begin
  If Oblige1=Oblige2 Then Begin
    F2:= F1(X,Oblige1);
    Exit;
  End;
  If Oblige1>Oblige2 Then Begin
    w:= Oblige1;           // В итоге будет обязательно выполняться условие
    Oblige1:= Oblige2;     // Oblige1<Oblige2. На скорость практически
    Oblige2:= w;           // не влияет, а код существенно упрощается.
  End;
  X0:= X;
  Repeat
    X1:= X;                // Этот цикл нужен потому, что после второго
    X:= Y(X, Oblige1);     // применения функции Y в числе X может снова
    X:= Y(X, Oblige2);     // появиться цифра Oblige1.
  Until (X=X1);
  Res:= 0;
  If Oblige1=0 Then Res:= 1;
  d8:= 1;
  While X>=10 Do Begin
    w:= X mod 10;
    X:= X div 10;
    If w>=Oblige2 Then Dec(w);
    If w>=Oblige1 Then Dec(w);
    Res:= Res + w*d8;
```

```
    d8:= d8*8;
    If Oblige1=0 Then Inc(Res,d8);
  End;
  If X>=Oblige2 Then Dec(X);
  If X>=Oblige1 Then Dec(X);
  Res:= Res + X*d8;
  F2:= X0-Res;
End;
```

А после этого никакие страшилки про подсчёт чисел с присутствующими/отсутствующими тройками цифр нас не пугают. Просто возьмём и переделаем функцию F2 на общий случай — функция F вычисляет количество тех натуральных чисел, не превосходящих X, которые содержат хотя бы одну из цифр заданного множества цифр. Результат этой операции перед вами, отличий от F2 очень мало:

```
     Type TDigitsSet = Set of 0..9;

     Function F (X: LongInt; ObligeSet: TDigitsSet): LongInt;
     Var Res: LongInt;
         { Res - количество натуральных чисел, не превосходящих }
         { X и не содержащих цифр из множества ObligeSet }
         d, X0: LongInt;
         optional: Byte; {количество необязательных цифр}
         w, i: Byte;
     Begin
       X0:= X;
       optional:= 10;
       For w:= 0 To 9 Do Begin
         If (w in ObligeSet) Then Begin
           Dec(optional);
         End;
       End;
       Repeat
         d:= X;        // здесь d используется как вспомогательная переменная
         For w:= 0 To 9 Do Begin
           If (w in ObligeSet) Then Begin
             X:= Y(X, w);
           End;
         End;
       Until (X=d);
       Res:= 0;
       If (0 in ObligeSet) Then Res:= 1;
       d:= 1;          // а начиная с этого момента d  - это последовательные
                       // степени основания системы счисления
       While X>9 Do Begin
         w:= X mod 10;
         X:= X div 10;
         For i:= w DownTo 0 Do Begin
           If (i in ObligeSet) Then Dec(w);
         End;
         Res:= Res + w*d;
         d:= d*optional;
         If (0 in ObligeSet) Then Inc(Res,d);
       End;
```

```
      For i:= X DownTo 0 Do Begin
        If (i in ObligeSet) Then Dec(X);
      End;
      Res:= Res + X*d;
      F:= X0-Res;
    End;
```

Остаётся только ещё раз применить формулу включений и исключений, чтобы найти количество чисел, содержащих одновременно и тройки, и девятки, и нули.