АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ТРИОНООБРАЗНИ ЧИСЛА

Наивното решение е пълно изчерпване – да се обходят всички N цифрени числа. Да се отделят цифрите на всяко число и да се провери дали е трионообразно. Такова решение ще получи около 25 точки – saw1.cpp.

Частично задачата може да се реши чрез броене, с помощта на търсене с връщане назад (backtracking). За по-малко от минута се обхождат всички 12-13 цифрени числа. Това е реализирано във файла saw2.cpp и може да вземе до 45 точки.

За 100 точки обаче трябва динамично оптимиране.

Създаваме две матрици g и l с N реда и 10 стълба, където:

- *g[i][j]* е брой на трионообразните последователности с дължина i, които завършват на цифрата j, а предходната цифра е по-малка от j.
- *l[i][j]* е брой на трионообразните последователности с дължина i, които завършват на цифрата j, а предходната цифра е по-голяма от j.

Сега може да напишем прехода от i към (i+1).

За всички двойки цифри k и j, където $k > j$:

```
l[i + 1][k] += g[i][j];
```

За всички двойки цифри k и j, където $k < j$:

```
g[i + 1][k] += l[i][j];
```

В първата формула става дума, че може да добавим цифрата k в края на всички i-цифрени числа, които завършват с цифрата j, а предната цифра е по-голяма от j. Тъй като тя вече ще стане по-малка от k, то резултатът се добавя към елемент на матрицата l. Аналогично за втората формула.

Забелязваме, че отговорът може да надхвърли диапазона на типа *int* и затова трябва да се използва *long long*.

Автор: Кинка Кирилова-Лупанова