Rotated Digits

X is a good number if after rotating each digit individually by 180 degrees, we get a valid number that is different from X. A number is valid if each digit remains a digit after rotation. 0, 1, and 8 rotate to themselves; 2 and 5 rotate to each other; 6 and 9 rotate to each other, and the rest of the numbers do not rotate to any other number.

Now given a positive number `N`, how many numbers X from `1` to `N` are good?

```
Example:
Input: 10
Output: 4
Explanation:
There are four good numbers in the range [1, 10] : 2, 5, 6, 9.
Note that 1 and 10 are not good numbers, since they remain unchanged after rotating.
```

Note:

- `N` will be in range `[1, 10000]`.

Solution 1

```java
class Solution {
    public int rotatedDigits(int N) {
        int count = 0;
        for (int i = 1; i <= N; i ++) {
            if (isValid(i)) count ++;
        }
        return count;
    }

    public boolean isValid(int N) {
        /*
         Valid if N contains ATLEAST ONE 2, 5, 6, 9
         AND NO 3, 4 or 7s
         */
        boolean validFound = false;
        while (N > 0) {
            if (N % 10 == 2) validFound = true;
            if (N % 10 == 5) validFound = true;
            if (N % 10 == 6) validFound = true;
            if (N % 10 == 9) validFound = true;
            if (N % 10 == 3) return false;
            if (N % 10 == 4) return false;
            if (N % 10 == 7) return false;
            N = N / 10;
        }
        return validFound;
    }
}
```

written by JLepere2 original link here

Solution 2

Clearly we can do better than O(N) by reuse results from shorter numbers.
Here is my O(len(N)) method. Took me some time in the contest...

```python
class Solution(object):
    def rotatedDigits(self, N):
        """
        :type N: int
        :rtype: int
        """
        same = [1, 2, 2, 2, 2, 2, 2, 2, 3, 3]
        diff = [0, 0, 1, 1, 1, 2, 3, 3, 3, 4]

        def calc(num):
            if len(num)==1:
                return same[int(num)], diff[int(num)]
            lead = int(num[0])
            if lead == 0:
                return calc(num[1:])
            else:
                n_s, n_d = calc('9'*(len(num)-1))
                nxt_same, nxt_diff = calc(num[1:])

                s = same[lead-1] * n_s
                if lead in [0, 1, 8]:
                    s += nxt_same

                d = (same[lead-1]+diff[lead-1]) * n_d + diff[lead-1] * n_s
                if lead in [0, 1, 8]:
                    d += nxt_diff
                if lead in [2, 5, 6, 9]:
                    d += nxt_diff + nxt_same

                return s, d

        return calc(str(N))[1]
```

written by 8939123 original link here

Solution 3

```
class Solution {
public:
    int rotatedDigits(int N) {
        static int goodsame[10] = {1,2,2,2,2,2,2,2,3,3};
        static int gooddiff[10] = {0,0,1,1,1,2,3,3,3,4};
        static int digittype[10]= {0,0,1,2,2,1,1,2,0,1};
        if (N>9999)
            return 0;
        int count=0;
        int foundfirst=0;
        int vartype=0;
        int nn=N;
        int d=nn/1000;
        if (d > 0) {
            foundfirst=1;
            vartype=digittype[d];
            int goods=goodsame[d-1]+gooddiff[d-1];
            count += (goods*7*7*7 - goodsame[d-1]*3*3*3);
        }
        nn -= d*1000;
        d=nn/100;
        if ((!foundfirst || (foundfirst && vartype<2)) && d > 0) {
            foundfirst=1;
            int goods=goodsame[d-1]+gooddiff[d-1];
            count += goods*7*7;
            if (vartype==0)
                count -= goodsame[d-1]*3*3;
            vartype=(digittype[d] > vartype)? digittype[d]:vartype;
        }
        nn -= d*100;
        d=nn/10;
        if ((!foundfirst || (foundfirst && vartype<2)) && d > 0) {
            foundfirst=1;
            int goods=goodsame[d-1]+gooddiff[d-1];
            count += goods*7;
            if (vartype==0)
                count -= goodsame[d-1]*3;
            vartype=(digittype[d] > vartype)? digittype[d]:vartype;
        }
        nn -= d*10;
        d = nn;
        if ((!foundfirst || (foundfirst && vartype<2)) && d >= 0) {
            int goods=goodsame[d]+gooddiff[d];
            count += goods;
            if (vartype==0)
                count -= goodsame[d];
        }
        return count;
    }
};
```

written by spchang2000 original link here