Python ugly backtracking solution

mujroom ★ 69 October 20, 2019 4:13 AM 3.8K VIEWS

61

```
class Solution(object):
    def confusingNumberII(self, N):
        """
        :type N: int
        :rtype: int
        """
        valid = [0,1,6,8,9]
        mapping = {0: 0,1: 1,6: 9,8: 8, 9: 6}

        self.count = 0

        def backtrack(v, rotation,digit):
            if v:
                if v != rotation:
                    self.count += 1
            for i in valid:
                if v*10+i > N:
                    break
                else:
                    backtrack(v*10+i, mapping[i]*digit + rotation, digit*10)

        backtrack(1,1, 10)
        backtrack(6,9,10)
        backtrack(8,8,10)
        backtrack(9,6,10)

        return self.count
```

Comments: 16 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)

Post

hialvin ★ 35 Last Edit: November 14, 2019 5:11 AM

Very impressive solution.
Try to make it not so "ugly" and without the global variable, lol

```
class Solution:
    def confusingNumberII(self, N: int) -> int:
        validDigit = [0, 1, 6, 8, 9]
        m = {0:0, 1:1, 6:9, 8:8, 9:6}
        def dfs(num, rotation, digit):
            res = 0
            if num != rotation: res += 1
            # add one more digit
```

Read More

15 Show 2 replies Reply

ninjaVic ★ 456 January 16, 2020 6:49 AM

This is really smart solution - generating rotated number on the fly. Port it to Java:

```
class Solution {

    private int[] nums = {0,1,6,8,9};
    private Map<Integer,Integer> map;
    int count = 0;
    int limit;

    public int confusingNumberII(int N) {
        limit = N;
```

Read More

12 Show 3 replies Reply

Code_King_666 ★ 6 January 19, 2020 12:36 AM