whissely ★ 48 June 13, 2019 8:21 PM 161 VIEWS

0

Time: O(5 ^ (n/2)) ~ O(2^n) - we have 5 choices for every position and we will do this selection n/2 times (because when we append one character, we also append the corresponding matching one)
Space: recursion stack - O(5 ^ (n/2)) ~ O(2^n)
Algorithm: Check if n is odd or even. For the even case, use DFS to append the (number, pair) to the front and end of the string constructed in all cases except if n == 2 and we hit a 0 (because 0 cannot be the start of the string). If n is greater than 2 then just append to front and end normally. If we have an odd number you can run the even DFS but this time the central number can be 0, 1, or 8 which themselves alone are strobogrammatic, so start the string with this value and run with n - 1.

```
def findStrobogrammatic(self, n):
        """
        :type n: int
        :rtype: List[str]
        """
        self.mapping = {"0": "0", "1": "1", "6": "9", "9": "6", "8": "8"}
        self.res = []
        if n % 2:
            for i in ["0", "8", "1"]:
                self.dfs(str(i), n - 1)
        else:
            self.dfs("", n)
        return self.res

    def dfs(self, num, n):
        if n == 0:
            self.res.append(num)
            return

        for number in self.mapping:
            if n == 2 and number != "0":
                self.dfs(number + num + self.mapping[number], n - 2)
            elif n > 2:
                self.dfs(number + num + self.mapping[number], n - 2)
```