Strobogrammatic Number II

A strobogrammatic number is a number that looks the same when rotated 180 degrees (looked at upside down).

Find all strobogrammatic numbers that are of length = n.

For example,
Given n = 2, return `["11","69","88","96"]`.

1. Try to use recursion and notice that it should recurse with$n$ - 2 instead of n - 1.

Solution 1

```
public List<String> findStrobogrammatic(int n) {
    return helper(n, n);
}

List<String> helper(int n, int m) {
    if (n == 0) return new ArrayList<String>(Arrays.asList(""));
    if (n == 1) return new ArrayList<String>(Arrays.asList("0", "1", "8"));

    List<String> list = helper(n - 2, m);

    List<String> res = new ArrayList<String>();

    for (int i = 0; i < list.size(); i++) {
        String s = list.get(i);

        if (n != m) res.add("0" + s + "0");

        res.add("1" + s + "1");
        res.add("6" + s + "9");
        res.add("8" + s + "8");
        res.add("9" + s + "6");
    }

    return res;
}
```

written by jeantimex original link here

Solution 2

```
public class Solution {
    public List<String> findStrobogrammatic(int n) {
        List<String> one = Arrays.asList("0", "1", "8"), two = Arrays.asList(""),
r = two;
        if(n%2 == 1)
            r = one;
        for(int i=(n%2)+2; i<=n; i+=2){
            List<String> newList = new ArrayList<>();
            for(String str : r){
                if(i != n)
                    newList.add("0" + str + "0");
                newList.add("1" + str + "1");
                newList.add("6" + str + "9");
                newList.add("8" + str + "8");
                newList.add("9" + str + "6");
            }
            r = newList;
        }
        return r;
    }

}
```

written by symgates original link here

Solution 3

```
public class Solution {
    public List<String> findStrobogrammatic(int n) {
        Map<Character, Character> map = new HashMap<Character, Character>();
        map.put('0', '0');
        map.put('1', '1');
        map.put('6', '9');
        map.put('8', '8');
        map.put('9', '6');
        List<String> result = new ArrayList<String>();
        char[] buffer = new char[n];
        dfs(n, 0, buffer, result, map);
        return result;
    }

    private void dfs(int n, int index, char[] buffer, List<String> result, Map<Ch
aracter, Character> map) {
        if (n == 0) {
            return;
        }
        if (index == (n + 1) / 2) {
            result.add(String.valueOf(buffer));
            return;
        }
        for (Character c : map.keySet()) {
            if (index == 0 && n > 1 && c == '0') {  // first digit cannot be '0'
when n > 1
                continue;
            }
            if (index == n / 2 && (c == '6' || c == '9')) {   // mid digit cannot
be '6' or '9' when n is odd
                continue;
            }
            buffer[index] = c;
            buffer[n - 1 - index] = map.get(c);
            dfs(n, index + 1, buffer, result, map);
        }
    }
}
```

written by stevenye original link here