Strobogrammatic Number III

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

Write a function to count the total strobogrammatic numbers that exist in the range of low

For example,
Given low = "50", high = "100", return 3. Because 69, 88, and 96 are three strobogrammatic numbers.

Note:
Because the range might be a large number, the *low* and *high* numbers are represented as string.

Solution 1

The basic idea is to find generate a list of strobogrammatic number with the length between the length of lower bound and the length of upper bound. Then we pass the list and ignore the numbers with the same length of lower bound or upper bound but not in the range.

I think it is not a a very optimized method and can any one provide a better one?

```
public class Solution{


    public int strobogrammaticInRange(String low, String high){
        int count = 0;
        List<String> rst = new ArrayList<String>();
        for(int n = low.length(); n <= high.length(); n++){
            rst.addAll(helper(n, n));
        }
        for(String num : rst){

            if((num.length() == low.length()&&num.compareTo(low) < 0 ) ||(num.len
gth() == high.length() && num.compareTo(high) > 0)) continue;
                count++;
        }
        return count;
    }

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

        List<String> rst = new ArrayList<String>();
        List<String> center = helper(cur - 2, max);

        for(int i = 0; i < center.size(); i++){
            String tmp = center.get(i);
            if(cur != max) rst.add("0" + tmp + "0");
            rst.add("1" + tmp + "1");
            rst.add("6" + tmp + "9");
            rst.add("8" + tmp + "8");
            rst.add("9" + tmp + "6");
        }
        return rst;
    }
}
```

written by czonzhu original link here

Solution 2

Construct char array from `lenLow` to `lenHigh` and increase `count` when `s` is between `low` and `high`. Add the stro pairs from outside to inside until `left > right`.

```
char[][] pairs = {{'0', '0'}, {'1', '1'}, {'6', '9'}, {'8', '8'}, {'9', '6'}};
int count = 0;

public int strobogrammaticInRange(String low, String high) {
    for(int len = low.length(); len <= high.length(); len++) {
        dfs(low, high, new char[len], 0, len - 1);
    }
    return count;
}

public void dfs(String low, String high, char[] c, int left, int right) {
    if(left > right) {
        String s = new String(c);
        if((s.length() == low.length() && s.compareTo(low) < 0) ||
           (s.length() == high.length() && s.compareTo(high) > 0)) return;
        count++;
        return;
    }

    for(char[] p : pairs) {
        c[left] = p[0];
        c[right] = p[1];
        if(c.length != 1 && c[0] == '0') continue;
        if(left < right || left == right && p[0] == p[1]) dfs(low, high, c, left
+ 1, right - 1);
    }
}
```

written by yavinci original link here

Solution 3

My clear Java solution. All comments are welcome.

```
    public class Solution {
Map<Character, Character> map = new HashMap<>();
{
    map.put('1', '1');
    map.put('8', '8');
    map.put('6', '9');
    map.put('9', '6');
    map.put('0', '0');
}
String low = "", high = "";
public int strobogrammaticInRange(String low, String high) {
    this.low = low;
    this.high = high;
    int result = 0;
    for(int n = low.length(); n <= high.length(); n++){
        int[] count = new int[1];
        strobogrammaticInRange(new char[n], count, 0, n-1);
        result += count[0];
    }
    return result;
}
private void strobogrammaticInRange(char[] arr, int[] count, int lo, int hi){
    if(lo > hi){
        String s = new String(arr);
        if((arr[0] != '0' || arr.length == 1) && compare(low, s) && compare(s, hi
gh)){
            count[0]++;
        }
        return;
    }
    for(Character c: map.keySet()){
        arr[lo] = c;
        arr[hi] = map.get(c);
        if((lo == hi && c == map.get(c)) || lo < hi)
            strobogrammaticInRange(arr, count, lo+1, hi-1);
    }
}
private boolean compare(String a, String b){
    if(a.length() != b.length())
        return a.length() < b.length();
    int i = 0;
    while(i < a.length() &&a.charAt(i) == b.charAt(i))
        i++;
    return i == a.length() ? true: a.charAt(i) <= b.charAt(i);
}
```

}

written by hyuna915 original link here