Restore IP Addresses

Given a string containing only digits, restore it by returning all possible valid IP address combinations.

For example:
Given `"25525511135"`,

return `["255.255.11.135", "255.255.111.35"]`. (Order does not matter)

Solution 1

```
public class Solution {
    public List<String> restoreIpAddresses(String s) {
        List<String> res = new ArrayList<String>();
        int len = s.length();
        for(int i = 1; i<4 && i<len-2; i++){
            for(int j = i+1; j<i+4 && j<len-1; j++){
                for(int k = j+1; k<j+4 && k<len; k++){
                    String s1 = s.substring(0,i), s2 = s.substring(i,j), s3 = s.s
ubstring(j,k), s4 = s.substring(k,len);
                    if(isValid(s1) && isValid(s2) && isValid(s3) && isValid(s4)){
                        res.add(s1+"."+s2+"."+s3+"."+s4);
                    }
                }
            }
        }
        return res;
    }
    public boolean isValid(String s){
        if(s.length()>3 || s.length()==0 || (s.charAt(0)=='0' && s.length()>1) ||
Integer.parseInt(s)>255)
            return false;
        return true;
    }
}
```

3-loop divides the string s into 4 substring: s1, s2, s3, s4. Check if each substring is valid. In isValid, strings whose length greater than 3 or equals to 0 is not valid; or if the string's length is longer than 1 and the first letter is '0' then it's invalid; or the string whose integer representation greater than 255 is invalid.

written by fiona_mao original link here

Solution 2

```
public List<String> restoreIpAddresses(String s) {
    List<String> solutions = new ArrayList<String>();
    restoreIp(s, solutions, 0, "", 0);
    return solutions;
}

private void restoreIp(String ip, List<String> solutions, int idx, String restore
d, int count) {
    if (count > 4) return;
    if (count == 4 && idx == ip.length()) solutions.add(restored);

    for (int i=1; i<4; i++) {
        if (idx+i > ip.length()) break;
        String s = ip.substring(idx,idx+i);
        if ((s.startsWith("0") && s.length()>1) || (i==3 && Integer.parseInt(s) >
= 256)) continue;
        restoreIp(ip, solutions, idx+i, restored+s+(count==3?"" : "."), count+1);
    }
}
```

written by greatgrahambini original link here

Solution 3

the basic idea is to make three cuts into the string, separating it into four parts, each part contains 1~3 digits and it must be <255.

```
static List<String> restoreIpAddresses(String s) {
    List<String> ans = new ArrayList<String>();
    int len = s.length();
    for (int i = 1; i <=3; ++i){  // first cut
        if (len-i > 9) continue;
        for (int j = i+1; j<=i+3; ++j){  //second cut
            if (len-j > 6) continue;
            for (int k = j+1; k<=j+3 && k<len; ++k){  // third cut
                int a,b,c,d;                // the four int's seperated by "."
                a = Integer.parseInt(s.substring(0,i));
                b = Integer.parseInt(s.substring(i,j)); // notice that "01" can b
e parsed into 1. Need to deal with that later.
                c = Integer.parseInt(s.substring(j,k));
                d = Integer.parseInt(s.substring(k));
                if (a>255 || b>255 || c>255 || d>255) continue;
                String ip = a+"."+b+"."+c+"."+d;
                if (ip.length()<len+3) continue;  // this is to reject those int'
s parsed from "01" or "00"-like substrings
                ans.add(ip);
            }
        }
    }
    return ans;
}
```

written by cbmbbz original link here