- ## 784. Letter Case Permutation

题目：搜索将字符串中所有字母大小写组成的可能加入list中

Hint：

1.dfs

2.用traverse实现dfs

```
class Solution {
    public List<String> letterCasePermutation(String S) {
        List<String> ans = new ArrayList<>();
        char [] ch = S.toCharArray();
        dfs(ch, 0, ans);
        return ans;
    }
    private void dfs(char[]ch, int pos, List<String>ans){
        if(pos >= ch.length){
            ans.add(new String(ch));
            return;
        }
        if(Character.isLetter(ch[pos])){
            char[] newch1 = new char[ch.length];
            char[] newch2 = new char[ch.length];
            for(int i = 0; i < ch.length; i++){
                newch1[i] = ch[i];
                newch2[i] = ch[i];
            }
            if(Character.isLowerCase(newch1[pos])){
                newch1[pos] = Character.toUpperCase(newch1[pos]);
            }
            else{
                newch1[pos] = Character.toLowerCase(newch1[pos]);
            }
            dfs(newch1, pos + 1, ans);
            dfs(newch2, pos + 1, ans);
        }
        else{
            dfs(ch, pos + 1, ans);
        }
    }
}
```

- 给定一个字符串s，你可以从中删除一些字符，使得剩下的串是一个回文串。如何删除才能使得回文串最长呢？输出需要删除的字符个数。

```
import java.util.*;
public class Main {
    public static void main(String[] argc) {
        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()) {
            String s = sc.nextLine();
            System.out.println(findMax(s));
        }
    }

    public static int findMax(String s) {
        int[][] temp = new int[s.length() + 1][s.length() + 1];
        for(int i = 1; i < s.length() + 1; i++) {
            for(int j = 1; j < s.length() + 1; j++) {
                if(s.charAt(i - 1) == s.charAt(s.length() - j)) {
                    temp[i][j] = temp[i - 1][j - 1] + 1;
                }
                else {
                    temp[i][j] = Math.max(temp[i - 1][j], temp[i][j - 1]);
                }
            }
        }
        return s.length() - temp[s.length()][s.length()];
    }
}
```

- 小Q最近遇到了一个难题：把一个字符串的大写字母放到字符串的后面，各个字符的相对位置不变，且不能申请额外的空间。

```
import java.util.*;
public class Main {
    public static void main(String[] argc) {
        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()) {
            String s = sc.nextLine();
            System.out.println(moveLetter(s));
        }
    }


    public static String moveLetter(String s) {
        char[] temp = s.toCharArray();
        int count = 0;
        for(int i = 0; i < temp.length - count; i++) {
            if(Character.isUpperCase(temp[i])) {
                char tmp = temp[i];
                for(int j = i; j < temp.length - 1; j++) {
                    temp[j] = temp[j + 1];
                }
                temp[temp.length - 1] = tmp;
                count++;
                i--;
            }
        }
        return new String(temp);
    }
}
```