## 剑指 Offer 38. 字符串的排列

子烁

```
class Solution {
    Set<String> resultSet;
    public String[] permutation(String s) {
        resultSet = new TreeSet<>();
        char[] chars = s.toCharArray();
        List<Character> arr = new ArrayList<>();
        for(char c : chars){
            arr.add(c);
        }
        compose(arr, new StringBuilder(), arr.size(), 0);
        String[] result = new String[resultSet.size()];
        int n = 0;
        for(String ss : resultSet){
            result[n] = ss;
            n++;
        }
        return result;
    }

    private void compose(List<Character> arr, StringBuilder builder, int size, int
index){
        if(index == size){
            resultSet.add(builder.toString());
            return;
        }
        for(int i = 0; i < arr.size(); i++){ //循环，依次在这个位置遍历所有之前没有出
现过的字符
            char c = arr.get(i);
            builder.append(c);
            arr.remove(i);  //去除当前遍历中已经使用过的字符传递给诶递归
            compose(arr, builder, size, index + 1);
            arr.add(i, c);  //加回来遍历下一个
            builder.deleteCharAt(builder.length() - 1); //当前循环中builder的一个位
置可能会存在中字符，刚刚加了是传递递归，现在还是当前则需要去掉
        }
    }
}
```

## 剑指 Offer 05. 替换空格

```
class Solution {
    public String replaceSpace(String s) {
        // return s.replace(" ","%20");

        StringBuilder res = new StringBuilder();
```

```java
        for(char ch : s.toCharArray()){
            if(ch == ' ')
                res.append("%20");
            else
                res.append(ch);
        }
        return res.toString();
    }
}
```

## 【回溯】N皇后

~~未成功~~

```java
class Solution {
    List<List<String>> res;
    public List<List<String>> solveNQueens(int n) {
        res = new LinkedList<>();
        List<String> board = new LinkedList<>();
        //初始化
        StringBuilder builder = new StringBuilder();
        int a = 0;
        while(a < n){
            builder.append(".");
        }
        board = Collections.nCopies(n, builder.toString());
        //执行
        backtrack(board, 0);
        return res;
    }

    private void backtrack(List<String> board, int row){
        // 触发结束条件
        if(row == board.size()){
            res.add(board);
            return;
        }

        int n = board.get(row).length();
        for(int col = 0; col < n; col++){
            if(!isValid(board, row, col))
                continue;

            StringBuilder builder = new StringBuilder(board.get(row));

            // 做选择
            builder.replace(col, col + 1, "Q");
            board.remove(row);
            board.add(row,builder.toString());

            // 进入下一行决策
```

```
                backtrack(board, row + 1);

                // 撤销选择
                builder.replace(col, col + 1, ".");
                board.remove(row);
                board.add(row,builder.toString());
            }
        }

    private boolean isValid(List<String> board, int row, int col) {
        // 检查列是否有皇后互相冲突
        int n = board.size();
        for (int i = 0; i < n; i++) {
            char c = board.get(i).charAt(col);
            if (c == 'Q')
                return false;
        }

        // 检查右上方是否有皇后互相冲突
        for (int i = row - 1, j = col + 1; i >= 0 && j < n; i--, j++) {
            char c = board.get(i).charAt(col);
            if (c == 'Q')
                return false;
        }

        // 检查左上方是否有皇后互相冲突
        for (int i = row - 1, j = col - 1;i >= 0 && j >= 0; i--, j--) {
            char c = board.get(i).charAt(col);
            if (c == 'Q')
                return false;
        }

        return true;
    }
}
```

## 面试题 17.17. 多次搜索

```
/**
 * Description
 * Author cloudr
 * Date 2020/8/21 19:21
 * Version 1.0
 **/
public class kmps {

    public static void main(String[] args) {
//        String txt = "mississippi";
//        String pat = "is";
//        List<Integer> res = kmp(txt, pat);
//        for(int re : res){
```

```
//            System.out.println(re);
//        }
//
//        int[][] test = new int[2][];
//        test[0] = new int[]{1, 2, 3};
//        test[1] = new int[]{4,5,6,6,7,8,9};

        String big = "mississippi";
        String[] smalls = {"is","ppi","hi","sis","i","ssippi"};
        int[][] res = multiSearch(big, smalls);
        System.out.println("res.length " + res.length);
        for(int i = 0; i < res.length; i++){
            for(int j = 0; j < res[i].length; j++)
                System.out.println(res[i][j]);
            System.out.println("----");
        }
    }

    public static int[][] multiSearch(String big, String[] smalls) {
        int[][] res = new int[smalls.length][];
        List<List<Integer>> resList = new ArrayList<>();
        for(String small : smalls){
            resList.add(kmp(big, small));
//            for(int num : kmp(big, small)){
//                System.out.println(num);
//            }
//            System.out.println("----");
        }
        for(int i = 0; i < smalls.length; i++){
            List<Integer> levelL = resList.get(i);
            int[] levelA = new int[levelL.size()];
            for(int j = 0; j < levelL.size(); j++){
                levelA[j] = levelL.get(j);
            }
            res[i] = levelA;
        }
        return res;
    }

    private static List<Integer> kmp(String txt, String pat){
        List<Integer> res = new ArrayList<>();
//        if(pat.length() == 0)
//            return 0;
        int[] next = getNext(pat);
        int i = 0, j = 0;
        while(i < txt.length() && j < pat.length()){
            if(j == -1 || txt.charAt(i) == pat.charAt(j)){
                i++;
                j++;
            }
            else{
                j = next[j];
            }
            if(j == pat.length()){
                res.add(i - j);
                j = 0;
                i++;
            }
        }
        return res;

//        if(j == pat.length())
//            return i - j;
//        else
//            return - 1;
    }


    private static int[] getNext(String s){
        char[] arr = s.toCharArray();
        int[] next = new int[arr.length];

        int i = 0, k = -1;
        next[0] = -1;

        while(i < arr.length - 1){  // 0 ~ i-2, next[i-1]是推出来的
            if(k == -1 || arr[i] == arr[k]){
                i++;
                k++;
                next[i] = k;
            }
            else{
                k = next[k];
            }
        }
        return next;
    }
}
```

## 【位运算】剑指 Offer 15. 二进制中1的个数

> 请实现一个函数，输入一个整数，输出该数二进制表示中 1 的个数。例如，把 9 表示成二进制是 1001，有 2 位是 1。因此，如果输入 9，则该函数输出 2。

```
public class Solution {
    // you need to treat n as an unsigned value
    public int hammingWeight(int n) {
        int res = 0;
        for(int i = 0; i < 32; i++){    //整型最大为32位
            if((n & 1) == 1)
                res++;
            n = n >> 1;
        }
        return res;
        }
    }
```