算法总结

斐波那契数

f[n] = f[n-1] + f[n-2] (n > 1)

递归思路

- 例题：给你二叉树的根节点 root 和一个表示目标和的整数 targetSum ，判断该树中是否存在 根节点到叶子节点 的路径，这条路径上所有节点值相加等于目标和 targetSum 。如果存在，返回 true ；否则，返回 false 。

```
public boolean hasPathSum(TreeNode root, int targetSum) {
    if (root == null) return false;
    int num = targetSum - root.val;
    if (root.left == null && root.right == null && num == 0)
        return true;
    if (num<0) return false;
    return hasPathSum(root.left, num) || hasPathSum(root.right, num);
}
```

链表相关的操作

排序

- 简单选择排序
- 插入排序：形如抽扑克
- 冒泡排序：不断进行左右比较交换
- 快速排序：两个字序列划分，大的放右边，小的话左边。要注意使用do~while~还是while~语句，这里的i,j至少要移动一次
- 两路合并排序：将初始数组视作多个升序数组，只不过每个数组中元素只有一个，然后进行合并
- 堆排序：看下面合并中内容。

合并

- 例题：给你一个链表数组，每个链表都已经按升序排列。请你将所有链表合并到一个升序链表中，返回合并后的链表。

```
//方法一：合并两个为一个，然后再与下一个合并

//方法二：两两一组，直到合并为一个

//方法三：每个链表都分配一个指针，指向第一个，把最小的拿出，指针后移（可以使用最小堆）
PriorityQueue<ListNode> minHeap = new PriorityQueue<>((a, b) -> a.val - b.val);//建立最小堆
```

```
minHeap.poll()；//输出并移除最小元素
minHeap.add(listNodeObject);//添加元素
minHeap.isEmpty();//检空
```

单调栈

- 例题：给定 n 个非负整数，用来表示柱状图中各个柱子的高度。每个柱子彼此相邻，且宽度为 1 。求在该柱状图中，能够勾勒出来的矩形的最大面积(难度较大)
- 例题：给定一个整数数组 temperatures ，表示每天的温度，返回一个数组 answer ，其中 answer[i] 是指对于第 i 天，下一个更高温度出现在几天后。如果气温在这之后都不会升高，请在该位置用 0 来代替。(难度适中)

```
public int[] dailyTemperatures(int[] temperatures) {
    int[] result = new int[temperatures.length];
    Stack<Integer> stack = new Stack<>();
    for (int i = 0; i < temperatures.length; i++) {
        if (stack.isEmpty())
            stack.push(i);
        while (temperatures[i] > temperatures[stack.peek()]) {
            result[stack.peek()] =i - stack.pop();
            if(stack.isEmpty()) break;
        }
        stack.push(i);
    }
    while(!stack.isEmpty()){
        result[stack.pop()] = 0;
    }
    return result;
}
```

深度与宽度优先算法(DFS,BFS)

- 寻找最短路径的话用宽度优先算法
- 给你一个由 '1'（陆地）和 '0'（水）组成的的二维网格，请你计算网格中岛屿的数量。岛屿总是被水包围，并且每座岛屿只能由水平方向和/或竖直方向上相邻的陆地连接形成。此外，你可以假设该网格的四条边均被水包围。

> 思路：利用深度宽度优先算法，把已经遍历过的元素标记为`true`，直到所有元素都被遍历，遍历次数就是岛屿数

- 树可以看成是一个连通且 无环 的 无向 图。给定往一棵 n 个节点 (节点值 1～n) 的树中添加一条边后的图。添加的边的两个顶点包含在 1 到 n 中间，且这条附加的边不属于树中已存在的边。图的信息记录于长度为 n 的二维数组 edges， edges[i] = [ai, bi] 表示图中在 ai 和 bi 之间存在一条边。请找出一条可以删去的边，删除后可使得剩余部分是一个有着 n 个节点的树。如果有多个答案，则返回数组 edges 中最后出现的那个。

> 思路：有宽度，深度优先，将可访问的节点遍历，当一天新边连接的是两个已经访问过的节点时，即为冗余连接

滑动窗口

- 例题：给定一个字符串 s ，请你找出其中不含有重复字符的 最长 子串的长度。

```
public int lengthOfLongestSubstring(String s) {
    Map<Character, Integer> map = new HashMap<>();
    int l = -1, length = 0;
    for(int i = 0; i< s.length(); i++){
        if (map.containsKey(s.charAt(i))) l = Math.max(l,
map.get(s.charAt(i)));
        map.put(s.charAt(i), i);
        length = Math.max(length, i - l);
    }
    return length;
}
```