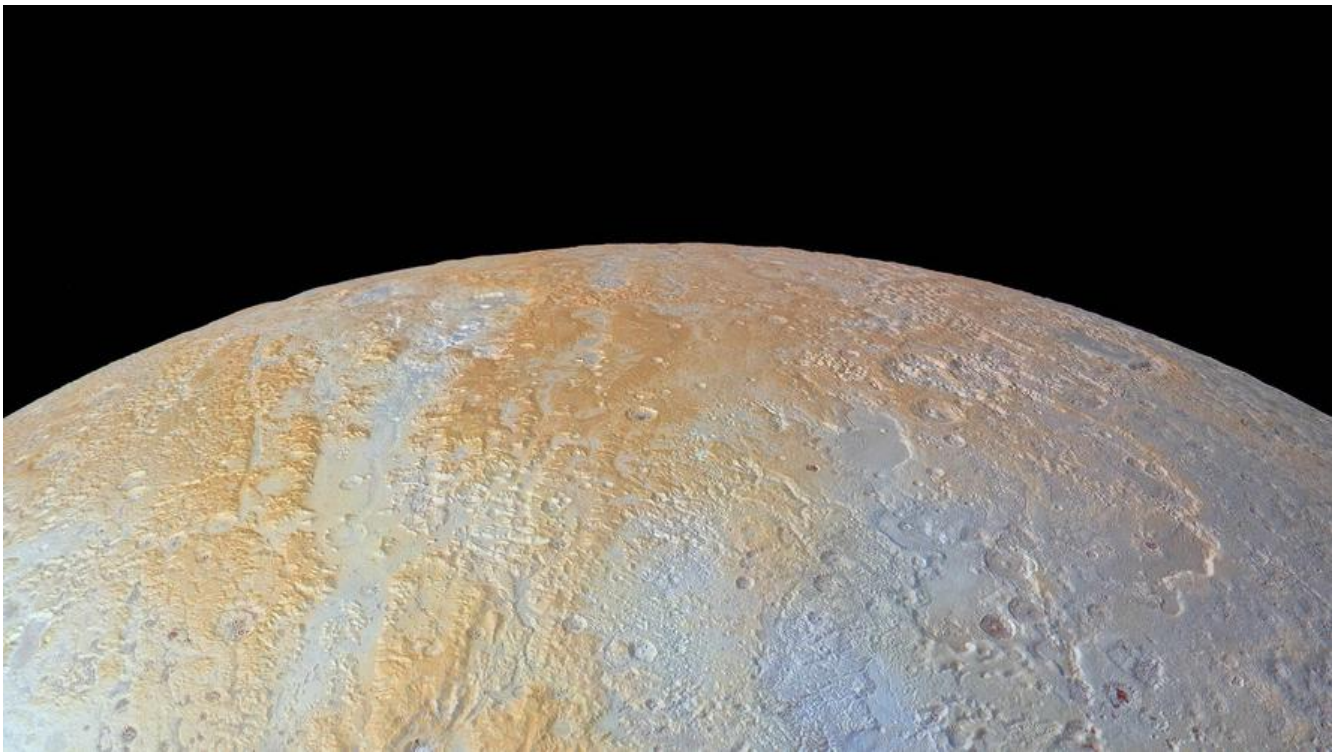
# 算法学习之路 | 分治思想

作者：qq692310342
原文链接：https://ld246.com/article/1566983324279
来源网站：链滴



# Question1 为运算表达式设计优先级

```
/**
 * 给定一个含有数字和运算符的字符串，为表达式添加括号，
 * 改变其运算优先级以求出不同的结果。你需要给出所有可能的组合的结果。
 * 有效的运算符号包含 +, - 以及 * 。
 * <p>
 * 分治思想：
 * 把一个复杂的问题分成两个或更多的相同或相似的子问题，
 * 再把子问题分成更小的子问题……直到最后子问题可以简单的直接求解，原问题的解即子问题的解合并
 */
public class Question1 {
    public List<Integer> diffWaysToCompute(String input) {
        List<Integer> ways = new ArrayList<>();
        for (int i = 0; i < input.length(); i++) {
            char c = input.charAt(i);
            if (c == '+' || c == '-' || c == '*') {
//                分割成若干个子问题
                List<Integer> left = diffWaysToCompute(input.substring(0, i));
                List<Integer> right = diffWaysToCompute(input.substring(i + 1));
                for (int l : left) {
                    for (int r : right) {
                        switch (c) {
                            case '+':
                                ways.add(l + r);
                                break;
                            case '-':
                                ways.add(l - r);
                                break;
                            case '*':
```

```
                        ways.add(l * r);
                        break;
                    }
                }
            }
        }
    }
//      说明无法分割了，直接返回数值
    if (ways.size() == 0) {
        ways.add(Integer.valueOf(input));
    }
    return ways;
  }
}
```

## Question 2 不同的二叉搜索树 II

```
/**
 * 给定一个整数 n，生成所有由 1 ... n 为节点所组成的二叉搜索树。
 *
 * 二叉搜索树的定义：对于数中的每个节点X，它的左子树中所有的项的值小于X，右子树中所有的项值大于X
 */
public class Question2 {
    public List<TreeNode> generateTrees(int n) {
        List<TreeNode> ans = new ArrayList<TreeNode>();
        if (n == 0) {
            return ans;
        }
        return getAns(1, n);

    }

    private List<TreeNode> getAns(int start, int end) {
        List<TreeNode> ans = new ArrayList<TreeNode>();
        //此时没有数字，将 null 加入结果中
        if (start > end) {
            ans.add(null);
            return ans;
        }
        //只有一个数字，当前数字作为一棵树加入结果中
        if (start == end) {
            TreeNode tree = new TreeNode(start);
            ans.add(tree);
            return ans;
        }
        //尝试每个数字作为根节点
        for (int i = start; i <= end; i++) {
//          分治！
            //得到所有可能的左子树
            List<TreeNode> leftTrees = getAns(start, i - 1);
            //得到所有可能的右子树
            List<TreeNode> rightTrees = getAns(i + 1, end);
            //左子树右子树两两组合
```

```
            for (TreeNode leftTree : leftTrees) {
                for (TreeNode rightTree : rightTrees) {
                    TreeNode root = new TreeNode(i);
                    root.left = leftTree;
                    root.right = rightTree;
                    //加入到最终结果中
                    ans.add(root);
                }
            }
        }
        return ans;
    }
}

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}
```

---

END

2019年8月28日17:08:36