# Leetcode 题解 - 分治

* [Leetcode 题解 - 分治](#leetcode-题解---分治)
  + [1. 给表达式加括号](#1-给表达式加括号)
  + [2. 不同的二叉搜索树](#2-不同的二叉搜索树)

## 1. 给表达式加括号

241. Different Ways to Add Parentheses (Medium)

[Leetcode](https://leetcode.com/problems/different-ways-to-add-parentheses/description/) / [力扣](https://leetcode-cn.com/problems/different-ways-to-add-parentheses/description/)

Input: "2-1-1".  
  
((2-1)-1) = 0  
(2-(1-1)) = 2  
  
Output : [0, 2]

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 == '\*') {  
 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;  
 }  
 }  
 }  
 }  
 }  
 if (ways.size() == 0) {  
 ways.add(Integer.valueOf(input));  
 }  
 return ways;  
}

## 2. 不同的二叉搜索树

95. Unique Binary Search Trees II (Medium)

[Leetcode](https://leetcode.com/problems/unique-binary-search-trees-ii/description/) / [力扣](https://leetcode-cn.com/problems/unique-binary-search-trees-ii/description/)

给定一个数字 n，要求生成所有值为 1...n 的二叉搜索树。

Input: 3  
Output:  
[  
 [1,null,3,2],  
 [3,2,null,1],  
 [3,1,null,null,2],  
 [2,1,3],  
 [1,null,2,null,3]  
]  
Explanation:  
The above output corresponds to the 5 unique BST's shown below:  
  
 1 3 3 2 1  
 \ / / / \ \  
 3 2 1 1 3 2  
 / / \ \  
 2 1 2 3

public List<TreeNode> generateTrees(int n) {  
 if (n < 1) {  
 return new LinkedList<TreeNode>();  
 }  
 return generateSubtrees(1, n);  
}  
  
private List<TreeNode> generateSubtrees(int s, int e) {  
 List<TreeNode> res = new LinkedList<TreeNode>();  
 if (s > e) {  
 res.add(null);  
 return res;  
 }  
 for (int i = s; i <= e; ++i) {  
 List<TreeNode> leftSubtrees = generateSubtrees(s, i - 1);  
 List<TreeNode> rightSubtrees = generateSubtrees(i + 1, e);  
 for (TreeNode left : leftSubtrees) {  
 for (TreeNode right : rightSubtrees) {  
 TreeNode root = new TreeNode(i);  
 root.left = left;  
 root.right = right;  
 res.add(root);  
 }  
 }  
 }  
 return res;  
}