> 给定一组不含重复元素的整数数组 nums，返回该数组所有可能的子集（幂集）。

> 说明：解集不能包含重复的子集。 输入: nums = [1,2,3] 输出:

```
[
  [3],
  [1],
  [2],
  [1,2,3],
  [1,3],
  [2,3],
  [1,2],
  []
]
```

```
class Solution {
    public List<List<Integer>> subsets(int[] nums) {
        List<List<Integer>>ans = new ArrayList<>();
        if(nums==null){return ans;}
        dfs(ans,nums,new ArrayList<Integer>(),0);
        return ans;
    }
    private void
dfs(List<List<Integer>>ans,int[]nums,List<Integer>list,int index){
//         terminator
        if(index==nums.length){
            ans.add(new ArrayList<Integer>(list));
            return ;
        }
        dfs(ans,nums,list,index+1);//not pick the number at this index
        list.add(nums[index]);
        dfs(ans,nums,list,index+1);//pick the number at this index
//         restore state
        list.remove(list.size()-1);
    }

}
```

> 给出 n 代表生成括号的对数，请你写出一个函数，使其能够生成所有可能的并且有效的括号组合。

> 例如，给出 n = 3，生成结果为：

```
[
  "((()))",
  "(()())",
  "(())()",
  "()(())",
  "()()()"
```

```
]
/**
 * @param {number} n
 * @return {string[]}
 */
var generateParenthesis = function(n) {
        let arrList = [];
    function generate(left, right, n, s) {
        if (left === n && right === n) {
            arrList.push(s);
            return;
        }
        if (left < n) generate(left + 1, right, n, s + '(');
        if (left > right) generate(left, right + 1, n, s + ')');
    }
    generate(0, 0, n, '');
    return arrList;
};
```