118 杨辉三角

Label：数组

给定一个非负整数 numRows，生成杨辉三角的前 numRows 行。

```
输入: 5
输出:
[
     [1],
    [1,1],
   [1,2,1],
  [1,3,3,1],
 [1,4,6,4,1]
]
```

- 迭代 (有点动态规划的味道)

```
class Solution {
    public List<List<Integer>> generate(int numRows) {

        List<List<Integer>> re = new ArrayList<>();
        if (numRows <= 0) {
            return re;
        }else if(numRows == 1){
            re.add(new ArrayList(Arrays.asList(1)));
            return re;
        }else if (numRows == 2) {
            re.add(new ArrayList(Arrays.asList(1)));
            re.add(new ArrayList(Arrays.asList(1,1)));
            return re;
        }

        re.add(new ArrayList(Arrays.asList(1)));
        re.add(new ArrayList(Arrays.asList(1,1)));

        for (int i = 2; i < numRows; i++) {  // 控制层数，从第三层开始
            List<Integer> curr = new ArrayList<>();
            List<Integer> pre = null;
            pre = re.get(i-1);

            curr.add(1);
            for (int j = 0; j < pre.size()-1 ; j++ ) {  // 行内控制
                curr.add(pre.get(j)+pre.get(j+1));
            }
            curr.add(1);
            re.add(curr);
        }

        return re;
    }
}
```

- 双重循环优化（速度最快）

```
class Solution {
    public List<List<Integer>> generate(int numRows) {
        List<List<Integer>> ret = new ArrayList<List<Integer>>();

        for (int i = 0; i < numRows; ++i) {
            List<Integer> row = new ArrayList<Integer>();
            for (int j = 0; j <= i; ++j) {
                if (j == 0 || j == i) {
                    row.add(1);
                } else {
                    row.add(ret.get(i - 1).get(j - 1) + ret.get(i - 1).get(j));
                }
            }
            ret.add(row);
        }
        return ret;
    }
}
```

- 递归

```
class Solution {
    public List<List<Integer>> generate(int numRows) {
        List<List<Integer>> dg = new ArrayList<>();
        if(numRows == 0) return dg;

        if(numRows == 1) {
            dg.add(new ArrayList<>(Arrays.asList(1)));
            return dg;
        }

        dg = generate(numRows-1);

        List<Integer> row = new ArrayList<>();
        row.add(1);  // 行首
        for (int j = 1; j < numRows - 1; j++) {
            row.add(dg.get(numRows-2).get(j-1) + dg.get(numRows-2).get(j)); // 减
2 是因为下标的原因
        }
        row.add(1);   // 行尾
        dg.add(row);

        return dg;
    }
}
```