1. 题目
2. 题干

给定一个可能包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。

1. 示例

**输入:** [1,2,2]

**输出:**

[

[2],

[1],

[1,2,2],

[2,2],

[1,2],

[]

]

1. 题解
2. 思路

求取子集，想到用递归+回溯，从空的子集开始，按照dfs深度优先搜索的方法一个个地往路径上添加节点。注意，题目中可能包含有重复元素，那么就可能会造成子集的重复。什么叫做子集的重复呢？用树的思想来解释的话，就是在同一层中你使用了和其他节点值相同的节点，也就是说，一层中有节点的值是相同的就是重复。

按照树的思想，每进入一个新的树，就是进入一个新的递归，就要遍历剩余的节点，将他们组建成新的树，即一个for循环，在这个循环中，如何来判断子集的重复呢？即判断同一层中是否有值相同的节点，同一层的节点就是for循环的每个值，如果原数组是有序的话（首先对数组进行排序），那么相同值的节点是连在一起的，那么通过判断当前节点nums[i]的值与前一个节点nums[i-1]的值是否相等即可。但是，这样又会出现问题，因为数组中是有重复的数字的，树的不同层之间的数是可能相等的，这种情况是合理的，而直接判断当前节点nums[i]的值与前一个节点nums[i-1]的值是否相等会把这种情况给排除掉。因此，我们就要想办法只让同一层中的节点进行比较，只要将for循环中第一个节点省略掉即可，从第二个节点开始比较，就不会造成错误了。这样，同一层中就不会有相同值的节点了，子集重复的问题解决了。

如： 下面这种情况，两个2在同一层中，这就是重复的子集。

1

/ \

2 2

/

5

下面这种情况，两个2在不同的层中，这种情况是合理的。

1

/

2

/

2

1. 代码实现

Java:

class Solution {

    public List<List<Integer>> subsetsWithDup(int[] nums) {

        List<List<Integer>> ans = new ArrayList<List<Integer>>();

        Arrays.sort(nums);

        dfs(nums, 0, new ArrayList<Integer>(), ans);

        return ans;

    }

    public void  dfs(int[] nums, int cur, List<Integer> temp, List<List<Integer>> ans){

        ans.add(new ArrayList<Integer>(temp));

        for(int i = cur; i < nums.length; ++i){

            if(i > cur && nums[i] == nums[i-1]){

                continue;

            }

            temp.add(nums[i]);

            dfs(nums, i+1, temp, ans);

            temp.remove(temp.size()-1);

        }

    }

}