给定一组**不同的**整数*nums*，返回所有可能的子集（功率集）。

**注意：**解决方案集不能包含重复的子集。

例如，  
如果***nums*** = [1,2,3]，解决方案是：

[

[3]，

[1]，

[2]，

[1,2,3]，

[1,3]，

[2,3]，

[1,2]，

[]

]

给出的数字序列为1，2，3，无重复。**写出所有子集，不包含重复的子集，包含空集。**

**代码：**

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

List<List<Integer>> list = new ArrayList<>(); //保存最终结果的ArrayList

Arrays.sort(nums); //给数组排序，按照自然顺序

backtrack(list, new ArrayList<>(), nums, 0); //递归开始，从nums[0]开始

//new ArrayList<>(),用于保存单个的结果的

return list;

}

private void backtrack(List<List<Integer>> list , List<Integer> tempList, int [] nums, int start){

list.add(new ArrayList<>(tempList));//添加上一次产生的结果

for(int i = start; i < nums.length; i++){//从start开始遍历

tempList.add(nums[i]); //遍历一个数据，就在当前结果中添加一个数据

backtrack(list, tempList, nums, i + 1); //遍历自己的子集

//回溯过程

tempList.remove(tempList.size() - 1); //遍历完自己子集将自己删除掉

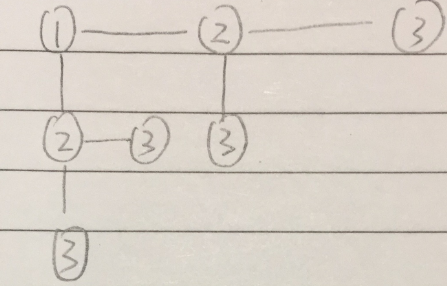
}

}

**示例**

给出的序列为1，2，3

可以将1，2，3的递归过程转化成



如图所示，横着为同级别，竖着的为子集

先访问子集，访问完子集，删除，然后访问同集

1.i开始=1，先访问1 [1]

2.进入1的子集2，访问2 [1、2]

3.进入2的子集3，访问3 [1、2、3]

4.3后边无子集，结束，删除，回溯到2 1、2

5.2无子集，访问结束，删除 1

6.访问2的同级3 [1、3]

7.3无子集访问结束删除3，回溯到1，1子集全部访问结束，删除1

8.访问1的同级2 [2]

9.访问2的子集3 [2、3]

10.3无子集访问结束删除，回溯到2 2

11.2无子集，访问结束，删除

12.访问同级3 [3]

13.3无子集访问结束，删除3

14.3无同级，整体访问结束。 []

每次访问前才会添加结果到list中。

**当给出序列包含重复元素的时候**

如果***nums*** = [1,2,2]，解决方案是：

[

[2]，

[1]，

[1,2,2]

[2,2]，

[1,2]，

[]

]

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

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

Arrays.sort(nums);

backtrack(list, new ArrayList<>(), nums, 0);

return list;

}

private void backtrack(List<List<Integer>> list, List<Integer> tempList, int [] nums, int start){

list.add(new ArrayList<>(tempList));

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

if(i > start && nums[i] == nums[i-1]) //i>start,说明是同级且不为起点

continue; // 当不是起点的同级，和前一个数据项相同的时候跳过

tempList.add(nums[i]);

backtrack(list, tempList, nums, i + 1);

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

}

}

**我们需要跳过同级重复的数据项。**