

思路，我们对于某个元素，只有两种选择，要么选，要么不选，这是一种思路，but it is kinda slow.

Cause the pseudo code will be like

result.add(current)

curr.add(nums[i])

Recursive(current,index+1,…..)

curr.remove(last)

recur(current,index+1)….

Then the remove one assume it always work

will add multiple '[]' into arrayList, will cause duplicate

class Solution {

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

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

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

subsetsRecursive(nums,0,curr,result);

Set<List<Integer>> temp=new HashSet<>(result);

result.clear();

result.addAll(temp);

return result;

}

public void subsetsRecursive(int[] nums,int index,List<Integer> curr,List<List<Integer>> Result){

Result.add(new ArrayList<>(curr));

if(index==nums.length){

return;

}

curr.add(nums[index]);

subsetsRecursive(nums,index+1,curr,Result);

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

subsetsRecursive(nums,index+1,curr,Result);

}

}

Another WAY::Recursive

1 2 3

2 3 3

3

need a for loop to build all three trees,

every tree, it potential child will be [i=index,nums.length-1)

class Solution {

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

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

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

subsetsRecursive(nums,0,curr,result);

return result;

}

public void subsetsRecursive(int[] nums,int index,List<Integer> curr,List<List<Integer>> Result){

Result.add(new ArrayList<>(curr));

if(index==nums.length){

return;

}

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

curr.add(nums[i]);

subsetsRecursive(nums,i+1,curr,Result); //use i+1 as new index, cause index should be bigger than current i

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

}

}

}