子集 II · Subsets II

中文

脸书 (/problem/?tags=facebook) 回溯法 (/problem/?tags=backtracking) 递归 (/problem/?tags=recursion)

描述

给定一个可能具有重复数字的列表，返回其所有可能的子集。

ℹ * 子集中的每个元素都是非降序的 * 两个子集间的顺序是无关紧要的 * 解集中不能包含重复子集

样例

样例 1：

```
输入：[0]
输出：
[
  [],
  [0]
]
```

样例 2：

```
输入：[1,2,2]
输出：
[
  [2],
  [1],
  [1,2,2],
  [2,2],
  [1,2],
  []
]
```

挑战

你可以同时用递归与非递归的方式解决么？

在线评测地址: https://www.lintcode.com/problem/subsets-ii/ (https://www.lintcode.com/problem/subsets-ii/)

收起题目描述 ^

语言类型 ALL (42) python (19) java (11) cpp (10) javascript (1) golang (1) 上传题解

令狐冲
更新于 12/20/2020, 8:47:00 PM

九章算法班中的方法

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param nums: A set of numbers.
    @return: A list of lists. All valid subsets.
    """
    def subsetsWithDup(self, nums):
        nums = sorted(nums)
        subsets = []
        self.dfs(nums, 0, [], subsets)
        return subsets

    def dfs(self, nums, index, subset, subsets):
        subsets.append(list(subset))

        for i in range(index, len(nums)):
            if i > index and nums[i] == nums[i - 1]:
                continue
            subset.append(nums[i])
            self.dfs(nums, i + 1, subset, subsets)
            subset.pop()
```


九章-小原
更新于 12/3/2020, 1:54:24 AM

解题思路

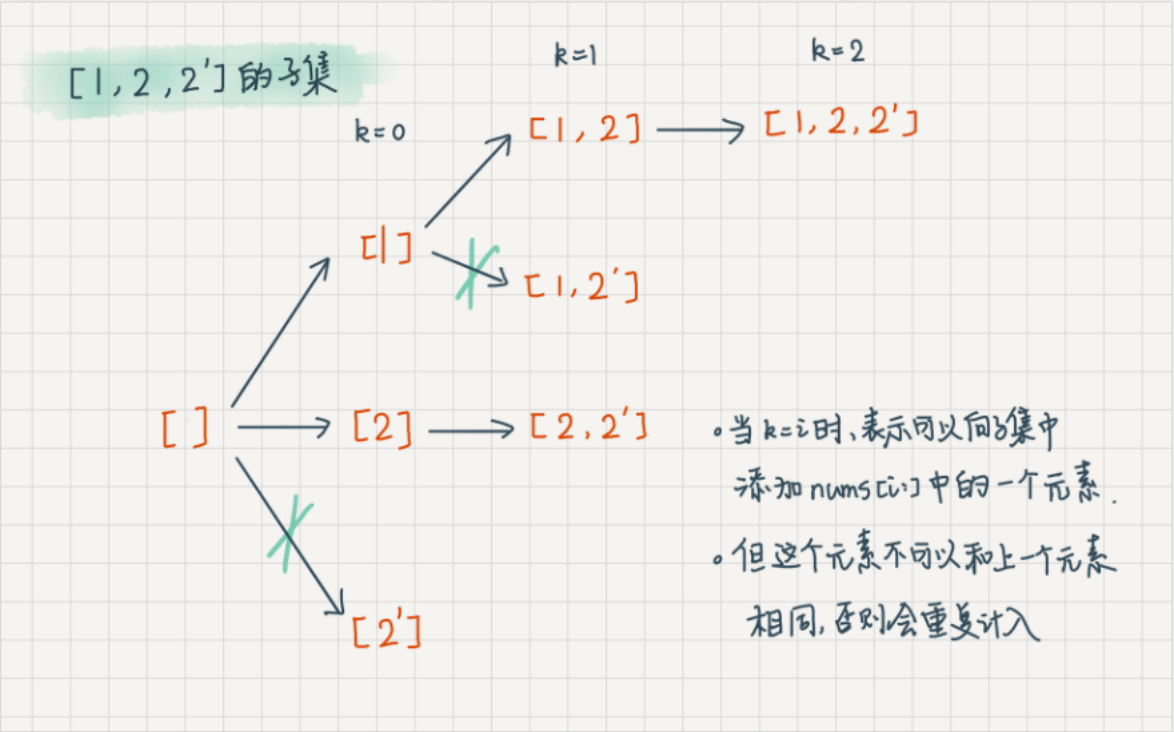
- 这道题我们需要使用dfs+回溯的方法来进行求解。
- 由于题目明确指出列表中可能有重复数字，所以我们在dfs的时候要进行剪枝。

算法

- 将数组进行升序排列。
- 定义一个递归方法 `dfs`，参数有：当前子集 `subset`，当前子集长度 `k`，返回结果 `res`。*将当前子集添加到 `res` 中。* 从 `k` 到 `len(nums)-1` 遍历索引 `i`。*如果 `i != k` 并且 `nums[i] == nums[i - 1]`，说明 `nums[i]` 是重复元素，所以我们跳过 `nums[i]`。* 将 `nums[i]` 添加到当前子集 `subset`。*进行下一层的递归搜索，继续向子集中添加元素，这时 `k` 要加一。* 从 `subset` 中删除 `nums[i]` 进行回溯。

举例分析

- 假设 `nums = [1, 2, 2']`，递归树如图所示。树每深一层，子集的长度就加一。每个节点都是满足条件的子集，需要记录到结果 `res` 中。
- 其中标叉的地方进行了剪枝。由于数组中有两个2，所以如果两者在同一层，只保留第一个。



复杂度分析

- 时间复杂度：$O(n * 2^n)$，其中 n 为 nums 的长度。生成所有子集，并复制到输出集合中。
- 空间复杂度：$O(n * 2^n)$，其中 n 为 nums 的长度。存储所有子集，共 n 个元素，每个元素都有可能存在或者不存在。

代码

python java c++

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param nums: A set of numbers.
    @return: A list of lists. All valid subsets.
    """
    def subsetsWithDup(self, nums):
        res = []
        # 排序
        nums.sort()
        # dfs搜索
        self.dfs(nums, 0, [], res)
        return res

    def dfs(self, nums, k, subset, res):
        # 当前组合存入res
        res.append(subset[:])
        # 为subset新增一位元素
        for i in range(k, len(nums)):
            # 剪枝
            if (i != k and nums[i] == nums[i - 1]):
                continue
            subset.append(nums[i])
            # 下一层搜索
            self.dfs(nums, i + 1, subset, res)
            # 回溯
            del subset[-1]
```

获赞 8　　添加评论

令狐冲
更新于 6/9/2020, 7:03:47 AM

Given a collection of integers that might contain duplicates, S, return all possible subsets.

Note: Elements in a subset must be in non-descending order. The solution set must not contain duplicate subsets. For example, If S = 1,2,2 (), a solution is:

[2 (), 1 (), 1,2,2 (), 2,2 (), 1,2 (), []]

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
// return List<List<Integer>>
class Solution {
    /**
     * @param nums: A set of numbers.
     * @return: A list of lists. All valid subsets.
     */
    public List<List<Integer>> subsetsWithDup(int[] nums) {
```

```
        // write your code here
        List<List<Integer>> results = new ArrayList<List<Integer>>();
        if (nums == null) return results;

        if (nums.length == 0) {
            results.add(new ArrayList<Integer>());
            return results;
        }
        Arrays.sort(nums);

        List<Integer> subset = new ArrayList<Integer>();
        helper(nums, 0, subset, results);

        return results;


    }
    public void helper(int[] nums, int startIndex, List<Integer> subset, List<List<Integer>> results){
        results.add(new ArrayList<Integer>(subset));
        for(int i=startIndex; i<nums.length; i++){
            if (i != startIndex && nums[i]==nums[i-1]) {
                continue;
            }
            subset.add(nums[i]);
            helper(nums, i+1, subset, results);
            subset.remove(subset.size()-1);
        }
    }
}

// return ArrayList<ArrayList<Integer>>
class Solution {
    /**
     * @param nums: A set of numbers.
     * @return: A list of lists. All valid subsets.
     */
    public List<List<Integer>> subsetsWithDup(int[] nums) {
        // write your code here
        List<List<Integer>> results = new ArrayList<>();
        if (nums == null) return results;

        if (nums.length == 0) {
            results.add(new ArrayList<Integer>());
            return results;
        }
        Arrays.sort(nums);

        ArrayList<Integer> subset = new ArrayList<>();
        helper(nums, 0, subset, results);

         return results;


    }
    public void helper(int[] nums, int startIndex, List<Integer> subset, List<List<Integer>> results){
        results.add(new ArrayList<Integer>(subset));
        for(int i=startIndex; i<nums.length; i++){
            if (i != startIndex && nums[i]==nums[i-1]) {
                continue;
            }
            subset.add(nums[i]);
            helper(nums, i+1, subset, results);
            subset.remove(subset.size()-1);
        }
    }
}
```

获赞 4 3 条评论

令狐冲
更新于 8/23/2020, 11:36:06 PM

用 "取或不取" 的方式进行 DFS 搜索的模板

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /**
     * @param nums: A set of numbers.
     * @return: A list of lists. All valid subsets.
     */
    public List<List<Integer>> subsetsWithDup(int[] nums) {
        List<List<Integer>> subsets = new ArrayList<>();

        Arrays.sort(nums);
        dfs(nums, 0, -1, new ArrayList<>(), subsets);

        return subsets;
    }

    private void dfs(int[] nums,
                     int index,
                     int lastSelectedIndex,
                     List<Integer> subset,
                     List<List<Integer>> subsets) {
        if (index == nums.length) {
            subsets.add(new ArrayList<Integer>(subset));
            return;
        }

        dfs(nums, index + 1, lastSelectedIndex, subset, subsets);

        if (index > 0 && nums[index] == nums[index - 1] && index - 1 != lastSelectedIndex) {
            return;
        }
        subset.add(nums[index]);
        dfs(nums, index + 1, index, subset, subsets);
        subset.remove(subset.size() - 1);
    }
}
```

获赞 2 添加评论

九章用户5173ZY
更新于 6/9/2020, 7:03:49 AM

Non-recursive version.

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param nums: A set of numbers.
    @return: A list of lists. All valid subsets.
    """
    def subsetsWithDup(self, nums):
        nums.sort()
        subsets = [[]]
        indexes = [-1]  # 记录subsets中每个集合结尾元素的下标

        for i in range(len(nums)):
            size = len(subsets)
            for s in range(size):
                if i > 0 and nums[i] == nums[i-1] and indexes[s] != i-1:
                    continue  # 去重，如果有重复数字出现，只有前上一个数字选了才能选当前数字
                subsets.append(list(subsets[s]))
                subsets[-1].append(nums[i])
                indexes.append(i)

        return subsets
```


获赞 2　　5 条评论

令狐冲
更新于 6/23/2020, 11:17:57 PM

Given a collection of integers that might contain duplicates, S, return all possible subsets.

Note: Elements in a subset must be in non-descending order. The solution set must not contain duplicate subsets. For example, If S = 1,2,2 (), a solution is:

[2 (), 1 (), 1,2,2 (), 2,2 (), 1,2 (), []]

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    def subsetsWithDup(self, S):
        # write your code here
        S.sort()
        p = [[S[x] for x in range(len(S)) if i>>x&1] for i in range(2**len(S))]
        func = lambda x,y:x if y in x else x + [y]
        p = reduce(func, [[], ] + p)
        return list(reversed(p))
```

获赞 1　　3 条评论

九章用户5173ZY
更新于 6/9/2020, 7:03:54 AM

Non-recursive version.

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    /**
     * @param nums: A set of numbers.
     * @return: A list of lists. All valid subsets.
     */
    vector<vector<int>> subsetsWithDup(vector<int> &nums) {
        sort(nums.begin(), nums.end());
        vector<vector<int>> subsets = {{}};
        vector<int> indexes = {-1}; // 记录subsets中每个集合结尾元素的下标

        for (int i = 0; i < nums.size(); ++i) {
            int size = subsets.size();
            for (int s = 0; s < size; ++s) {
                if (i > 0 && nums[i] == nums[i-1] && indexes[s] != i-1) {
                    continue;  // 去重，如果有重复数字出现，只有前上一个数字选了才能选当前数字
                }
                subsets.push_back(subsets[s]);
                subsets.back().push_back(nums[i]);
                indexes.push_back(i);
            }
        }

        return subsets;
    }
};
```

获赞 1　　添加评论

助教-咖啡蛇
更新于 12/29/2020, 10:49:45 PM

使用哈希表去重的解法

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param nums: A set of numbers.
    @return: A list of lists. All valid subsets.
    """
    def subsetsWithDup(self, nums):
        subsets = []
        visited = {}
        nums.sort()
        self.dfs(nums, 0, [], subsets, visited)

        return subsets

    def get_hash(self, subset):
        hash_string = "-".join([str(num) for num in subset])
        return hash_string

    def dfs(self, nums, start_index, subset, subsets, visited):
        hash_string = self.get_hash(subset)
        if hash_string in visited:
            return;

        visited[hash_string] = True
        subsets.append(list(subset))
        for i in range(start_index, len(nums)):
            subset.append(nums[i])
            self.dfs(nums, i + 1, subset, subsets, visited)
            subset.pop()
```

获赞 0 添加评论

令狐冲
更新于 6/9/2020, 7:04:30 AM

Given a collection of integers that might contain duplicates, S, return all possible subsets.

Note: Elements in a subset must be in non-descending order. The solution set must not contain duplicate subsets. For example, If S = 1,2,2 (), a solution is:

[2 (), 1 (), 1,2,2 (), 2,2 (), 1,2 (), []]

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
#include<vector>
#include<algorithm>
using namespace std;


class Solution {
public:
    vector<vector<int> > result;
    bool equls( vector<int> a, vector<int> b){
        if (a.size() != b.size())
            return false;
        int cnt = a.size();
        for (int i = 0; i < cnt;++i)
            if (a[i]!=b[i]) return false;
        return true;
    }
    void dfs(vector<int> tmp, int x , vector<int> nums) {
        if (x == nums.size()) {
            int cnt = result.size();
            for ( int i = 0; i  < cnt ; ++i)
                if ( equls(result[i], tmp) )
                        return ;
            result.push_back(tmp);
            return ;
        }
        dfs(tmp, x + 1, nums);
        tmp.push_back(nums[x]);
        dfs(tmp, x + 1, nums);
    }
    vector<vector<int> > subsetsWithDup(vector<int> &nums){
        sort(nums.begin(), nums.end());
        vector<int> tmp;
        dfs(tmp, 0 , nums) ;
        // write your code here
        return result;
    }
};
```

获赞 0　　添加评论

九章用户5173ZY
更新于 6/9/2020, 7:04:17 AM

Non-recursive version.

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /**
     * @param nums: A set of numbers.
     * @return: A list of lists. All valid subsets.
     */
    public List<List<Integer>> subsetsWithDup(int[] nums) {
        List<List<Integer>> subsets = new ArrayList<List<Integer>>();
        ArrayList<Integer> indexes = new ArrayList<Integer>(); // 记录subsets中每个集合结尾元素的下标

        Arrays.sort(nums);
        subsets.add(new ArrayList<Integer>());
        indexes.add(-1);

        for (int i = 0; i < nums.length; ++i) {
            int size = subsets.size();
            for (int s = 0; s < size; ++s) {
                if (i > 0 && nums[i] == nums[i-1] && indexes.get(s) != i-1) {
                    continue;  // 去重，如果有重复数字出现，只有前上一个数字选了才能选当前数字
                }
                subsets.add(new ArrayList<Integer>(subsets.get(s)));
                subsets.get(subsets.size()-1).add(nums[i]);
                indexes.add(i);
            }
        }

        return subsets;
    }
}
```

获赞 0 添加评论

九章用户OPC79M
更新于 6/9/2020, 7:03:46 AM

continue the loop if duplicate found logic is the same as the origianl subset

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution(object):
    def subsetsWithDup(self, nums):
      self.ans = []
      nums.sort()
      self.dfs(nums, [])
      return self.ans

    def dfs(self, nums, path):
      self.ans.append(path)
      for i, n in enumerate(nums):
        if i > 0 and n == nums[i-1]:
          continue
        self.dfs(nums[i+1:], path + [n])
```

获赞 5　　添加评论

Tin
更新于 8/6/2020, 5:13:27 AM

如果你计划用深搜的办法解这个题，你应该以经做了之前第17题，知道深搜怎么用。

这道题的唯一难点在于如何去重，所有其它解中，去重的逻辑都一样但程序并不直接易懂。这里 展示了一个不同的视角，那就是，如果上一次递归调用已经用过的数字和for循环里下一个数字 相同，那就要把这个数字跳过去，否则就会得到重复的结果。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param nums: A set of numbers.
    @return: A list of lists. All valid subsets.
    """
    def subsetsWithDup(self, nums):
        subsets = []

        self.dfsWithDedup(sorted(nums), 0, [], subsets)

        return subsets

    def dfsWithDedup(self, nums, startIndex, workingSet, subsets):
        subsets.append(list(workingSet))

        popped = None
        for i in range(startIndex, len(nums)):
            if nums[i] == popped:
                continue
            workingSet.append(nums[i])
            self.dfsWithDedup(nums, i+1, workingSet, subsets)
            popped = workingSet.pop()
```

(/)