喜刷刷

Monday, November 17, 2014

[LeetCode] Subsets I, II

Subsets I

Given a set of distinct integers, *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*** = `[1,2,3]` , a solution is:

```
[
  [3],
  [1],
  [2],
  [1,2,3],
  [1,3],
  [2,3],
  [1,2],
  []
]
```

Subsets II

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],
  []
]
```

思路：Subsets I

方法1：backtracking

与combination/combination sum I, II思路一样。区别在于单层扫描时不用跳过重复数字，而在进入下一层递归前就需要把当前subset压入结集中。

```
class Solution {
public:
    vector<vector<int> > subsets(vector<int> &S) {
        vector<vector<int>> allSets;
        vector<int> sol;
        allSets.push_back(sol);
        sort(S.begin(),S.end());
        findSubsets(S, 0, sol, allSets);
        return allSets;
    }

    void findSubsets(vector<int> &S, int start, vector<int> &sol, vector<vector<int>> &allSets) {
        for(int i=start; i<S.size(); i++) {
            sol.push_back(S[i]);
            allSets.push_back(sol);
            findSubsets(S, i+1, sol, allSets);
            sol.pop_back();
        }
    }
};
```

方法2：添加数字构建subset

起始subset集为：[]
添加S0后为：[], [S0]
添加S1后为：[], [S0], [S1], [S0, S1]
添加S2后为：[], [S0], [S1], [S0, S1], [S2], [S0, S2], [S1, S2], [S0, S1, S2]
红色subset为每次新增的。显然规律为添加Si后，新增的subset为克隆现有的所有subset，并在它们后面都加上Si。

```
class Solution {
public:
    vector<vector<int> > subsets(vector<int> &S) {
        vector<vector<int>> allSets;
        vector<int> sol;
        allSets.push_back(sol);
        sort(S.begin(), S.end());
        for(int i=0; i<S.size(); i++) {
            int n = allSets.size();
            for(int j=0; j<n; j++) {
                sol = allSets[j];
                sol.push_back(S[i]);
                allSets.push_back(sol);
            }
        }
        return allSets;
    }
};
```

方法3：bit manipulation

由于S[0: n-1]组成的每一个subset，可以看成是对是否包含S[i]的取舍。S[i]只有两种状态，包含在特定subset内，或不包含。所以subset的数量总共有2^n
可以用0~2^n-1的二进制来表示一个subset。二进制中每个0/1表示该位置的S[i]是否包括在当前subset中。

```
class Solution {
public:
    vector<vector<int> > subsets(vector<int> &S) {
        vector<vector<int>> allSets;
        sort(S.begin(), S.end());
        unsigned long long maxNum = pow(2, S.size()) - 1;
        for(unsigned long long i=0; i<=maxNum; i++)
            allSets.push_back(num2subset(S, i));
        return allSets;
    }

    vector<int> num2subset(vector<int> &S, unsigned long long num) {
        vector<int> sol;
        int i=0;
        while(num) {
            if(num & 1) sol.push_back(S[i]);
            num >>= 1;
            i++;
        }
        return sol;
    }
};
```

思路：Subsets II

和3sum, combination sum II一样的去重思路。这类问题两个细节千万不能粗心：

1. 一定要先排序
2. 调用下一层递归时(ln 17)，起始index 一定是i+1而不能错写成start+1

```
 1 class Solution {
 2 public:
 3     vector<vector<int> > subsetsWithDup(vector<int> &S) {
 4         vector<vector<int>> allSets;
 5         vector<int> sol;
 6         allSets.push_back(sol);
 7         sort(S.begin(), S.end());
 8         findSubsetsWithDup(S, 0, sol, allSets);
 9         return allSets;
10     }
11 
12     void findSubsetsWithDup(vector<int> &S, int start, vector<int> &sol, vector<vector<int>> &allSets) {
13         for(int i=start; i<S.size(); i++) {
14             if(i>start && S[i]==S[i-1]) continue;
15             sol.push_back(S[i]);
16             allSets.push_back(sol);
17             findSubsetsWithDup(S, i+1, sol, allSets);
18             sol.pop_back();
19         }
20     }
21 };
```

Posted by Yanbing Shi at 8:23 PM G+1 +3 Recommend this on Google

Labels: algorithm, backtracking, bit manipulation, Leetcode

2 comments:

Unknown December 19, 2016 at 3:10 AM

subset 1 不需要 sort吧

Reply

Replies

Shaosong Li January 14, 2017 at 9:19 PM

需要的，因为题目要求输出结果为升序排列。

Reply

Enter your comment...

Comment as: Unknown (Goog ⬍ Si

Publish Preview ☐ I

Newer Post Home

Subscribe to: Post Comments (Atom)