

```

1  /*
2  给定一个可包含重复数字的序列，返回所有不重复的全排列。
3
4  示例：
5
6  输入：[1,1,2]
7  输出：
8  [
9      [1,1,2],
10     [1,2,1],
11     [2,1,1]
12 ]
13
14 来源：力扣（LeetCode）
15 链接：https://leetcode-cn.com/problems/permutations-ii
16 著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
17 */

```

分析:

- 与[046 全排列](#)思想类似,都是一个数字一个数字的填,直到填充缓存的长度等于数组的长度就保存一遍.
- 考虑到与[046 全排列](#)不同的是,这个数组里面是有重复元素的,难点在于如何去除重复的排列.
- 第一种思路是先全排列再利用set去重,虽然很慢,但是可以学习到set要怎么用.
- 第二种思路是先对原始数组进行排序,最简单的情况的就是2个重复,遇到重复两个的第一个正常explore,遇到重复两个的第二个就直接unchoose或者不进入choose流程,这样就能筛选出来了.
- 方法二致谢:[happygirlzt](#)

方法一:C++\_不筛选回溯+set去重.

```

1  class Solution
2  {
3
4      private:
5
6          void helper(    set<vector<int>>&    svi        ,
7                        vector<int>&        cur        ,
8                        vector<int>&        nums        ,
9                        vector<int>&        visited
10                     )
11      {
12          if(cur.size() == nums.size())
13          {
14              svi.insert(cur);
15              return;
16          }
17
18          for(int i = 0 ; i < nums.size() ; i++)
19          {
20

```

```

21         if(visited[i])
22         {
23             continue;
24         }
25         else
26         {
27             cur.push_back(nums[i]);           // choose
28             visited[i] = 1;
29             helper(svi,cur,nums,visited);     // explore
30             visited[i] = 0;                   // unchoose
31             cur.pop_back();
32         }
33     }
34 }
35
36 public:
37
38     vector<vector<int>> permuteUnique(vector<int>& nums)
39     {
40         vector<vector<int>> vvi;
41         set<vector<int>> svi;
42         vector<int> cur;
43         vector<int> visited(nums.size(),0);
44
45         if(nums.size() < 1)
46         {
47             return vvi;
48         }
49         helper(svi, cur, nums , visited);
50
51         set<vector<int>> ::iterator iter = svi.begin();
52         while(iter != svi.end())
53         {
54             vvi.push_back(*iter);
55             iter++;
56         }
57
58         return vvi;
59     }
60 };
61
62 /*
63 执行结果:
64 通过
65 显示详情
66 执行用时 :208 ms, 在所有 cpp 提交中击败了11.33% 的用户
67 内存消耗 :10.7 MB, 在所有 cpp 提交中击败了53.01%的用户
68 */
69

```

方法二:C++\_排序+筛选回溯

```

1  class Solution
2  {
3      private:
4          void helper(    vector<vector<int>>&    vvi,

```

```

5         vector<int>& cur,
6         vector<int>& nums,
7         vector<int>& visited
8     )
9 {
10     if(cur.size() == nums.size())
11     {
12         vvi.push_back(cur);
13         return;
14     }
15
16     for(int i = 0 ; i < nums.size() ; i++)
17     {
18         if(visited[i])
19         {
20             continue;
21         }
22
23         /* 跳过与上一个重复的,最基本的重复情况就是重复两个*/
24         if(i > 0 && nums[i] == nums[i-1] && !visited[i-1]) //
25         {
26             continue;
27         }
28
29         cur.push_back(nums[i]);          /* choose*/
30         visited[i] = 1;
31         helper(vvi,cur,nums,visited);    /* explore*/
32         visited[i] = 0;                  /* unchoose*/
33         cur.pop_back();
34
35     }
36
37 }
38
39 public:
40     vector<vector<int>> permuteUnique(vector<int>& nums)
41     {
42         vector<vector<int>> vvi          ;
43         vector<int> cur                  ;
44         vector<int> visited(nums.size(),0) ;
45         if(nums.size() < 1)
46         {
47             return vvi;
48         }
49         /*先排序一下*/
50         sort(nums.begin(),nums.end());
51         helper(vvi,cur,nums,visited);
52         return vvi;
53     }
54 }
55 };
56
57 /*
58 执行结果:
59 通过
60 显示详情
61 执行用时 :28 ms, 在所有 cpp 提交中击败了93.70% 的用户
62 内存消耗 :9.8 MB, 在所有 cpp 提交中击败了93.27%的用户

```

AlimyBreak  
2019.11.23