```
/*
给定一组不含重复元素的整数数组 nums，返回该数组所有可能的子集（幂集）。

说明：解集不能包含重复的子集。

示例:

输入: nums = [1,2,3]
输出:
[
  [3],
  [1],
  [2],
  [1,2,3],
  [1,3],
  [2,3],
  [1,2],
  []
]

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/subsets
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。
*/
```

分析:

- 方法一:根据二项式展开公式,若$nums$的长度为n,则子集合的个数共有2^n.我们可以用二进制的1或0表示在子集中是否包含.
  - 缺点: $unsigned int$只能表示32个二进制位,只能表示$n \leq 32$的数组长度的全集.
  - 上面的缺点可以根据数据的长度申请更多的二级制位进行表示.
- 方法二:回溯法
  - 递归遍历每个数据,每个数据又分为选中和没有选中两种情况分别递归,直到数组的所有元素完成递归遍历.
  - 致谢happygirlzt

方法一:C++_二进制位遍历

```
class Solution
{
    public:
        vector<vector<int>> subsets(vector<int>& nums)
        {
            vector<vector<int>> ret_val                         ;
            vector<int>         temp                            ;
            unsigned long int   i           = 0                 ;
            unsigned char       j           = 0                 ;
```

```
            unsigned long int   size        = nums.size()       ;
            unsigned long int   size_pow    = pow(2, size)      ;

            for (i = 0; i < size_pow; i++)
            {
                temp.clear();
                for (j = 0; j < size; j++)
                {
                    if (i & (0x01 << j))
                    {
                        temp.push_back(nums[j]);
                    }
                }
                ret_val.push_back(temp);
            }
            return ret_val;
        }
};

/*
执行结果：
通过
显示详情
执行用时 :8 ms, 在所有 C++ 提交中击败了91.24% 的用户
内存消耗 :9 MB, 在所有 C++ 提交中击败了84.04%的用户
*/
```

方法二:C++_回溯法

```
// https://leetcode-cn.com/problems/subsets/submissions/
// https://www.bilibili.com/video/av76286065

class Solution
{
    private:
        void helper(    vector<vector<int>>&    vvi,
                        vector<int>&            cur,
                        vector<int>&            nums,
                        int                     index
                    )
        {
            if(index == nums.size())
            {
                vvi.push_back(cur);
                return;
            }

            cur.push_back(nums[index]);
            helper(vvi,cur,nums,index+1);
            cur.pop_back();
            helper(vvi,cur,nums,index+1);
        }

    public:
        vector<vector<int>> subsets(vector<int>& nums)
        {
```

```
            vector<vector<int>> vvi;
            vector<int>         cur;

            if(nums.size() < 1)
            {
                return vvi;
            }
            helper(vvi,cur,nums,0);
            return vvi;
        }
};



/*
执行结果：
通过
显示详情
执行用时 :4 ms, 在所有 cpp 提交中击败了99.78% 的用户
内存消耗 :12.7 MB, 在所有 cpp 提交中击败了11.02%的用户
*/
```

AlimyBreak
2019.10.03
2019.11.22 增加回溯法