```
/*
有 n 位用户参加活动，他们的 ID 从 0 到 n - 1，每位用户都 恰好 属于某一用户组。给你一个长度为 n 的数组 groupSizes，其中包含每位用户所处的用户组的大小，请你返回用户分组情况（存在的用户组以及每个组中用户的 ID）。

你可以任何顺序返回解决方案，ID 的顺序也不受限制。此外，题目给出的数据保证至少存在一种解决方案。



示例 1：

输入：groupSizes = [3,3,3,3,3,1,3]
输出：[[5],[0,1,2],[3,4,6]]
解释：
其他可能的解决方案有 [[2,1,6],[5],[0,4,3]] 和 [[5],[0,6,2],[4,3,1]]。

示例 2：

输入：groupSizes = [2,1,3,3,3,2]
输出：[[1],[0,5],[2,3,4]]



提示：

    groupSizes.length == n
    1 <= n <= 500
    1 <= groupSizes[i] <= n

在真实的面试中遇到过这道题？

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/group-the-people-given-the-group-size-they-belong-to
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:

- 遍历$i=0\sim n-1$,在若发现已有vvi的对应长度的子数组是否还有空间存放当前i,若还有空间存放就存入对应子数组，若没有空间存放，就重新压入一个子数组。
- 需要增加两个数组，$vi_capacity$用来记录当前二维数组的各子数组的总容量，vi_count用来记录各子数组已经存放了的数据的个数，当$vi_capacity[j]==groupSize(i)$且$vi_count[j]<vi_capacity[j]$,则元素$i$,可以分配到$vvi[j][vi_count[j]]$,对应的$vi_count[j]$需要自加；若条件不满足，且需要给$vvi$增加一个容量为$groupSize[i]$的子数组，并把这个子数组的第一个元素设置为$i$,对应$vi_capacity$和$vi_count$数组也要压入数据标记.

方法一:C++_遍历插入

```
class Solution
```

```
{
    public:
        vector<vector<int>> groupThePeople(vector<int>& groupSizes)
        {
            vector<vector<int>> vvi                     ;
            int                 n   = groupSizes.size() ;

            if(n < 1)
            {
                return vvi;
            }

            vector<int>         vi_capacity             ;
            vector<int>         vi_count                ;
            int                 i           =   0       ;
            int                 j           =   0       ;
            int                 flag        =   0       ;



            for(i = 0 ; i < n ; i++)
            {
                flag = -1; /*需要重新建立一个temp*/
                for(j=vi_capacity.size()-1;j>=0;j--)
                {
                    if(    ( vi_capacity[j] == groupSizes[i] )
                        && ( vi_count[j] < vi_capacity[j]    )
                    )
                    {
                        flag = j;
                        break;
                    }
                }

                if(flag == -1) /*需要新建一个来存放 i */
                {
                    vector<int> vi_temp(groupSizes[i],0);
                    vi_temp[0] = i;
                    vi_capacity.push_back(groupSizes[i]);
                    vi_count.push_back(1);
                    vvi.push_back(vi_temp);

                }
                else /*在flag对应vvi位置存放i */
                {
                    vvi[j][vi_count[j]] =   i                   ;
                    vi_count[j]         =   vi_count[j] + 1     ;
                }

            }

            return vvi;


        }
};

/*
```

```
执行结果:
通过
显示详情
执行用时 :44 ms, 在所有 cpp 提交中击败了100.00% 的用户
内存消耗 :12 MB, 在所有 cpp 提交中击败了100.00%的用户
*/
```

AlimyBreak
2019.12.18