Algorithm 1 找出多重集合中出现次数 n/4 的元素

实现思路是:1. 直接使用哈希表存放，遍历集合，对每个出现的元素计数，最后返回出现次数大于 n/4 的元素。时间复杂度显然是 $O(n)$

2. 使用摩尔投票法的拓展方法：因为出现次数大于 n/4 的元素最多为 3 个，所以我们我们采用抵消的策略，每 4 个元素，如果各不相同，则都不会为候选元素，如果有相同的元素，则其是答案的概率增加。最后判断候选元素是不是满足条件。最主要的思路是：不是找满足条件的元素，而是去除所有不满足条件的元素（用抵消的方法），那么剩下的就是候选元素。时间复杂度：只需要遍历两次集合，第一次选出候选元素，第二次判断候选元素是否满足条件，所以时间复杂度为 $O(n)$

输入： 集合 S，大小为 n

初始化 can1,can2,can3,vote1,vote2,vote3 为 0, 用来记录候选元素

for i in S **do**
  if vote1 > 0 **and** i == can1 **then**
    vote1++
  else if vote2 > 0 **and** i == can2 **then**
    vote2++
  else if vote3 > 0 **and** i == can3 **then**
    vote3++
  else if vote1 == 0 **then**
    can1 = i
    vote1++
  else if vote2 == 0 **then**
    can2 = i
    vote2++
  else if vote3 == 0 **then**
    can3 = i
    vote3++
  else
    vote1, vote2, vote3 均 -1
  end if
end for

=0

2 结果和相关说明

```
ans1
(0, 2)
(2, 8)
(4, 9)
(9, 4)
(8, 0)
(3, 1)
ans2
(0, 2)
(2, 8)
(4, 9)
(9, 4)
(8, 0)
(3, 1)
ans3
(0, 2)
(2, 8)
(4, 9)
(9, 4)
(8, 0)
(3, 1)
```

3个ans分别对应3种算法输出的凸包答案。

需要解释的是：

一开始使用了自己的框架，所以前两个函数定义如下：

vector insert_hull(int index)

vector merge_hull(int left,int right)

使用了一个全局数组Points存放相关点，

index代表计算Points[0-index]的元素的凸包

left和right代表计算Points[left-right]的元素的凸包

第三个函数使用了原本的框架

因为写完前两个函数才发现这个问题，且这不影响代码实现逻辑，只影响了元素的访问和存取（实际上后来我才发现用list可以很方便的访问前后元素，相关代码就可以简化），所以并没有修改。