做了这周的十个题目有些心得体会：

因为我没有看过CC，所以只有一些模糊的概念，欢迎大家批评指正。

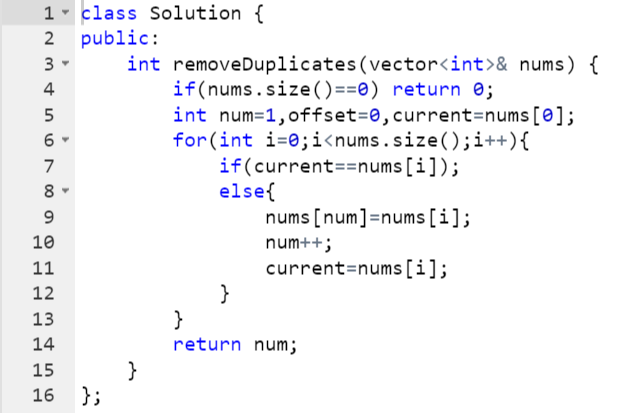
首先很多题目用了两个指针法。就是一个指针在前，一个在后，这样可以起到加快查找的功能。还有题目两个指针都在前面，也可以让题目变得非常简单。

再有就是一个题目使用了哈希表，哈希表主要是用空间换时间，让我们在O(1)的时间里找到N个元素。

有一个题目用到了动态规划，貌似这是第一个DP的题目，其实简单的说也就是用到了递推的思想，防止重复计算已经计算过的东西。

大概的重要信息我觉得就在这里，具体知识点还要参看CC。

## 1 Remove Duplicates from Sorted Array



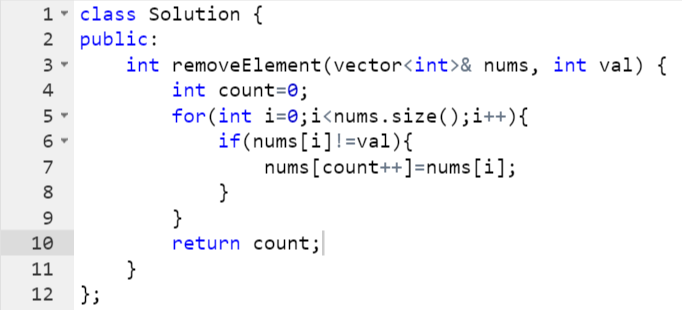
用两个指针分别控制需要查找的和需要修改的。如果发现当前元素和上一个记录的元素相同，直接跳过，发现不相同，直接将需要修改的那个指针替换成这个元素（如果当前元素并不重复也只是简单的A=A而已），这样和一发现重复就进行替换来比在很多元素重复的测试下明显快很多（122222....33）。

需要注意的测试用例有：（空），（12345）没有重复的和（12222...33）大量重复的

## 2 Remove Element



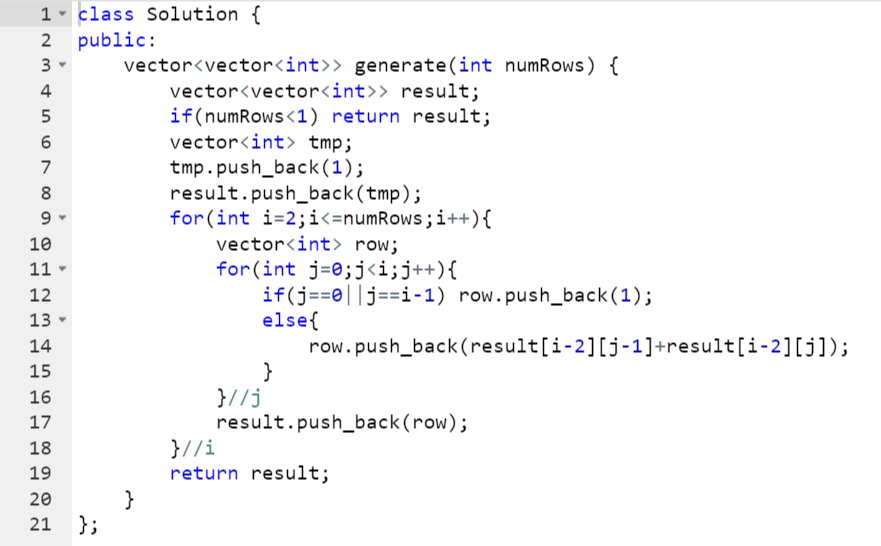
这是第一个解，想的是发现不一样就移动，而且是从后往前移动。虽然能做出来但是非常繁琐。当完成了第一个题目以后再来审视这个题目结果非常简单。



同样是用两个指针的方法，由于COUNT走得速度一定比不超过I，所以IF里面的语句一定是有意义的，其他同第一个题目。

需要注意的测试用例：（空），<(2,2,3,3),2>

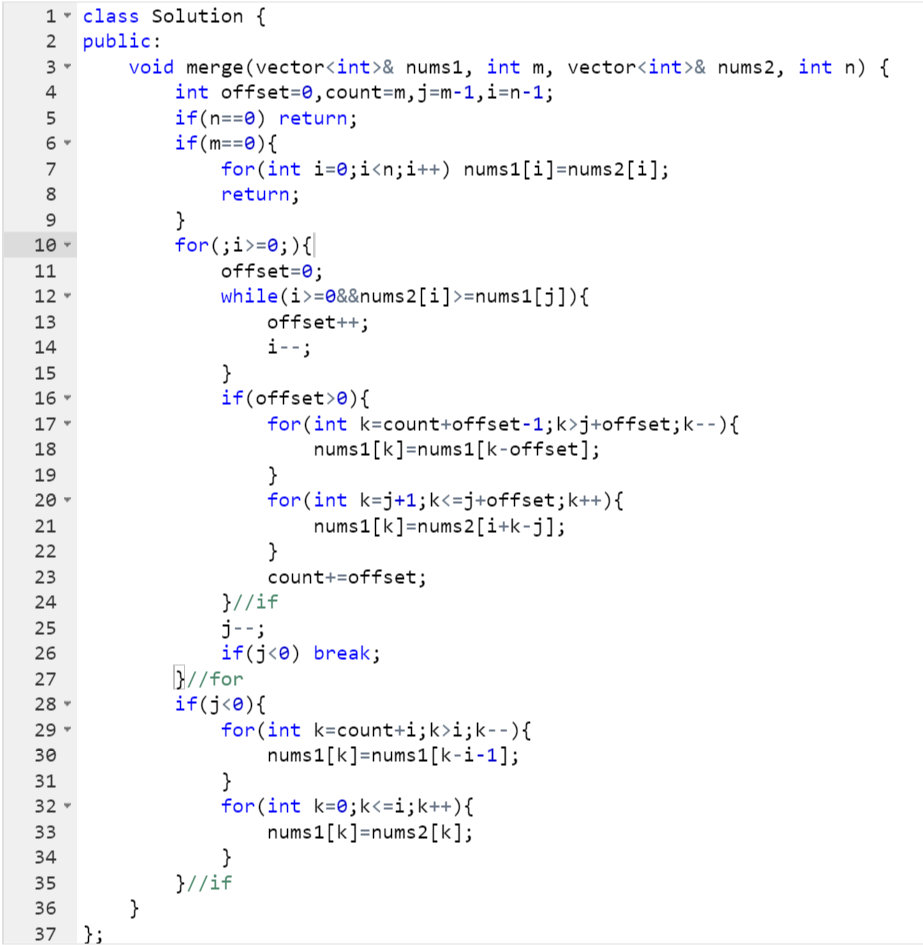
## 3 Pascal's Triangle



一道发现规律的题目，注意第一行和第一列必须直接给出即可，然后用递推公式。注意输入小于2的情况。

需要注意的用例有：0,1,5

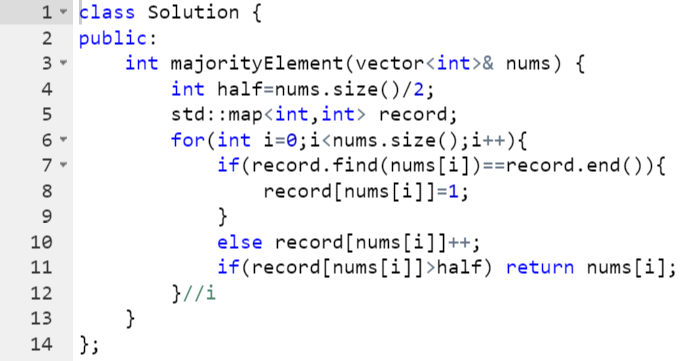
## 4 Merge Sorted Array



这个题目和1，2有相似之处，上面代码大致思路是直接合并，如果第二个数组比较大，则移动第一个数组的指针，否则将第一个数组整体向后移动。这个方法很容易想到但是比较繁琐就不多叙述。一种简单的办法是开辟新的空间，用三个指针，原来两个数组和新的数组，新的数组指针一直移动，旧的两个移动小的一个。但是要注意移动完了一个以后需要把剩下的全部移动到新的里面，然后再把新的移动到第一个数组返回即可。

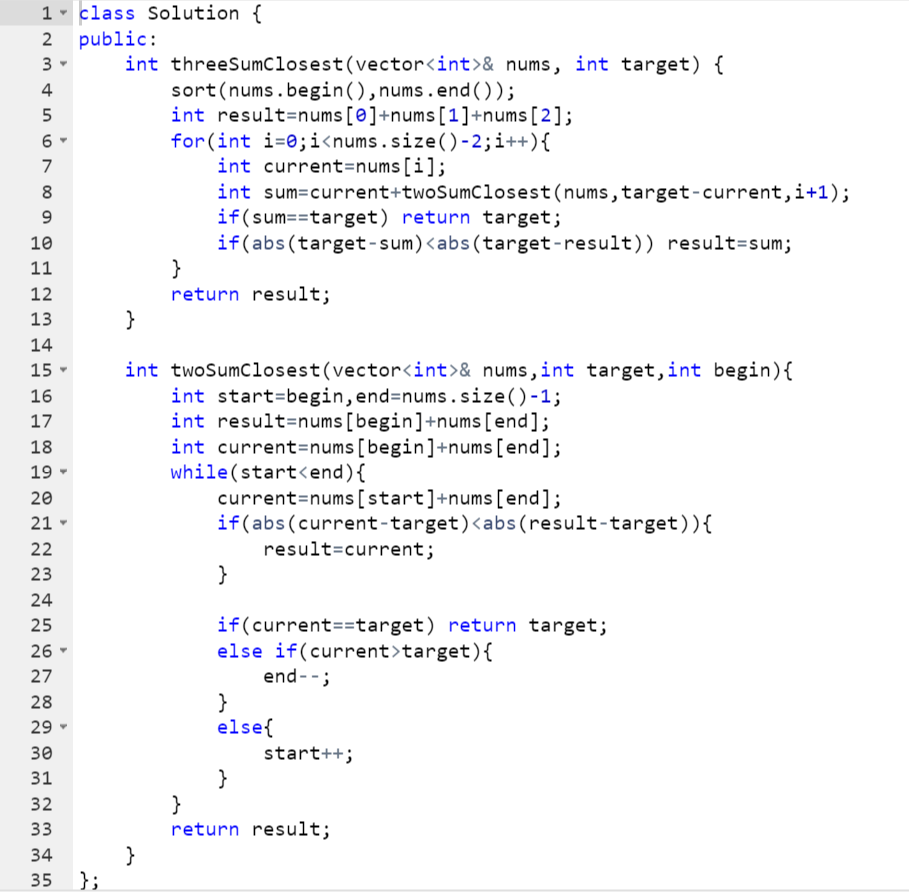
需要注意的测试用例：<（空），（空）>，<（1）,1，（空）>，<（空），（1），1>

## 5 Majority Element



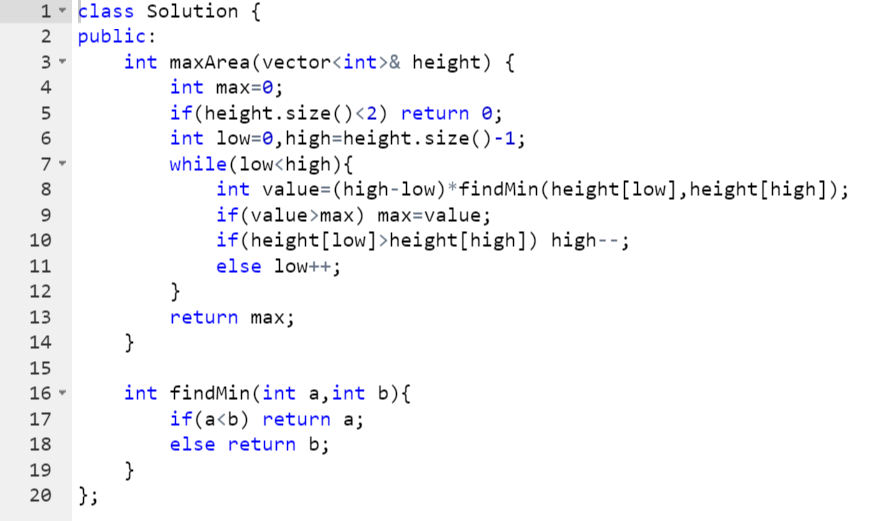
这个题目的关键是运用哈希表，直接简历哈希表然后看是不是存在当前元素，存在则加一看是否超过总元素一半，不存在则建立哈希索引并赋值为1.由于题目保证移动有可行解，所以没有特殊用例

## 6 3 Sum Closest



这个题目比较容易超时，我也是看了提示才达到要求。总体来说用暴力搜索时间复杂读是N立方，如果用两个指针法师N平方。思路是将3SumClosest变成2SumClosest。先排序，然后遍历每个数，把target变成target-nums[i].然后对从i+1到nums.size()-1的空间里面，用两个指针法操作，如果和大于new target，将大的指针减一，否则将小的指针加一，途中记录最接近的数即可，同理由于题目保证一定有可行解，没有很容易错的边界条件

## 7 Container With Most Water



这个题目的核心还是两个指针，开始一个在0，一个在末尾，移动的时候，如果首大于尾，则移动尾，否则移动首，因为如果首指针大于尾指针移动首指针，宽度减一了而且下次得到的长度一定小于尾指针的值，所以乘积小于上次的值。另外一个方向同理。这里所说的移动，对于首是往上加，尾是往下减，当碰到一起时值为零，所以返回。

需要注意的用例：（空），（0），（1，2，3）

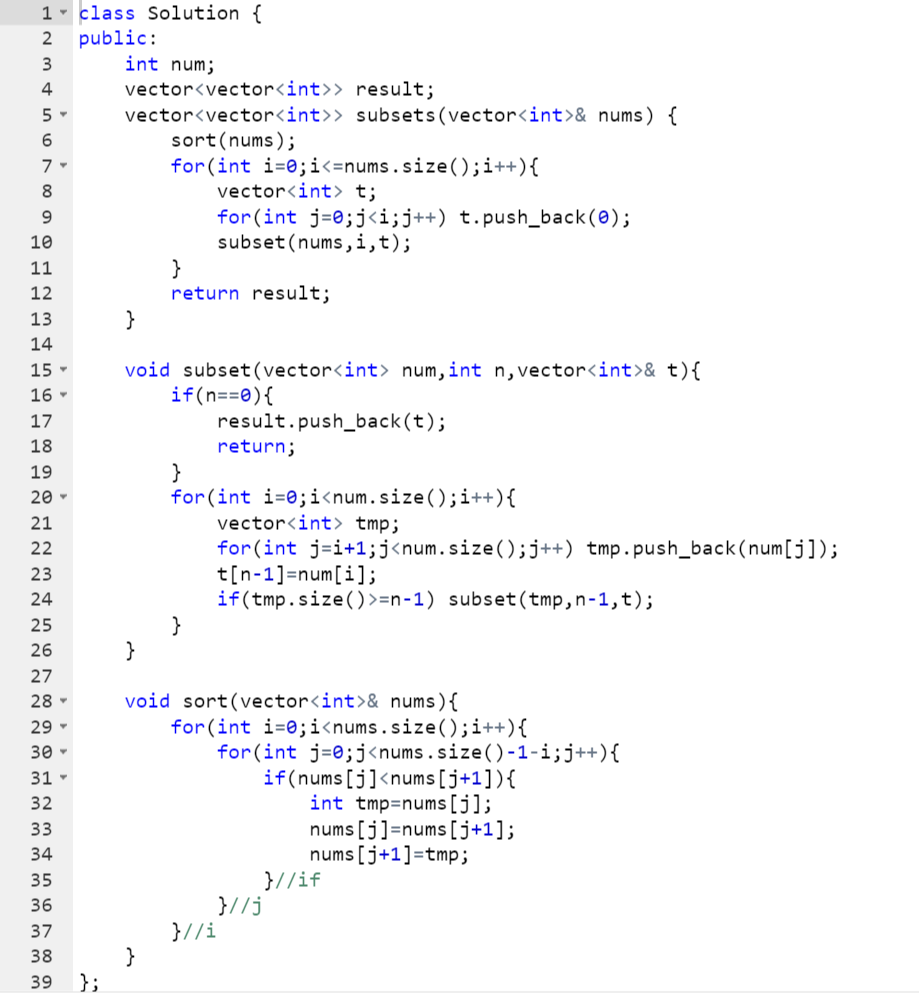
## 8 Minimum Path Sum



这个题目表面上是我们学过的加法原理，实际上是一个动态规划（DP）。如果直接搜索所有路径，时间复杂度是阶乘级非常不现实。因为题目规定不能向左上走，所以要到达一个点只有可能从它的上面或者左边来（边界情况除外）。因此只需要保证当F（M,N-1）和F（M-1,N）是最小的时候，F(M,N)就是GRID[M][N]+MIN(F(M,N-1),F(M-1,N)。因此只要把最左边和最上边的值初始化然后依次一排一排从上到下从左到右按照递推公式做就好了（这样能保证递推公式里面的上一层一定有值）

需要注意的测试用例：（空），（（空））

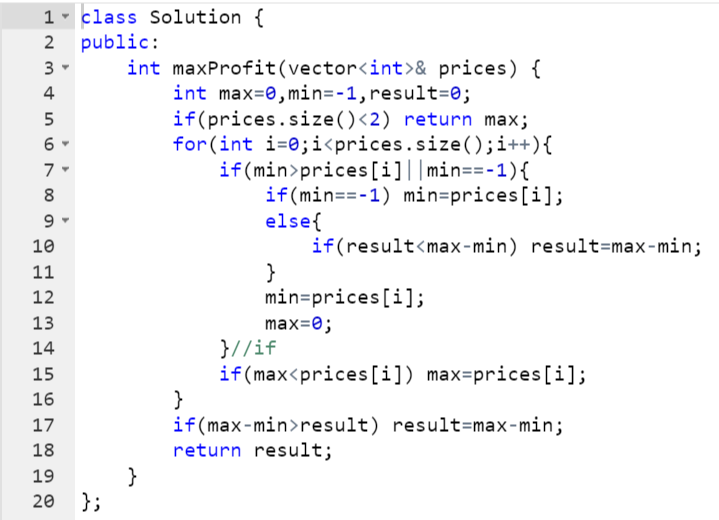
## 9 Subsets



这个题目很常见，因为原始数组原始长度未知，所以不能用FOR循环。其中上面的SUBSET函数是递归函数，NUM是剩余数组，N表示NUM数组仍然需要的个数，因此从NUMS里面取出N个数即可，T是当前的集合。如果不需要取更多的数，就把T直接放到RESULT里面（RESULT是全局的）。否则对剩余数组的所有数，一次执行SUBSET递归。当然要注意由于控制SUBSET元素个数的是要在非递归的函数里面控制的。当然因为题目的要求开始需要排序。貌似可以直接用库里面的排序，而且我用的还是冒泡法（-\_-|||），排序部分大家直接略过O\_\_O"…

需要注意的测试用例：（空）

## 10 Best Time to Buy and Sell Stock



这个题目意思很清晰，貌似只是在全局里面寻找最大和最小值，因此非常容易出错。这里隐藏的内在条件是，最大值的索引一定要大于最小值的索引。因此不能直接找全局最大和最小。首先给定两个数MAX和MIN记录最大和最小。如果当前的值大于MAX，则计算差值，如果比迄今为止的差值大就更新。如果当前的值比MIN小，就自动把MAX和MIN都重新设置成当前的这个值。因为首先在此之前我们已经记录了可能的最大差值所以不用担心前面的数，就算后面有值比前面最大值还大（可以得到比前面更大的差值），我们再选取最小值的时候也一定会选当前的这个数（因为它比前面的最小值还小）。因此此时前面的数就根本不需要再去考虑，直接好像把前面的数删掉了一样从当前的这个数开始用相同的规律计算。当然如果当前的数在MAX和MIN之间（包含），直接跳过即可。