[[LeetCode]Container With Most Water](http://blog.csdn.net/xshalk/article/details/8182978)

分类： [LeetCode](http://blog.csdn.net/xshalk/article/category/1274402)2012-11-14 16:12 156人阅读 [评论](http://blog.csdn.net/xshalk/article/details/8182978#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/xshalk/article/details/8182978#report)

Given n non-negative integers a1, a2, ..., an, where each represents a point at coordinate (i, ai). n vertical lines are drawn such that the two endpoints of line i is at (i, ai) and (i, 0). Find two lines, which together with x-axis forms a container, such that the container contains the most water.

Note: You may not slant the container.

简单翻译： 有一些竖线，坐标分别是  (i,ai);  问选 其中两条线 作为容器的两侧 ，X轴为底边，最大容量是多少。

                 如果选择  （i, a）(j,b) 那么容量就是     |i-j| \* min(a,b) 。

初级思路，遍历，所有坐标，  复杂度 O（n^2） ，选最大值。 小数据能过，大数据果断超时。

**/\*中级思路以下我觉得分析有点问题\*/**

中级思路，  这样的值 有两个性质，  不妨设 （i,a）为左边    （j,b）为右边为最大值，

1.若有坐标( j+k，c)，k>0 必然  b>c  ，否则（i,a）(j+k, c)   的容积 (k+j-i)\*min(a,c) 大于     （j-i）\*min(a,b) 即b 比右边的坐标都大

2. 同理，  a 比 左边的坐标值都大所以，可以做预处理，把 比左边的大的坐标，标记出来，   把比右边都大的坐标也标记出来。

       例如：如下数组，第一个坐标是（0，5），第二坐标（1，2），最后一个是（9，4）；

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 2 | 12 | 1 | 5 | 3 | 4 | 11 | 9 | 4 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

 第一行是ai， 第二行是左标记，第三行    右标记 ，标记的复杂度是O（n）     。

 求解的遍历方式：i 从0 开始  j 从  size（）-1 开始，    i<j  ; i和j 仅取相应的标记值的值。

   这是相当于一个较大的剪枝， 大数据能过。

代码如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/xshalk/article/details/8182978)

1. **class** Solution {
2. **public**:
3. **int** maxArea(vector<**int**> &height) {
4. // Start typing your C/C++ solution below
5. // DO NOT write int main() function
6. **int** max = 0;
7. vector<**int**> flag;
8. flag.resize(height.size());
9. **int** lmax = 0;
10. **for**(**int** i = 0; i < height.size();i++)
11. {
12. **if**(height[i]<lmax)
13. flag[i]=0;
14. **else**{
15. flag[i]=1;
16. lmax = height[i];
18. }
20. }
21. **int** rmax = 0;
22. **for**(**int** i = height.size()-1; i >= 0;i--)
23. {
24. **if**(height[i]>=rmax)
25. {
26. flag[i]+=2;
27. rmax = height[i];
29. }
31. }
32. **for**(**int** i = 0 ; i < height.size() ; i++)
33. {
34. **if**(flag[i]%2==0)
35. **continue**;
37. **for** (**int** j = height.size()-1 ; j >i  ; j--)
38. {
39. **if**( flag[j] <2)
40. **continue**;
41. **int** sum = min(height[i],height[j]) \* (j- i);
42. **if** ( max < sum)
43. max =sum;
44. }
45. }
46. **return** max;
47. }
48. };

进一步思路：  上面的例子中，从 （0，5），（9，4）开始

                                            得出36，然后是（0，5），（8，9）；得出40，之后是（0，5），（7，11）得出35，实际上这已经不需要继续遍历 右边了，

                                          右边标记位是递增的，当9比5大之后，右边能遍历到的数都比5大， 而坐标依次减少，所以得出的容积，只会越来越小。

                       结论：          抽象的说，在（i,a）,(j,b)的情况下，  若 a  <b   ， （i，a）的情况已经遍历完成，j无需移动， i向右移动，i++;

                                                                                    同样 若 b <a   ,     (i, a ) 的 情况要继续遍历，或者说，（j，b）的情况已经求解完成， j--；

复杂度O（n）

        上面的代码大数据要600多ms，下面的仅需104ms

Run Status: **Accepted!**  
Program Runtime: 104 milli secs

 代码如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/xshalk/article/details/8182978)

1. **class** Solution {
2. **public**:
3. **int** maxArea(vector<**int**> &height) {
4. // Start typing your C/C++ solution below
5. // DO NOT write int main() function
6. **int** max = 0;
7. vector<**int**> flag;
8. flag.resize(height.size());
9. **int** lmax = 0;
10. **for**(**int** i = 0; i < height.size();i++)
11. {
12. **if**(height[i]<lmax)
13. flag[i]=0;
14. **else**{
15. flag[i]=1;
16. lmax = height[i];
18. }
19. //     cout <<i << "L:"<< flag[i] << endl;
21. }
22. **int** rmax = 0;
23. **for**(**int** i = height.size()-1; i >= 0;i--)
24. {
25. **if**(height[i]>=rmax)
26. {
27. flag[i]+=2;
28. rmax = height[i];
30. }
32. }
33. **for**(**int** i = 0, j = height.size()-1 ; i < j ;)
34. {
35. **if**(flag[i]%2==0)
36. {
37. i++;
38. **continue**;
40. }
41. **if** (flag[j]<2)
42. {
43. j--;
44. **continue**;
45. }
46. **int** sum = min(height[i],height[j]) \* (j- i);
47. **if** ( max < sum)
48. max =sum;
49. **if**(height[i]<height[j])
50. {
51. i++;
52. }**else**{
53. j--;
54. }
56. }
57. **return** max;
58. }
59. };