**[[LeetCode] Container With Most Water](http://www.cnblogs.com/remlostime/archive/2012/10/29/2745018.html)**

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.

类似于2Sum的思想，两边设一个指针，然后计算area，如果height[i] <= height[j]，那么i++，因为在这里height[i]是瓶颈，j往里移只会减少面积，不会再增加area。

这是一个贪心的策略，每次取两边围栏最矮的一个推进，希望获取更多的水。

一个不严格的证明：

当height[i] <= height[j]时，为什么是i++，而不是j++来获取可能更多的水？

假设j' > j，之所以j'往左移，是因为存在height[i'] > height[j'] (i’ <= i), 而那时area' = (j' - i') \* min(height[i'], height[j'])，

因为height[j'] == min(height[i'], height[j'])，所以area' = (j' - i') \* height[j']。

而i 和 j'构成的面积area = (j' - i) \* min(height[i], height[j'])。

area' >= area，所以j不需要往右移。

[复制代码](javascript:void(0);)

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 i = 0;

7 int j = height.size() - 1;

8

9 int ret = 0;

10 while(i < j)

11 {

12 int area = (j - i) \* min(height[i], height[j]);

13 ret = max(ret, area);

14

15 if (height[i] <= height[j])

16 i++;

17 else

18 j--;

19 }

20

21 return ret;

22 }

23 };

[复制代码](javascript:void(0);)

时间复杂度O(n)