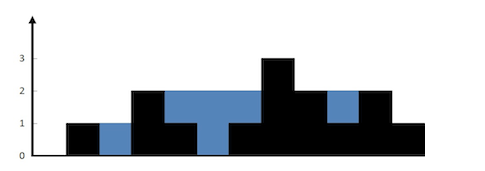
Given *n* non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

For example,   
Given [0,1,0,2,1,0,1,3,2,1,2,1], return 6.



如图，我们需要找到能存水的大小，我们需要找到两个存水的栅栏，就是比较高的。我们从头开始遍历，找到一个逆序的，就是nums[i]>nums[i+1]，这个就可以充当第一个栅栏，然后我们从i开始，找比大于等于i的另一半，遇到这个另一个栅栏，然后停下，这就是一个存水的容器，存水高度我们取最小的栅栏高度。如果没有找到另一个大于等于第一个栅栏的说明当前的栅栏是最高的，并不能作为栅栏起点来使用，我们掠过它查找下一位i++。

代码：

public static int trappingRainWater(int[] nums) {

int key = 0;

int sum = 0;

int i = 0;

while(i<nums.length) {

while(i<nums.length-1&&nums[i]<=nums[i+1]) { //查找第一个栅栏的位置

i++;

}

if(i<nums.length) { //位置有效

int j = i+1;//j为第二个栅栏

while (j < nums.length && nums[i] > nums[j]) { //当高度大于等于当前前一根栅栏就可以作为第二根栅栏

j++;

}

if (j < nums.length) { //j的位置有效，说明存在第二根栅栏

key = Math.min(nums[i],nums[j]); //取低的栅栏为水位高度

while (i < j) { //计算容量

sum = sum + (key - nums[i]);

i++;

}

}

else //如果没有找到第二根栅栏，说明上一根栅栏是最高的，并不能作为第一根栅栏

i++;//掠过开始下一位。

}

}

return sum;

}

时间复杂度为O(n)