**4. 寻找两个正序数组的中位数**

给定两个大小为 m 和 n 的正序（从小到大）数组 nums1 和 nums2。请你找出并返回这两个正序数组的中位数。

进阶：你能设计一个时间复杂度为 O(log (m+n)) 的算法解决此问题吗？

输入：nums1 = [1,3], nums2 = [2]

输出：2.00000

解释：合并数组 = [1,2,3] ，中位数 2

输入：nums1 = [1,2], nums2 = [3,4]

输出：2.50000

解释：合并数组 = [1,2,3,4] ，中位数 (2 + 3) / 2 = 2.5

|  |
| --- |
| class Solution:  def findMedianSortedArrays(self, nums1: List[int], nums2: List[int]) -> float:  # 1. 组合2个列表，并排序  num\_all = nums1 + nums2  num\_all.sort()  # 【特殊情况】：组合列表长度为0，即2个列表都空（根据提示，没有这个情况）  # if len(num\_all) == 0:  # return float(0)  # 2. 判断组合列表长度的奇偶性  if len(num\_all) % 2 == 0:  # 3. 若偶数，则有2个元素组成中位数（取整//2得到索引位置）  first = len(num\_all) // 2 - 1  second = len(num\_all) // 2  return (num\_all[first] + num\_all[second]) / 2  else:  # 4. 若奇数，则只有1个元素（取整//2得到索引位置）  first = len(num\_all) // 2  return num\_all[first] |

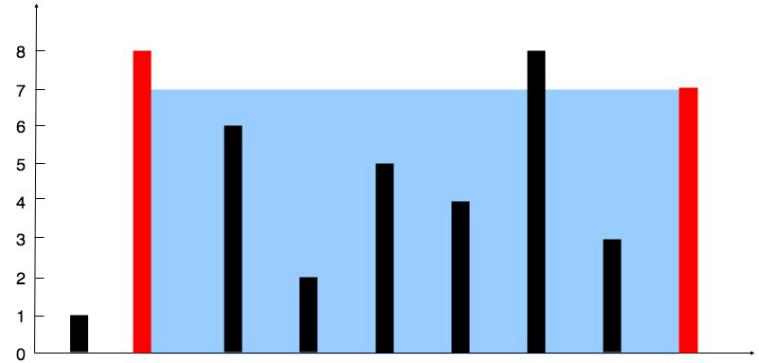
|  |
| --- |
| def findMedianSortedArrays(self, nums1: List[int], nums2: List[int]) -> float:  def findKthElement(arr1,arr2,k):  len1,len2 = len(arr1),len(arr2)  if len1 > len2:  return findKthElement(arr2,arr1,k)  if not arr1:  return arr2[k-1]  if k == 1:  return min(arr1[0],arr2[0])  i,j = min(k//2,len1)-1,min(k//2,len2)-1  if arr1[i] > arr2[j]:  return findKthElement(arr1,arr2[j+1:],k-j-1)  else:  return findKthElement(arr1[i+1:],arr2,k-i-1)  l1,l2 = len(nums1),len(nums2)  left,right = (l1+l2+1)//2,(l1+l2+2)//2  return (findKthElement(nums1,nums2,left)+findKthElement(nums1,nums2,right))/2 |

题目可以归结到寻找第k小(大)元素问题，思路可以总结如下：取两个数组中的第k/2个元素进行比较，如果数组1的元素小于数组2的元素，则说明数组1中的前k/2个元素不可能成为第k个元素的候选，所以将数组1中的前k/2个元素去掉，组成新数组和数组2求第k-k/2小的元素，因为我们把前k/2个元素去掉了，所以相应的k值也应该减小。另外就是注意处理一些边界条件问题，比如某一个数组可能为空或者k为1的情况。

**11. 盛最多水的容器**

给你 n 个非负整数 a1，a2，...，an，每个数代表坐标中的一个点 (i, ai) 。在坐标内画 n 条垂直线，垂直线 i 的两个端点分别为 (i, ai) 和 (i, 0)。找出其中的两条线，使得它们与 x 轴共同构成的容器可以容纳最多的水。

说明：你不能倾斜容器，且 n 的值至少为 2。



图中垂直线代表输入数组 [1,8,6,2,5,4,8,3,7]。在此情况下，容器能够容纳水（表示为蓝色部分）的最大值为 49。

输入：[1,8,6,2,5,4,8,3,7]

输出：49

|  |
| --- |
| class Solution {  public:  int maxArea(vector<int>& height) {  int size = height.size();  int left=0, right=size-1;  int ans = 0;  while(left < right){  ans = max(ans, (right-left)\*min(height[left], height[right]));  if(height[left] > height[right]) --right;  else ++left;  }  return ans;  }  }; |

一开始两个指针一个指向开头一个指向结尾，此时容器的底是最大的，接下来随着指针向内移动，会造成容器的底变小，在这种情况下想要让容器盛水变多，就只有在容器的高上下功夫。 那我们该如何决策哪个指针移动呢？我们能够发现不管是左指针向右移动一位，还是右指针向左移动一位，容器的底都是一样的，都比原来减少 1。这种情况下我们想要让指针移动后的容器面积增大，就要使移动后的容器的高尽量大，所以我们选择指针所指的高较小的那个指针进行移动，这样我们就保留了容器较高的那条边，放弃了较小的那条边，以获得有更高的边的机会。

**15. 三数之和**

给你一个包含 n 个整数的数组 nums，判断 nums 中是否存在三个元素 a，b，c ，使得 a + b + c = 0 ？请你找出所有满足条件且不重复的三元组。

注意：答案中不可以包含重复的三元组。

给定数组 nums = [-1, 0, 1, 2, -1, -4]，

满足要求的三元组集合为：

[

[-1, 0, 1],

[-1, -1, 2]

]

|  |
| --- |
| class Solution {  public:  vector<vector<int>> threeSum(vector<int>& nums) {  int target;  vector<vector<int>> ans;  sort(nums.begin(), nums.end());  for (int i = 0; i < nums.size(); i++) {  if (i > 0 && nums[i] == nums[i - 1]) continue;  if ((target = nums[i]) > 0) break;  int l = i + 1, r = nums.size() - 1;  while (l < r) {  if (nums[l] + nums[r] + target < 0) ++l;  else if (nums[l] + nums[r] + target > 0) --r;  else {  ans.push\_back({target, nums[l], nums[r]});  ++l, --r;  while (l < r && nums[l] == nums[l - 1]) ++l;  while (l < r && nums[r] == nums[r + 1]) --r;  }  }  }  return ans;  }  }; |

**16. 最接近的三数之和**

给定一个包括 n 个整数的数组 nums 和 一个目标值 target。找出 nums 中的三个整数，使得它们的和与 target 最接近。返回这三个数的和。假定每组输入只存在唯一答案。

输入：nums = [-1,2,1,-4], target = 1

输出：2

解释：与 target 最接近的和是 2 (-1 + 2 + 1 = 2) 。

先排序, 然后遍历, 然后内部使用双指针, 时间复杂度应该是O(n²), 代码如下:

|  |
| --- |
| class Solution {  public int threeSumClosest(int[] nums, int target) {  // 排序  Arrays.sort(nums);  int closestNum = nums[0] + nums[1] + nums[2];  for (int i = 0; i < nums.length - 2; i++) {  int l = i + 1, r = nums.length - 1;  while (l < r){  int threeSum = nums[l] + nums[r] + nums[i];  if (Math.abs(threeSum - target) < Math.abs(closestNum - target)) {  closestNum = threeSum;  }  if (threeSum > target) {  r--;  } else if (threeSum < target) {  l++;  } else {  // 如果已经等于target的话, 肯定是最接近的  return target;  }  }  }  return closestNum;  }  } |

双指针二分问题。对于排好序的数组，首先确定三个数位置在中间的那个数字，然后在它左右搜索另外两个数，如果大于目标，则右指针左移，否则左指针右移。

**18. 四数之和**

给定一个包含 n 个整数的数组 nums 和一个目标值 target，判断 nums 中是否存在四个元素 a，b，c 和 d ，使得 a + b + c + d 的值与 target 相等？找出所有满足条件且不重复的四元组。

注意：答案中不可以包含重复的四元组。

给定数组 nums = [1, 0, -1, 0, -2, 2]，和 target = 0。

满足要求的四元组集合为：

[

[-1, 0, 0, 1],

[-2, -1, 1, 2],

[-2, 0, 0, 2]

]

|  |
| --- |
| class Solution {  public:  vector<vector<int>> fourSum(vector<int>& nums, int target) {  sort(nums.begin(),nums.end());  vector<vector<int>> result;  int size=nums.size();  for(int a=0;a<size-3;++a)  {  if(a>0&&nums[a]==nums[a-1])continue;  for(int b=a+1;b<size-2;++b)//以下代码与三数之和一样的  {  if(b>a+1&&nums[b]==nums[b-1])continue;  int i=b+1,j=size-1;  while(i<j)  {  int sum=nums[a]+nums[b]+nums[i]+nums[j];  if(sum<target)  while(i<j&&nums[i]==nums[++i]);  else if(sum>target)  while(i<j&&nums[j]==nums[--j]);  else{  result.push\_back(vector<int>{nums[a],nums[b],nums[i],nums[j]});  while(i<j&&nums[i]==nums[++i]);  while(i<j&&nums[j]==nums[--j]);  }  }  }  }  return result;  }  };  双指针二分问题。和3sum相比，要先选定两个数，而不是一个数。 |

**33. 搜索旋转排序数组**

给你一个升序排列的整数数组 nums ，和一个整数 target 。

假设按照升序排序的数组在预先未知的某个点上进行了旋转。（例如，数组 [0,1,2,4,5,6,7] 可能变为 [4,5,6,7,0,1,2] ）。

请你在数组中搜索 target ，如果数组中存在这个目标值，则返回它的索引，否则返回 -1 。

输入：nums = [4,5,6,7,0,1,2], target = 0

输出：4

输入：nums = [4,5,6,7,0,1,2], target = 3

输出：-1

输入：nums = [1], target = 0

输出：-1

|  |
| --- |
| class Solution {  public int search(int[] nums, int target) {  int len = nums.length;  int left = 0, right = len-1;  while(left <= right){  int mid = (left + right) / 2;  if(nums[mid] == target)  return mid;  else if(nums[mid] < nums[right]){  if(nums[mid] < target && target <= nums[right])  left = mid+1;  else  right = mid-1;  }  else{  if(nums[left] <= target && target < nums[mid])  right = mid-1;  else  left = mid+1;  }  }  return -1;  }  } |

思路：如果中间的数小于最右边的数，则右半段是有序的，若中间数大于最右边数，则左半段是有序的，我们只要在有序的半段里用首尾两个数组来判断目标值是否在这一区域内，这样就可以确定保留哪半边了

**34. 在排序数组中查找元素的第一个和最后一个位置**

给定一个按照升序排列的整数数组 nums，和一个目标值 target。找出给定目标值在数组中的开始位置和结束位置。

你的算法时间复杂度必须是 O(log n) 级别。

如果数组中不存在目标值，返回 [-1, -1]。

输入: nums = [5,7,7,8,8,10], target = 8

输出: [3,4]

输入: nums = [5,7,7,8,8,10], target = 6

输出: [-1,-1]

|  |
| --- |
| class Solution {  public:  vector<int> searchRange(vector<int>& nums, int target) {  vector<int> res(2,-1);  if(nums.empty()) return res;  int n=nums.size(),l=0,r=n-1;  while(l<r){  int m=l+(r-l)/2;  if(nums[m]>=target) r=m;  else l=m+1;  }  if(nums[l]!=target) return res;  res[0]=l;  r=n;  while(l<r){  int m=l+(r-l)/2;  if(nums[m]<=target) l=m+1;  else r=m;  }  res[1]=l-1;  return res;  }  }; |

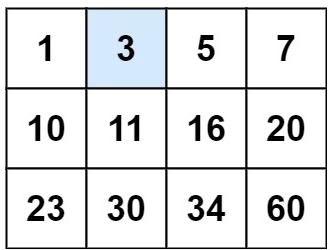
利用二分思想先找其左边界，再找其右边界即可，注意找左边界的时候，由右侧逼近；找右边界的时候，由左侧逼近，即可。

根据下标二分问题。要点是改变low或high的时候把当前数字mid也包含进来，因为它也可能是结果。

**74. 搜索二维矩阵**

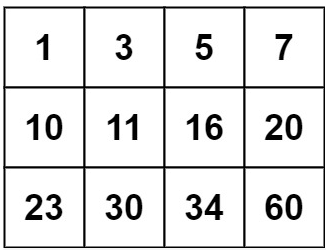
编写一个高效的算法来判断 m x n 矩阵中，是否存在一个目标值。该矩阵具有如下特性：

每行中的整数从左到右按升序排列。每行的第一个整数大于前一行的最后一个整数。



输入：matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,50]], target = 3

输出：true



输入：matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,50]], target = 13

输出：false

输入：matrix = [], target = 0

输出：false

这道题的关键点在于利用矩阵升序的特性，需要选择合适的切入点与目标值相比较以缩小数据返围，很明显如果选择左上角或右下角当做切入点并不能起到只能分别排除一个选项，而如果选择右上角或者左下角进行比较即可起到更明显的效果。

|  |
| --- |
| public boolean searchMatrix(int[][] matrix, int target) {  if(matrix.length == 0)  return false;  int row = 0, col = matrix[0].length-1;  while(row < matrix.length && col >= 0){  if(matrix[row][col] < target)  row++;  else if(matrix[row][col] > target)  col--;  else  return true;  }  return false;  } |

**162. 寻找峰值**

峰值元素是指其值大于左右相邻值的元素。给定一个输入数组 nums，其中 nums[i] ≠ nums[i+1]，找到峰值元素并返回其索引。

数组可能包含多个峰值，在这种情况下，返回任何一个峰值所在位置即可。你可以假设 nums[-1] = nums[n] = -∞。

输入: nums = [1,2,3,1]

输出: 2

解释: 3 是峰值元素，你的函数应该返回其索引 2。

输入: nums = [1,2,1,3,5,6,4]

输出: 1 或 5

解释: 你的函数可以返回索引 1，其峰值元素为 2；或者返回索引 5， 其峰值元素为 6。

|  |
| --- |
| int findPeakElement(vector<int>& nums) {  int left = 0, right = nums.size() - 1;  for (; left < right; ) {  int mid = left + (right - left) / 2;  if (nums[mid] > nums[mid + 1]) {  right = mid;  } else {  left = mid + 1;  }  }  return left;  } |

【笔记】由于题目假设nums[-1]=nums[n]=-∞。所以，我们从0开始往后遍历元素，如果某个元素大于其后面的元素，则该元素就是峰值元素。（但是它时O(n)，不符合题意）O(logN)一般考虑二分搜索。有如下规律：

规律一：如果nums[i] > nums[i+1]，则在i之前一定存在峰值元素

规律二：如果nums[i] < nums[i+1]，则在i+1之后一定存在峰值元素

<https://blog.csdn.net/smile_watermelon/article/details/47267089>

**[4. 寻找两个正序数组的中位数](https://blog.csdn.net/smile_watermelon/article/details/47267089)**

[这道题让我们求两个有序数组的中位数，而且限制了时间复杂度为O(log (m+n))，看到这个时间复杂度，自然而然的想到了应该使用二分查找法来求解。那么回顾一下中位数的定义，如果某个有序数组长度是奇数，那么其中位数就是最中间那个，如果是偶数，那么就是最中间两个数字的平均值。这里对于两个有序数组也是一样的，假设两个有序数组的长度分别为m和n，由于两个数组长度之和 m+n 的奇偶不确定，因此需要分情况来讨论，对于奇数的情况，直接找到最中间的数即可，偶数的话需要求最中间两个数的平均值。为了简化代码，不分情况讨论，我们使用一个小trick，我们分别找第 (m+n+1) / 2 个，和 (m+n+2) / 2 个，然后求其平均值即可，这对奇偶数均适用。假如m+n 为奇数的话，那么其实 (m+n+1) / 2 和 (m+n+2) / 2 的值相等，相当于两个相同的数字相加再除以2，还是其本身。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[这里我们需要定义一个函数来在两个有序数组中找到第K个元素，下面重点来看如何实现找到第K个元素。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[1，为了避免产生新的数组从而增加时间复杂度，我们使用两个变量i和j分别来标记数组nums1和nums2的起始位置。然后来处理一些边界问题，比如当某一个数组的起始位置大于等于其数组长度时，说明其所有数字均已经被淘汰了，相当于一个空数组了，那么实际上就变成了在另一个数组中找数字，直接就可以找出来了。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[2.还有就是如果K=1的话，那么我们只要比较nums1和nums2的起始位置i和j上的数字就可以了。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[3.难点就在于一般的情况怎么处理？因为我们需要在两个有序数组中找到第K个元素，为了加快搜索的速度，我们要使用二分法，对K二分，意思是我们需要分别在nums1和nums2中查找第K/2个元素，注意这里由于两个数组的长度不定，所以有可能某个数组没有第K/2个数字，所以我们需要先检查一下，数组中到底存不存在第K/2个数字，如果存在就取出来，否则就赋值上一个整型最大值。如果某个数组没有第K/2个数字，那么我们就淘汰另一个数字的前K/2个数字即可。有没有可能两个数组都不存在第K/2个数字呢，这道题里是不可能的，因为我们的K不是任意给的，而是给的m+n的中间值，所以必定至少会有一个数组是存在第K/2个数字的。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[4.最后就是二分法的核心啦，比较这两个数组的第K/2小的数字midVal1和midVal2的大小，如果第一个数组的第K/2个数字小的话，那么说明我们要找的数字肯定不在nums1中的前K/2个数字，所以我们可以将其淘汰，将nums1的起始位置向后移动K/2个，并且此时的K也自减去K/2，调用递归。反之，我们淘汰nums2中的前K/2个数字，并将nums2的起始位置向后移动K/2个，并且此时的K也自减去K/2，调用递归即可。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [class Solution {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public double findMedianSortedArrays(int[] nums1, int[] nums2) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int m = nums1.length;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int n = nums2.length;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int left = (m + n + 1) / 2;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int right = (m + n + 2) / 2;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return (findKth(nums1, 0, nums2, 0, left) + findKth(nums1, 0, nums2, 0, right)) / 2.0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [//i: nums1的起始位置 j: nums2的起始位置](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public int findKth(int[] nums1, int i, int[] nums2, int j, int k){](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if( i >= nums1.length) return nums2[j + k - 1];//nums1为空数组](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if( j >= nums2.length) return nums1[i + k - 1];//nums2为空数组](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if(k == 1){](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return Math.min(nums1[i], nums2[j]);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int midVal1 = (i + k / 2 - 1 < nums1.length) ? nums1[i + k / 2 - 1] : Integer.MAX\_VALUE;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int midVal2 = (j + k / 2 - 1 < nums2.length) ? nums2[j + k / 2 - 1] : Integer.MAX\_VALUE;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if(midVal1 < midVal2){](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return findKth(nums1, i + k / 2, nums2, j , k - k / 2);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}else{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return findKth(nums1, i, nums2, j + k / 2 , k - k / 2);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

**[315. 计算右侧小于当前元素的个数](https://blog.csdn.net/smile_watermelon/article/details/47267089)**

[给定一个整数数组 nums，按要求返回一个新数组 counts。数组 counts 有该性质： counts[i] 的值是 nums[i] 右侧小于 nums[i] 的元素的数量。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入：nums = [5,2,6,1]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出：[2,1,1,0]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[解释：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[5 的右侧有 2 个更小的元素 (2 和 1)](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[2 的右侧仅有 1 个更小的元素 (1)](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[6 的右侧有 1 个更小的元素 (1)](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[1 的右侧有 0 个更小的元素](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [class Solution {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [vector<int> countSmaller(vector<int>& nums)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [vector<int> res;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [vector<int> tmp = nums;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [sort(tmp.begin(), tmp.end());](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [for (size\_t i = 0; i < nums.size(); i++) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [auto it = lower\_bound(tmp.begin(), tmp.end(), nums[i]);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [res.push\_back(it - tmp.begin());](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [tmp.erase(it);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return res;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

[先额外存一个排序好的数组。然后查找lower\_bound，看有多少个数比它小，并把这个数从排序数组中移除。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

**[475. 供暖器](https://blog.csdn.net/smile_watermelon/article/details/47267089)**

[冬季已经来临。 你的任务是设计一个有固定加热半径的供暖器向所有房屋供暖。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[在加热器的加热半径范围内的每个房屋都可以获得供暖。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[现在，给出位于一条水平线上的房屋 houses 和供暖器 heaters 的位置，请你找出并返回可以覆盖所有房屋的最小加热半径。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[说明：所有供暖器都遵循你的半径标准，加热的半径也一样。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入: houses = [1,2,3], heaters = [2]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出: 1](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[解释: 仅在位置2上有一个供暖器。如果我们将加热半径设为1，那么所有房屋就都能得到供暖。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入: houses = [1,2,3,4], heaters = [1,4]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出: 1](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[解释: 在位置1, 4上有两个供暖器。我们需要将加热半径设为1，这样所有房屋就都能得到供暖。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入：houses = [1,5], heaters = [2]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出：3](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [class Solution:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [def findRadius(self, houses: List[int], heaters: List[int]) -> int:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [houses.sort()](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [heaters.sort()](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [heaters = [float("-inf")] + heaters + [float("inf")]](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [i = 1](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [res = 0](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [for house in houses:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [while i < len(heaters) - 1 and house > heaters[i]:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [i += 1](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [res = max(res, min(heaters[i] - house, house - heaters[i - 1]))](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return res](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

|  |
| --- |
| [class Solution:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [def findRadius(self, houses: List[int], heaters: List[int]) -> int:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [heaters.sort()](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [heaters = [float("-inf")] + heaters + [float("inf")]](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [res = 0](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [for house in houses:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [loc = bisect.bisect\_left(heaters, house)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [res = max(res, min(house - heaters[loc - 1], heaters[loc] - house))](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return res](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

|  |
| --- |
| [模版：](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int LowerBound(vector<int> nums, int target)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int left, right;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int len = nums.size();](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [left = 0, right = len - 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int mid;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [while (left <= right)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [mid = left + (right - left) / 2;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if (nums[mid] >= target)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{ // 这里的比较运算符与题目要求一致](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [right = mid - 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [else](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [left = mid + 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if (left >= len || nums[left] != target)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{ // 判断一下是否越界，或者不相等](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return -1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return left; // 返回下界的下标](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [class Solution](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int findRadius(vector<int> &houses, vector<int> &heaters)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int res = 0, n = heaters.size();](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [sort(heaters.begin(), heaters.end());](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [for (int house : houses)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int left = 0, right = n - 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [// 找数组中目标元素第一次出现的位置](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [while (left <= right)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int mid = left + (right - left) / 2;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if (heaters[mid] < house)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [left = mid + 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [else](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [right = mid - 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [// 供暖器在房子右边：](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int dist1 = (left == n) ? INT\_MAX : heaters[left] - house;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [// 供暖器在房子左边：](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int dist2 = (left == 0) ? INT\_MAX : house - heaters[right];](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [res = max(res, min(dist1, dist2));//将距离房子最近的供暖器距离纳入统计](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return res;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

[二分查找上下界问题。对加热器进行排序，通过二分查找距离最近的热水器，并求所有最近距离的最大值。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

**[528. 按权重随机选择](https://blog.csdn.net/smile_watermelon/article/details/47267089)**

[给定一个正整数数组 w ，其中 w[i] 代表下标 i 的权重（下标从 0 开始），请写一个函数 pickIndex ，它可以随机地获取下标 i，选取下标 i 的概率与 w[i] 成正比。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[例如，对于 w = [1, 3]，挑选下标 0 的概率为 1 / (1 + 3) = 0.25 （即，25%），而选取下标 1 的概率为 3 / (1 + 3) = 0.75（即，75%）。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[也就是说，选取下标 i 的概率为 w[i] / sum(w) 。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[["Solution","pickIndex"]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[[[1]],[]]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[null,0]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[解释：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[Solution solution = new Solution([1]);](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[solution.pickIndex(); // 返回 0，因为数组中只有一个元素，所以唯一的选择是返回下标 0。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[["Solution","pickIndex","pickIndex","pickIndex","pickIndex","pickIndex"]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[[[1,3]],[],[],[],[],[]]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[null,1,1,1,1,0]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[解释：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[Solution solution = new Solution([1, 3]);](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[solution.pickIndex(); // 返回 1，返回下标 1，返回该下标概率为 3/4 。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[solution.pickIndex(); // 返回 1](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[solution.pickIndex(); // 返回 1](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[solution.pickIndex(); // 返回 1](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[solution.pickIndex(); // 返回 0，返回下标 0，返回该下标概率为 1/4 。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[由于这是一个随机问题，允许多个答案，因此下列输出都可以被认为是正确的:](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[null,1,1,1,1,0]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[null,1,1,1,1,1]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[null,1,1,1,0,0]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[null,1,1,1,0,1]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[null,1,0,1,0,0]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[......](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[诸若此类。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [class Solution {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [vector<int> w\_;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [Solution(vector<int>& w) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [w\_.push\_back(w[0]);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [for (int i = 1; i < w.size(); ++i) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [w\_.push\_back(w\_.back() + w[i]);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int pickIndex() {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if (w\_.size() == 1) return 0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int i = rand() % w\_.back() + 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int pos = lower\_bound(w\_.begin(), w\_.end(), i) - w\_.begin();](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return pos;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

|  |
| --- |
| [class Solution {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [vector<int>\* data;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [Solution(vector<int>& w) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [data = new vector<int>(w.size());](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int sum = 0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [for (int i = 0; i < w.size(); ++i)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [(\*data)[i] = (sum += w[i]);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int pickIndex() {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int r = rand() % data->back();](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int start = 0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int end = data->size() - 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [while (start < end) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int mid = start + (end - start) / 2;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if ((\*data)[mid] <= r)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [start = mid + 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [else](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [end = mid;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return start;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

[二分查找上下界问题。首先计算一个累积的频数，根据总频数来进行随机，之后通过二分查找得到当前随机数对应的下标。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

**[875. 爱吃香蕉的珂珂](https://blog.csdn.net/smile_watermelon/article/details/47267089)**

[珂珂喜欢吃香蕉。这里有 N 堆香蕉，第 i 堆中有 piles[i] 根香蕉。警卫已经离开了，将在 H 小时后回来。珂珂可以决定她吃香蕉的速度 K （单位：根/小时）。每个小时，她将会选择一堆香蕉，从中吃掉 K 根。如果这堆香蕉少于 K 根，她将吃掉这堆的所有香蕉，然后这一小时内不会再吃更多的香蕉。珂珂喜欢慢慢吃，但仍然想在警卫回来前吃掉所有的香蕉。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[返回她可以在 H 小时内吃掉所有香蕉的最小速度 K（K 为整数）。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入: piles = [3,6,7,11], H = 8](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出: 4](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入: piles = [30,11,23,4,20], H = 5](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出: 30](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [class Solution {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int minEatingSpeed(vector<int>& piles, int H) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int left = 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int right = \*max\_element(piles.begin(), piles.end()) + 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [while (left < right) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int mid = left + (right - left) / 2;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if (CanEate(piles, H, mid)) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [right = mid;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [} else {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [left = mid + 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return left;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [bool CanEate(vector<int>& piles, int H, int speed) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int totalHours = 0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [for (size\_t i = 0; i < piles.size(); i++) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [totalHours += (piles[i] / speed) + ((piles[i] % speed == 0) ? 0 : 1);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return totalHours <= H;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

[二分猜答案问题。也就是在一定的求解空间中找到最小满足条件的值，使得KoKo能够以最慢的速度在特定时间内吃完所有香蕉。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

**[911. 在线选举](https://blog.csdn.net/smile_watermelon/article/details/47267089)**

[在选举中，第 i 张票是在时间为 times[i] 时投给 persons[i] 的。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[现在，我们想要实现下面的查询函数： TopVotedCandidate.q(int t) 将返回在 t 时刻主导选举的候选人的编号。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[在 t 时刻投出的选票也将被计入我们的查询之中。在平局的情况下，最近获得投票的候选人将会获胜。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入：["TopVotedCandidate","q","q","q","q","q","q"], [[[0,1,1,0,0,1,0],[0,5,10,15,20,25,30]],[3],[12],[25],[15],[24],[8]]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出：[null,0,1,1,0,0,1]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[解释：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[时间为 3，票数分布情况是 [0]，编号为 0 的候选人领先。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[时间为 12，票数分布情况是 [0,1,1]，编号为 1 的候选人领先。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[时间为 25，票数分布情况是 [0,1,1,0,0,1]，编号为 1 的候选人领先（因为最近的投票结果是平局）。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[在时间 15、24 和 8 处继续执行 3 个查询。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [class TopVotedCandidate {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [map<int, int> ret;//两个int分别对应时间和那个时间获胜的人](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [TopVotedCandidate(vector<int>& persons, vector<int>& times) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [vector<int>tickets(persons.size());//每个人的得分情况](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int n = persons.size();](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int max = 0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [for (int i = 0; i < n; i++) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [//如果当前这个人的票数+1 >= 当前的获胜者的票数，更新获胜者并记录他为当前时间戳的获胜者](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [max = (++tickets[persons[i]] >= tickets[max]) ? persons[i] : max;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [ret[times[i]] = max;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int q(int t) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return (--ret.upper\_bound(t))->second;//upper\_bound对应时间戳前一个就是答案](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

[二分查找上下界问题。主要是根据时间来二分，先预先计算好当前时间点对应的选举人，存到hashmap中。之后通过二分找到时间，再通过hash找到对应选举人。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

**[1011. 在 D 天内送达包裹的能力](https://blog.csdn.net/smile_watermelon/article/details/47267089)**

[传送带上的包裹必须在 D 天内从一个港口运送到另一个港口。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[传送带上的第 i 个包裹的重量为 weights[i]。每一天，我们都会按给出重量的顺序往传送带上装载包裹。我们装载的重量不会超过船的最大运载重量。返回能在 D 天内将传送带上的所有包裹送达的船的最低运载能力。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入：weights = [1,2,3,4,5,6,7,8,9,10], D = 5](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出：15](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[解释：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[船舶最低载重 15 就能够在 5 天内送达所有包裹，如下所示：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 1 天：1, 2, 3, 4, 5](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 2 天：6, 7](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 3 天：8](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 4 天：9](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 5 天：10](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[请注意，货物必须按照给定的顺序装运，因此使用载重能力为 14 的船舶并将包装分成 (2, 3, 4, 5), (1, 6, 7), (8), (9), (10) 是不允许的。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入：weights = [3,2,2,4,1,4], D = 3](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出：6](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[解释：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[船舶最低载重 6 就能够在 3 天内送达所有包裹，如下所示：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 1 天：3, 2](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 2 天：2, 4](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 3 天：1, 4](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入：weights = [1,2,3,1,1], D = 4](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出：3](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[解释：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 1 天：1](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 2 天：2](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 3 天：3](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[第 4 天：1, 1](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [class Solution {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int shipWithinDays(vector<int>& weights, int D) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int left = \*min\_element(weights.begin(), weights.end());](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int right = accumulate(weights.begin(), weights.end(), 0) + 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [while (left < right) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int mid = left + (right - left) / 2;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if (CanDo(weights, D, mid)) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [right = mid;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [} else {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [left = mid + 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return left;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [bool CanDo(vector<int>& weights, int D, int cap) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int index = 0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [for (size\_t i = 0; i < D; i++) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int eachCap = cap;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [while (eachCap - weights[index] >= 0) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [eachCap -= weights[index];](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [index++;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if (index == weights.size()) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [//printf("cap: %d day: %d can\n", cap, i);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return true;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return false;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

[二分猜答案问题。也就是在一定的求解空间中找到最小满足条件的值，使得货船能够以最小的容量在特定时间内运完所有货物。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

**[1146. 快照数组](https://blog.csdn.net/smile_watermelon/article/details/47267089)**

[实现支持下列接口的「快照数组」- SnapshotArray：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[SnapshotArray(int length) - 初始化一个与指定长度相等的 类数组 的数据结构。初始时，每个元素都等于 0。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[void set(index, val) - 会将指定索引 index 处的元素设置为 val。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[int snap() - 获取该数组的快照，并返回快照的编号 snap\_id（快照号是调用 snap() 的总次数减去 1）。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[int get(index, snap\_id) - 根据指定的 snap\_id 选择快照，并返回该快照指定索引 index 的值。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入：["SnapshotArray","set","snap","set","get"]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[[3],[0,5],[],[0,6],[0,0]]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出：[null,null,0,null,5]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[解释：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[SnapshotArray snapshotArr = new SnapshotArray(3); // 初始化一个长度为 3 的快照数组](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[snapshotArr.set(0,5); // 令 array[0] = 5](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[snapshotArr.snap(); // 获取快照，返回 snap\_id = 0](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[snapshotArr.set(0,6);](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[snapshotArr.get(0,0); // 获取 snap\_id = 0 的快照中 array[0] 的值，返回 5](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [//对于一个下标而言，我们不需要储存每一次snap的结果，只有它的值改变了，才有必要去储存, 比如说，下标1在snap\_id=10，100,1000的时候值被改变，那么snap\_id在10-100的查询就是snap\_id=10的值,snap\_id在100-1000的查询就是snap\_id=100的值,只需要查询upper\_bound即可](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [class SnapshotArray {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [map<int, map<int, int>>ret;//三个int分别表示index，snap\_id，val，表示index在snap\_id时刻的值](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [SnapshotArray(int length) {}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int cur=0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [void set(int index, int val) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if(ret[index].size()==0) ret[index][0]=0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [ret[index][cur] = val;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int snap() {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return cur++;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int get(int index, int snap\_id) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if(ret[index].size()==0) ret[index][0]=0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return (--ret[index].upper\_bound(snap\_id))->second;//upper\_boun之前一个就是答案](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

[二分查找上下界问题。每次修改值的时候记录一下当前快照的值，然后每次查找当前下标特定快照id时的值时，二分查找小于等于这一快照id对应的值即可。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

**[250. 统计同值子树](https://blog.csdn.net/smile_watermelon/article/details/47267089)**

[给定一个二叉树，统计该二叉树数值相同的子树个数。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[同值子树是指该子树的所有节点都拥有相同的数值。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入: root = [5,1,5,5,5,null,5]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[5](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[/ \](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[1 5](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[/ \ \](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[5 5 5](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[思路1：节点node若是同值子树点，则其左右子树首先都是同值子树点，并且左右孩子的val与node的val相同。介于此，遍历node的时候，对左右子树dfs返回一个bool值，若都为真，再将三者的val进行对比，否则直接返回false。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [class Solution {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int countUnivalSubtrees(TreeNode\* root) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int sum=0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [helper(root,sum);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return sum;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [bool helper(TreeNode\* node,int& sum){](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if(node==0){](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return true;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [bool le=helper(node->left,sum),ri=helper(node->right,sum);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if(le and ri){](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if(node->left and node->left->val!=node->val){](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return false;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if(node->right and node->right->val!=node->val){](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return false;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [++sum;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return true;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return false;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

[思路2：我觉得这个解法不好理解，但题解里写了我就写下。对于node节点的左右子树递归调用，将node的val直接作为参数传下去，如果不等那么之后返回false，但这个false仅意味着node不是同值子树。node的左右子树仍可能是。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [class Solution {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int countUnivalSubtrees(TreeNode\* root) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int res=0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [helper(root,0,res); //这里第二个参数写多少都行，因为root没有父节点](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return res;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [bool helper(TreeNode\* node,int val,int& sum){](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if(node==0){](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return true;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [bool le=helper(node->left,node->val,sum),ri=helper(node->right,node->val,sum);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if(le and ri){ //node是同值子树](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [++sum;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return node->val==val;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return false;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |

**[655. 输出二叉树](https://blog.csdn.net/smile_watermelon/article/details/47267089)**

[在一个 m\*n 的二维字符串数组中输出二叉树，并遵守以下规则：](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[行数 m 应当等于给定二叉树的高度。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[列数 n 应当总是奇数。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[根节点的值（以字符串格式给出）应当放在可放置的第一行正中间。根节点所在的行与列会将剩余空间划分为两部分（左下部分和右下部分）。你应该将左子树输出在左下部分，右子树输出在右下部分。左下和右下部分应当有相同的大小。即使一个子树为空而另一个非空，你不需要为空的子树输出任何东西，但仍需要为另一个子树留出足够的空间。然而，如果两个子树都为空则不需要为它们留出任何空间。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[每个未使用的空间应包含一个空的字符串""。](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入:](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[1](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[/ \](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[2 3](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[\](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[4](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出:](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[["", "", "", "1", "", "", ""],](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[["", "2", "", "", "", "3", ""],](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[["", "", "4", "", "", "", ""]]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输入:](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[1](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[/ \](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[2 5](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[/](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[3](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[/](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[4](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[输出:](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[[["", "", "", "", "", "", "", "1", "", "", "", "", "", "", ""]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[["", "", "", "2", "", "", "", "", "", "", "", "5", "", "", ""]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[["", "3", "", "", "", "", "", "", "", "", "", "", "", "", ""]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

[["4", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]]](https://blog.csdn.net/smile_watermelon/article/details/47267089)

|  |
| --- |
| [class Solution {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [vector<vector<string>> res;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [public:](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [vector<vector<string>> printTree(TreeNode\* root)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if (root == nullptr) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return {};](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int m = GetMaxDepth(root);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int n = pow(2, m) - 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [res = vector<vector<string>>(m, vector<string>(n, ""));](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [Dfs(root, 0, 0, n);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return res;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [void Dfs(TreeNode\* root, int depth, int start, int end)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if (root == nullptr) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [string val = to\_string(root->val);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int pos = (start + end) / 2;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [res[depth][pos] = val;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [Dfs(root->left, depth + 1, start, pos);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [Dfs(root->right, depth + 1, pos + 1, end);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int GetMaxDepth(TreeNode\* root)](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [{](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [if (root == nullptr) {](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return 0;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int left = GetMaxDepth(root->left);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [int right = GetMaxDepth(root->right);](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [return max(left, right) + 1;](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [}](https://blog.csdn.net/smile_watermelon/article/details/47267089)  [};](https://blog.csdn.net/smile_watermelon/article/details/47267089) |