

GeekBand 极客班

互联网人才加油站!



C++系统工程师



iOS开发工程师



Android开发工程师



PM产品经理

5. Sorting and Binary Search

大纲

1. 内排序
2. 外排序
3. 堆介绍
4. 二分搜索
5. 例题解析
6. Scalability & Memory Limits问题

内排序

所谓的内排序是指所有的数据已经读入内存，在内存中进行排序的算法。排序过程中不需要对磁盘进行读写。同时，内排序也一般假定所有用到的辅助空间也可以直接存在于内存中。与之对应地，另一类排序称作外排序，即内存中无法保存全部数据，需要进行磁盘访问，每次读入部分数据到内存进行排序。

- a. Merge Sort
- b. Quick Sort
- c. Heap Sort
- d. Bucket Sort (桶排序) 和 Radix Sort (基数排序)

Merge Sort

Merge sort是一种典型的排序算法，应用“分而治之(divide and conquer)”的算法思路：将线性数据结构（如array, vector或list）分为两个部分，对两部分分别进行排序，排序完成后，再将各自排序好的两个部分合并还原成一个有序数组。由于merge sort不依赖于随机读写，因此具有很强的普适性，适用于list等数据结构。算法的时间复杂度 $O(n\log n)$ ，需要额外 $O(n)$ 空间。

Quick Sort

Quick sort是最为常用的排序算法，C++自带的排序算法实现的就是quick sort。该算法以其高效性，简洁性，被评为20世纪十大算法之一。Quick sort的算法核心与merge sort类似，也采用“分而治之”的想法：随机选定一个元素作为轴值，利用该轴值将数组分为左右两部分，左边元素都比轴值小，右边元素都比轴值大，但它们不是完全排序的。在此基础上，分别对左右两部分分别递归调用quick sort，使得左右部分完全排序。算法的平均时间复杂度是 $O(n\log n)$ ，在最坏情况下为 $O(n^2)$ ，额外空间复杂度 $O(\log n)$ 。

Heap Sort

Heap sort利用了heap作为逻辑储存结构，将输入array变成一个最大值堆。然后，我们反复进行堆的弹出操作。回顾之前所述的弹出过程：将堆顶元素与堆末元素交换，堆的大小减一，向下移动新的堆顶以维护堆的性质。事实上，该操作等价于每次将剩余的最大元素移动到数组的最右边，重复这样的操作最终就能获得由小到大排序的数组。初次建堆的时间复杂度 $O(n)$ ，删除堆顶元素并维护堆的性质需要 $O(\log n)$ ，这样的操作一共进行 n 次，故最终时间复杂度 $O(n \log n)$ 。我们不需要利用额外空间，故空间复杂度 $O(1)$ 。

```
void heapSort(int array[], int size) {  
    Heapify(array, size);  
    for (int i = 0; i < size - 1; i++)  
        popHeap(array);  
}
```


Bucket Sort & Radix Sort

Bucket sort 和 Radix sort 不需要进行数据之间的两两比较，但是需要事先知道数组的一些具体情况。特别地，bucket sort 适用于知道待排序数组大小范围的情况。其特性在于将数据根据其大小，放入合适的“桶(容器)”中，再依次从桶中取出，形成有序序列。

GeekBand

外排序

外排序算法的核心思路在于把文件分块读到内存，在内存中对每块文件依次进行排序，最后合并排序后的各块数据，依次按顺序写回文件。相比于内排序，外排序需要进行多次磁盘读写，因此执行效率往往低于内排序，时间主要花费于磁盘读写上。算法步骤如下：

假设文件需要分成 k 块读入，需要从小到大进行排序

- 1) 依次读入每个文件块，在内存中对当前文件块进行排序(应用恰当的内排序算法)。此时，每块文件相当于一个由小到大排列的有序队列
- 2) 在内存中建立一个最小值堆，读入每块文件的队列头
- 3) 弹出堆顶元素，如果元素来自第 i 块，则从第 i 块文件中补充一个元素到最小值堆。弹出的元素暂存至临时数组
- 4) 当临时数组存满时，将数组写至磁盘，并清空数组内容。
- 5) 重复过程3)，4)，直至所有文件块读取完毕

Quick Selection

快速选择算法能够在平均 $O(n)$ 时间内从一个无序数组中返回第 k 大的元素。算法实际上利用了快速排序的思想，将数组依照一个轴值分割成两个部分，左边元素都比轴值小，右边元素都比轴值大。由于轴值下标已知，则可以判断所求元素落在数组的哪一部分，并在那一部分继续进行上述操作，直至找到该元素。与快排不同，由于快速选择算法只在乎所求元素所在的那一部分，故效率可以从 $O(n\log n)$ 进一步提升至 $O(n)$ 。

Heap

iff it is empty or the key in the root is larger than that in either child and both subtrees have the heap property.

Operations

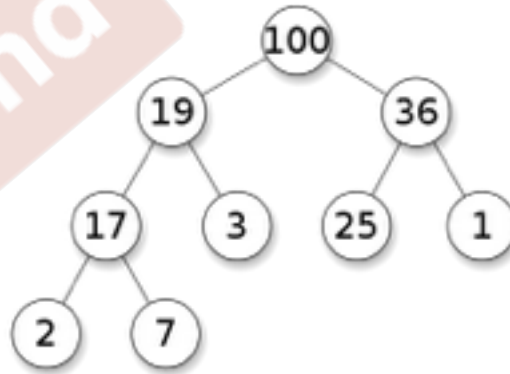
Add $O(\log N)$

Remove $O(\log N)$

Min/Max $O(1)$

Build heap: $O(n)$ why?

Heap sort: $O(n \log n)$



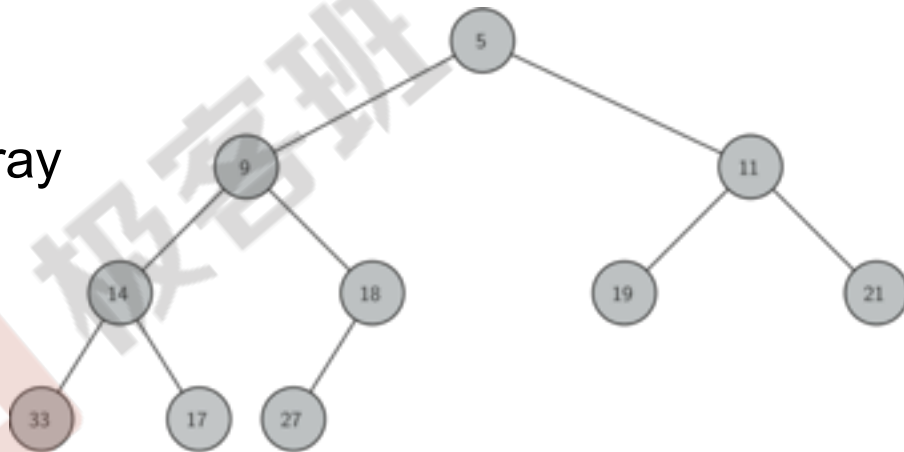
Heap - Implementation

Low level data structure: Dynamic Array

```
Heap {  
    elems[], size;  
}
```

elems[0] - root,
also the minimum elem in elems.
i's left child: $i*2$, right child: $i*2+1$

Internal Method:
siftup, siftdown



0	5	9	11	14	18	19	21	33	17	27	
0	1	2	3	4	5	6	7	8	9	10	11

Heap 特点

能够很方便地维护动态数据的最大值、最小值或中位数。

`priority_queue`相比简单的`queue`更适用于需要优先级的情境。

与`queue`类似，该结构适合 `push`和`pop`，但不方便`update`和检索(仅仅部分有序)。

Heap & Priority Queue

C++:

```
priority_queue<int> myPriorityQueue;  
myPriorityQueue.push(30);  
myPriorityQueue.push(10);  
myPriorityQueue.push(50);  
myPriorityQueue.push(40);
```

```
cout << "Popping out elements...";  
while (!myPriorityQueue.empty()) {  
    cout << ' ' << myPriorityQueue.top();  
    myPriorityQueue.pop();  
}  
cout << endl;
```

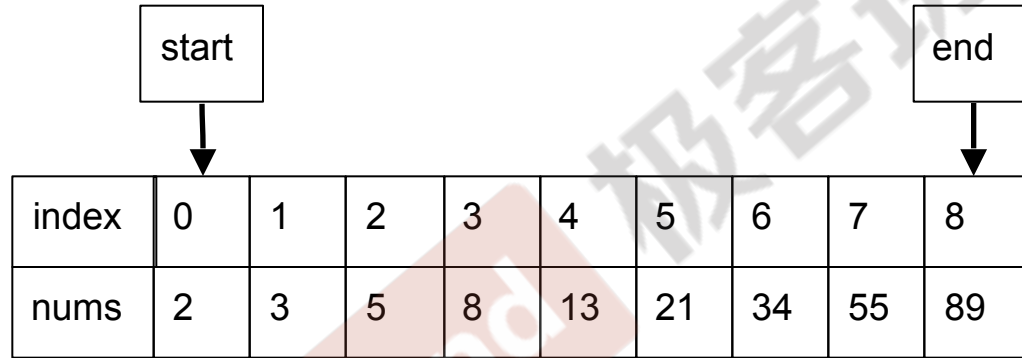
// output: Popping out elements... 50 40 30 10

Binary Search

Given an sorted integer array - nums, and an integer - target. Find the first position of target in nums, return -1 if target doesn't exist.

```
public int binarySearch(int[] nums, int target)
```


How we do binary search?

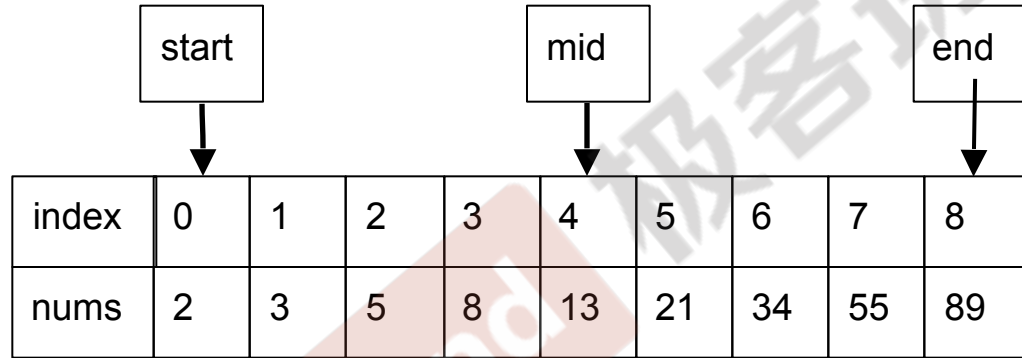


The diagram illustrates the initial state of a binary search. A 'start' box points to index 0 and an 'end' box points to index 8 of a sorted array. The array contains the values [2, 3, 5, 8, 13, 21, 34, 55, 89].

start									end
	↓								↓
index	0	1	2	3	4	5	6	7	8
nums	2	3	5	8	13	21	34	55	89

1. Find 5

How we do binary search?

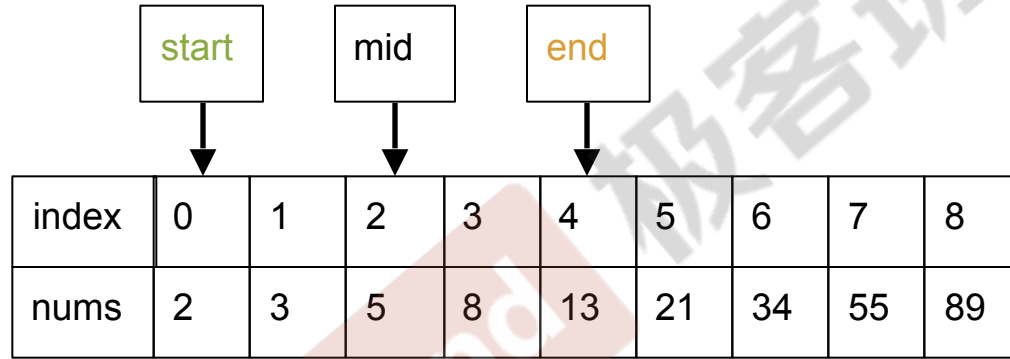


The diagram illustrates the initial state of a binary search on a sorted array. Three boxes labeled 'start', 'mid', and 'end' are positioned above the array. Arrows point from 'start' to index 0, from 'mid' to index 4, and from 'end' to index 8. The array itself is a table with two rows: 'index' and 'nums'. The 'index' row contains values from 0 to 8, and the 'nums' row contains the values 2, 3, 5, 8, 13, 21, 34, 55, and 89. The columns for index 4 and value 13 are highlighted with a light red background.

	start				mid				end
	↓				↓				↓
index	0	1	2	3	4	5	6	7	8
nums	2	3	5	8	13	21	34	55	89

1. Find 5, mid=4

How we do binary search?

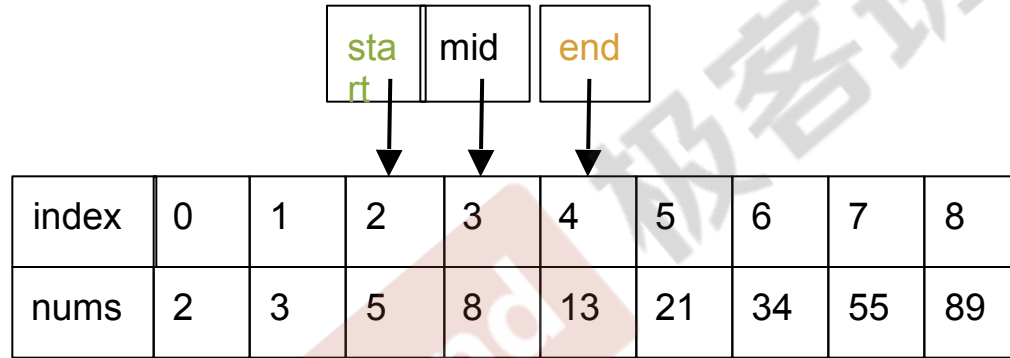


The diagram illustrates the initial state of a binary search on a sorted array. Three boxes labeled 'start', 'mid', and 'end' are positioned above the array. Arrows point from 'start' to index 0, from 'mid' to index 2, and from 'end' to index 4. The array itself is a table with two rows: 'index' and 'nums'. The 'index' row contains values from 0 to 8, and the 'nums' row contains the values 2, 3, 5, 8, 13, 21, 34, 55, and 89. The cell at index 2 containing the value 5 is highlighted in red.

	start		mid		end				
	↓		↓		↓				
index	0	1	2	3	4	5	6	7	8
nums	2	3	5	8	13	21	34	55	89

1. Find 5, mid=4, 2. Find it!

How we do binary search?



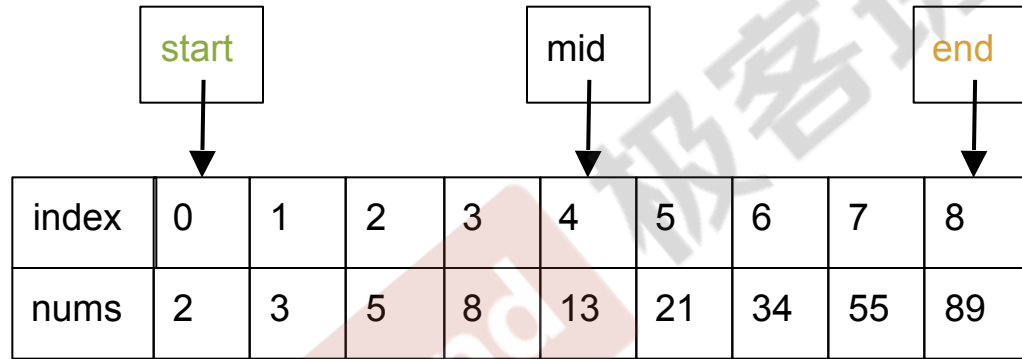
The diagram illustrates the initial state of a binary search on a sorted array. Above the array, three boxes represent the search range: 'start' (green text, split as 'sta' and 'rt'), 'mid' (black text), and 'end' (orange text). Arrows point from the 'start' box to index 2, from the 'mid' box to index 3, and from the 'end' box to index 4. The array itself is a table with two rows: 'index' and 'nums'. The 'index' row contains values from 0 to 8. The 'nums' row contains the values 2, 3, 5, 8, 13, 21, 34, 55, and 89. The columns for index 2, 3, and 4 are highlighted with a light red background.

index	0	1	2	3	4	5	6	7	8
nums	2	3	5	8	13	21	34	55	89

1. Find 5, mid=4, 2. Find it!

2. Find 8, mid=4, 2, 3. Find it!

How we do binary search?

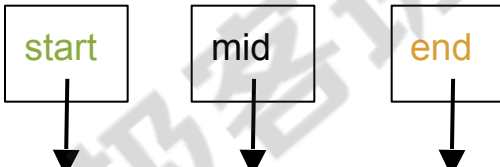


The diagram illustrates the initial state of a binary search on a sorted array. Three boxes labeled 'start', 'mid', and 'end' are positioned above the array. Arrows point from 'start' to index 0, from 'mid' to index 4, and from 'end' to index 8. The array itself is a table with two rows: 'index' and 'nums'.

index	0	1	2	3	4	5	6	7	8
nums	2	3	5	8	13	21	34	55	89

1. Find 5, mid=4, 2. Find it!
2. Find 8, mid=4, 2, 3. Find it!
3. Find 14, mid=4

How we do binary search?



The diagram illustrates the initial state of a binary search on a sorted array. Three boxes labeled 'start' (green), 'mid' (black), and 'end' (orange) are positioned above the array. Arrows from each box point to the element at index 4, which has the value 13. The array itself is a table with two rows: 'index' and 'nums'.

index	0	1	2	3	4	5	6	7	8
nums	2	3	5	8	13	21	34	55	89

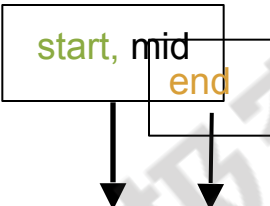
1. Find 5, mid=4, 2. Find it!
2. Find 8, mid=4, 2, 3. Find it!
3. Find 14, mid=4, 6

How we do binary search?

					<table><tr><td>start</td><td></td><td>mid</td><td></td><td>end</td></tr><tr><td>↓</td><td></td><td>↓</td><td></td><td>↓</td></tr></table>			start		mid		end	↓		↓		↓					
start		mid		end																		
↓		↓		↓																		
index	0	1	2	3	4	5	6	7	8													
nums	2	3	5	8	13	21	34	55	89													

1. Find 5, mid=4, 2. Find it!
2. Find 8, mid=4, 2, 3. Find it!
3. Find 14, mid=4, 6, 5

How we do binary search?



index	0	1	2	3	4	5	6	7	8
nums	2	3	5	8	13	21	34	55	89

1. Find 5, mid=4, 2. Find it!
2. Find 8, mid=4, 2, 3. Find it!
3. Find 14, mid=4, 6, 5, 4. Return -1

Recursion or While-Loop?

对于有序线性容器的搜索，二分查找或其变种基本上是解题的最佳方法

binary search template in recursion

```
int binarySearch(int *array, int left, int right, int value) {  
    if (left > right) {  
        // value not found  
        return -1;  
    }  
  
    int mid = left + (right - left) / 2;  
    if (array[mid] == value) {  
        return mid;  
    } else if (array[mid] < value) {  
        return binarySearch(array, mid + 1, right, value);  
    } else {  
        return binarySearch(array, left, mid - 1, value);  
    }  
}
```

Keypoints:

1. $\text{start} + 1 < \text{end}$
2. $\text{left} + (\text{right} - \text{left}) / 2$
3. $A[\text{mid}] ==, <, >$
4. $A[\text{start/end}] == \text{target}$

A generic binary search template

```
int binary_search(const int a[], const int size, const int val) {  
    int lower = 0;  
    int upper = size-1;  
    /* invariant: if a[i]==val for any i, then lower <= i <= upper */  
    while (lower <= upper) {  
        int i = lower + (upper-lower)>>1;  
        if (val == a[i]) {  
            return i;  
        } else if (val < a[i]) {  
            upper = i-1;  
        } else { /* val > a[i] */  
            lower = i+1;  
        }  
    }  
    return -1;  
}
```

From Programming Pearls

模式识别

1. 当题目中出现“前 k 个”，“合并 / 排序 k 组数据”或者“数据流(即不知道全部数据，而是每次读入一个)”时，可以考虑使用heap，动态地维护最大值 / 最小值信息

GeekBand

Merge K Sorted List

Suppose you are given k sorted linked lists, try to implement a function to merge these linked lists, and return one sorted list containing all elements.

GeekBand

K-th Largest

Design an algorithm for computing the k-th largest element in a sequence of elements, one presented at a time

GeekBand

Closest to Origin

There are n points on a 2D plan, find the k points that are closest to origin ($x= 0$, $y= 0$).

GeekBand

极客班

模式识别

2. 对于有序 / 部分有序容器的搜索

GeekBand

极客班

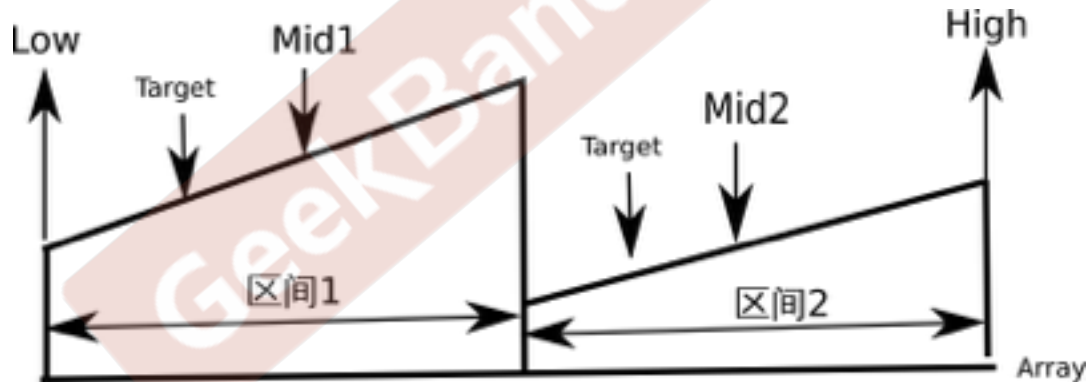
Find i in a given sorted array that $arr[i] == i$.

在此例中， $A[3] = 3$ 。同时，不难发现一个规律： $A[3]$ 左侧的数据满足 $value < index$ ， $A[3]$ 右侧的数据满足 $value > index$ 。

Index	0	1	2	3	4	5	6	7	8
Value	- 7	-2	0	3	7	9	10	12	13

Search in Rotated Sorted Array

An array is sorted without duplicates. However, someone mysteriously shifted all the elements in this array (e.g. 1,2,3,4,5 -> 5,1,2,3,4). Implement a function to find an element in such array (return -1 if no such element).



Range Search

Given a sorted array of integers with duplicates. Implement a function to get the start and end position of a given value.

GeekBand

极客班

矩阵搜索

Check if an element is in a $M \times N$ matrix, each row and column of which is sorted.

我们可以构造一个矩阵：

1	5	10	20
2	6	11	30
7	9	12	40
8	15	31	41

如果要在上述矩阵中找到9，应该如何计算？最简单的方法显然是遍历每行每列，这样的时间复杂度是 $O(n^2)$ ，而且完全没有利用到矩阵已经部分有序的特性。

Search a 2D Matrix II

Write an efficient algorithm that searches for a value in an $m \times n$ matrix. This matrix has the following properties:

Integers in each row are sorted from left to right.

The first integer of each row is greater than the last integer of the previous row.

For example,

Consider the following matrix:

```
[  
  [1, 3, 5, 7],  
  [10, 11, 16, 20],  
  [23, 30, 34, 50]  
]
```

Given target = 3, return true.

Find the Square Root

Implement `sqrt(x)`, which returns the square root of value `x`.

http://www.nowamagic.net/algorithm/algorithm_EfficacyOfFunctionSqrt.php

Median of Two Sorted Array

Given two sorted arrays A and B of size m and n respectively. Find the median of these two sorted arrays. The overall run time complexity should be $O(\log(m + n))$.

何谓"Median"? 由题目意思可得即为两个数组中一半数据比它大, 另一半数据比它小的那个数。题中已有信息两个数组均为有序, 题目要求时间复杂度为 $O(\log)$, 因此应该往二分法上想。

在两个数组中找第k大数->找中位数即为找第k大数的一个特殊情况——第 $(A.length + B.length) / 2$ 大数。因此首先需要解决找第k大数的问题

使用归并的思想逐个比较找出中位数的复杂度为 $O(n)$, 显然不符要求, 接下来考虑使用二分法。由于是找第k大数, 使用二分法则比较 $A[k/2 - 1]$ 和 $B[k/2 - 1]$, 并思考这两个元素和第k大元素的关系。

$A[k/2 - 1] \leq B[k/2 - 1] \Rightarrow$ A和B合并后的第k大数中必包含 $A[0] \sim A[k/2 - 1]$, 可使用归并的思想去理解。若 $k/2 - 1$ 超出A的长度, 则必取 $B[0] \sim B[k/2 - 1]$

Given a server that has requests coming in. Design a data structure such that you can fetch the count of the number requests in the last second, minute and hour.

GeekBand

模式识别

3. 数据范围有限(或存在大量重复数据)的排序问题，一般可以使用 Bucket Sort。对于有限位数的数据(如string, vector<int>, int)，可以利用 Radix Sort 进行数值序或词典序排序。

Sort a large number of people by their ages.

GeekBand

极客班

Sort an array of strings that all the anagrams are next to each other.

GeekBand

极客班

Scalability & Memory Limits 问题

对这类问题一般采用Divide & Conquer策略，即对问题进行预处理，将问题的输入进行分割、归类（**sorting**），放入相应的Bucket（单机上的某一块**Chunk**，或者分布式系统中的一台单机），再对每个Bucket进行后期处理，最后合并结果。

整个过程中应该用到hash function: 对于Memory Limits问题，一般可以直接利用hash function建立对象到索引的直接映射；对Scalability问题，一般可以用hash table来记录对象与储存该对象的机器之间的映射，在该机器上进一步做映射以获得索引。

A library is trying to build up a smart computer-aided look up system: user may input a list of key words, and the system shall provide all books that contain these words. How to implement such query? (A library may have millions of books)

GeekBand

Design the data structures for a very large social network and an algorithm to show the connection between two people.

GeekBand

Suppose you are given an extremely large set of URLs, 10 billion for example. How do you design an algorithm to detect the duplicate ones?

GeekBand

Homework

GeekBand

极客班

Find the First Bad Version

The code base version is an integer and start from 0 to n. One day, someone commit a bad version in the code case, so it caused itself and the following versions are all failed in the unittests. You can determine whether a version is bad by the following interface:

```
boolean isBadVersion(int version);
```

Find Peak Element

There is an integer array which has the following features:

- * The numbers in adjacent positions are different.
- * $A[0] < A[1]$ && $A[A.length - 2] > A[A.length - 1]$.

We define a position P is a peak if $A[P] > A[P-1]$ && $A[P] > A[P+1]$.
Find a peak element in this array. Return the index of the peak.

Note

The array may contains multiple peaks, find any of them.

Example

[1, 2, 1, 3, 4, 5, 7, 6]

return index 1 (which is number 2) or 6 (which is number 7)

Nuts and Bolts

Given a set of n nuts of different sizes and n bolts of different sizes. There is a one-one mapping between nuts and bolts. Comparison of a nut to another nut or a bolt to another bolt is not allowed. It means nut can only be compared with bolt and bolt can only be compared with nut to see which one is bigger/smaller.

We will give you a compare function to compare nut with bolt.

Example

Given nuts = ['ab','bc','dd','gg'], bolts = ['AB','GG', 'DD', 'BC'].

Your code should find the matching bolts and nuts.

one of the possible return:

nuts = ['ab','bc','dd','gg'], bolts = ['AB','BC','DD','GG'].

we will tell you the match compare function. If we give you another compare function.

the possible return is the following:

nuts = ['ab','bc','dd','gg'], bolts = ['BC','AA','DD','GG'].

So you must use the compare function that we give to do the sorting.

The order of the nuts or bolts does not matter.

You just need to find the matching bolt for each nut.