Welcome to Interview Kickstart !

Sorting Live Class

INTERVIEW
KICKSTART

{ik} INTERVIEW KICKSTART

Agenda

Introduction to Problem-solving and Coding Patterns (30 min)

Interview problems on Sorting:

1. Presorting (45 min)
2. Extensions of Merge Sort (45 min)
   Lunch/Dinner break (20 min)
3. Extensions of Quicksort (1 hr)
4. Extensions of Heapsort (1 hr)

The two halves of interview prep

Problem-solving

Coding fluency

Problem-solving

A problem is a question for which you have no idea (at least initially) what method will work. You *will* feel clueless at the beginning.

Problems are not exercises.

Example exercise: 3456789 X 43289763 = ?

It is hard, but there is no doubt what method to use.

A problem requires sustained investigation, deploying *problem-solving strategies*.

Mathematical problems, scientific problems, and

computational problems.

The intent of an ideal technical interview

An *algorithm* is the method to solve a computational problem.

Implemented as a *program*.

Can you design a correct and efficient algorithm that solves the problem, and code it up?

This tests your problem-solving skills, your coding skills and your knowledge of computer science and engineering fundamentals.

“But a real interview is a timed test!”

How do we tackle the tension between problem-solving (which needs time) and a severe time limit?

1. Mug up solutions to ~ 1300 Leetcode questions and hope every question you will face will be an exercise?
2. Become a problem-solver and develop inner confidence, while getting exposed to popular interview questions grouped by coding patterns?

news* & views*

Research suggests timed tests cause math anxiety

JO BOALER, PROFESSOR OF MATHEMATICS EDUCATION, STANFORD UNIVERSITY

Teachers in the United States are often forced to follow directives that make little sense to them and are far removed from research evidence. One of the initiatives mandated by many school districts that I place high in the category of uninformed policy is the use of timed tests to assess math facts and fluency. Teachers and administrators use these tests with the very best of intentions, but they use them without knowledge of the important evidence that is emerging from neuroscience.

IN MY OPINION

early onset of math anxiety. Indeed, researchers now know that students experience stress on timed tests that they do not experience even when working on the same math questions in untimed conditions (Engle 2002).

In a recent study of 150 first and second graders, researchers measured students' levels of math anxiety, finding that children as young as first grade experienced it and that levels of math anxiety did not correlate with grade level, reading level, or parental income (Ramirez et al. 2013). Other researchers analyzed

‹ Back | Anyone having massive anxiety?

Anonymous User Last Edit: November 17, 2019 10:34 PM 12.2K VIEWS

196

I don't know why, but for the last several months I was having massive anxiety in preparation of upcoming interviews. I'm a veteran engineers (almost 20 years in the industry), and had lost my job this year that was out of my control, and the idea of being asked coding questions drove me crazy. I wish your years of experience, work references, and the fact that you have a college degree actually meant something.

Although I've learned to accept it. I even had to learn relaxation techniques to remain calm. I'll even admit this anxiety has caused me some physical health issue.

Its strange because I've always been a highly productive and outstanding engineering in any role that I've taken (large company or startup).

In any case, I'm curious to hear if you've had this problem. How did you deal with it? How do you see these interviews?

anxiety stress

Comments: 53 Best | Most Votes | Newest to Oldest | Oldest to Newest

Type comment here... (Markdown is supported)

Post

cool_shark ★ 297 Last Edit: July 25, 2019 5:38 AM

I also consider myself as a veteran engineer (20 years in the industry). I get calls, emails and LinkedIn requests from big companies and I usually ignore those, and while I'm perfectly happy at my job, this year, I responded to the interview requests, and reluctantly accepted to do an interview for several big companies. I was lucky to pass phone interviews for couple of big companies, and then I had to go for on-site interviews (some required flying). Then, I notice I have an anxiety starting. Practicing LeetCode and studying did not help reducing the anxiety. I've done the preparation (more of a hobby) for almost 6 months including 4 months of LeetCode. In my case, the anxiety was due to my inner telling that I was not ready. I was in a position where, even if I pass an on-site interview, I would not be able to accept the offer however good the offer is. So, I told recruiters to contact me later, and cancel on-site interviews. The anxiety was gone, completely. I decided to do more preparation, and then have interviews when I know I'm ready.

The job market has changed drastically -- In the past, companies were measuring knowledge of design patterns (Gang of Four), general software concepts and basic coding skills. These days, even an unknown company tries to ask questions from LeetCode, Cracking the Coding Interview or EPI. and system design (Design Twitter in 20 minutes - Duh!). In other words, the bar has been raised very high, and the interview became like a college entrance exam. It requires dedicated time to prepare for interview. The question is how much preparation is needed. Some people brag that they only solved 80 questions in LeetCode and got a job at Facebook, etc., and it might be possible if the person had PhD specialized in algorithms. But for the rest of us who have daily jobs and became rusty with algorithms and interview crap, I honestly think it requires *minimum six months to two years* of grinding to pass interview at big tech companies (your mileage may vary). You may take a short cut, but what that means is that you're underprepared. I think your massive

https://leetcode.com/discuss/career/343147/anyone-having-massive-anxiety

Approach 1

Mug up solutions to ~ 1300 Leetcode questions and hope every question you will face will be an exercise?

https://www.reddit.com/r/cscareerquestions/comments/7nkk10/feel_like_im_just_trying_to_memorize_the/

14

Posted by u/cs_throw_away110 1 year ago

feel like im just trying to memorize the solutions from leetcode?

hi guys (and gals), I've been doing leetcode for the past two month-ish.

At first, the struggle was real but now I can comfortably solve easy and some mediums (depending on the condition & questions). However, it seems that a lot of medium questions have their own tricks to solve and I just feel like I'm just trying to memorize these tricks rather than figuring out the solution myself. Is this normal? The problem with this is that if I come across a completely new question (that requires a new trick to solve), I am way off the tangent when I'm trying to solve the question on my own and feel like there would would be just too many to memorize.

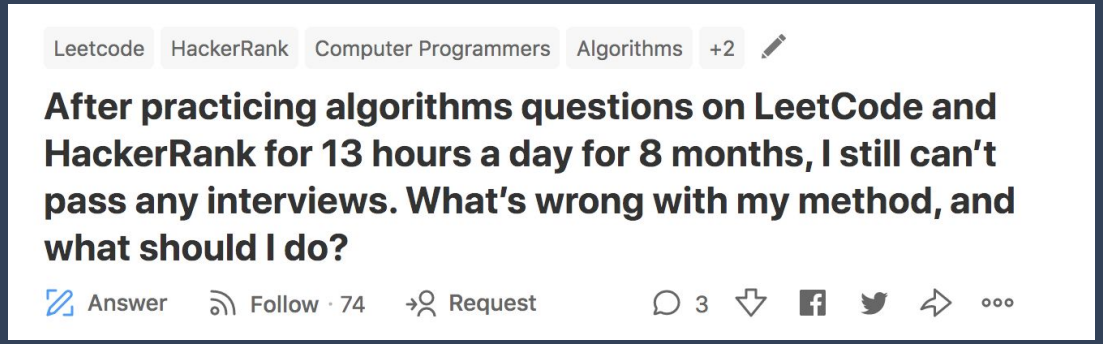
I've already seen other posts on how they went through this process and landed their dream job but they did not really specify on this topic; was wondering if I could get your experience and 2 cents, cheers!

9 Comments Share Save Hide Report

Approach 1

Mug up solutions to ~ 1300 Leetcode questions and hope every question you will face will be an exercise?

Needs too much time and you're still dependent on luck.



Doesn't help you become a better engineer.

Approach 2

Become a problem-solver and develop *inner confidence*, while getting exposed to popular interview questions grouped by coding patterns.

At IK, you should be learning problem-solving (for new problems) and coding fluency (for exercises)

Learn to tackle questions you have not seen before. Be prepared for the worst-case.

We will be giving you popular interview questions as problems. Avoid looking up solutions and train yourself to become a problem-solver when you see a new question.

Many popular interview questions (problems) will become exercises. But do not forget the problem-solving path you followed to first solve it.

Once a problem has become an exercise, try to improve your coding fluency on it (coding patterns help here).

912. Sort an Array

Medium 94 107 Favorite Share

Given an array of integers `nums`, sort the array in ascending order.

Example 1:

```
Input: [5,2,3,1]
Output: [1,2,3,5]
```

Example 2:

```
Input: [5,1,1,2,0,0]
Output: [0,0,1,1,2,5]
```

Is this a problem or an exercise?

How fast can you code this up?

```python
class Solution(object):
    def sortArray(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """

        def helper(A,start,end):
            if start >= end:
                return
            #Divide
            mid = (start+end)/2
            helper(A,start,mid)
            helper(A,mid+1,end)

            #Merge
            i = start
            j = mid+1
            aux = []
            while i <= mid and j <= end:
                if A[i] <= A[j]:
                    aux.append(A[i])
                    i += 1
                else: #A[i] > A[j]
                    aux.append(A[j])
                    j += 1
            while i <= mid:
                aux.append(A[i])
                i += 1
            while j <= end:
                aux.append(A[j])
                j += 1
            #Copy the aux array back into the original array
            A[start:end+1] = aux

        helper(nums,0,len(nums)-1)
        return nums
```

Mergesort: Target < 10 min

{ik} | INTERVIEW KICKSTART

```python
import random
class Solution(object):
    def sortArray(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """

        def helper(A,start,end):
            if start >= end:
                return
            pivotindex = random.randint(start,end)
            A[start],A[pivotindex] = A[pivotindex],A[start]
            orange = start
            for green in range(start+1,end+1):
                if A[green] < A[start]:
                    orange += 1
                    A[orange],A[green] = A[green],A[orange]
            A[start],A[orange] = A[orange],A[start]

            helper(nums,start,orange-1)
            helper(nums,orange+1,end)

        helper(nums,0,len(nums)-1)

        return nums
```

Quicksort:
Target < 10 min

Heapsort: Target < 5 min

```python
import heapq

class Solution(object):
    def sortArray(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """
        heapq.heapify(nums)
        result = []
        while len(nums) != 0:
            result.append(heapq.heappop(nums))
        return result

```

Problem-solving mindset

Mental discipline: Long, wild-goose chases and open-ended experimentation.
- Timescale of concentration required needs to increase.
- Failure: Investigation is *always* worthwhile even if it leads to a dead-end.

Emotional attitude: Mountaineer vs Gym rat
“The explorer is the person who is lost”

Problem-solving is a craft that can be learnt

- Build background (playground) knowledge
- Learn how other people solved problems
- Active engagement

Takeaways from pre-class videos

1. General algorithm design strategies: brute-force, decrease-and-conquer, divide-and-conquer, transform-and-conquer
2. Algorithm analysis (correctness and efficiency)
   - Big-Oh notation
   - Worst-case, Average-case, Best-case
3. Sorting problem as a case study
   - Merge Sort (Divide-and-conquer), worst-case O(n log n)
   - Quick Sort (Divide-and-conquer), average-case O(n log n)
   - Heap Sort (Transform-and-conquer), worst-case O(n log n)

Part 1

Presorting

Sorting as a pre-processing step / building block of more complex algorithms

1. Searching
2. Closest pair
3. Element uniqueness (check for duplicates)
4. Frequency distribution
5. Select kth largest/kth smallest/median

Two sum

You have an array of n numbers and a number *target*. Find out whether the array contains two elements whose sum is *target*.

For example, for the array [5, 9, 1, 3] and target = 6, the answer is yes.

For the same array and target = 7, the answer is no.

1. Two Sum

Easy 11768 406 Favorite Share

Given an array of integers, return **indices** of the two numbers such that they add up to a specific target.

You may assume that each input would have ***exactly*** one solution, and you may not use the *same* element twice.

Example:

```
Given nums = [2, 7, 11, 15], target = 9,

Because nums[0] + nums[1] = 2 + 7 = 9,
return [0, 1].
```

167. Two Sum II - Input array is sorted

Easy 1207 487 Favorite Share

Given an array of integers that is already ***sorted in ascending order***, find two numbers such that they add up to a specific target number.

The function twoSum should return indices of the two numbers such that they add up to the target, where index1 must be less than index2.

Note:

- Your returned answers (both index1 and index2) are not zero-based.
- You may assume that each input would have *exactly* one solution and you may not use the *same* element twice.

Example:

```
Input: numbers = [2,7,11,15], target = 9
Output: [1,2]
Explanation: The sum of 2 and 7 is 9. Therefore index1 = 1, index2 = 2.
```

252. Meeting Rooms

Easy 361 26 Favorite Share

Given an array of meeting time intervals consisting of start and end times `[[s1,e1],[s2,e2],...]` ($s_i < e_i$), determine if a person could attend all meetings.

Example 1:

```
Input: [[0,30],[5,10],[15,20]]
Output: false
```

Example 2:

```
Input: [[7,10],[2,4]]
Output: true
```

Presorting problems

Common patterns:
Sorting plus binary search
Sorting plus one pass
Sorting plus two-pointer pass

Part 2

1213. Intersection of Three Sorted Arrays

Easy 37 3 Favorite Share

Given three integer arrays `arr1`, `arr2` and `arr3` **sorted** in **strictly increasing** order, return a sorted array of **only** the integers that appeared in **all** three arrays.

Example 1:

```
Input: arr1 = [1,2,3,4,5], arr2 = [1,2,5,7,9], arr3 = [1,3,4,5,8]
Output: [1,5]
Explanation: Only 1 and 5 appeared in the three arrays.
```

Constraints:

- `1 <= arr1.length, arr2.length, arr3.length <= 1000`
- `1 <= arr1[i], arr2[i], arr3[i] <= 2000`

Intersection of two sorted arrays

Write a program that takes as input two sorted arrays, and returns a new array containing elements that are present in both of the input arrays.

The input arrays may have duplicate entries, but the returned array should be free of duplicates.

For example, if the input is
A1 = [2,3,3,5,5,6,7,7,8,12]
A2 = [5,5,6,8,8,9,10,10]
the output should be [5,6,8]

#What if arrays are of similar sizes? What if they are of very different sizes?

```
def intersect(a1,a2):

  i = 0
  j = 0
  aux = []
  while i < len(a1) and j < len(a2):
    if a1[i] < a2[j]:
      i += 1
    elif a1[i] > a2[j]:
      j += 1
    else: #a1[i] == a2[j]
      if len(aux) == 0 or aux[len(aux)-1] != a1[i]:
        aux.append(a1[i])
      i += 1
      j += 1
  return aux

a1 = [2,3,3,5,5,6,7,7,8,12]
a2 = [5,5,6,8,8,9,10,10]
```

88. Merge Sorted Array

Easy 1275 3041 Favorite Share

Given two sorted integer arrays *nums1* and *nums2*, merge *nums2* into *nums1* as one sorted array.

Note:

- The number of elements initialized in *nums1* and *nums2* are *m* and *n* respectively.
- You may assume that *nums1* has enough space (size that is greater or equal to *m* + *n*) to hold additional elements from *nums2*.

Example:

```
Input:
nums1 = [1,2,3,0,0,0], m = 3
nums2 = [2,5,6],       n = 3

Output: [1,2,2,3,5,6]
```

Part 3

215. Kth Largest Element in an Array

Medium 2229 183 Favorite Share

Find the **k**th largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.

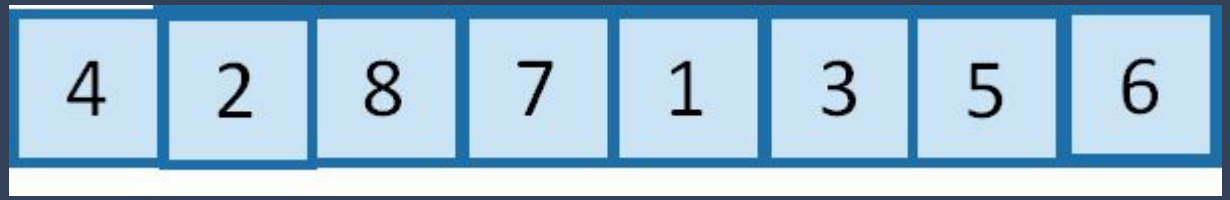
Example 1:

```
Input: [3,2,1,5,6,4] and k = 2
Output: 5
```

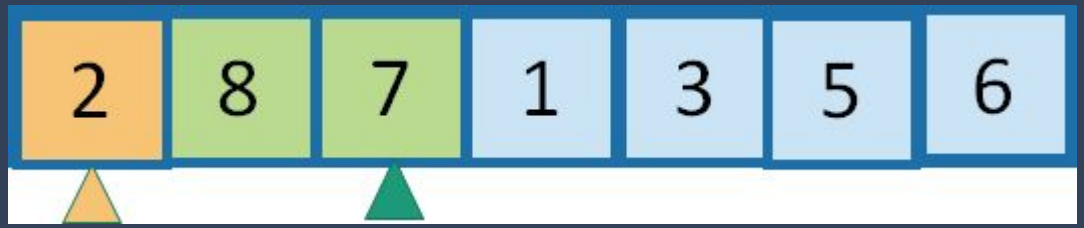
Example 2:

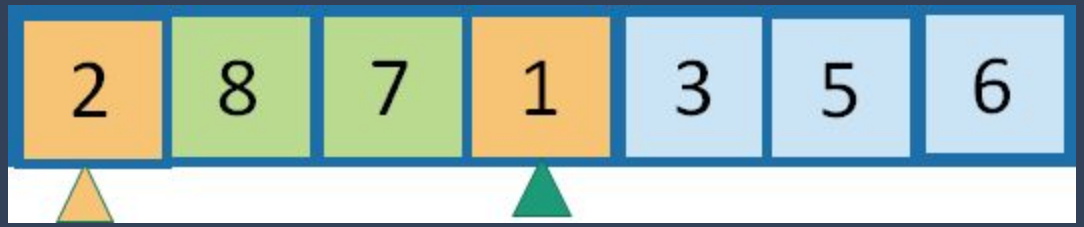
```
Input: [3,2,3,1,2,4,5,5,6] and k = 4
Output: 4
```

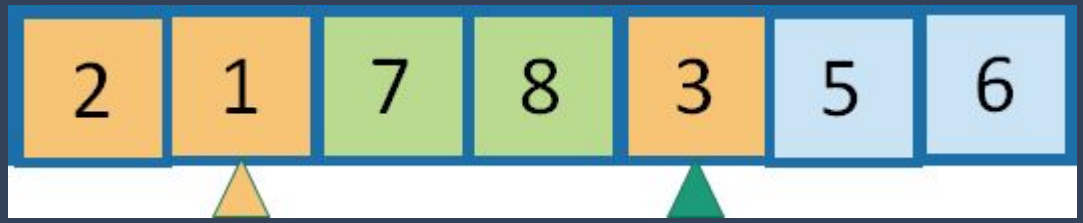
Note:
You may assume k is always valid, 1 ≤ k ≤ array's length.

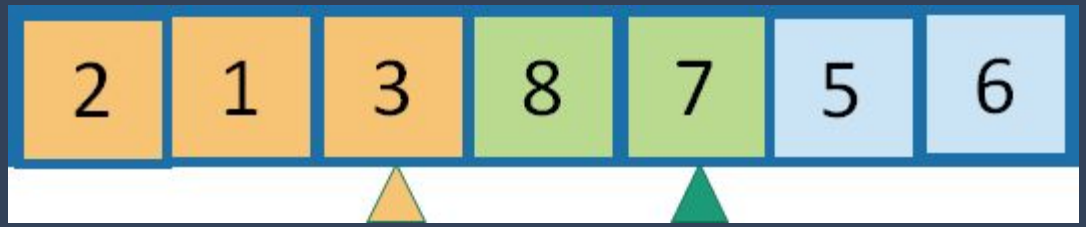


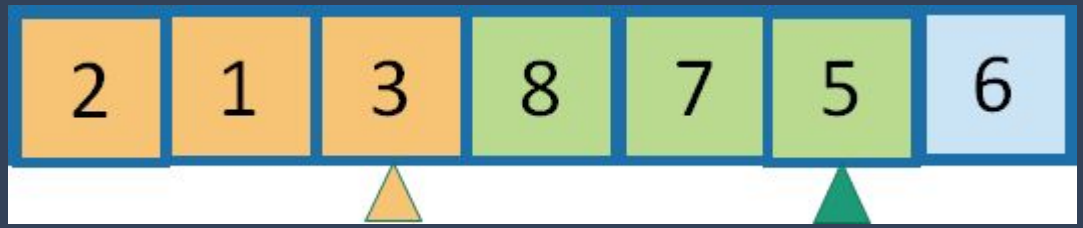
< value > value

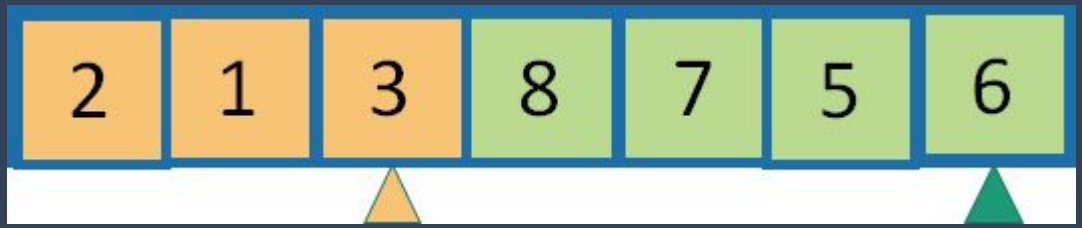


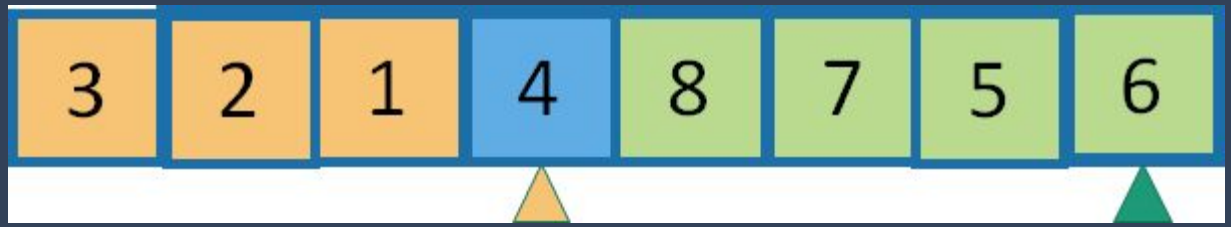












Lomuto’s partitioning: O(n) time, in place

```
import random

def quickselect(A,k):
  if len(A) == 0 or (k-1 < 0) or (k-1 >= len(A)):
    print "Invalid input"
  else:
    return quickselecthelper(A,k,0,len(A)-1)

def quickselecthelper(A,k,start,end):
  if start == end: #One element subarray
    if start != k-1:
      print "There is some error here"
    else:
      return A[k-1]
  #Pick a random element as pivot
  randindex = random.randint(start, end)
  (A[start],A[randindex]) = (A[randindex],A[start])
  pivot = A[start]
  smaller = start
  bigger = start
  for bigger in range(start+1,end+1):
    if A[bigger] <= pivot:
      smaller += 1
      (A[smaller],A[bigger]) = (A[bigger],A[smaller])
  #Now place pivot in the right spot
  (A[start],A[smaller]) = (A[smaller],A[start])
  if k-1 == smaller:
    return A[k-1]
  elif k-1 < smaller:
    return quickselecthelper(A,k,start,smaller-1)
  else:
    return quickselecthelper(A,k,smaller+1,end)
```

Best case?

Worst case?

Average case?

973. K Closest Points to Origin

Medium 547 56 Favorite Share

We have a list of `points` on the plane. Find the `K` closest points to the origin `(0, 0)`.

(Here, the distance between two points on a plane is the Euclidean distance.)

You may return the answer in any order. The answer is guaranteed to be unique (except for the order that it is in.)

Example 1:

```
Input: points = [[1,3],[-2,2]], K = 1
Output: [[-2,2]]
Explanation:
The distance between (1, 3) and the origin is sqrt(10).
The distance between (-2, 2) and the origin is sqrt(8).
Since sqrt(8) < sqrt(10), (-2, 2) is closer to the origin.
We only want the closest K = 1 points from the origin, so the answer is just [[-2,2]].
```

347. Top K Frequent Elements

Medium 2130 140 Favorite Share

Given a non-empty array of integers, return the ***k*** most frequent elements.

Example 1:

```
Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]
```

Example 2:

```
Input: nums = [1], k = 1
Output: [1]
```

Note:

- You may assume *k* is always valid, 1 ≤ *k* ≤ number of unique elements.
- Your algorithm's time complexity **must be** better than O(*n* log *n*), where *n* is the array's size.

Solution

Intuition

If `k = 1` the linear-time solution is quite simple. One could keep the frequency of elements appearance in a hash map and update the maximum element at each step.

When `k > 1` we need a data structure that has a fast access to the elements ordered by their frequencies. The idea here is to use the heap which is also known as priority queue.

Approach 1: Heap

The first step is to build a hash map `element -> its frequency`. In Java we could use data structure `HashMap` but have to fill it manually. Python provides us both a dictionary structure for the hash map and a method `Counter` in the `collections` library to build the hash map we need.
This step takes $\mathcal{O}(N)$ time where `N` is number of elements in the list.

The second step is to build a heap. The time complexity of adding an element in a heap is $\mathcal{O}(\log(k))$ and we do it `N` times that means $\mathcal{O}(N \log(k))$ time complexity for this step.

The last step to build an output list has
$\mathcal{O}(k \log(k))$ time complexity.

In Python there is a method `nlargest` in `heapq` library (check here the source code) which has the same $\mathcal{O}(k \log(k))$ time complexity and combines two last steps in one line.

Complexity Analysis

- Time complexity : $\mathcal{O}(N \log(k))$. The complexity of `Counter` method is $\mathcal{O}(N)$. To build a heap and output list takes $\mathcal{O}(N \log(k))$. Hence the overall complexity of the algorithm is $\mathcal{O}(N + N \log(k)) = \mathcal{O}(N \log(k))$.

- Space complexity : $\mathcal{O}(N)$ to store the hash map.

Side Notes

Following the complexity analysis, the approach is optimal for small `k`. In the case of large `k`, one could revert the procedure by excluding the less frequent elements from the output.

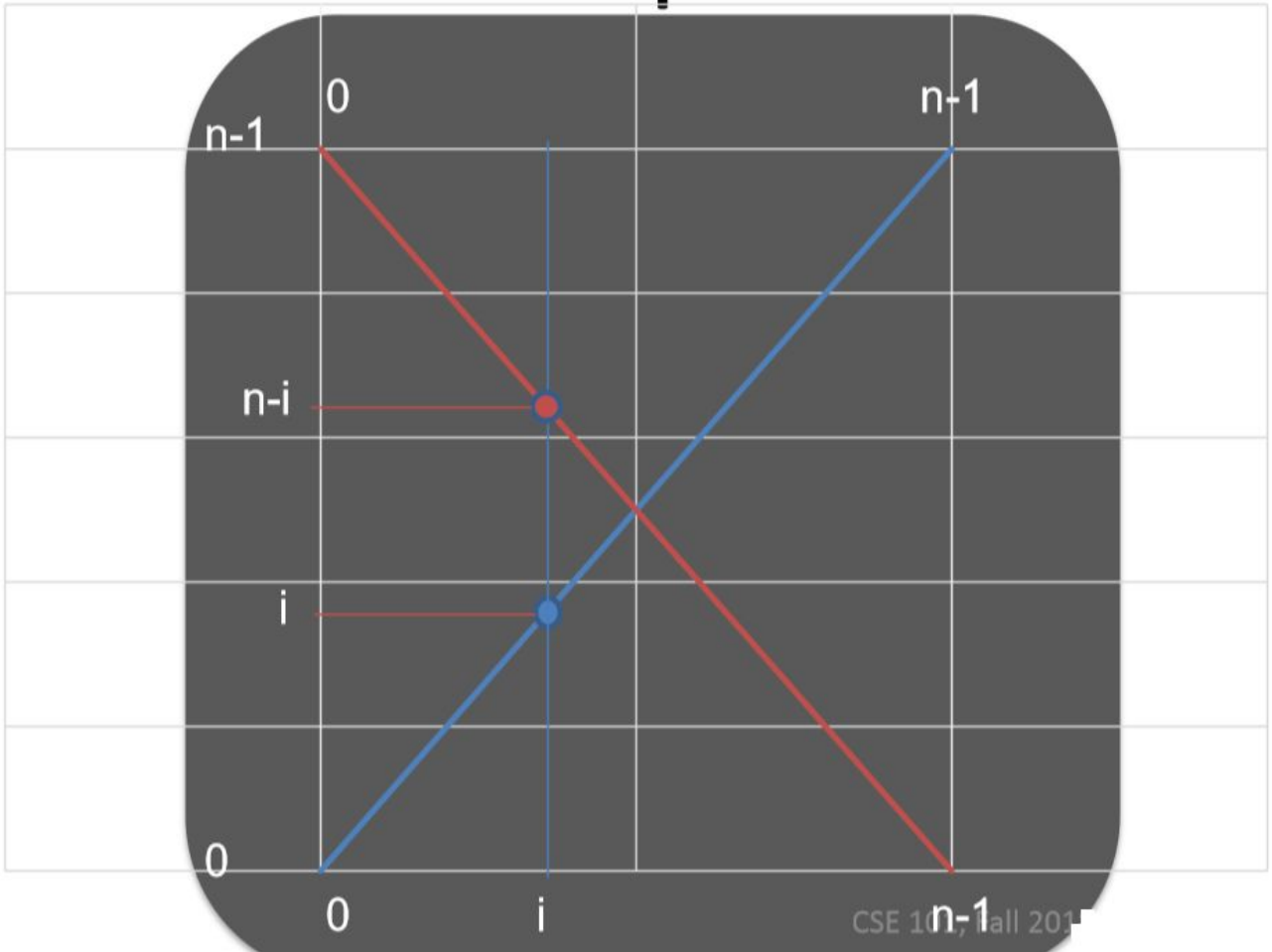
Analysis written by @liaison and @andvary

JPV ★ 515 December 28, 2018 12:41 PM

The problem specifies that:
"Your algorithm's time complexity must be better than O(n log n), where n is the array's size."
LeetCode, why do you have an official solution that fails to meet your own requirement? Please either post a solution that meets the requirement, or change the problem description.
(And, no, O(n log k) is not better than O(n log n), since we have no reason to think k isn't, say, n/2.)

67 Show 12 replies Reply

Expected runtime

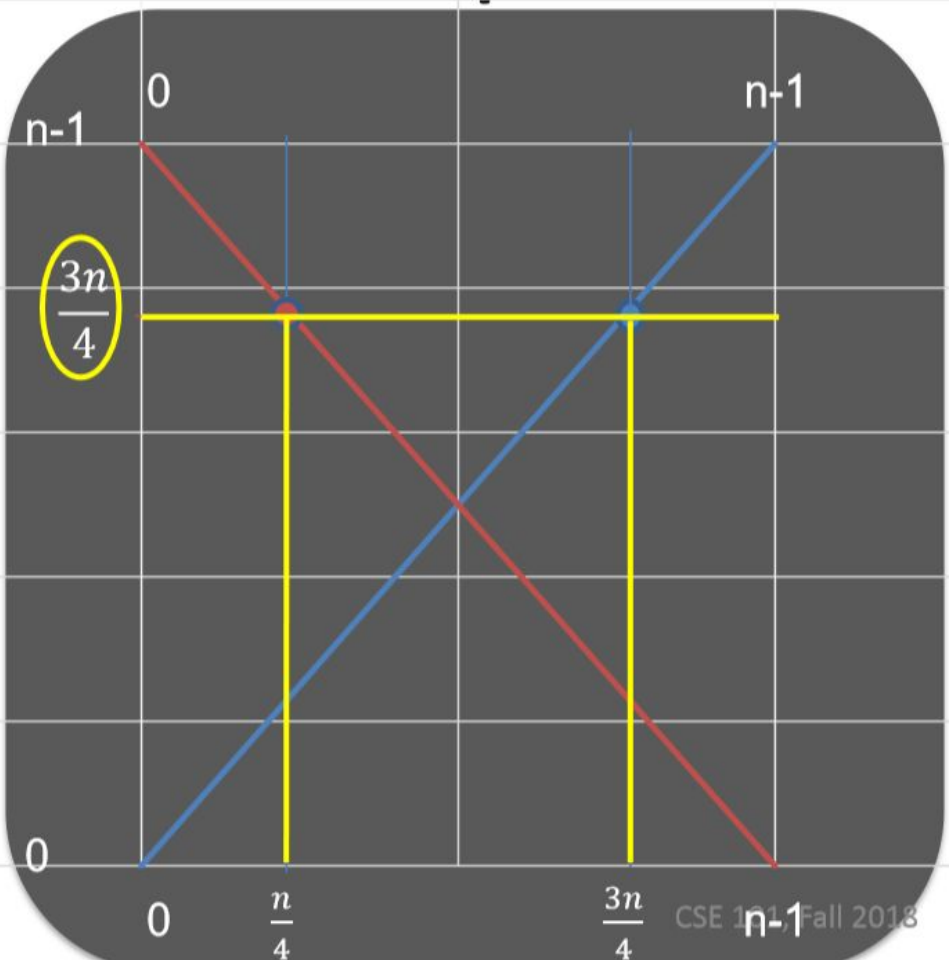


If you randomly select the ith element, then your list will be split into a list of length i and a list of length n-i

So when we recurse on the smaller list, it will take time proportional to

$$\max(i, n - i)$$

Expected runtime



What is the probability of choosing a value from 1 to n in the interval $\left[\frac{n}{4}, \frac{3n}{4}\right]$ if all values are equally likely?

Median

- The purpose of the median is to summarize a set of numbers. The average is also a commonly used value. The median is more typical of the data.
- For example, suppose in a company with 20 employees, the CEO makes $1 million and all the other workers each make $50,000
- Then the average is $97,500 and the median is $50,000, which is much closer to the typical worker's salary

Median, algorithm

- Can you think of an efficient way to find the median?
  - How long would it take?
  - Is there a lower bound on the runtime of all median selection algorithms?

Median, algorithm

- Can you think of an efficient way to find the median?
  - How long would it take?
  - Is there a lower bound on the runtime of all median selection algorithms?
- Sort the list then find the $\lceil n/2 \rceil$th element $O(n \log n)$
- You can never have a faster runtime than $O(n)$ because you at least have to look at every element

Part 4

703. Kth Largest Element in a Stream

Easy 351 164 Favorite Share

Design a class to find the **k**th largest element in a stream. Note that it is the kth largest element in the sorted order, not the kth distinct element.

Your `KthLargest` class will have a constructor which accepts an integer `k` and an integer array `nums`, which contains initial elements from the stream. For each call to the method `KthLargest.add`, return the element representing the kth largest element in the stream.

Example:

```
int k = 3;
int[] arr = [4,5,8,2];
KthLargest kthLargest = new KthLargest(3, arr);
kthLargest.add(3);   // returns 4
kthLargest.add(5);   // returns 5
kthLargest.add(10);  // returns 5
kthLargest.add(9);   // returns 8
kthLargest.add(4);   // returns 8
```

Note:
You may assume that `nums`' length ≥ `k-1` and `k` ≥ 1.

Write a program that takes a sequence of strings in “streaming” fashion, and computes the k longest strings in the sequence.

Time complexity after seeing n strings = ?

Write a program that takes a sequence of strings in “streaming” fashion, and computes the k longest strings in the sequence.

```
minheap = []

for s in sequence:

  minheap.insert(s,priority=len(s))
  if len(minheap) > k:
     minheap.extractmin()

//minheap always maintains the k longest strings at any time
```

```python
import heapq

class KthLargest(object):

    def __init__(self, k, nums):
        """
        :type k: int
        :type nums: List[int]
        """
        self.array = []
        heapq.heapify(self.array)
        self.maxsize = k
        #Maintain only k elements
        for element in nums:
            if len(self.array) != self.maxsize:
                #Insert the next element into the heap
                heapq.heappush(self.array,element)
                #Every time you add an element to the heap, increase size by 1
            elif len(self.array) == self.maxsize:
                heapq.heappushpop(self.array,element)


    def add(self, val):
        """
        :type val: int
        :rtype: int
        """
        if len(self.array) != self.maxsize:
            #Insert the next element into the heap
            heapq.heappush(self.array,val)
            #Every time you add an element to the heap, increase size by 1
        elif len(self.array) == self.maxsize:
            heapq.heappushpop(self.array,val)

        return self.array[0]

```

A heap is a good choice when you are maintaining the k largest or k smallest elements in any collection.

But what about the median finding problem?

Compute the Median of Online data

Compute the running median of a sequence of numbers. The sequence is presented to you in a streaming fashion - you cannot back up to read an earlier value, and you need to output the median after reading in each new element.

Example input: 1,0,3,5,2,0,1

Output: 1, 0.5, 1, 2, 2, 1.5, 1

295. Find Median from Data Stream

Hard 1306 25 Favorite Share

Median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value. So the median is the mean of the two middle value.

For example,
`[2,3,4]`, the median is `3`

`[2,3]`, the median is `(2 + 3) / 2 = 2.5`

Design a data structure that supports the following two operations:

- void addNum(int num) - Add a integer number from the data stream to the data structure.
- double findMedian() - Return the median of all elements so far.

Example:

```
addNum(1)
addNum(2)
findMedian() -> 1.5
addNum(3)
findMedian() -> 2
```

```python
import heapq

class MedianFinder(object):

    def __init__(self):
        """
        initialize your data structure here.
        """
        self.maxheap = []
        self.minheap = []
        heapq.heapify(self.maxheap)
        heapq.heapify(self.minheap)
        self.median = 0.0

    def addNum(self, num):
        """
        :type num: int
        :rtype: None
        """
        if num <= self.median:
            #Insert it into the left heap (maxheap)
            heapq.heappush(self.maxheap,-num)
            if len(self.maxheap) - len(self.minheap) == 2:
                #Rebalancing needed. Remove one element from maxheap and put into the minheap
                heapq.heappush(self.minheap,-heapq.heappop(self.maxheap))
        else:
            #Insert it into the right heap (minheap)
            heapq.heappush(self.minheap,num)
            if len(self.minheap) - len(self.maxheap) == 2:
                #Rebalancing needed. Transfer one element from minheap to maxheap
                heapq.heappush(self.maxheap,-heapq.heappop(self.minheap))

        self.findMedian()

```

```
36      def findMedian(self):
37          """
38          :rtype: float
39          """
40          if len(self.maxheap) > len(self.minheap):
41              self.median = -self.maxheap[0]
42              return self.median
43          elif len(self.maxheap) < len(self.minheap):
44              self.median = self.minheap[0]
45              return self.median
46          else: #Both are of the same height
47              self.median = (-self.maxheap[0] + self.minheap[0])/2.0
48              return self.median
49
```

Thank You!