GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

K'th Smallest/Largest Element in Unsorted Array | Set 3 (Worst Case Linear Time)

We recommend to read following posts as a prerequisite of this post.

K'th Smallest/Largest Element in Unsorted Array | Set 1
K'th Smallest/Largest Element in Unsorted Array | Set 2 (Expected Linear Time)

Given an array and a number k where k is smaller than size of array, we need to find the k'th smallest element in the given array. It is given that ll array elements are distinct.

Examples:

```
Input: arr[] = {7, 10, 4, 3, 20, 15}
```

```
       k = 3
Output: 7

Input: arr[] = {7, 10, 4, 3, 20, 15}
       k = 4
Output: 10
```

In previous post, we discussed an expected linear time algorithm. In this post, a worst case linear time method is discussed. *The idea in this new method is similar to quickSelect(), we get worst case linear time by selecting a pivot that divides array in a balanced way (there are not very few elements on one side and many on other side)*. After the array is divided in a balanced way, we apply the same steps as used in quickSelect() to decide whether to go left or right of pivot.

Following is complete algorithm.

```
kthSmallest(arr[0..n-1], k)
1) Divide arr[] into ⌈n/5rceil; groups where size of each group is 5
   except possibly the last group which may have less  than 5 elements.

2) Sort the above created ⌈n/5⌉ groups and find median
   of all groups. Create an auxiliary array 'median[]' and store medians
   of all ⌈n/5⌉ groups in this median array.

// Recursively call this method to find median of median[0..⌈n/5⌉-1]
3) medOfMed = kthSmallest(median[0..⌈n/5⌉-1], ⌈n/10⌉)

4) Partition arr[] around medOfMed and obtain its position.
     pos = partition(arr, n, medOfMed)

5) If pos == k return medOfMed
6) If pos < k return kthSmallest(arr[l..pos-1], k)
7) If poa > k return kthSmallest(arr[pos+1..r], k-pos+l-1)
```

In above algorithm, last 3 steps are same as algorithm in previous post. The first four steps are used to obtain a good point for partitioning the array (to make sure that there are not too many elements either side of pivot).

Following is C++ implementation of above algorithm.

```
// C++ implementation of worst case linear time algorithm
// to find k'th smallest element
#include<iostream>
#include<algorithm>
#include<climits>
using namespace std;

int partition(int arr[], int l, int r, int k);

// A simple function to find median of arr[].  This is called
// only for an array of size 5 in this program.
int findMedian(int arr[], int n)
{
    sort(arr, arr+n);  // Sort the array
    return arr[n/2];   // Return middle element
```

```
}

// Returns k'th smallest element in arr[l..r] in worst case
// linear time. ASSUMPTION: ALL ELEMENTS IN ARR[] ARE DISTINCT
int kthSmallest(int arr[], int l, int r, int k)
{
    // If k is smaller than number of elements in array
    if (k > 0 && k <= r - l + 1)
    {
        int n = r-l+1; // Number of elements in arr[l..r]

        // Divide arr[] in groups of size 5, calculate median
        // of every group and store it in median[] array.
        int i, median[(n+4)/5]; // There will be floor((n+4)/5) groups;
        for (i=0; i<n/5; i++)
            median[i] = findMedian(arr+l+i*5, 5);
        if (i*5 < n) //For last group with less than 5 elements
        {
            median[i] = findMedian(arr+l+i*5, n%5);
            i++;
        }

        // Find median of all medians using recursive call.
        // If median[] has only one element, then no need
        // of recursive call
        int medOfMed = (i == 1)? median[i-1]:
                                 kthSmallest(median, 0, i-1, i/2);

        // Partition the array around a random element and
        // get position of pivot element in sorted array
        int pos = partition(arr, l, r, medOfMed);

        // If position is same as k
        if (pos-l == k-1)
            return arr[pos];
        if (pos-l > k-1)  // If position is more, recur for left
            return kthSmallest(arr, l, pos-1, k);

        // Else recur for right subarray
        return kthSmallest(arr, pos+1, r, k-pos+l-1);
    }

    // If k is more than number of elements in array
    return INT_MAX;
}

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

```
// It searches for x in arr[l..r], and partitions the array
// around x.
int partition(int arr[], int l, int r, int x)
{
    // Search for x in arr[l..r] and move it to end
    int i;
    for (i=l; i<r; i++)
        if (arr[i] == x)
           break;
    swap(&arr[i], &arr[r]);

    // Standard partition algorithm
    i = l;
    for (int j = l; j <= r - 1; j++)
    {
        if (arr[j] <= x)
        {
            swap(&arr[i], &arr[j]);
            i++;
        }
    }
    swap(&arr[i], &arr[r]);
    return i;
}

// Driver program to test above methods
int main()
{
    int arr[] = {12, 3, 5, 7, 4, 19, 26};
    int n = sizeof(arr)/sizeof(arr[0]), k = 3;
    cout << "K'th smallest element is "
         << kthSmallest(arr, 0, n-1, k);
    return 0;
}
```

Output:

```
K'th smallest element is 5
```

Time Complexity:
The worst case time complexity of the above algorithm is O(n). Let us analyze all steps.

The steps 1) and 2) take O(n) time as finding median of an array of size 5 takes O(1) time and there are n/5 arrays of size 5.
The step 3) takes T(n/5) time. The step 4 is standard partition and takes O(n) time.
The interesting steps are 6) and 7). At most, one of them is executed. These are recursive steps. What is the worst case size of these recursive calls. The answer is maximum number of elements greater than medOfMed (obtained in step 3) or maximum number of elements smaller than medOfMed.
How many elements are greater than medOfMed and how many are smaller?
At least half of the medians found in step 2 are greater than or equal to medOfMed. Thus, at least half of the n/5 groups contribute 3 elements that are greater than medOfMed, except for the one group that has fewer than 5 elements. Therefore, the number of elements greater than medOfMed is at least.

$$3\left(\left\lceil\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil\right\rceil - 2\right) \geq \frac{3n}{10} - 6$$

Similarly, the number of elements that are less than medOfMed is at least 3n/10 – 6. In the worst case, the function recurs for at most n – (3n/10 – 6) which is 7n/10 + 6 elements.

Note that 7n/10 + 6 < n for n > 20 and that any input of 80 or fewer elements requires O(1) time. We can therefore obtain the recurrence

$$T(n) \leq \begin{cases} \Theta(1) & \text{if } n \leq 80\,, \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n) & \text{if } n > 80\,. \end{cases}$$

We show that the running time is linear by substitution. Assume that T(n) cn for some constant c and all n > 80. Substituting this inductive hypothesis into the right-hand side of the recurrence yields

```
T(n)  <= cn/5 + c(7n/10 + 6) + O(n)
     <= cn/5 + c + 7cn/10 + 6c + O(n)
    <= 9cn/10 + 7c + O(n)
    <= cn,
```

since we can pick c large enough so that c(n/10 - 7) is larger than the function described by the O(n) term for all n > 80. The worst-case running time of is therefore linear (Source: http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap10.htm).

Note that the above algorithm is linear in worst case, but the constants are very high for this algorithm. Therefore, this algorithm doesn't work well in practical situations, randomized quickSelect works much better and preferred.

Sources:
MIT Video Lecture on Order Statistics, Median
Introduction to Algorithms by Clifford Stein, Thomas H. Cormen, Charles E. Leiserson, Ronald L.
http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap10.htm

This article is contributed by **Shivam**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Topics:

- Find Union and Intersection of two unsorted arrays
- Pythagorean Triplet in an array
- Maximum profit by buying and selling a share at most twice
- Design a data structure that supports insert, delete, search and getRandom in constant time
- Print missing elements that lie in range 0 – 99
- Iterative Merge Sort
- Group multiple occurrence of array elements ordered by first occurrence
- Given a sorted and rotated array, find if there is a pair with a given sum

Writing code in comment? Please use ideone.com and share the link here.

23 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

neer1304 • 15 days ago

C++ implementation for kth largest element in an unsorted array

http://ideone.com/KiCiOR

• Reply • Share ›

devakar verma • a month ago

I have a doubt, int calculation of medOfMed

int medOfMed = (i == 1)? median[i-1]:kthSmallest(median, 0, i-1, i/2);

but if i!=1 then we can call getMedian(median,i)

So, why kthsmallest function was called? Anyone please clarify it.

• Reply • Share ›

Guest • a month ago

I have a doubt, int calculation of medOfMed
int medOfMed = (i == 1)? median[i-1]:kthSmallest(median, 0, i-1, i/2);

but if i!=1 then we can call getMedian(median,i)
So, why kthsmallest function was called? Anyone please clarify it.

• Reply • Share ›

Ashish Ani • 3 months ago

Try this code .... Working in O(n) time, and O(n) space

//

//FIND K'th Smallest Number from an unsorted array

//Considering all positive numbers & range is small

class KSmallest

{

public static void main(String[] args)

```
{

int arr[]={7, 10, 4, 3, 20, 15, 9, 3, 6};

System.out.println("5th smallest : "+ small(arr,5));

}
```

see more

• Reply • Share ›

Namnori • 3 months ago

Ok, ignore my previous coment, I missed your point of calculate the median of a 5 in a O(1), so the total complexity of median calculation be a O(1*n/5).

• Reply • Share ›

Namnori • 3 months ago

The step 2 realizes an ordering to get the median, so this step complexity would be at least O((n/5)*log(n/5)), and not O(n).

• Reply • Share ›

RK- An Unproven Theorem • 3 months ago

Is it possible to sort an array, and get the kth element from array? I believe that we can sort array in log n time and, then we can find the element in n time. So, the complexity of that approach would be n log n.

Let me know, whether it can work or not.

• Reply • Share ›

guestorita ↗ RK- An Unproven Theorem • 3 months ago

Which algoritm sorts an array in log n time?

3 • Reply • Share ›

lebron ↗ guestorita • 2 months ago

merge sort

• Reply • Share ›

devakar verma ↗ lebron • a month ago

merge sort take o(n log n) time

• Reply • Share ›

Priya • 4 months ago

How is computation of median in step 1 & 2 O(n) when it involves sorting?

• Reply • Share ›

lebron ➚ Priya • 2 months ago

since the array size is 5, it takes constant time. If you don't know the size of the array, sorting would cause the nlogn. It takes O(n) time because he does O(1) median calculation n/5 times which causes O(n/5) and O(n)

• Reply • Share ›

randomguy202 • 4 months ago

How did you choose to divide the array into groups of 5? Why not 3 or 7; why 5?

• Reply • Share ›

Sreenivas Putta • 4 months ago

my code in Java: http://ideone.com/3oJGZj

• Reply • Share ›

Mitesh • 4 months ago

For the last method discussed, if we put k=4 in driver program,
we should get 7 as output.
But we are getting 26.

• Reply • Share ›

Varun Anand • 4 months ago

You can also use randomized selection algorithm, which in general is of the order of O(n) unless the random pivot selector chooses bad pivots in which case the complexity increases to n^2.

Randomized Selection algorithm can also be used in cases where you need to find top 'K' elements in an unsorted array. Below implementation samples both the cases

Java Implementation:

```
import java.util.Random;

/* Randomized select - Select ith element */
public class Select<t extends="" comparable<t="">> {

int partition(T arr[], int left, int right, int pivotindex)
{
T pivotvalue = arr[pivotindex];
if(pivotindex != right)
{
```

see more

• Reply • Share ›

Harish • 4 months ago

Harish • 4 months ago

When you pass the input array to findMedian(arr+l+i*5, 5) you are sorting a portion of input array which I think that should not happen. Partition should be the only method that should move the elements in the array as per the original algorithm.

• Reply • Share ›

nishantfirst • 4 months ago

Correct this line(typo):

case, the function recurs for at most n – (3n/6-6) which is 7n/10 + 6 elements to

case, the function recurs for at most n – (3n/10-6) which is 7n/10 + 6 elements

which is just confusing

• Reply • Share ›

GeeksforGeeks Mod ➚ nishantfirst • 4 months ago

Thanks for pointing this out. We have updated the expression.

• Reply • Share ›

bishwanath mandal • 5 months ago

why constant time for n<80?

• Reply • Share ›

Guest • 5 months ago

I think in the complete algorithms :
1. suppose k=n/5. then each of the five groups is having k size except the last . The size of the group will not be 5 only.
suppose n=100 then k=20 and number of groups is 5.

• Reply • Share ›

Kartik ➚ Guest • 5 months ago

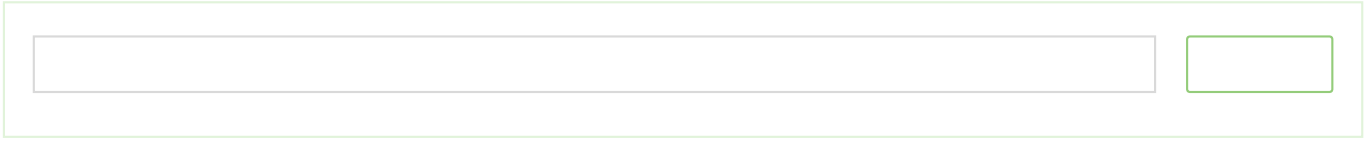
Thanks for pointing this out. We have updated the step 1) of above algorithm.

• Reply • Share ›

Gomathi Ganesan • 5 months ago

Form a minheap/maxheap with the elements of the array and then call deletemin()/deletemax() 'k' times.

• Reply • Share ›



- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding "extern" keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 1 hour ago

- Piyush

What is the purpose of else if (recStack[*i])...

Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 2 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 2 hours ago