k largest(or smallest) elements in an array | added Min Heap method

Question: Write an efficient program for printing k largest elements in an array. Elements in array can be in any order.

For example, if given array is [1, 23, 12, 9, 30, 2, 50] and you are asked for the largest 3 elements i.e., k = 3 then your program should print 50, 30 and 23.

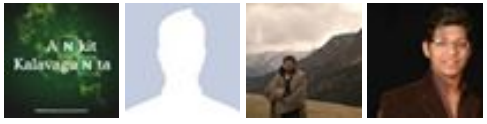
Method 1 (Use Bubble k times)
Thanks to Shailendra for suggesting this approach.
1) Modify Bubble Sort to run the outer loop at most k times.
2) Print the last k elements of the array obtained in step 1.

Time Complexity: O(nk)

Like Bubble sort, other sorting algorithms like Selection Sort can also be modified to get the k largest elements.

Method 2 (Use temporary array)
K largest elements from arr[0..n-1]

1) Store the first k elements in a temporary array temp[0..k-1].
2) Find the smallest element in temp[], let the smallest element be *min*.
3) For each element *x* in arr[k] to arr[n-1]
If *x* is greater than the min then remove *min* from temp[] and insert *x*.
4) Print final k elements of *temp[]*



Time Complexity: O((n-k)*k). If we want the output sorted then O((n-k)*k + klogk)

Thanks to nesamani1822 for suggesting this method.

Method 3(Use Sorting)
1) Sort the elements in descending order in O(nLogn)
2) Print the first k numbers of the sorted array O(k).

Time complexity: O(nlogn)

Method 4 (Use Max Heap)
1) Build a Max Heap tree in O(n)
2) Use Extract Max k times to get k maximum elements from the Max Heap O(klogn)

Time complexity: O(n + klogn)

Method 5(Use Oder Statistics)
1) Use order statistic algorithm to find the kth largest element. Please see the topic selection in worst-case linear time O(n)
2) Use QuickSort Partition algorithm to partition around the kth largest number O(n).
3) Sort the k-1 elements (elements greater than the kth largest element) O(kLogk). This step is needed only if sorted output is required.

Time complexity: O(n) if we don't need the sorted output, otherwise O(n+kLogk)

Thanks to Shilpi for suggesting the first two approaches.

Method 6 (Use Min Heap)
This method is mainly an optimization of method 1. Instead of using temp[] array, use Min Heap.

Thanks to geek4u for suggesting this method.

1) Build a Min Heap MH of the first k elements (arr[0] to arr[k-1]) of the given array. O(k)

2) For each element, after the kth element (arr[k] to arr[n-1]), compare it with root of MH.
……a) If the element is greater than the root then make it root and call heapify for MH
……b) Else ignore it.
// The step 2 is O((n-k)*logk)

3) Finally, MH has k largest elements and root of the MH is the kth largest element

Geometric Algorithms

Popular Posts

Time Complexity: O(k + (n-k)Logk) without sorted output. If sorted output is needed then O(k + (n-k)Logk + kLogk)

All of the above methods can also be used to find the kth largest (or smallest) element.

Please write comments if you find any of the above explanations/algorithms incorrect, or find better ways to solve the same problem.

References:
http://en.wikipedia.org/wiki/Selection_algorithm

Asked by geek4u

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort

- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

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

70 Comments GeeksforGeeks

Sort by Newest

Guest • a month ago

/*

k largest(or smallest) elements in an array | added Min Heap method

Question: Write an efficient program for printing k largest elements in an array order.

For example, if given array is [1, 23, 12, 9, 30, 2, 50] and you are asked for the program should print 50, 30 and 23.

Method 1 (Use selection k times)

1) Modify selection Sort to run the outer loop at most k times.
2) Print the firts k elements of the array obtained in step 1.

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 24 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 28 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 53 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 54 minutes ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

Time Complexity: O(nk)

*/

#include<stdio.h>
void main()

see more

Guest • a month ago

/*
/*
k largest(or smallest) elements in an array | added Min Heap method

Question: Write an efficient program for printing k largest elements in an array order.

For example, if given array is [1, 23, 12, 9, 30, 2, 50] and you are asked for the program should print 50, 30 and 23.

Method 1 (Use selection k times)

1) Modify selection Sort to run the outer loop at most k times.
2) Print the firts k elements of the array obtained in step 1.

Time Complexity: O(nk)

*/

#include<stdio.h>

see more

1

foo • 3 months ago

> Time Complexity: O(k + (n-k)Logk) without sorted output. If sorted output is r

No. The total complexity is O(k + n log k).

meh • 3 months ago

I don't quite understand the min heap approach, if n = k, then the root is the sm one. Shouldn't it be a max-heap?

Say, you insert the first k elements into the max-heap, for all consecutive elem root then ignore them, otherwise, replace current root with each element and h

KJ ➚ meh • 2 months ago

if k = n, then the smallest number IS the kth largest one.
[1,4,2,3], 4 is the largest and 1 is the 4th largest.
If you are finding kth smallest number, then a max heap should be use

1

Thomas • 3 months ago

Solution below in C++, linear time :

```
#include <cstdlib>
#include <iostream>
#include <vector>

using namespace std;

vector<int> smallest(vector<int> L, int k) {
int sup;
vector<int> sublist;
```

```
if (L.size() <= k) {
return L;
}
for (int i = 0; i < L.size(); ++i) {
if (i < k) {
if (i == 0) {
sup = L[i];
} else if (L[i] > sup) {
```

see more

4

mizhao • 3 months ago

I have an approach cost only O(n) times, and it can print the results in sorting
It is based on Method 4.
1) Build a Max Heap tree in O(n), call it h1.
2) important observation: the second largest element can only appear as a chi
largest element can only appear under the largest one or the second largest o
3) We put the largest element of the h1 into another empty heap h2.
4) We output the maximum element of heap h2, and put his children in h1 to h
5) Repeat step 3,4,5 until we output k elements

Time complexity:
We get 2*k elements from heap h1, which cost O(k). Each operation is O(1) b
element in h1.
We do 2*k insertion in heap h2. The total cost is O(k).
We do k poll operation to heap h2 which cost O(k).
The total time complexity is: O(n+k) in case k is less than n is O(n).

What do you think?

Alok Kumar ➦ mizhao • 10 days ago

I think your explanation requires more clarity. Statements like "put his c

mizhao ➚ Alok Kumar • 7 days ago

Suppose the heap is stored in an array starting from index 1. If
children will be located at index 2*k and 2*k+1. Thus, accessing
element costs O(1). And you add them to h2 using the standard

Suraj • 5 months ago

Method 4 has wrong time complexity
1) Build a Max Heap tree in O(n)? creating Max heap is O(nlogn)
2) Use Extract Max k times to get k maximum elements from the Max Heap O
O(1) so its O(k)

Time complexity should be : O(k + nlogn) ~ O(nlogn)

Coder011 ➚ Suraj • 4 months ago

@Suraj: plz check your facts before posting, http://www.geeksforgeeks
max is O(1) , but heapify procedure would still have to be called , in orc
thus you get O(logn) for one call to extract max (+heapify) and this don
complexity is O(n + k*logn).

1

gourav pathak ➚ Coder011 • 3 months ago

Doesn't building a max heap take O(nlogn) time for n keys??...i

Babrael ➚ gourav pathak • 3 months ago

Build heap essentially takes O(n) time.A good explanati
algorithms by

Cormen

1 ^ | v •

gourav pathak ➦ Babrael • 3 months ago

Got it..Thanks

^ | v •

Aditya Tirodkar • 6 months ago

Wait... why can't you just use Quick Select and just extract the values till k? Th

4 ^ | v •

Code_Addict ➦ Aditya Tirodkar • 4 months ago

What suraj told is perfectly correct. Worst Case complexity of quicksel

http://en.wikipedia.org/wiki/Q...

^ | v •

Suraj ➦ Aditya Tirodkar • 5 months ago

O(n) is average case using Quick Select.. Worst case is O(n^2).. I agr decently enough large compared to the size of input.. if k is minuscule

1 ^ | v •

Raj • 6 months ago

How to solve this problem

N machines and each machine contains M integers sorted in increasing order example Machine A : 5, 60, 70 Machine B : 10, 20, 30

K = 2

Output should be 5, 10

Best time and space complexity?

Vinod ➚ Raj • 6 months ago

@Raj...Refer to another post : http://www.geeksforgeeks.org/t...

Alien • 8 months ago

Could you please post code for this.

Satyarth • 9 months ago

There is a typo mistake in method 6. First line should be "This method is main of "This method is mainly an optimization of method 1".

Thanks!

Akhil • 10 months ago

Simple Code for MinHeap Method

```
#include<stdio.h>

void swap(int *a, int *b)
{
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
}

void minHeapify(int a[], int size, int i)
{
    int l = 2*i;
    int r = 2*i+1;
```

```
    int smallest = i;
    if(l<size && a[l]<a[smallest])
        smallest = l;
```

see more

Bharath G M ↗ Akhil • 8 months ago

Very good one. I think there is little redundancy. In MinHeapify..
if(smallest!=i)
{
swap(&a[i],&a[smallest]);
minHeapify(a,size,smallest);
//I dont think we need to call minHeapify again. "swap" is fine
}

1

Nizamuddin Saifi ↗ Bharath G M • 5 months ago

No its fine . If there is no recursive call in minHeapify , In kthLarg
call minHeapify there would be problem.

Ronny • 11 months ago

Can anyone explain Method 5.
I am having problem grabbing the concept behind that method.
Thanks.

Asap ↗ Ronny • 10 months ago

First u have to get Kth rank element in array.
Then u can partition array based on this Kth rank element.

Check this video
http://www.youtube.com/watch?v...

Ronny Asap • 10 months ago
@Asap
Thanks a lot. was of great help.
here is the link LINK TO FIND RANK OF AN ELEMENT IN AN A

Ronny • 11 months ago
@geeksforgeeks
I guess there is a typo in description of Method 6
"This method is mainly an optimization of method 1. Instead of using temp[] ar

It should be "method 2".
Method 1 stated above uses sorting techniques k times.
Method 2 uses a temporary array to store first k elements.
1

joker • 11 months ago
Method 6 : O(k +(n-k)logk)

```
{
    int n,k,x,i;
    vector<int> a,ans;
    priority_queue<int,vector<int>,greater<int> > p;     // min_heap

    scanf("%d %d",&n,&k);
    for(i=0;i<n;i++)
          scanf("%d",&x), a.push_back(x);
```

```
    for(i=0;i<k;i++)
          p.push(a[i]);

    for(i=k;i<n;i++)
          if(a[i]>(x=p.top()))
             p.pop() , p.push(a[i]);
```

see more

1

Ujjwal • a year ago

We could alsouse this algorithm :

-find out the element with rank 'k', using median finding algo in O(n)
-once we have found out the element, just traverse the array again to print all e
print elements greater than 'kth' largest..

correct me if i am wrong..!!

Manolis Lourakis • a year ago

```
/* Find the kth smallest element of an array using Method 5 (Order Si
 *
 * This is a C/C++ version of the Java implementation above. The code
 * made more compact by inlining partition() and eliminating recursion
 *
 * Written by Manolis Lourakis, Nov. 2012
 */

/* Select the kth smallest element among elements a[l]..a[r] of array
 * For an array of n elements invoke as quickSelect(a, 0, n-1, k);
```

```
 */
int quickSelect(int a[], int l, int r, int k)
{
register int i, j;
int s;
int pivot, temp;

  //if(k<1 || k>=r-l+1) return -1; // k out of range
```

see more

Manolis Lourakis • a year ago

```
 /* Find the kth smallest element of an array using Method 5 (Order Si
  *
  * This is a C/C++ version of the Java implementation above. The code
  * made more compact by inlining partition() and eliminating recursion
  *
  * Written by Manolis Lourakis, Nov. 2012
  */

/* Select the kth smallest element among elements a[l]..a[r] of array
  * For an array of n elements invoke as quickSelect(a, 0, n-1, k);
  */
int quickSelect(int a[], int l, int r, int k)
{
register int i, j;
int s;
int pivot, temp;

  //if(k<1 || k>=r-l+1) return -1; // k out of range
```

see more

Kasim • 3 years ago

```
class ThreeBiggestDemo{
public static void main (String args[]){
int ar [] = new int[10];
int big1, big2, big3, temp;

ar[0] = 29;
ar[1] = 2;
ar[2] = 43;
ar[3] = 8;
ar[4] = 72;
ar[5] = 17;
ar[6] = 92;
ar[7] = 113;
ar[8] = 11;
ar[9] = 0;

big1 = ar[0];
big2 = ar[1];
```

see more

Nithish • 3 years ago

How about randomized QuickSort with a small tweak?

Choose the pivot for the QuickSort as any number from the given array and sa
QuickSort, it was found the pivot element belonged to index 'X' of the say N ar

If we had to find the Kth smallest element, then if X was greater an K - 1, apply

than X.

If we had to find the Kth largest element, then the element we have to find the
- 1 and apply the same logic.

2 ^ | v •

geek • 3 years ago

Building a heap of n elements require nLogn operations . How can you build a

^ | v •

shanky • 3 years ago

in method 4 how can we build a max heap in O(n).it requires O(nlogn)

^ | v •

Sandeep ➦ shanky • 3 years ago

@shanky: Build Heap takes O(n) time. See this G-Fact

^ | v •

Imran • 3 years ago

```
/* Based on Method 5 of Order Statistics. It finds kth Smallest elemer
* Java Code using Partition as the first element.
* Comments and suggestions are appreciated.
* We can enhance this code by calculating Median of Medians to find p:
* For reference consult this explanation.
* http://www.comp.dit.ie/rlawlor/Prob_Solv/Imperative_Algs/Quick%20So
*/
public static int quickSelect( int a[], int l, int r, int x)
        {
                // if x is outOfRange return -1
                if(x < 1 || x > r) return -1;
```

```
            if(l==r) return a[l];
            if(l < r)
            {
          // divide and conquer
           int j = partition( a, l, r);
```

see more

lalu ➚ lmran • 6 months ago

Can you explain , how its O(n) ?

WgpShashank • 3 years ago

here you can also get Min-Max Heap

http://forestofcode.blogspot.c...

laxman • 3 years ago

@sandeep...maderator,, venki

hi geeks please provide the working code fro the 6th method..this is highly in d asap...everyone looking forward.

Thanks
Rahul

Sandeep ➚ laxman • 3 years ago

@Rahul: Following is the code for method 6.

```
#include<iostream>
#include<stdio.h>
using namespace std;

void swap(int *x, int *y) {
  int temp = *x;
  *x = *y;
  *y = temp;
}

class Heap
{
  int *arr; // pointer to array of elements in heap
  int heap_size;
public:
```

see more

wgpshashank ➦ Sandeep • 3 years ago

@sandeep u haven't done any boundary checking its not a goo
in java u will get exception..at fisrt step itsel...hope u will rerpos
with some test case as well

Algoseekar • 3 years ago

@geeksforgeeks,venki,,all geeks everyone know that method 6 using heap is
exact implementation of that..

Thank
Algoseekar

reg_frenzy • 3 years ago

We could use Winner trees approach. At each time, we compare two adjacen
comparisons by 2 at each iteration.

Eg:
12 5 8 1 78 90

Outcome of First iteration:
(Compare adjacent elements, winner is the bigger of the 2 elements)

If it is odd, retain the last element.

12 8 90

Proceeding similarly,

Outcome of Second iteration:

12 90

Outcome of Third iteration:

see more

bunty • 4 years ago

Another algorithm which will take O(n(1+k))
- Scan through the original array and create a temp array, "temp", of k elemen
ascending order. This temp array will have our k largest elements.
- Arr[n]
- Make and array temp[k] with k (here 3) elements:
temp[i] = 0 for i = 0 to 1

```
temp[] = {0,0, Arr[0]);
// Comparing each element of Arr with temp[k-1] and place the //larger one in t
ascending order.
for (count=0;count<n;count++)
{// n comparison
if (temp[k-1] < Arr[count])
{
// Assigning Arr[count] to temp[k-1] and keeping array in
// order and over writing the lowest element in temp.
count2=0;
```

see more

RK • 4 years ago

@ Method 5

I am not sure why we need to sort the elements(step 3)once we have already question says the elements larger then kth largest element can be in any orde

In my opinion, the running time be O(n).

Please correct me if I am wrong.

GeeksforGeeks ➚ RK • 4 years ago

@RK: Thanks for sharing your thoughts, we have added a note for this

Mahesh • 4 years ago

Link in method 5 broken.

GeeksforGeeks ➚ Mahesh • 4 years ago

@Mahesh: Thanks for pointing this out. We ave fixed it.

Load more comments

Subscribe Add Disqus to your site