K Largest Elements

Courses

**Tutorials** 

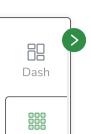
Johs

Practice

Contests







Αll

 $\Box$ Articles

Videos

</>

Problems

(?)

Quiz

« Prev

Next >>

# Method 1(Use Sorting)

- 1) Sort the elements in descending order in O(n\*log(n))
- 2) Print the first k numbers of the sorted array O(k).



Following is the implementation of the above.



```
C++
        Java
```

```
9
      // Java code for k largest elements in an array
      import java.util.Arrays;
-,0(-
      import java.util.Collections;
      import java.util.ArrayList;
      class GFG {
          public static void kLargest(Integer[] arr, int k)
              // Sort the given array arr in reverse order
              // This method doesn't work with primitive data
              // types. So, instead of int, Integer type
              // array will be used
              Arrays.sort(arr, Collections.reverseOrder());
              // Print the first kth largest elements
              for (int i = 0; i < k; i++)</pre>
                  System.out.print(arr[i] + " ");
        //This code is contributed by Niraj Dubey
```



```
public static ArrayList<Integer> kLargest(int[] arr, int k)
        //Convert using stream
       Integer[] obj array = Arrays.stream( arr ).boxed().toArray( Integer[] :: new);
        Arrays.sort(obj array, Collections.reverseOrder());
       ArrayList<Integer> list = new ArrayList<>(k);
        for (int i = 0; i < k; i++)</pre>
           list.add(obj array[i]);
        return list;
   public static void main(String[] args)
        Integer arr[] = new Integer[] { 1, 23, 12, 9,
                                        30, 2, 50 };
       int k = 3;
       kLargest(arr, k);
       //This code is contributed by Niraj Dubey
       //What if primitive datatype array is passed and wanted to return in ArrayList<Integer>
       int[] prim_array = { 1, 23, 12, 9, 30, 2, 50 };
         System.out.print(kLargest(prim array, k));
// This code is contributed by Kamal Rawal
```

#### Output

50 30 23

Time complexity: O(n\*log(n))

**Auxiliary Space:** O(1)

### Method 2 (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(k\*log(n))

Time complexity: O(n + k\*log(n))

#### Method 3 (Use Min Heap)

Using Min Heap.

Q



```
C++
       Java
```



```
import java.io.*;
      import java.util.*;
      class GFG{
public static void firstKElements(int arr[], int n, int k)
-,0,-
          PriorityQueue<Integer> minHeap = new PriorityQueue<>>();
          for(int i = 0; i < k; i++)</pre>
               minHeap.add(arr[i]);
          for(int i = k; i < n; i++)</pre>
               if (minHeap.peek() > arr[i])
                   continue;
               else
                   minHeap.poll();
                   minHeap.add(arr[i]);
```



```
Iterator iterator = minHeap.iterator();

while (iterator.hasNext())
{
    System.out.print(iterator.next() + " ");
}

public static void main (String[] args)
{
    int arr[] = { 11, 3, 2, 1, 15, 5, 4, 45, 88, 96, 50, 45 };
    int size = arr.length;
    int k = 3;
    firstKElements(arr, size, k);
}
```

## Output

50 88 96

**Time Complexity:** O(nlogn) **Auxiliary Space:** O(n)

Mark as Read

Report An Issue

If you are facing any issue on this page. Please let us know.





