Dash
All
Articles
Videos
Problems
Quiz
Prev
Next

Merge K Sorted Arrays

Given k sorted arrays of possibly different sizes, merge them and print the sorted output.

Examples:

```
Input: k = 3
      arr[][] = { {1, 3},
                  {2, 4, 6},
                  {0, 9, 10, 11}} ;
Output: 0 1 2 3 4 6 9 10 11

Input: k = 2
      arr[][] = { {1, 3, 20},
                  {2, 4, 6}} ;
Output: 1 2 3 4 6 20
```

A **simple solution** is to create an output array and one by one copy all k arrays to it. Finally, sort the output array. This approach takes O(N Log N) time where N is the count of all elements.

An **efficient solution** is to use a heap data structure. The time complexity of the heap-based solution is O(N Log k).

1. Create an output array.
2. Create a min-heap of size k and insert 1st element in all the arrays into the heap
3. Repeat the following steps while the priority queue is not empty.

.....a) Remove the minimum element from the heap (minimum is always at the root) and store it in the output array.

.....b) Insert the next element from the array from which the element is extracted. If the array doesn't have any more elements, then do nothing.

Below is the implementation of the above approach:

C++ Java Python3 C#

```java
/*package whatever //do not write package name here */

import java.util.ArrayList;
import java.util.PriorityQueue;

public class MergeKSortedArrays {
    private static class HeapNode
        implements Comparable<HeapNode> {
        int x;
        int y;
        int value;

        HeapNode(int x, int y, int value)
        {
            this.x = x;
            this.y = y;
            this.value = value;
        }

        @Override public int compareTo(HeapNode hn)
        {
            if (this.value <= hn.value) {
                return -1;
            }
            else {
                return 1;
            }
        }
    }

    // Function to merge k sorted arrays.
    public static ArrayList<Integer>
    mergeKArrays(int[][] arr, int K)
    {
```

```
        ArrayList<Integer> result
            = new ArrayList<Integer>();
        PriorityQueue<HeapNode> heap
            = new PriorityQueue<HeapNode>();

        // Initially add only first column of elements. First
        // element of every array
        for (int i = 0; i < arr.length; i++) {
            heap.add(new HeapNode(i, 0, arr[i][0]));
        }

        HeapNode curr = null;
        while (!heap.isEmpty()) {
            curr = heap.poll();
            result.add(curr.value);

            // Check if next element of curr min exists,
            // then add that to heap.
            if (curr.y < (arr[curr.x].length - 1)) {
                heap.add(
            }
        }

        return result;
    }

    public static void main(String[] args)
    {

        int[][] arr = { { 2, 6, 12 },
                        { 1, 9 },
                        { 23, 34, 90, 2000 } };
        System.out.println(
            MergeKSortedArrays.mergeKArrays(arr, arr.length)
                .toString());
    }
}
// This code has been contributed by Manjunatha KB
```

Dash
All
Articles
Videos
Problems
Quiz

Output

```
Merged array is
1 2 6 9 12 23 34 90 2000
```

Time Complexity: O(N Log k)
Auxiliary Space: O(N)