Kth smallest element in two sorted arrays

jitsceait 6 years ago

We have already solved a problem to find kth smallest element in an array using quicksort modification and min-heap. Today, our problem is to find K^{th} smallest element in two sorted arrays. For example if A = [10, 20, 40, 60] and B =[15, 35, 50, 70, 100] and K = 4 then solution should be 35 because union of above arrays will be C = [10,15,20,35,40,50,60,70,100] and fourth smallest element is 35.

K^{th} smallest element in two sorted arrays

As I always insist on doing: what will be the most brute force solution for this problem. Simple, merge two sorted arrays into one and find the kth smallest element in the array. There are different ways to merge two sorted arrays, merge part of merge sort will be most efficient when two parts are already sorted. The time complexity of merge is $O(n+m)$ and additional space complexity of $O(m+n)$ where m and m are sizes of two sorted arrays.

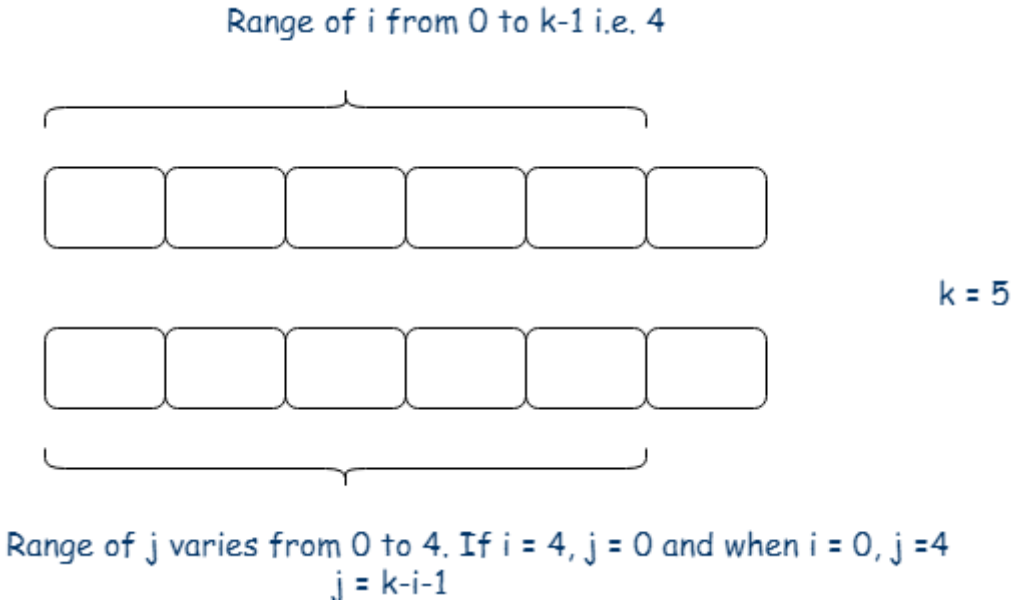
If you are preparing for an interview, you can signup for a free session to find a coach to help you with your preparation.

Book a free session

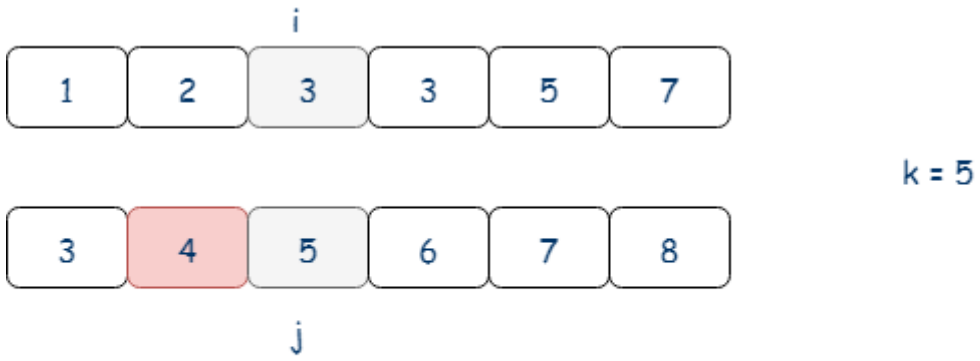
Now that we already know brute force solution, can we do better than this? The k-th smallest element can be present in any one of the two arrays. How many elements can we include from first array **A**, with size m, it can be max(k-1, m).

Let's suppose we took i elements from the first array, how many maximum numbers of elements we can take from array B? Since, we have zero-based indexing of array,

```
i+j = k-1
```

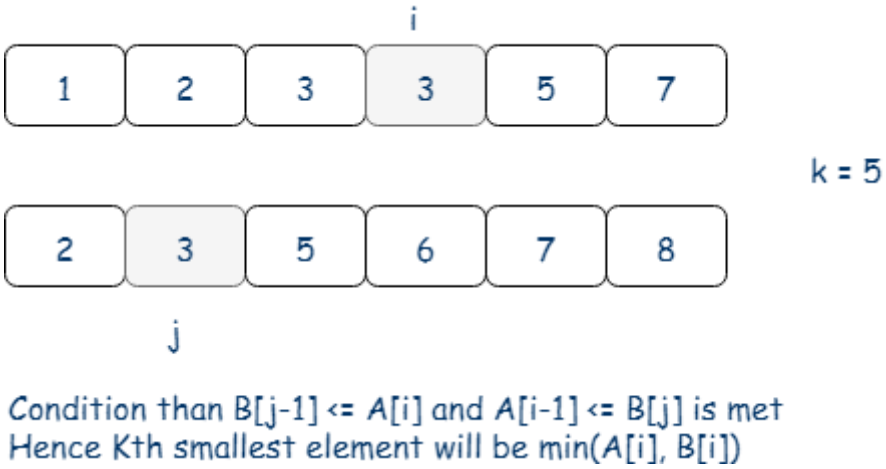


Now, we need to find combination of *i* and *j* such that all elements from 0 to *i* are less than **B[j]** and all elements from 0 to *j* should be smaller than **A[i]**, and min(A[i], B[j]) will be the k^{th} smallest element in the combined array.



For **A[i]**, all elements from index 0 to *i-1* are smaller, all we need to check is that all elements in array B from index 0 to index *j-1* are smaller too. Therefor A[i] >= B[j-1]

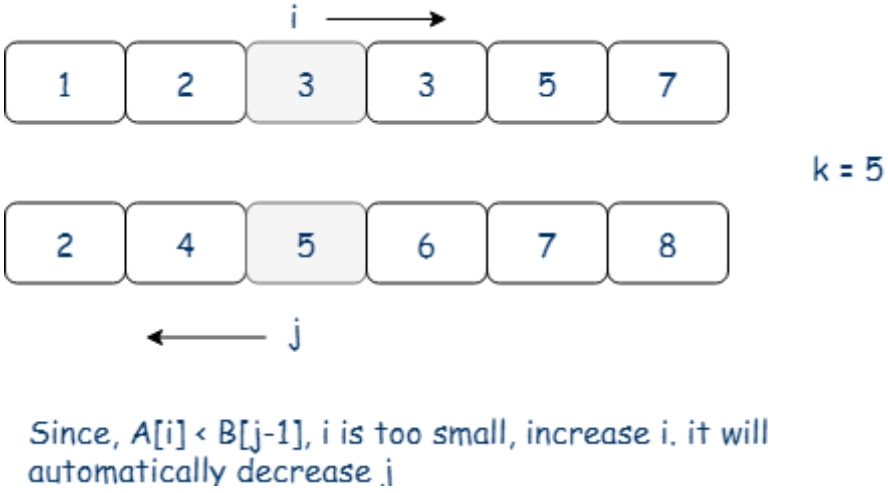
Similarly for B[j], it must satisfy the condition B[j] >= A[i-1].



In order to be kth smallest element, index i and j have to satisfy two conditions:

> 1. *A[i] >= B[j-1] and B[j] >= A[i-1]*
> 2. *i+j = k-1 or j = k-1-i*

What if A[i] < B[j-1] as we saw in the first picture? That means i is too small, we need to increase *i* and when we increase *i* and A[i], *j* and B[j] is decreased automatically, which makes it possible to satisfy A[i] >= B[j-1].



In the same vain when B[j] < A[i-1], *i* is too big and it's a good choice to decrease it.

This is where binary search comes into the picture. We can start *i* as mid of array A, *j* = k-i, and see if this *i* satisfies the condition. There can be three possible outcomes for the condition.

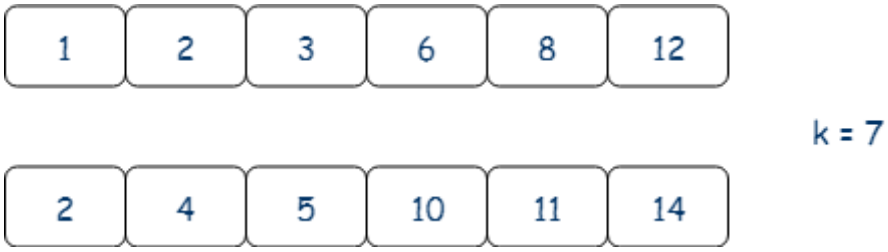
1. *A[i-1] >= B[j]* and *B[j-1] <=A[i]* is true, we return the index min(A[i], B[j])
2. If*B[j-1] > A[i]*, in this case, A[i] is too small. How can we increase it? by moving towards right. If *i*

is increased, value A[i] is bound to increase, and also it will decrease *j*. In this case, B[j-1] will decrease and A[i] will increase which will make B[j-1] >=A[i] is true. So, limit search space for i to mid+1 to min(k, m)

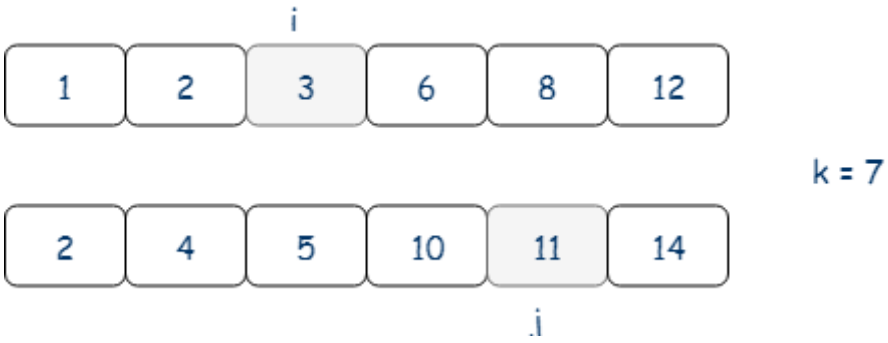
3. *A[i-1] > B[j]*, means A[i-1] is too big. And we must decrease i to get A[i-1] >=B[j]. Limit search space for *i* from 0 to *i-1*.

Since, *i* is bound by 0 to min(k-1,m), *j* will never go below 0. When *i* or *j* is 0, return min(A[i], B[j]).

Let's take an example and see how it works. Below are the two sorted arrays and K = 7.

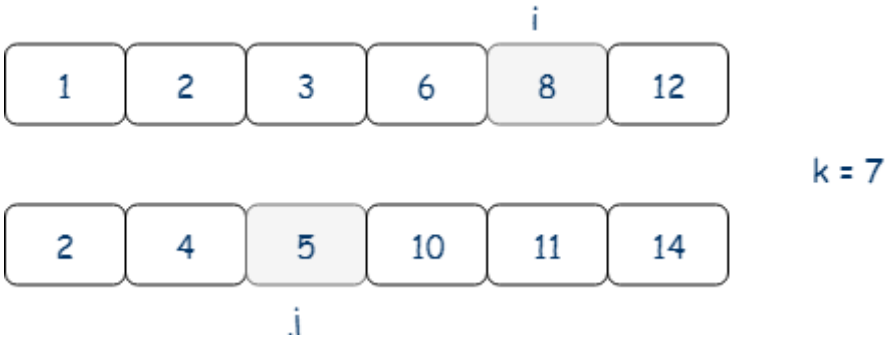


The first thing to notice is that k is greater than m, size of array A. Hence the range of *i* will be 0 to m i.e. 5. Find mid, i = 2 and j = k – 2-1 = 4

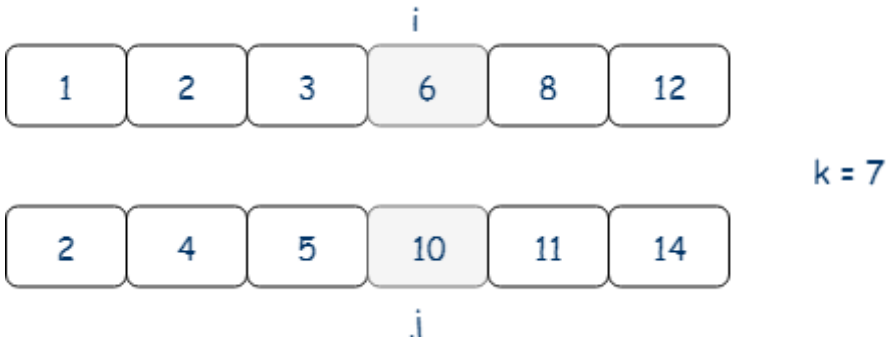


Now, A[i] < B[j-1], it is evident that we chose i too small. Search space is now from i+1 to m.

Again we find mid which is index 4 of array A. Index j = 2. At this point condition A[i-1] <= B[j] is false. This means we chose *i* too big and hence we decrease range till i-1 which now 3.



New *i* and *j* are shown below. Condition A[i-1] <= B[j] and B[j-1] <= A[j] is true. Return min(A[i], B[j]) as kth smallest element which is 6.



Implemmentation to find kth smallest element

```
package com.company;


/**
 * Created by sangar on 19.4.18.
 */
public class KthSmallestElement {

    public static double findKthSmallestElement(int[] A, int[] B, int k){

        int[] temp;

        int lenA = A.length;
        int lenB = B.length;

        if(lenA + lenB < k) return -1;

        int iMin = 0;
        int iMax = Integer.min(A.length, k-1);

        int i = 0;
        int j = 0;

        while (iMin <= iMax) {
            i = (iMin + iMax) / 2;
```

```
            j = k - 1 - i; // because of zero based index
            if (B[j - 1] > A[i]) {
                // i is too small, must increase it
                iMin = i + 1;
            } else if (i > 0 && A[i - 1] > B[j]) {
                // i is too big, must decrease it
                iMax = i - 1;
            } else {
                // i is perfect
               return Integer.min(A[i], B[j]);
            }
        }
        return -1;
    }

    public static void main(String[] args){
        int[] a = {1,3,5,6,7,8,9,11};
        int[] b = {1,4,6,8,12,14,15,17};

        double smallest = findKthSmallestElement(a,b, 9);
        System.out.println("Kth smallest element is : " + smallest);
    }
}
```

Complexity of algorithm to find K^{th} smallest element two sorted arrays is *O(log(N+M))*.

Please share if there is something wrong or missing. we would be glad to hear from you. If you are preparing for an interview and want to understand how to approach it, please book a free coaching session.

Categories: Algorithms, Amazon Interview questions, Arrays

Tags: find kth element, find kth element in two sorted arrays, find kth smallest element, find kth smallest element in sorted arrays, find kth smallest element in two sorted arrays

Leave a Comment