

North Rivers

Tuesday, April 30, 2013

K-th Smallest Element of Two Sorted Arrays

K-th Smallest Element of Two Sorted Arrays

Problem Description:

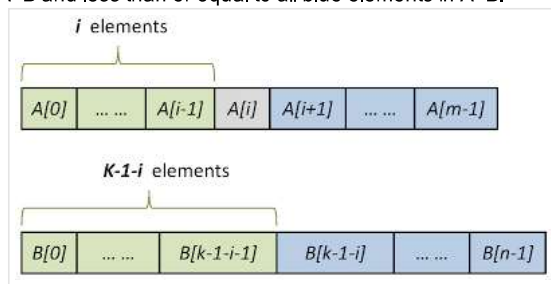
Given two sorted arrays A and B of size m and n, respectively. Find the k -th ($1 \leq k \leq m+n$) smallest element of the total $(m+n)$ elements in $O(\log(m)+\log(n))$ time.

Analysis:

This is a popular interview question and one special case of this problem is finding the median of two sorted arrays. One simple solution is using two indices pointing to the head of A and B, respectively. Then increase the index pointing to the smaller element by one till k elements have been traversed. The last traversed element is the k -th smallest one. It is simple but the time complexity is $O(m+n)$.

To get $O(\log(m)+\log(n))$ time complexity, we may take advantage of binary search because both of the two arrays are sorted. I first read the binary search method from this MIT handout, which aims to find the median of two sorted arrays. This method can be easily modified to solve the k -th smallest problem:

First of all, we assume that the k -th smallest element is in array A and ignore some corner cases for basic idea explanation. Element $A[i]$ is greater than or equal to i elements and less than or equal to all the remaining elements in array A ($A[i] \geq A[0..i-1]$ & $A[i] \leq A[i+1..m-1]$). If $A[i]$ is the k -th smallest element of the total $(m+n)$ elements ($A[i] \geq k-1$ elements in A+B and \leq all other elements in A+B), it must be greater than or equal to $(k-1-i)$ elements in B and less than or equal to all the remaining elements in B ($A[i] \geq B[k-1-i-1]$ & $A[i] \leq B[k-1-i]$). Below figure shows that $A[i]$ (in gray) is greater than or equal to all green elements ($k-1$ in all) in A+B and less than or equal to all blue elements in A+B.




Then it becomes very easy for us to check whether $A[i] \geq B[k-1-i-1]$ and $A[i] \leq B[k-1-i]$. If so, just return $A[i]$ as the result; if not, there are two cases:


1). $A[i] > B[k-1-i]$, which means $A[i]$ is greater than more than k elements in both A and B. The k -th smallest element must be in the lower part of A ($A[0..i-1]$);



2). otherwise, the k -th smallest element must be in the higher part of A ($A[i+1..m-1]$).

The search begins with range $A[0..m-1]$ and every time i is chosen as the middle of the range, therefore we can reduce the search size by half until we find the result. 

The above algorithm looks good, but it won't give you correct answer because there are many corner cases need to be addressed:

1). The simplest one may be that the k -th smallest element is in B rather than in A. When the entire array A has been traversed and no answer is returned, it means the k -th smallest element must be in B (if k is valid, i.e. $1 \leq k \leq m+n$). We just need to run another "binary search" in B and this time the correct answer is guaranteed to be returned. 

2). $i \geq k$. This means $A[i]$ is greater than or equal to k elements in A and will be at least the $(k+1)$ -th element in A+B. In this case, we need to "binary search" in the lower part of A.

About Me



nrive

Hong Kong, China

[View my complete profile](#)



Followers

Followers (2)



[Follow](#)

Blog Archive

▼ 2013 (5)

► May (1)

▼ April (4)

[K-th Smallest Element of Two Sorted Arrays](#)

[Longest Repeated Substring](#)

[Merge k Sorted Lists](#)

[Zigzag Conversion](#)



3). $i + n < k - 1$. Similarly to case 2), this means $A[i]$ will be at most the $(k-1)$ -th element in A+B. In this case, we need to "binary search" in the higher part of A.

4). At any time when we refer $B[k - 1 - i - 1]$ and $B[k - 1 - i]$, we should assure the indices are in range $[0, n)$ to avoid out of array bounds error.

The binary search on A and B takes $O(\log(m))$ and $O(\log(n))$ time respectively, so the worst case time complexity is $O(\log(m) + \log(n))$. For the problem of finding median of two sorted arrays, the $((m+n)/2 + 1)$ -th element should be returned if $(m+n)$ is odd. If $(m+n)$ is even, the average of the $((m+n)/2)$ -th and $((m+n)/2 + 1)$ -th elements is returned. Below is the code in c++.

Code (C++)

```
/**
 * Search the k-th element of A[0..m-1] and B[0..n-1] in A[p..q]
 */
int kth_elem(int A[], int m, int B[], int n, int k, int p, int q) {
    if (p > q)
        return kth_elem(B, n, A, m, k, 0, n-1); //search in B

    int i = p + (q - p) / 2;
    int j = k - 1 - i - 1;

    if ((j < 0 || (j < n && A[i] >= B[j]))
        && (j+1 >= n || (j+1 >= 0 && A[i] <= B[j+1]))) {
        return A[i];
    } else if (j < 0 || (j+1 < n && A[i] > B[j+1])) {
        return kth_elem(A, m, B, n, k, p, i-1);
    } else {
        return kth_elem(A, m, B, n, k, i+1, q);
    }
}

double median_two_arrays(int A[], int m, int B[], int n) {
    if ((m + n) % 2 == 1) {
        return kth_elem(A, m, B, n, (m+n)/2+1, 0, m-1);
    } else {
        return (kth_elem(A, m, B, n, (m+n)/2, 0, m-1) +
            kth_elem(A, m, B, n, (m+n)/2+1, 0, m-1)) / 2.0;
    }
}
```

-End-

Posted by nriver at 12:50 AM

 Recommend this on Google

Labels: Algorithm, Coding

1 comment:



Sida Chen August 15, 2015 at 12:16 PM

can u explain the corner cases more?

Reply

Enter your comment...

Comment as: Unknown (Google) ▼

Sign out

Publish

Preview

☐ Notify me

Newer Post

Home

Older Post

Subscribe to: Post Comments (Atom)

