Theory Assignment-1: ADA Winter-2024

Shobhit Raj (2022482)   Vashu (2022606)

1 Preprocessing

Not Applicable

2 Assumption

1-n based indexing is followed.

3 Algorithm Description

The algorithm employs a binary search-based approach to efficiently find the k^{th} smallest element in the union of three sorted arrays A, B, and C of size n. The algorithm begins by initializing the search range based on the minimum and maximum values in the union of three sorted arrays A, B, and C. It then enters a binary search loop, where it calculates the mid-point value (of the lowest and highest value) and uses a counting function to determine the total number of elements smaller than or equal to the mid-point in the union of arrays. The comparison of this total count with the target value k adjusts the size of the search range, either to the right or left. This binary search loop continues until the search range is narrowed down, ultimately identifying the k^{th} smallest element. The algorithm achieves a time complexity of O($(logn)^2$) by efficiently using binary search and counting operations.

4 Recurrence Relation

The recurrence relation for FINDPIVOT function is T(n)=T(n/2)+O(1) as it recursively uses binary search on array of n/2 elements which indicates the time complexity of this function O(logn). The recurrence relation for the KSMALLEST function is T(n) = T(n/2) + 3*log(n) as it recursively uses binary search on n/2 elements and uses the FINDPIVOT function 3 times to compute the count which gives 3*log(n) using above complexity of FINDPIVOT. Hence, the overall recurrence relation is

T(n) = T(n/2) + 3*log(n)

5 Complexity Analysis

Time complexity : O(log(max(A[n],B[n],C[n])- min(A[1],B[1],C[1])*3log(n)).
log(max(A[n],B[n],C[n]) - min(A[1],B[1],C[1])) is the time taken to optimize the space for binary search.
3*log(n) time is taken to count the number of elements smaller than the specific target in each array A,B,C of size n. By using master's theorem on above recurrence relation
T(n) = O($(logn)^2$)

Auxilliary Space : O(1), The algorithm does not use any extra space.

6 Proof of Correctness

The provided algorithm for finding the k-th smallest element in the union of three sorted arrays (A, B, and C) is based on the principles of binary search and counting. The key idea is to efficiently search for the k^{th} smallest element within a certain range of values. The ‘FINDPIVOT‘ function is employed to count the number of elements smaller than or equal to a given value in a sorted array. The main ‘KSMALLEST‘ function employs a binary search approach on the range of possible values for the k^{th} smallest element. The search range is iteratively narrowed down based on the counts obtained from the ‘FINDPIVOT‘ function. The algorithm is designed such that, at the termination of the binary search loop, the ‘low‘ value represents the k^{th} smallest element. This algorithm works because it leverages binary search for efficiency, focusing on the search space where the k^{th} smallest element is likely to be found. We don’t work on finding the elements and don’t worry what they are, we just divide and narrow the search space based on the number of elements using the binary search algorithm and ’mid’ value. The counting mechanism ensures correctness by providing the total count of smaller elements, guiding the search range adjustments until it reaches the bases case where low=high and count=k and search space is concised to k elements. So in k elements search space, the largest element is the k^{th} smallest element of the unions of arrays. Overall, the algorithm involve narrowing down the search range and counting the number of elements smaller than or equal to a mid-point value.

7 Pseudocode

Algorithm 1 Your Algorithm

```
1: function FINDPIVOT(A,m,num)
     Initialize count $\leftarrow 0, low \leftarrow 1, high \leftarrow m, mid \leftarrow 0$;
     $while$low $\leq$high :
     mid $\leftarrow (low + high)$;
     $if$(A[mid] $\leq$ num) then
     count $\leftarrow mid + 1$;
     $left \leftarrow mid + 1$;
     $end$
     $else\ \ then$
     $right \leftarrow mid - 1$;
     $end$
     $return\ count$

2:   function KSMALLEST(A,B,C,n,k)
   Initialize low $\leftarrow min(A[1], B[1], C[1])$;
   $Initialize\ high \leftarrow max(A[n], B[n], C[n])$;
   $Initialize\ mid$;
   $while\ low < high$ :
   mid $\leftarrow (low + high)/2$;
   $count \leftarrow FINDPIVOT(A, n, mid) + FINDPIVOT(B, n, mid) + FINDPIVOT(C, n, mid)$;
   $if$ count $\leq k$ then
   low $\leftarrow mid + 1$;
   $end$
   $else$
   $high \leftarrow mid$;
   $end$
   $return\ low$;
   end function
```