Theory Assignment-1: ADA Winter-2024

Ashutosh Dwivedi 2022116 Pandillapelly Harshvardhini 2022345

1 Preprocessing

NA

2 Algorithm Description

We are given three sorted arrays A, B, and C, each having n numbers. The goal is to find the k-th smallest element in the union of these three arrays. The algorithm uses a binary search approach, maintaining a range of possible values for the k-th smallest element.

Algorithm 1: Binary search

Binary search efficiently finds a target element in a sorted array by iteratively reducing the search range. Returns the index of the value or 0 if not found.

Algorithm 2: Count the elements less than or equal to

This algorithm uses Binary Search to count elements less than or equal to the target in a sorted order, and gives count as a result.

Algorithm 3: Kth Smallest Element

Finds the kth smallest element among three sorted arrays using Binary Search to count elements less than or equal to a midpoint. The search space is adjusted until convergence, returning the final lower bound as the k-th smallest element.

3 Recurrence Relation

NA

4 Complexity Analysis

The time complexity of the algorithm is $O(\log^2 n)$, where n is the size of the input arrays.

5 Pseudocode

Algorithm 1 Binary Search

```
function BinarySearch(arr, target)
    $l1 \leftarrow 0$
    $h1 \leftarrow$ size(arr) $- 1$
    $result \leftarrow -1$
    while $l1 \leq h1$ do
        $mid \leftarrow l1 + \frac{h1 - l1}{2}$
        if arr[mid] $\leq$ target then
            $result \leftarrow$ mid
            $l1 \leftarrow$ mid $+ 1$
        else
            $h1 \leftarrow$ mid $- 1$
        end if
    end while
    return $result + 1$
end function
```

P.T.O

Algorithm 2 Count Elements Less Than or Equal

```
function CountElementsLessThanEqual(arr, target)
    $count$ ← BinarySearch(arr, target)
    return $count$
end function
```

Algorithm 3 Kth Smallest Element

```
function KthSmallestElement(A, B, C, k)
    $l1$ ← min(A[0], B[0], C[0])
    $h1$ ← max(A[last], B[last], C[last])
    while $l1 < h1$ do
        $mid$ ← $l1 + \frac{h1-l1}{2}$
        $countA$ ← CountElementsLessThanEqual(A, mid)
        $countB$ ← CountElementsLessThanEqual(B, mid)
        $countC$ ← CountElementsLessThanEqual(C, mid)
        $totalCount$ ← $countA + countB + countC$
        if $totalCount$ < k then
            $l1$ ← mid + 1
        else
            $h1$ ← mid
        end if
    end while
    return $l1$
end function
```

6 Proof of correctness

The correctness of the combined algorithm is based on binary search principles. The binary search in Algorithm 2 guarantees that the index of the target element will be correctly found in the array. Algorithm 3 uses Algorithm 2 to find the kth smallest element among three sorted arrays, ensuring that it is correct with careful initialization and reasonable modifications of the search procedure The proof depends on invariants and by maintaining binary search termination conditions.