CSE221 Quiz 1 Solution

Date:........................

set A

1. **first loop:**

$0 - n$ $\quad$ $i = i + 3$

$\therefore$ runs $n/3$ times

2nd loop:

~~0~~ $n - 1$ $\quad$ $j = j/5$

runs $\log_5 n$

3rd loop:

$k = 1 - n$ $\quad$ $k = k * 5$

runs $\log_5 n$

$\therefore$ total complexity

$$= O\left(\frac{n}{3} \log_5 n \cdot \log_5 n\right)$$

$$= O\left(n (\log_5 n)^2\right)$$

Q3. **1st loop:**

$i = n/2 - 1$ $i /= 6$

$\therefore$ runs $n/12$ times.

2nd loop:

$j = 2 - i$ $j ~~*~~ = 4$

$\log_4 i$

3rd loop:

$k = 0$, so $\boxed{\text{infinite}}$

$\therefore$ time complexity undetermine

Inspiring Excellence

set B

Date:........................

1. 1st loop:

$i = 0 - n$ $\quad$ $i += 4$

$n/4$

2nd loop:

$j = 1 - n$ $\quad$ $j *= 2$

$\log_2 n$

3rd loop:

$\textcircled{1}$ 30 times, constant

$\textcircled{2}$ $m = n - 0$ $\quad$ $m -= 2$

$n/2$

$\therefore$ $O(30 + n/2)$

$\therefore$ total complexity

$= O(n/4)\, O(\log n)\, O(30 + n/2)$

$= O(n)\, O(\log n)\, O(n)$

$= O(n^2 \log n)$

Set A:

3. To find the index i where the array transitions from one increasing sequence to another, you can follow these step-by-step instructions:

1. **Initialize Variables**:
   Set two pointers, `left = 0` and `right = N - 2`, where NNN is the size of the array. We only need to go up to `N-2` because we're looking for a transition between two elements.
2. **Binary Search Setup**:
   Use binary search to efficiently find the transition index. The idea is to locate the position where the array goes from larger values (first increasing sequence) to smaller values (start of the second increasing sequence).
3. **Binary Search Loop**:
   - While `left <= right`:
     - Calculate the middle index: `mid = left + (right - left) // 2`
     - **Check the Transition Condition**:
       - If `array[mid] > array[mid + 1]`, then:
         - You have found the transition point, so set `i = mid + 1` and exit the loop.
     - **Adjust Pointers**:
       - If `array[mid] < array[mid + 1]`, then the transition point is further to the right, so set `left = mid + 1`.
       - Otherwise, set `right = mid - 1`.
4. **Return Result**:
   The value of `i` after exiting the loop is the index where the transition occurs.

Time complexity: O(logn)

Set B:

3. Initialize Pointers:

- Set two pointers, `left = 0` and `right = N - 1`, where NNN is the number of elements in the array.

Binary Search Loop:

- While `left` is less than or equal to `right`:
  - Calculate the middle index: `mid = left + (right - left) // 2`.
  - **Check Peak Condition**:

        - If `array[mid] > array[mid - 1]` (left neighbor) and `array[mid] > array[mid + 1]` (right neighbor), then `array[mid]` is the peak (maximum), and you can return `array[mid]` as the result.
    - **Adjust Pointers Based on Slope**:
        - If `array[mid] < array[mid + 1]`, the maximum must be to the right of `mid` (as the sequence is still increasing), so set `left = mid + 1`.
        - If `array[mid] > array[mid + 1]`, the maximum must be to the left of `mid` (as the sequence is now decreasing), so set `right = mid - 1`.

Return Result:

- The loop will terminate when `left` and `right` converge on the maximum element's index. After the loop, the maximum element will be at `array[left]` or `array[right]`, so return `array[left]` as the maximum.