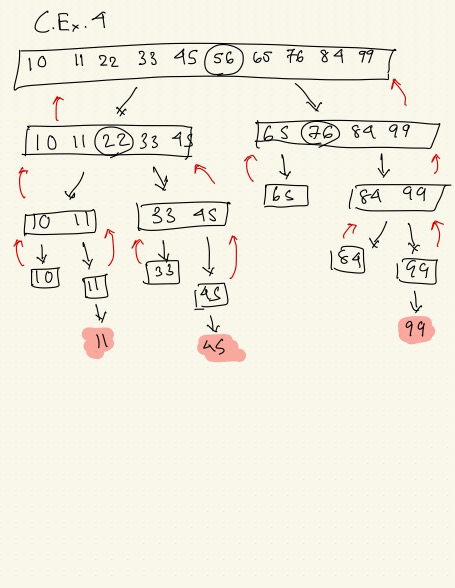
Class Ex 4

Q1)

Q2)

**Case Type: Best Case**

Explanation:

* In quicksort, the best case occurs when the pivot evenly splits the list into two approximately equal parts at every step, ensuring optimal time complexity of O(n log n).
* Here, using the **median-of-three** pivot selection and the naturally sorted order of the list, the pivots (56, 22, 76) divide the subarrays into balanced halves, minimizing recursion depth and comparisons.
* This balanced partitioning makes the process efficient, exemplifying the **best-case scenario**.

### **Task 5: Quicksort Upper Bound for Already Sorted Input**

* **Observation**: When the input is already sorted, the choice of the pivot can significantly impact the performance.
* **Worst Case**: If the pivot is the smallest or largest element in every partition, the recursive depth becomes n, resulting in quadratic behavior.
* **Upper Bound**: T(n)=O(n^2), as each partition results in one empty and one full sublist.

**Task 6: Quicksort Upper Bound for Reverse Sorted Input**

* **Observation**: A reverse-sorted input behaves similarly to an already sorted input if the pivot selection is poor.
* **Worst Case**: Each partition leads to an empty and a full sublist, leading to quadratic behavior.
* **Upper Bound**: T(n) = O(n^2)

**Task 7: Quicksort Upper Bound for Random Input**

* **Observation**: With a random pivot, the partitions are likely balanced, reducing the depth of recursion.
* **Expected Case**: For well-balanced partitions, the recursive depth is log n, and each level requires O(n) work.
* **Upper Bound**: T(n) = O(n \log n)

**Task 8: Comparing Quicksort and Mergesort**

**Advantages of Quicksort**:

1. **In-Place Sorting**: Quicksort operates in-place, requiring less additional memory compared to mergesort.
2. **Cache Efficiency**: Quicksort accesses memory more linearly, improving cache performance.

**Disadvantages of Quicksort**:

1. **Worst-Case Performance**: Quicksort can degrade to O(n^2) in its worst case.
2. **Unstable**: Quicksort does not preserve the relative order of equal elements.

**Advantages of Mergesort**:

1. **Stable**: Mergesort preserves the order of equal elements.
2. **Guaranteed** O(n \log n) **Performance**: Mergesort has a consistent runtime, regardless of input order.

**Disadvantages of Mergesort**:

1. **Space Requirement**: Mergesort requires extra space for merging.
2. **Overhead**: Mergesort can have higher constant factors due to copying operations.

**Task 9: Proposing an Optimized Quicksort**

**Goal**: Minimize the chance of hitting the worst-case scenario.

1. **Randomized Pivot Selection**: Randomly select the pivot to ensure that partitions are less predictable and more balanced on average.
2. **Median-of-Three Pivot**: Choose the median of the first, middle, and last elements as the pivot to improve balance.
3. **Hybrid Approach**:
   * Switch to insertion sort for small sublists (e.g., n ≤ 10).
   * Insertion sort is more efficient for small datasets due to lower overhead.