MOST IMPORTANT QUESTIONS 🔥

7-Mark Questions

1. Explain Bubble sort.
2. Explain Selection sort.
3. Explain Insertion sort.
4. Explain Quick sort.
5. Explain Merge sort.
6. Explain Heap sort.
7. Explain Sequential search.
8. Explain Binary search.
9. Differentiate searching methods.
10. Explain hashing.

14-Mark Questions

1. Compare different sorting techniques.
2. Explain Quick sort with example.
3. Explain Merge sort with example.
4. Explain Heap sort with heapify.
5. Explain Binary search with example.
6. Compare searching methods.
7. Explain hashing and indexing.

1. Compare Different Sorting Techniques

Introduction

Sorting techniques data ko ascending ya descending order me arrange karti hain.

Different sorting algorithms ki speed, memory usage aur working different hoti hai.

Comparison Table

Sorting Technique	Time Complexity	Stable	Extra Memory	Method
Bubble Sort	$O(n^2)$	Yes	No	Swapping
Selection Sort	$O(n^2)$	No	No	Selection
Insertion Sort	$O(n^2)$	Yes	No	Insertion
Quick Sort	O(n log n)	No	Less	Divide & Conquer
Merge Sort	O(n log n)	Yes	Yes	Divide & Conquer
Heap Sort	O(n log n)	No	No	Heap Tree

Bubble Sort

- Adjacent elements compare karta hai
- Large element end me move hota hai

Advantages

- Simple

Disadvantages

- Slow

Selection Sort

- Minimum element select karta hai

Advantages

- Less swapping

Disadvantages

- More comparisons

Insertion Sort

- Correct position par insertion karta hai

Advantages

- Efficient for small data

Disadvantages

- Large data ke liye slow

Quick Sort

- Pivot select karta hai
- Fast sorting

Advantages

- Very efficient

Disadvantages

- Worst case $O(n^2)$

Merge Sort

- Divide and merge technique

Advantages

- Stable sorting

Disadvantages

- Extra memory

Heap Sort

- Heap tree use karta hai

Advantages

- Efficient

Disadvantages

- Complex implementation

Best Sorting Technique

Situation	Best Sorting
Small Data	Insertion
Large Data	Quick Sort
Stable Sorting	Merge Sort
Memory Efficient	Heap Sort

Conclusion

Different sorting techniques different situations me useful hoti hain. Quick sort aur merge sort most efficient techniques hain.

---

2. Explain Quick Sort with Example

Introduction

Quick sort divide and conquer sorting algorithm hai.

Ek pivot select hota hai aur array divide hota hai.

Working Principle

```
1. Select pivot
2. Smaller elements left
3. Larger elements right
4. Recursively sort
```

Algorithm

```
QUICKSORT(arr, low, high)

Step 1: Select pivot

Step 2: Partition array

Step 3: Recursively sort left subarray

Step 4: Recursively sort right subarray
```

Example

Array:

```
50 20 10 40
```

Pivot:

```
40
```

Partition

Elements smaller than 40:

```
20 10
```

Elements larger than 40:

```
50
```

After partition:

```
20 10 40 50
```

Recursive Sorting

Left side:

```
20 10
```

Sorted:

```
10 20
```

Final Sorted Array

```
10 20 40 50
```

Advantages

- Very fast
- Efficient for large datasets

Disadvantages

- Worst case $O(n^2)$

Time Complexity

```
Best = O(n log n)
Worst = O(n²)
```

Applications

- Databases
- System libraries
- Searching optimization

Conclusion

Quick sort highly efficient divide and conquer sorting technique hai.

---

3. Explain Merge Sort with Example

Introduction

Merge sort divide and conquer sorting technique hai.

Array ko divide karke merge kiya jata hai.

Working Principle

```
1. Divide array
2. Sort subarrays
3. Merge sorted arrays
```

Algorithm

```
MERGESORT(arr)

Step 1: Divide array into halves

Step 2: Recursively sort halves

Step 3: Merge sorted halves
```

Example

Array:

```
40 20 10 30
```

Divide

```
40 20
10 30
```

Further divide:

```
40 | 20
10 | 30
```

Merge

```
20 40
10 30
```

Final merge:

```
10 20 30 40
```

Advantages

- Stable sorting
- Efficient for large data

Disadvantages

- Extra memory required

Time Complexity

```
O(n log n)
```

Applications

- External sorting
- Large databases

Conclusion

Merge sort efficient and stable sorting algorithm hai using divide and conquer.

---

4. Explain Heap Sort with Heapify

Introduction

Heap sort heap tree based sorting algorithm hai.

Heap complete binary tree hota hai.

Types of Heap

1. Max Heap

Parent > children

---

2. Min Heap

Parent < children

Heapify

Heapify process heap property maintain karta hai.

Working Principle

```
1. Build max heap
2. Swap root with last element
```

```
3. Heapify remaining elements
4. Repeat
```

Example

Array:

```
40 20 10 30
```

Build max heap:

```
40
/ \
30 10
/
20
```

Swap root with last:

```
20 30 10 40
```

Heapify again.

Final sorted array:

```
10 20 30 40
```

Advantages

- Efficient
- No extra memory

Disadvantages

- Complex implementation

Time Complexity

```
O(n log n)
```

Applications

- Priority queue
- Scheduling systems

Conclusion

Heap sort efficient sorting technique hai jo heap tree aur heapify process use karta hai.