



# Sorting and Searching in Java

**With Rohit Mazumder**

Let's crack Competitive Programming together!

# Lecture - 1

# Introduction to Sorting

---

1. Which of the following sorting algorithms in its typical implementation gives best performance when applied on an array which is almost sorted (1 or 2 elements are misplaced) ?


- A. Selection Sort
- B. Bubble Sort
- C. Insertion Sort
- D. None of the above

- A. Selection Sort
- B. Bubble Sort
-  C. Insertion Sort
- D. None of the above

Solution : Since insertion sort only does extra work for the elements out of place. It will be most efficient in this case.

2. What is the best case time complexity we can achieve in bubble sort?

- A.  $O(n^2)$
- B.  $O(n \log n)$
- C.  $O(n)$
- D.  $O(n^3)$

- A.  $O(n^2)$
- B.  $O(n \log n)$
-  C.  $O(n)$
- D.  $O(n^3)$

Solution : By using a boolean swap variable to break when the array is already sorted we can achieve  $O(n)$   
Otherwise it is  $O(n^2)$

3. Selection Sort never makes more than  $n$  swaps?

A. True

B. False


- A. True
- B. False

Solution : Because we only swap after finding the index of minimum element and at worst all the elements could be out of their place.



4. What is the best auxiliary space complexity a sorting algorithm can have?


- A.  $O(n)$
- B.  $O(1)$
- C.  $O(n \log n)$
- D.  $O(n^2)$

- A.  $O(n)$
-  B.  $O(1)$
- C.  $O(n \log n)$
- D.  $O(n^2)$

Solution : Bubble, insertion and selection sort have  $O(1)$  space complexity.

5. Consider the array  $\{4,3,5,2\}$  how many swaps will be needed to sort the array using selection sort?

- A. 1
- B. 2
- C. 3
- D. 4

- A. 1
-  B. 2
- C. 3
- D. 4

Solution : First we swap 2 and 4, then we swap 4 and 5. So we require 2 swaps.

# Lecture - 2

## Merge Sort

---


1. What is the maximum number of inversions possible in an array of length  $n$ ?

A.  $n^2$

B.  $n*(n-1)/2$

C.  $n*(n+1)/2$

D.  $n+1$

- A.  $n^2$
-  B.  $n*(n-1)/2$
- C.  $n*(n+1)/2$
- D.  $n+1$

Solution : Worst case the array is sorted in decreasing order.

2. Which of the following mentioned algorithms works on the divide and conquer paradigm?

- A. Merge Sort
- B. Selection Sort
- C. Bubble Sort
- D. Insertion Sort



- ✓ A. Merge Sort
- B. Selection Sort
- C. Bubble Sort
- D. Insertion Sort

Solution : Merge Sort works on divide and conquer as discussed in class.

3. You have to sort 1GB of data with only 100MB of main memory available. Which sorting technique will be most appropriate?


- A. Bubble Sort
- B. Merge Sort
- C. Selection Sort
- D. Insertion Sort

- A. Bubble Sort
-  B. Merge Sort
- C. Selection Sort
- D. Insertion Sort

Solution : Merge sort works for external sorting

4. What is the total number of levels in the recursion tree while using merge sort on an array of 8 elements?

- A. 1
- B. 2
- C. 3
- D. 4

- A. 1
- B. 2
- C. 3
-  D. 4

Solution : Level -1  $\Rightarrow 8$


Level-2  $\Rightarrow 4+4$

Level-3  $\Rightarrow 2+2+2+2$

Level-4  $\Rightarrow 1+1+1+1+1+1+1+1$

5. What is the worst case time complexity of merge sort?

- A.  $O(n)$
- B.  $O(n \log n)$
- C.  $O(n^2)$
- D.  $O(\sqrt{n})$

- A.  $O(n)$
-  B.  $O(n \log n)$
- C.  $O(n^2)$
- D.  $O(\sqrt{n})$

Solution : Derived from recurrence relation

$$T(n) = 2 * T(n/2) + O(n)$$

6. Which of the following algorithms works best for sorting arrays?

- A. Bubble Sort
- B. Selection Sort
- C. Merge Sort
- D. Insertion Sort



- A. Bubble Sort
- B. Selection Sort
-  C. Merge Sort
- D. Insertion Sort

Solution : Worst case for merge sort is  $O(n \log n)$  while for others it is  $O(n^2)$