





Ans 3 Average case complexities of sorting algo:

- \* bubble =  $O(n^2)$
- \* insertion =  $O(n^2)$
- \* selection =  $O(n^2)$
- \* quick =  $O(n \log n)$
- \* heap =  $O(n \log n)$
- \* merge =  $O(n \log n)$

Ans 4

	Stable (where records)	Efficient ( $O(1)$ )
bubble	✓	✓
selection	✗	✓
insertion	✓	✓
merge	✓	✗
quick	✗	✗
Heap	✗	✓

Ans 5 Pseudocode for binary search

```

int start = 0
int end = size - 1
while (start <= end)
{
    int mid = (start + (end - start)) / 2
    if (key == arr[mid])
        return mid
    else if (key < arr[mid])
        end = mid - 1
    else
        start = mid + 1
}
return -1

```

Time complexity =  $O(\log n)$

Space complexity =  $O(1)$



Ans 6 Recurrence relation of binary search:  

$$T(n) = T(n/2) + 1$$

Ans 8 Quick sort: The best sorting algo in practical use as it follows the locality of reference and also its best case Time complexity is  $O(n \log n)$

Ans 9 NO. of inversions: tells us how far is the array is from being sorted

If  $a[i] > a[j]$  and  $i < j$

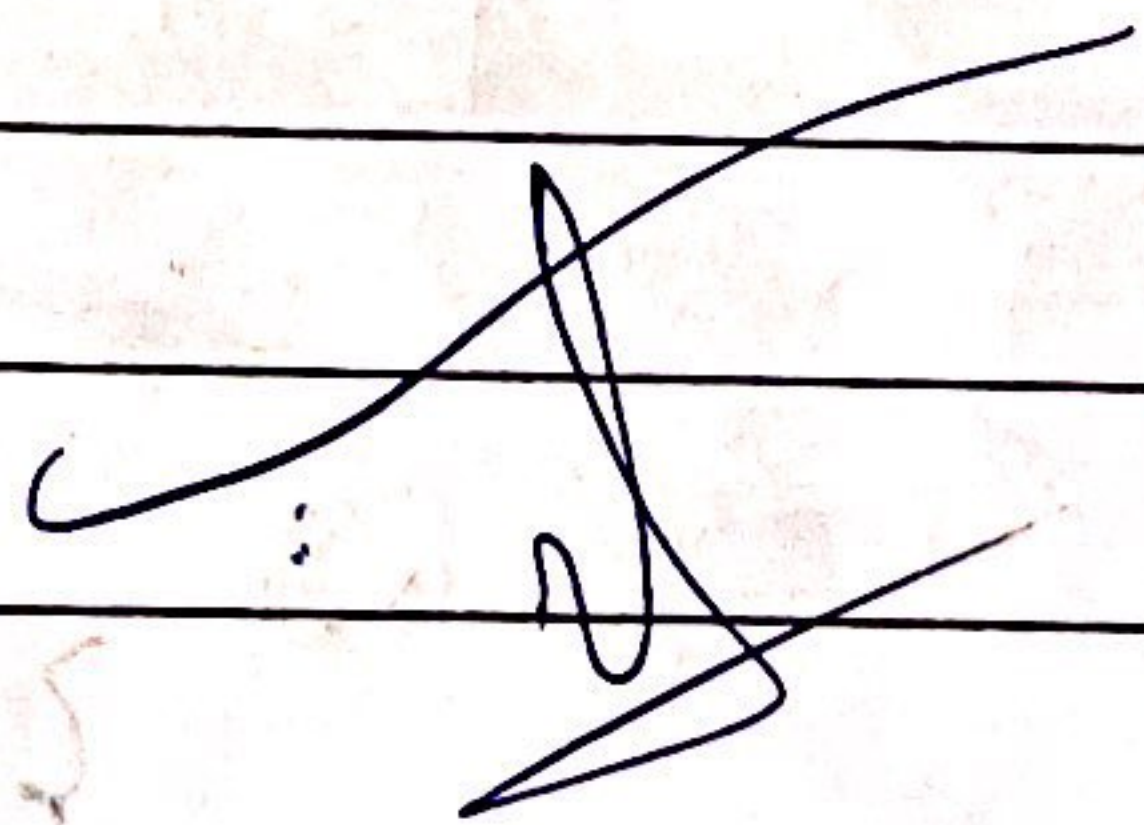
$\Rightarrow$  7 2 1 3 8 10 1 20 4 5

no. of inversions:  $4 + 7 + 1 + 4 + 4 + 3 + 2 = 31$

Ans 10 Quick sort will give:

\* best case complexity: when array is already sorted

\* worst complexity: when array is  $\log$  or reverse sorted





Recurrence rel<sup>n</sup> of:  
merge sort

Divide sort

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

Best  $\rightarrow 2T(n/2) + O(n)$   $T(n) = T(n-1) + O(n)$   
Worst  $\rightarrow$

Similarly: Both are of type divide and conquer

Difference: worst case complexity of merge sort is  $O(n \log n)$  where as of quick sort is  $O(n^2)$

13. Optimal Bubble Sort

for  $i = 0; i < n; i++$

{ swap = false

for  $j = 0; j < n - i - 1; j++$

{ if (arr[j] > arr[j+1])

{ swap (arr[j], arr[j+1])

swap = true

}