HANDWRITTEN PAPER

① Approach to solve the first problem i.e (Insertion Sort).

Ans → First of all, I created an integer array and filled it in unsorted order.

→

0	1	2	3	4
5	4	3	2	1

Sorted sublist ↑ ↑ Unsorted sublist.

→ Then I took the first position element in temp variable & compared it to all the elements present on the left side.

→ If on comparision the left element of the temp variable (first positioned) found greater than the first positioned element then this left element (i.e. element at 0^{th} position is now stored under the first position of the array.

→ This comparision is done to all the elements present behind the first variable & after this we will move to 2^{nd} position.

→ All the steps would be repeated until we reach the end.

Nikhil Kumar Arora
IT
11912063
IIIT SONEPAT

NIKHIL KUMAR ARORA

Date
Page No.

```
\*  Analysis on time complexity
    of insertion sort algorithm in
           best case                 */
Void isort(int *a, int n)
 {
   int temp, i;
   for (i=1; i<n; i++)          O(n)
   {
     temp = a[i];
     int j = i-1;
   While (j>=0 && a[j]>temp) ≃ O(n)
    {
      a[j+1] = a[j];
        j--;
     }
     a[j+1] = temp;
   }
 }
```

In best case only one comparision would be done here so time complexity would get reduced to O(n).

→ Since there are two loops present (one inside other) and both of them have time complexity of $O(n)$.

→ So in worst case, the time complexity would be $O(n^2)$.

→ But in best case like sorted array (1,2,3,4,5), time complexity would be $O(n)$. Because in best case, the inside loop would not do or perform its operations as we will not enter in that loop So the time complexity would only be $O(n)$.

NIKHIL KUMAR ARORA

Date
Page No.

Binary Insertion Sort

- Yes Some modifications can be performed to reduce the time complexity of insertion sort from $O(n^2)$ to $O(n\log n)$
- This technique is known as binary insertion sort.
- In this technique, we will be using binary search to find a proper place for inserting an element.

Time comp. for Binary search = $O(\log n)$

So time comp. of Binary insertion sort $= n \times O(\log n)$

$= O(n\log n)$