Vaibhav Bhaskar 11912060
IT.

Insertion Sort Analysis:

initial algorithm

```
for (i from 1 to n-1)
    j = i-1 ;
    temp = arr[i] // element to be inserted.
    while(i>=0 and temp < arr[j]) // comparison
        arr[i+1] = arr[j];
            j--;
        arr[j+1] = temp
```

① Best case ; n=5 ; 1,2,3,4,5 ;

i=1, j=0 → 1 comparison , 0 movements
i=2, j=1 → 1 comparison , 0 movements
i=3, j=2 → 1 comparison , 0 movements
i=4, j=3 → 1 comparison , 0 movements.
~~i=5, j=4~~

⇒ 4 comparisons, 0 movements

for n in general = n-1 comparisons,
(Best case) 0 movements

$$\boxed{T(n) \approx O(n)}$$

② Worst case, n=5, 5, 4, 3, 2, 1

i=1, j=0 → 1 comparisons , 1 movements
i=2, j=1 → 2 comparisons , 2 movements
i=3, j=2 → 3 comparisons , 3 movements
i=4, j=3 → 4 comparisons , 4 movements

→ 4+3+2+1 comp., 4+3+2+1 mov.

for n general = $\frac{n(n-1)}{2}$ comparisons , $\frac{n(n-1)}{2}$ movements.

Time Complexity ≈ $O(n^2)$

(3) Average case

n = 5, 3, 5, 4, 2, 1

In average case also it would be n^2

Time Complexity ≈ $O(n^2)$

Improved Algorithm

* proposal → • Take another array of size double the given array ⇒ space complexity increases from O(1) to O(n)
• use binary search to insert element.

* Algorithm n, arr1 = org. array, arr = new array

```
for(i from 0 to n-1)
    e = arr1[i]
    arr[end] = arr1[0]          // new element already at end.
    if (end != start)
        index = bsearch(start, end-1, arr, e)
        if(index == start-1) // descending order
            arr[index] = e;
            start = start-1;
        else if (index == end) // ascending order
            end++;
        else                    // random order
            temp = end
            while (temp != index)
                arr[temp] = arr[temp-1]
                temp--
            arr[temp+1] = e;
            end++
    else
        end++;
    finally copy arr1 to arr;
```

① Best case ; n=5 1, 2, 3, 4, 5

comparisons or searching

i=0	0 comparisons
i=1	log 1
i=2	log 2
i=3	log 3
i=4	log 4

movements = 0, comp = log 1 + log 2 + log 3 + log 4

In general for n, $\approx O(n \log n)$

② worst case n=5, 5, 4, 3, 2, 1

i=0	0 comparisons
i=1	log 1 comparisons
i=2	log 2 comparisons
i=3	log 3 comparisons
i=4	log 4 comparisons.

⇒ movements = 0 , comparisons = log 1 + log 2 + log 3 + log 4

In general → $\approx O(n \log n)$

③ Average case

Somewhere b/w $O(n \log n)$ and $O(n^2)$

T= less than $O(n^2)$