Date → 17/04/2020. Time: 3:00PM to 4:00PM

Insertion sort: →

- A sublis (or sorted array) is maintained which is already sorted.
- (n-1) pass are required to sort n elements
- Pick left most element of the unsorted array as the first element of the sorted array.
- Pick up next element from left unsorted list/array and insert it into its correct Position in the already sorted array by comparing it with each elements in the sorted array from right to left. This Process will continue until the unsorted list is empty.
- Not suitable for large data set
- Best case time complexity $O(n)$
- Worst case time complexity, $O(n^2)$
- Average case time complexity $O(n^2)$

Note → Insertion sort is used when we sort playing cards in our hands.

example: I/P 74 32 89 55 21 64 , n = 6

Pick 74 as the first element of the sorted array

Required No. of passes = 5

Pass 1 : 32, 74 , No. of comparisions = 1

Pass 2 : 32, 74, 89 , No. of comparisions = 1

Pass 3 : 32, 55, 74, 89 , No. of comparisions = 3

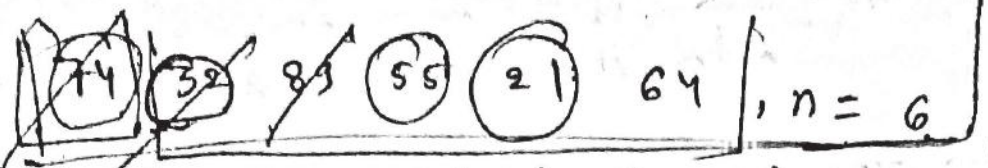
Pass 4 : 21, 32, 55, 74, 89 , No. of comparisions = 4

Pass 5 : 21, 32, 55, 64, 74, 89 , No. of comparisions = 3

O/P: 21 32 55 64 74 89 .

Analysis of Insertion sort

↳ Best case: In Insertion sort, the best case occurs if the array is already sorted.

I/P: 

~~first of all Pick first element of the unsorted array~~ as

→ Pick 74 as first element of the sorted array.

Pass 1: 32 74. No. of comparision = 1

Pass 2: 32 74 89., No of comparision = 1

Pass 3: 32 55 74 89., No. of comparisions = 3

Pass 4: 21 32 55 74 89., No. of comparisions = 4

Pass 5: 21 32 55 64 74 89., No. of Comparision = 3

↳ Sorted data :

Analysis of Insertion sort:

↳ Best Case: → In Insertion sort, the best case occurs if the array is already sorted.

Pass 1: 1 (comparision)

Pass 2: 1 " Total No. of comparisions

Pass 3: 1 " = (n-1) x)

Pass 4: 1 " = n-1

Pass n-1: 1 " Time complexity = O(n).

Analysis of Insertion sort.

↳ Best case: ~~example~~ example: Already sorted data

ex: → (10) 12 (14) (16) (18) (20)

Pick (10) as a first element of the sorted list

$n = 6$,

Number of passes required = 6 − 1 = 5

Pass 1: 10, 12 , No. of comparisons = 1

Pass 2: 10 | 12, 14 , " " = 1

Pass 3: 10 | 12, 14, 16 " " = 1

Pass 4: 10 | 12, 14 | 16, 18 " " = 1

Pass 5: 10 | 12, 14, 16, 18 | 20 " " = 1.

O/P: 10, 12, 14, 16, 18, 20.

$2+2 = 2\times 2$

~~Sorting → Either decorate the given~~

$2+2+2 = 2\times 3$

↳ **Best case**

No. of data = n

No. of passes required = n − 1

Pass 1 : 1 no. of comparisions.

Pass 2 : 1 "

Pass 3 : 1 "

— — — — — — — — — — — 2n−1

Pass n−2 : 1 " "

Pass n−1 : 1 " "

Total No. of comparisions

$= 1 + 1 + 1 + \cdots + n-1$

$= 1 \times (n-1)$ times

$= n-1$

No. of comparisions $= n-1$

Time complexity $= \underset{n\to\infty}{O(n)}$, n is too large