Insertion Sort

```
Insertion (A)
{
  for (j=2 to A(lenght))
  {
    Key = A[j];
    i = j-1;
    while ( i>0 && A[i] > key]
    {
      A(i+1] = A[i];
      i = i-1;
    }
    A[i+1] = key ;
  }
```

Time Complexity

A = 5 4 3 2

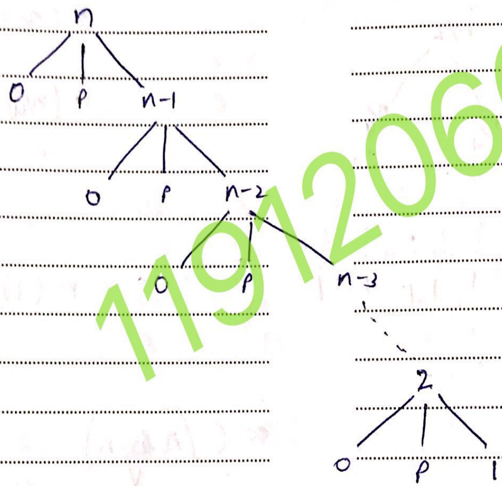
		1	2	3	4
Value of i	0	~~1~~ ~~1~~	~~2~~ ~~2~~	~~3~~	
Element		~~5~~ ~~4~~ ~~3~~ 2	4 ~~5~~ ~~4~~ 3	~~3~~ ~~8~~ 4	~~7~~ 5
j			~~2~~	~~3~~	4
Key			~~4~~	~~3~~	2
No. of comparisons			1	1+1	1+1+1
No. of Movements			1	1+1	1+1+1
			2(1)	2(2)	2(3)

05 sunday

$$\text{Time complexity} = 2\,\frac{n(n-1)}{2} \approx O(n^2)$$

Quick Sort

Worst Case eg (0, 2, 5, 8, 20, 50)
(50, 20, 8, 5, 2, 0)



Time taken

n

$n-1$

$n-2$

$n-3$

$\vdots$

2

0

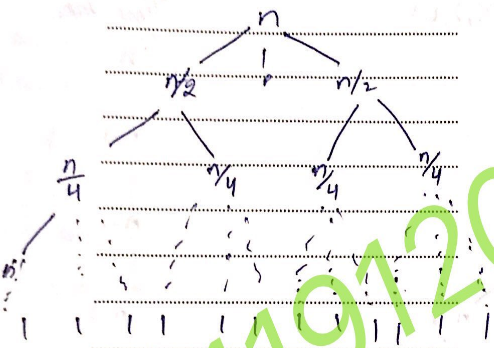
$0 + 2 + 3 + \ldots + n$

$= \frac{n(n+1)}{2}$

$\approx \boxed{O(n^2)}$

Best Case

When middle element is picked as pivot.



Time taken

n

$2(n/2) = n$

$4(n/4) = n$

$n(1) = n$

$\approx O(n \log n)$

It is an in-place sorting algorithm as it takes extra memory only for recursive function calls.

Merge Sort

Eg:-

Sorted Array

Merge Sort

In Worst case

At

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

$$\propto O(n \log n)$$

Merge & Quick

- In average case, both merge sort and quick sort are $O(n \log n)$
- ~~At~~ In worst case

  Merge sort – $O(n \log n)$
  Quick sort – $O(n^2)$
- Space

  Merge sort – $2n$ space required
  Quick sort – n spa requ

Bubble & Insertion sort

- Both have same time complexity for worst case i.e. $O(n^2)$
- Both have same time complexity for best case i.e. $O(n)$
- Bubble sort have constant space complexity.