Data Structures (Theory)

SORTING:

→ Makes searching fast.

Linear Data Structures → Linear Search

Linear Algorithms:

Non Recursive +
- Bubble Sort
- Insertion Sort
- Selection Sort

→ Easy coding

→ We will calculate time complexity of every sorting algorithm.
→ Response time of processor.

There is no best, worst or average case so there is every case. If algo has same time for sorted, unsorted & mixed data.

Non Linear Sort:

1. BST Sort
2. Heap Sort
3. Merge Sort

Max heap tab when we need ascending data
Min heap tab when we need descending order.

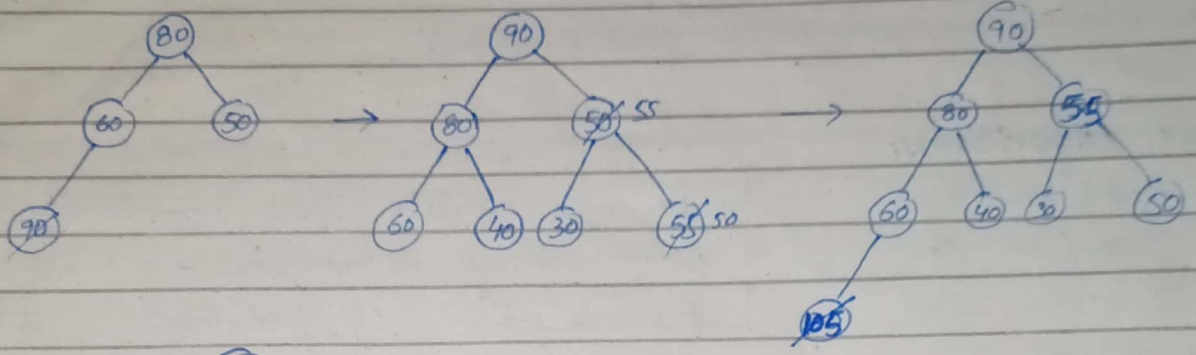
→ Recursive code + difficult coding

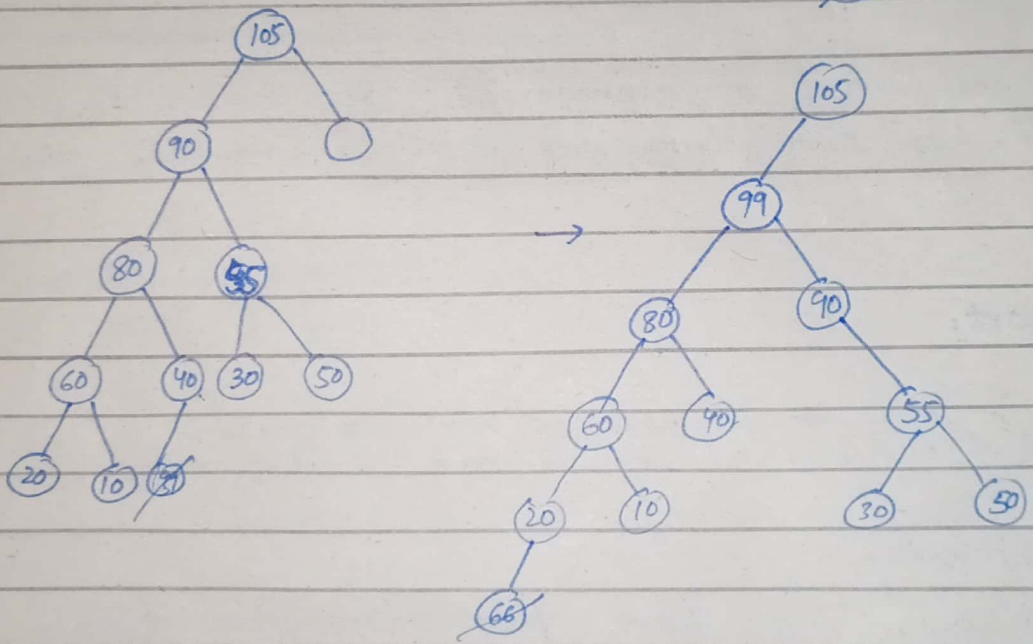
Linear Sorting algorithms have better time complexity than non-linear.

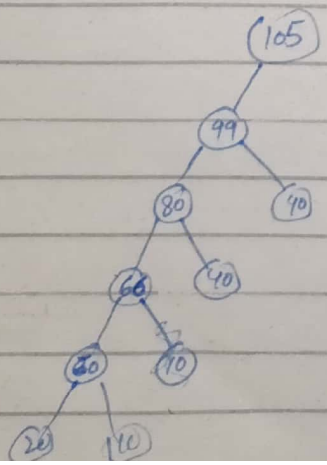
Max Heap Tree:

→ Every root is greater than its children.
→ Left to right node insertion.

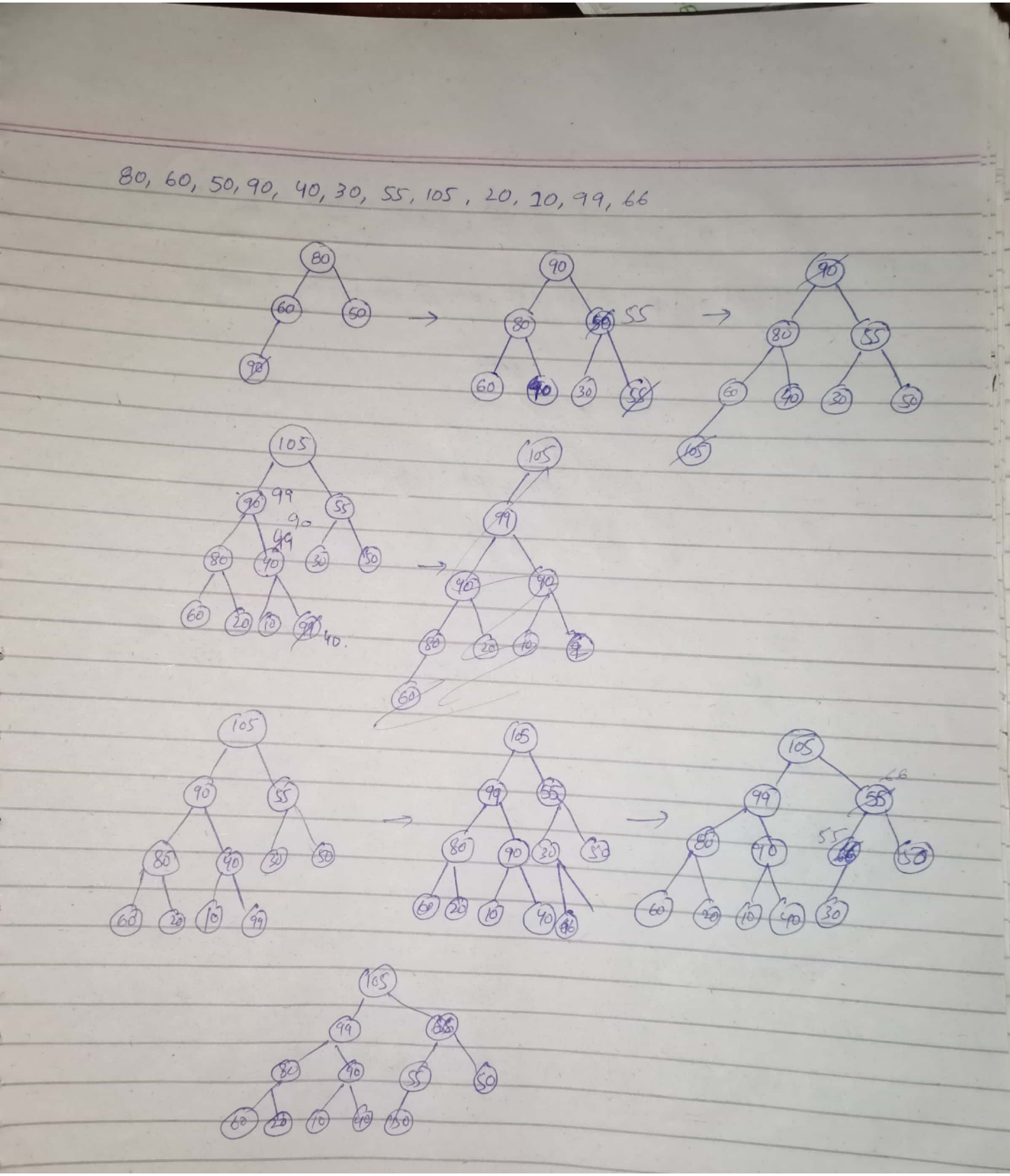
80, 60, 50, 90, 40, 30, 55, 105, 20, 10, 99, 66







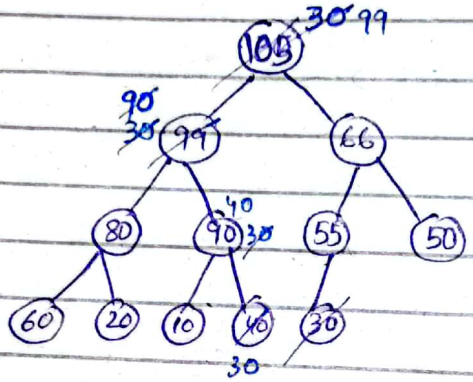
80, 60, 50, 90, 40, 30, 55, 105, 20, 10, 99, 66

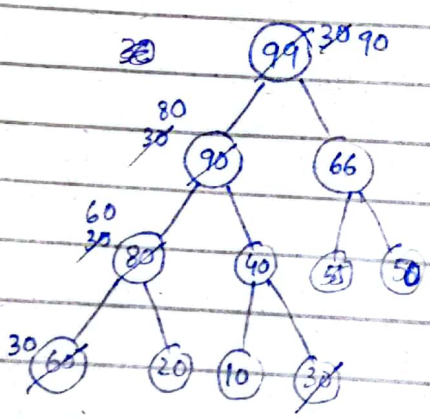


105	99	66	80	90	55	50	60	20	10	40	30

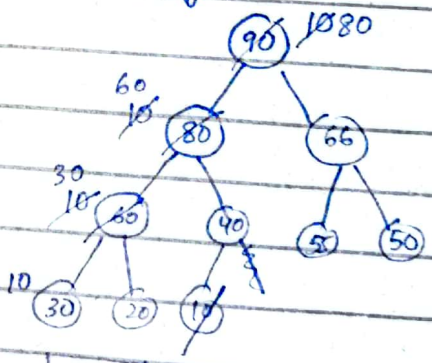
Replace with last node.

30	99	66	80	90	55	50	60	20	10	40	105

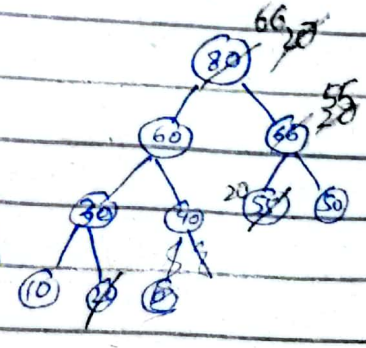




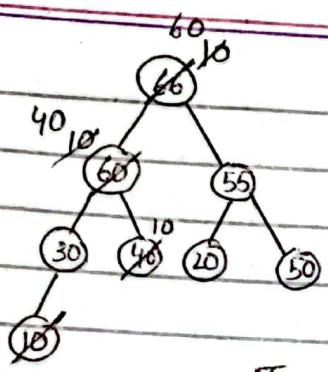
99	90	66	80	40	55	50	60	20	10	30	105



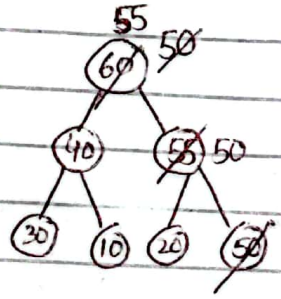
90	80	66	60	40	30	20	10	99	105



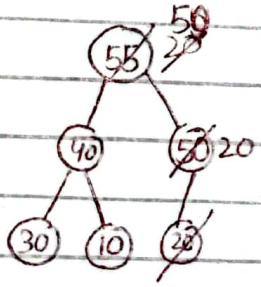
80	60	66	30	40	55	50	10	20	90	99	105



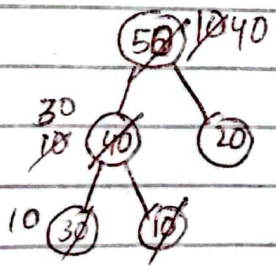
66	60	55	30	40	20	50	10	80	90	99	105



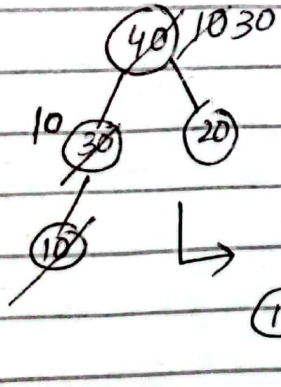
60	40	55	30	10	20	50	66	80	90	99	105



55	40	50	30	10	20	60	66	80	90	99	105

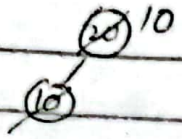


50	40	20	30	10	55	60	66	80	90	99	105



40	30	20	10	50	55	60	66	80	90	99	105

30	10	20	50	50	55	60	66	80	90	99	105

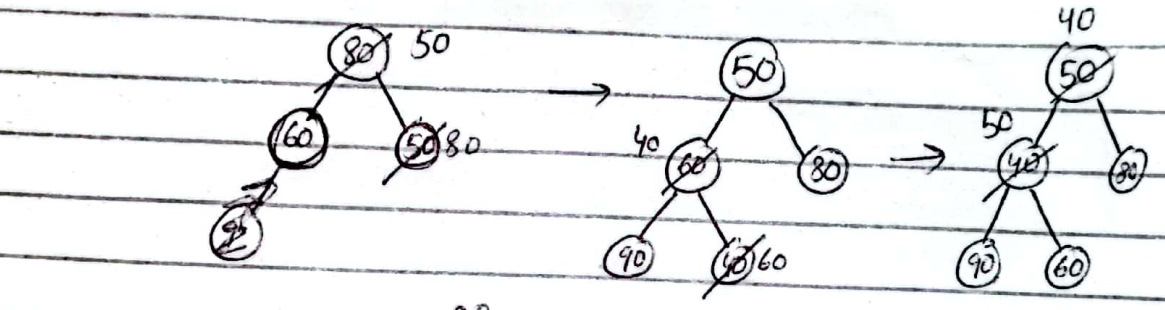


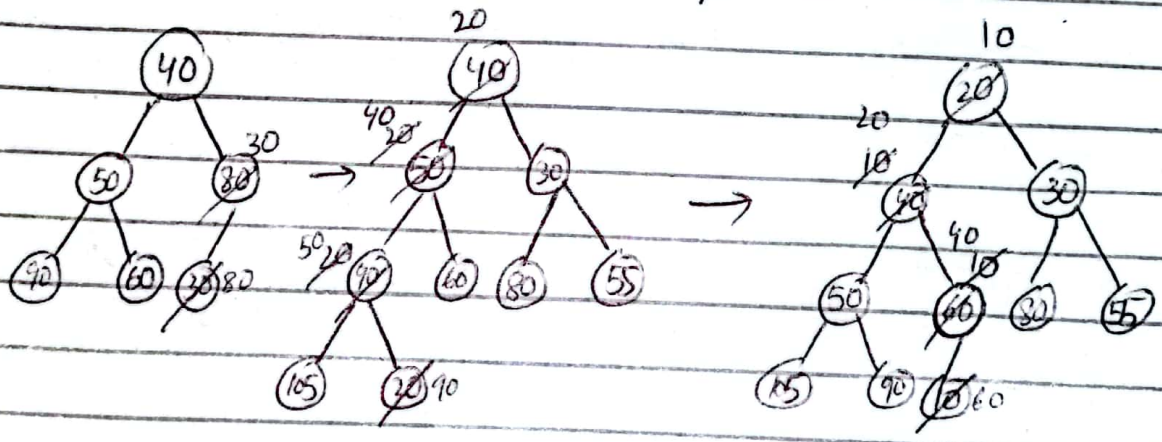
20	10	30	40	50	55	60	66	80	90	99	105

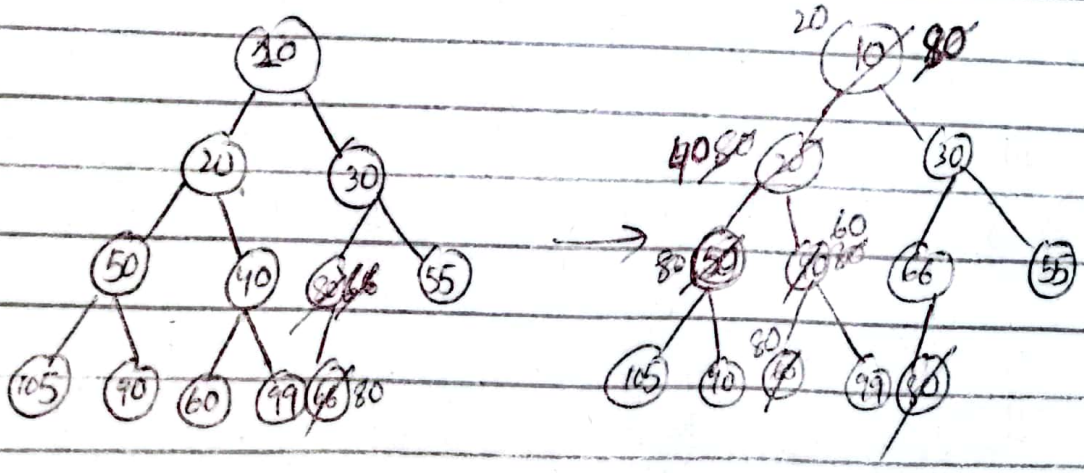
10	20	30	40	50	55	60	66	80	90	99	105

Min heap tree:

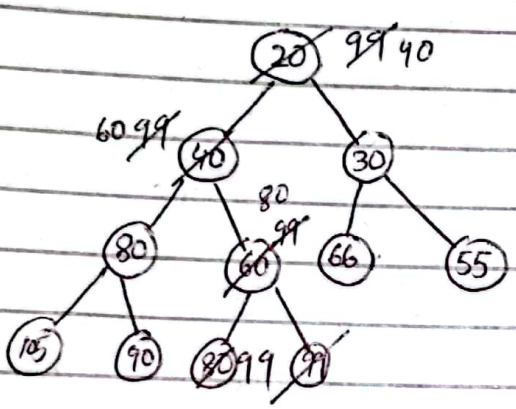
80, 60, 50, 90, 40, 30, 55, 105, 20, 10, 99, 66







10	20	30	50	40	66	55	105	90	60	99	80

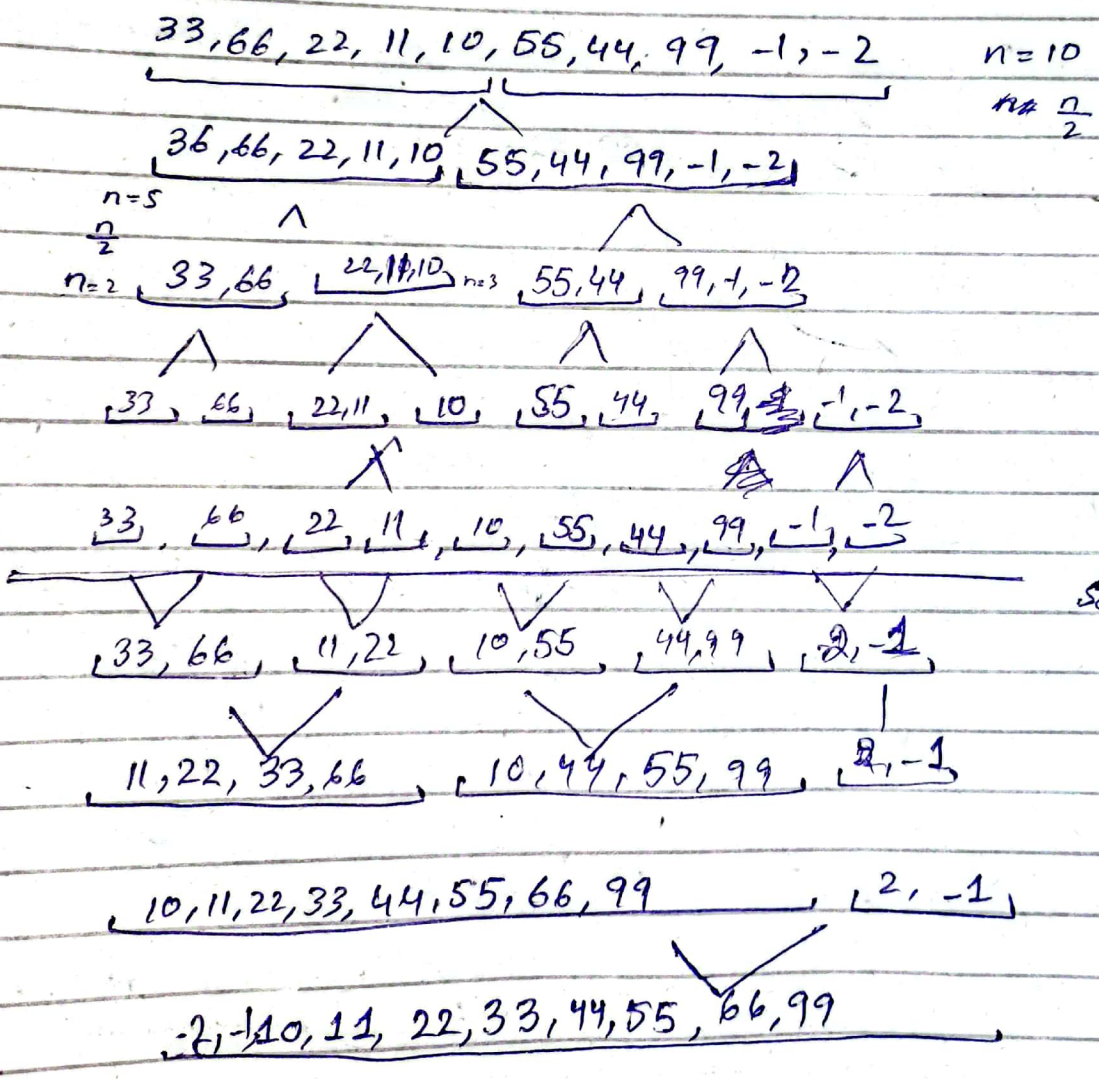


20	40	30	80	60	66	55	105	90	80	99	10

9 ~~July~~ June-2022 Data Structures (Theory) Thursday

★ Non-linear Sorting Algorithms:

1. BST with inorder traversal.
2. Heap sort with Max heap / Min heap.
3. Merge sort



Divide and conquer
Sorting while merging

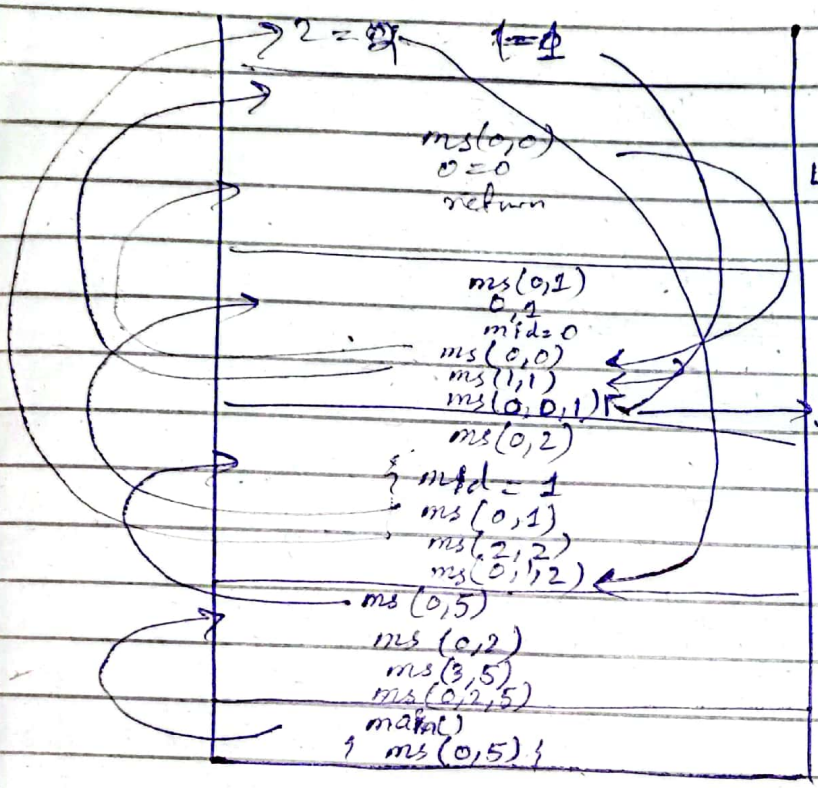
$\frac{5}{2} = 2$

low ↑ high ↑

0 1 2 3 4 5

33, 22, 9, 66, 5, 10

- uniform division
- uniform repeating
- uniform merging



```
merge-sort(int low, int high)
{ int mid,
  if (low != high)
  {
    mid = (low+high)/2
    merge_sort (low, mid);
    merge_sort (mid+1, high);
    merge (low, mid, high);
```

```
merge (int low, int mid,
       int high)
{ int temp[MAX]
  int i = low;
  int j = mid+1;
  int k = low;
  while (i<=mid) && (j<=high) (1<=1)
  {
    if (array[i] <= array[j])
      temp[k++] = array[i++]
    else
      temp[k++] = array[j++]
```

Kisi dono array mn se ek element rah jaye.

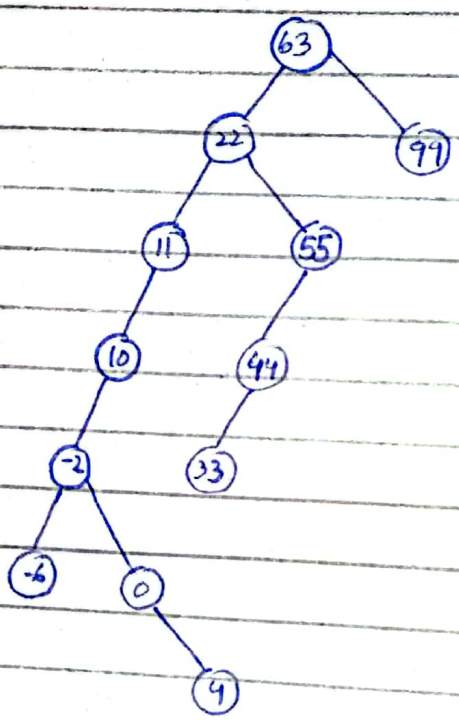
```
while (i<=mid)
  temp[k++] = array[i++]
while (j<=high)
  temp[k++] = array[j++]
for(i=low, i<=high; i++)
  array[i] = temp[i];
}
```

DSA (Theory)

Non linear Sorting Algorithms:

① BST

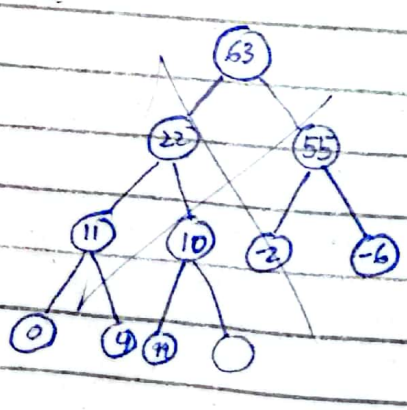
63, 22, 55, 11, 10, -2, -6, 0, 4, 99, 44, 33

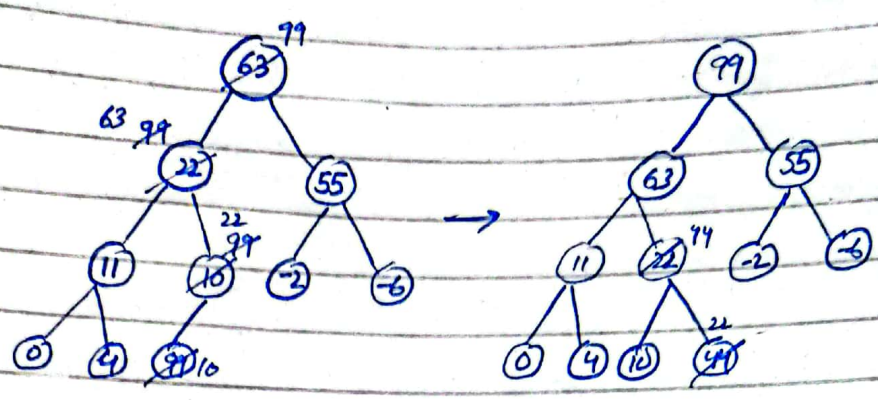


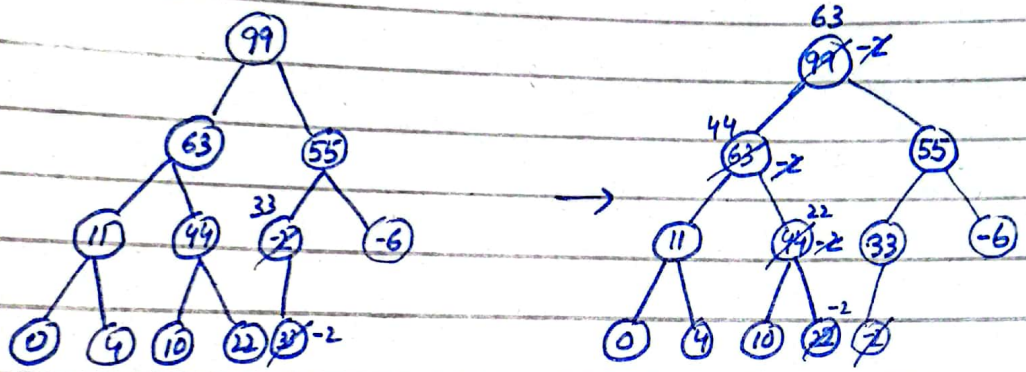
Inorder Traversal:

-6, -2, 0, 4, 10, 11, 22, 33, 44, 55, 63, 99

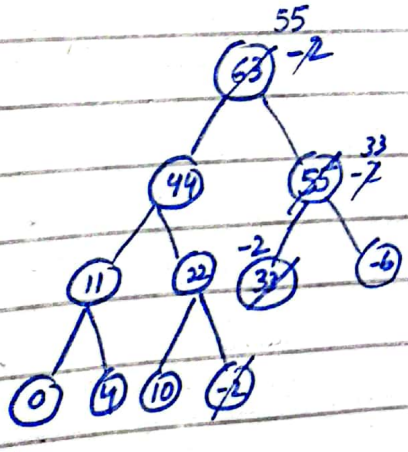
② Heap Sort:



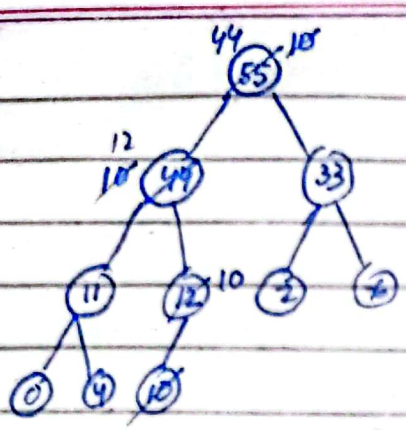




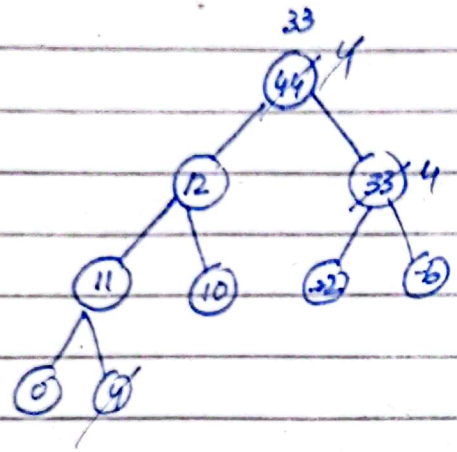
99	63	55	11	44	33	-6	0	4	10	22	-2



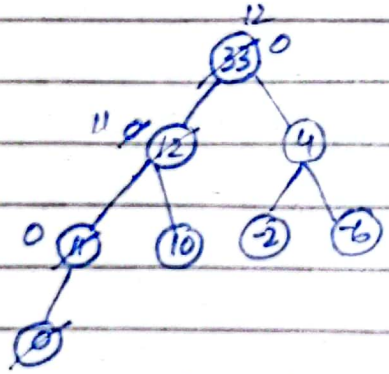
63	44	55	11	22	33	-6	0	4	10	-2	99



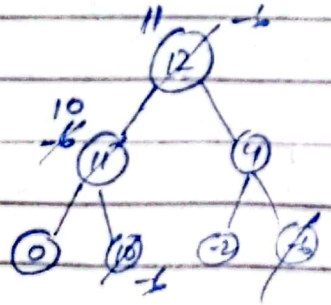
55	44	33	11	12	-2	-6	0	4	10	63	99



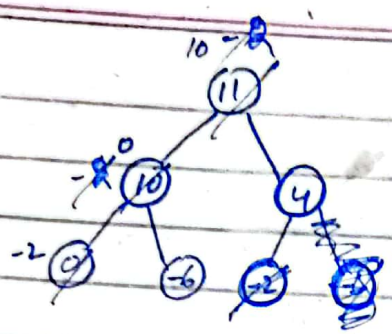
44	12	33	11	10	-2	-6	0	4	55	63	99



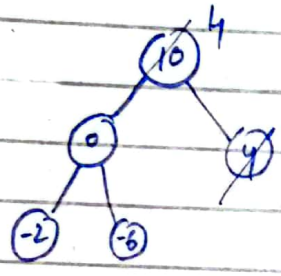
33	12	4	11	10	-2	-6	0	44	55	63	99



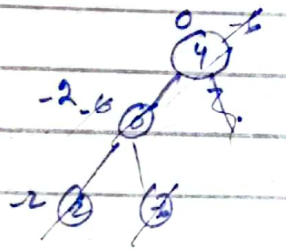
12	11	4	0	10	-2	-6	33	44	55	63	99



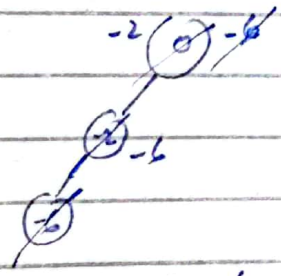
→ | 11 | 10 | 4 | 0 | -6 | -2 | 12 | 33 | 44 | 55 | 63 | 99 |



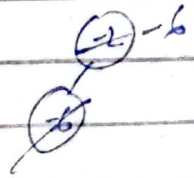
| 10 | 0 | 4 | -2 | -6 | 11 | 12 | 33 | 44 | 55 | 63 | 99 |



| 4 | 0 | -2 | -6 | 11 | 12 | 33 | 44 | 55 | 63 | 99 |



| 0 | -2 | -6 | 4 | 11 | 12 | 33 | 44 | 55 | 63 | 99 |



| -2 | -6 | 0 | 4 | 11 | 12 | 33 | 44 | 55 | 63 | 99 |

| -6 | -2 | 0 | 4 | 11 | 12 | 33 | 44 | 55 | 63 | 99 |

10 June 21

Time Complexity Analysis:

$T(n) =$

- $O(1)$
- $O(\log n)$
- $O(n)$
- $O(n \log n)$
- $O(n^2)$
- $O(n^3)$
- $O(n!)$
- $O(e^n)$

Merge Sort Same steps for all types of data.

division — Merging ↑ for every step

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + n$$

$$\frac{n}{2} + \frac{n}{2} = \frac{2n}{2} = n$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n \quad \text{——} \textcircled{1}$$

Substitution Method: $n = \frac{n}{2}$ in eq ①

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{n}{2}$$

Put eq ② in eq ①

$$T(n) = 2\left[2T\left(\frac{n}{4}\right) + \frac{n}{2}\right] + n$$

$$= 2^2 T\left(\frac{n}{2^2}\right) + \frac{n}{2} + n$$

$$= 2^2 T\left(\frac{n}{2^2}\right) + 2n$$

Subtituting $n = n/4$ in eq ①

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \frac{n}{4} \Rightarrow \textcircled{4}$$

① Best Case $B(n)$
② Worst Case $\Omega(n)$ → sara ASC ya DSC
③ Average Case $A(n)$
④ Every Case $T(n)$ → mixed data

Put eq ④ in eq ③

$$T(n) = 2^2\left[2T\left(\frac{n}{8}\right) + \frac{n}{4}\right] + n$$

$$= 2^3 T\left(\frac{n}{2^3}\right) + 3n$$

$$T(n) = 2^4 T\left(\frac{n}{2^4}\right) + 4n$$

$$T(n) = 2^i T\left(\frac{n}{2^i}\right) + in \longrightarrow \text{General Term.}$$

$T(1) = 1 \longrightarrow$ One cycle for one one.

$$\frac{n}{2^i} = 1$$

$$n = 2^i$$

$$\log_2 n = \log_2 2^i$$

$$\log_2 n = 2 \log_2 i$$

$$\log_2 n = i$$

$$T(n) = n\,T(1) + n\log_2 n$$

$$= n + n\log_2 n$$

const ko ignore kr dete hai $= n(\log_2 n)$

$$T(n) = O(n \log n)$$

6 5 4 3 2 1 $n=6$

$\frac{n}{2} + \frac{n}{2} + n$

$\frac{6}{2} + \frac{6}{2} + 6$

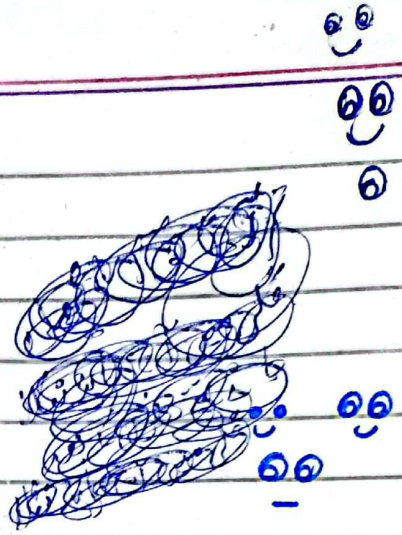
Merge sort is everycase algorithm.

$O(n \log n)$

n: No of elements

→ A good time complexity

Non-linear data structure



Sorting → Comparison + Interchage

Heap Sort:

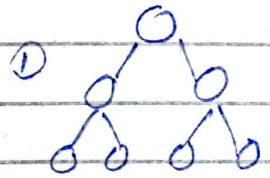
Reason:

Bcz heap sort is tree ~~BST~~

$O(\log_2 n)$

$O(n \log_2 n)$ → bcz sorting.

BST



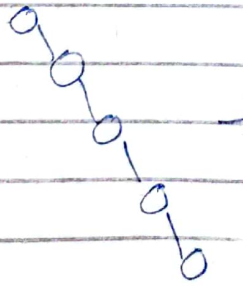
① → Not unbalanced → hr level pr node present

↳ So its best case

T

③ Adha idhr adha udhar :D

Average case



② → unbalanced

↳ worst case

Comparison.

$W(n) = 1+2+3+4+5+6+\ldots\ldots+n$

$T(n) = 2^i$

$= n \log_2 n$

$B(n) \in O(n \log_2 n)$

$$W(n) = 1+2+3+4+5+6+\cdots+n$$

$$= \frac{n}{2}\left[2n + (n-1)d\right]$$

$$= \frac{n}{2}\left[2\times 1 + (n-1)1\right]$$

$$= \frac{n}{2}\left[2+n-1\right]$$

$$= \frac{n(n-1)}{2} = \frac{n^2-n}{2} \Rightarrow \frac{n^2}{2} - \frac{n}{2}$$

Highest term which is in n → constant term ignore

$$= \frac{n^2}{2}$$

→ $O(n^2)$ worst case

14 June 2022 — DSA (Theory) — Tuesday

BST sort
insert
find
inorder

→ Non-linear Sorting Algorithms:
* Merge sort
* Binary Tree sort
* Heap sort

Linear Sorting Algorithms:
① Bubble Sort

Loop for Passes

```
for (i=1; i<=n-1; i++)
{  for (j=1; j<=n-i, j++)   → for comparison
   {  if a[j] (<) a[j+i]
```

less than for descending order

(n-2) (n-1) 65, 22, 33, 11, 20, -1, 0, 21, 14, 15 n=9

(n-1) → Passes → Access

8 comparisons.

Pass 1	8	22, 65 33, 65 11, 65 20, 65 -1, 65 0, 65 21, 65 14, 65 15
(n-2) Pass 2	7	22, 33 11, 33 20, 33 -1, 33 0, 33 21, 33 14, 33, 65
(n-3) Pass 3	6	22 11, 22 20, 22 -1, 22 0, 22 21, 22 14, 22, 33, 65
(n-4) Pass 4	5	11, 20 -1, 20 0, 20 21, 20 14, 21, 22, 33, 65
(n-5) Pass 5	4	-1, 11 0, 11, 20 14, 20, 21, 22, 33, 65
(n-6) Pass 6	3	-1, 0, 11, 14, 20, 21, 22, 33, 65
(n-7) Pass 7	2	-1, 0, 11, 14, 20, 21, 22, 33, 65
(n-8) Pass 8	1	-1, 0, 11, 14, 20, 21, 22, 33, 65

comparisons

Jitni times data par jana hoga.

Comparison is greatest no of times operation.

cutting kar k likhna hai or phir neat kar k likhna hai.

Pass 1	65 22, 65 33, 65 11, 65 20, 65 -1, 65 0, 65 21, 65 21, 65 14, 65 15, 65
Pass 2	22, 33, 11, 20, -1, 0, 21, 14, 15 | 65
	22, 33, 11, 33 20, 33 -1, 33 0, 33 21, 33 14, 33 15, | 33, 65
Pass 3	22, 11, 20, -1, 0, 21, 14, 15 | 33, 65
	22 11, 22 20, 22 -1, 22 0, 22 21, 22 14, 22 15, 22, 33, 65
Pass 4	11, 20, -1, 0, 21, 14, 15, | 22, 33, 65
	11, 20 -1, 20 0, 20, 21 14, 21 14, 21 15, | 21, 22, 33, 65
Pass 5	11, -1, 0, 20, 14, 15 | 21, 22, 33 65
Pass 6	
Pass 7	
Pass 8	

There are 3 cases of time complexity of Bubble sort

→ Best Case (Already Sorted) (Pehly Pass mn hi sorted data)

→ Worst Case (Totally different (unsorted) (Last pass mn sort ho)

→ Average Case (Data sorted in any middle Pass)
↳ Rest of Passes nhe krny chyen)

Processor ko frq nhi pata isliye jo zada no of times operation perform hota hai to basic operation usy longer to calculate T(n) Bqi operations km times hee longer to find, nhi pata

```
for (i=1; i<=n-1; i++)
{   xchanges=0;
    for (j=1; j<=n-2; j++)
    {
      if a[j] > a[j+1]
        interchange;
        xchanges++;
    }
    if xchanges == 0
      break;
}
```

For Passes.

If in 1st Pass xchanges == 0 → Best case.

→ Maxxed Passes nhi chyen.

Passes + Comparison + Xchanges

In Best Case:

1 time outer loop (Passes) and inner loop (Comparison) executes so

Greatest no of operations

Time complexity $1 + (n-1) = T(n) \in O(n)$ ~~$\Omega(\log n)$~~

Worst Case:

$$T(n) = (n-1)+(n-2)+(n-3)+\ldots+1$$

$$= 1+2+3+\ldots+(n-1)$$

First term ← 1 ; ↳ highest term

$$= \frac{n-1}{2}\left[2a+(n-1)d\right]$$

$$= \frac{n-1}{2}\left[2\times 1+(n-1-1)\right]$$

↳ differences

$= \frac{n-1}{2}[2 + n - 2]$

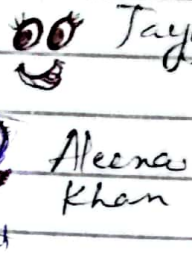
$= \frac{n}{2}(n-1)$

$= \frac{n^2}{2} - \frac{n}{2}$ constt term $= \frac{n^2}{2} \in O(n^2)$

Average Case

↳ Also belongs to $O(n^2)$

$n-1$ comparisons



Selection Sort: Every case

Every comparison

r 33, 22, 5, 6, -1, 36, 42, 55, 44. $n = 9$ $(n-1)$ Passes

$n-1$	Pass 1	[-1], 22, 5, 6, [33], 36, 42, 55, 44
$n-2$		-1, [5], [22], 6, 33, 36, 42, 55, 44
$n-3$		-1, 5, [6], [22], 33, 36, 42, 55, 44
$n-4$		-1, 5, 6, 22, 33, 36, 42, 55, 44
$n-5$		-1, 5, 6, 22, 33, 36, 42, 55, 44
$n-6$		-1, 5, 6, 22, 33, 36, 42, 55, 44
$n-7$		-1, 5, 6, 22, 33, 36, 42, 55, 44
$n-8$		-1, 5, 6, 22, 33, 36, 42, [44], [55]

1 Pass.

```
for(i = 1; i<=n-1; i++)
{
    smallest = a[i]
    for(j = i+1 ; j<=n ; j++)
    { if (a[j]< smallest)
        smallest = a[j]
        interchange a[j] and smallest;
    }
}
```

Selection Sort:

Everycase (a ki sort ho bhi jaye to say passes perform krega)

Same as worst case of bubble sort ki time complexity
Comparison is greatest no of times operation performed

③ **Insertion Sort:**

33, 66, 22, 11, -5, 60, 77, 88, 44. ($n=9$, $(n-1)$ passes)

1 Passes ($i=1$) | 33 (a_0) | 66, 22, 11, -5, 60, 77, 88, 44.

Pass 2 | 33 (a_0), 66 (a_1) | , 22, 11, -5, 60, 77, 88, 44 — right se compare krna hai

Pass 3 | 33, ~~66~~ 22, 66 |, 11, -5, 60, 77, 88, 44

| 22, 33, 66 | 11, -5, 60, 77, 88, 44 .

Sorting Modified

Sortide Modified pdf

Pass 4	22, 33, 11, 66 | 60, 77, 88, 44 22, 11, 33, 66 | -5, 60, 77, 88, 44 11, 22, 33, 66 | -5, 60, 77, 88, 44 11, 22, 33, 66 | 60, 77, 88, 44
Pass 5	11, 22, 33, 60, 66 | 77, 88, 44 11, 22, 33, 60, 66 | 77, 88, 44
Pass 6	11, 22, 33, 60, 66, 77 | 88, 44
Pass 7	11, 22, 33, 60, 66, 77, 88 | 44
Pass 8	11, 22, 33, 60, 66, 77, 44 | 88 11, 22, 33, 60, 66, 44 | 77, 88 11, 22, 33, 60, 44 | 66, 77, 88 11, 22, 33, 44 | 60, 66, 77, 88 11, 22, 33, 44, 60, 66, 77, 88

```
for (i=1; i<=n-1; i++)
{   smallest = a[i];
    j = i+1
    while (a[j] >= 0 && a[j] < smallest)
    interchange smallest with a[j];
}
```

data already sorted rai to loop nhi chly ga.

Worst case $O(n^2)$ Average case also $O(n^2)$.

Comparisons = $1+2+3+4+\ldots\ldots+n-1$.

```
void Bubble_sort (node **start, int count)
{
    struct node **h;
    int i,j, xchange;

    for (i=0; i<=count ; i++)
    {
        *h = start;
        xchange =0
    for (j=0; j<=count-i-1 ; j++)
    {
        p1 = *h;
        p2 = p1->link;
        if (p1->info > p2->info)
        { *h = swap(p1, p2);
            xchange = 1;
        }
        h = &(*h)->link;
    }
    if(xchange =0)
        break;
    }
}
```

```
swap( node ptr1, ptr2)
{ struct node temp;
  temp = ptr2->link;
  ptr2->link = ptr1;
  ptr1->link = temp
  return ptr2;
}
```