Lecture - 6

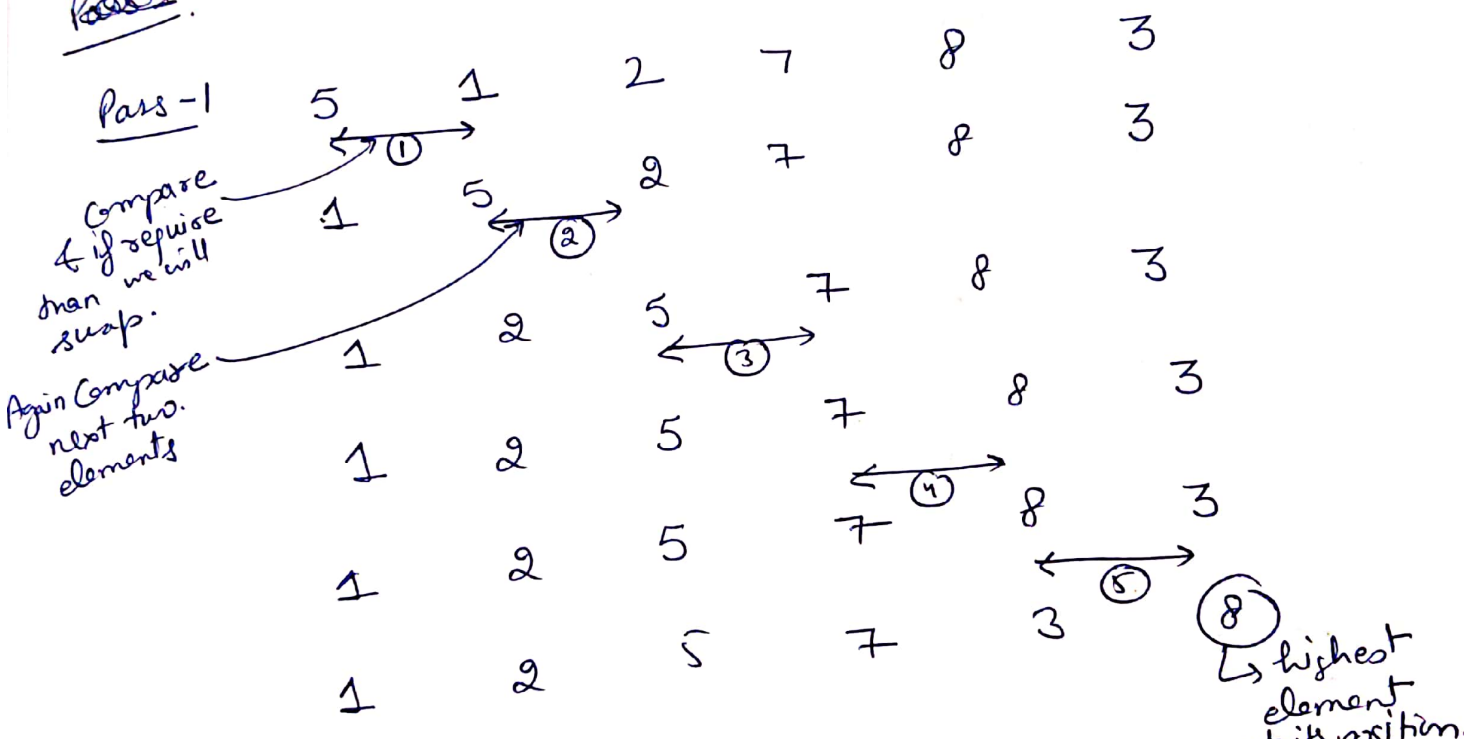
Bubble Sort

Consider a given unsorted array

5, 1, 2, 7, 8, 3

- Observations :-
- If we have n elements in an array then Bubble sort will take n-1 passes to sort the given array.
- After every pass the highest element in the array will be fitted to its position.
- Since in given array we have 6 elements so it will take 6-1 ⇒ 5 passes.

Pass-1

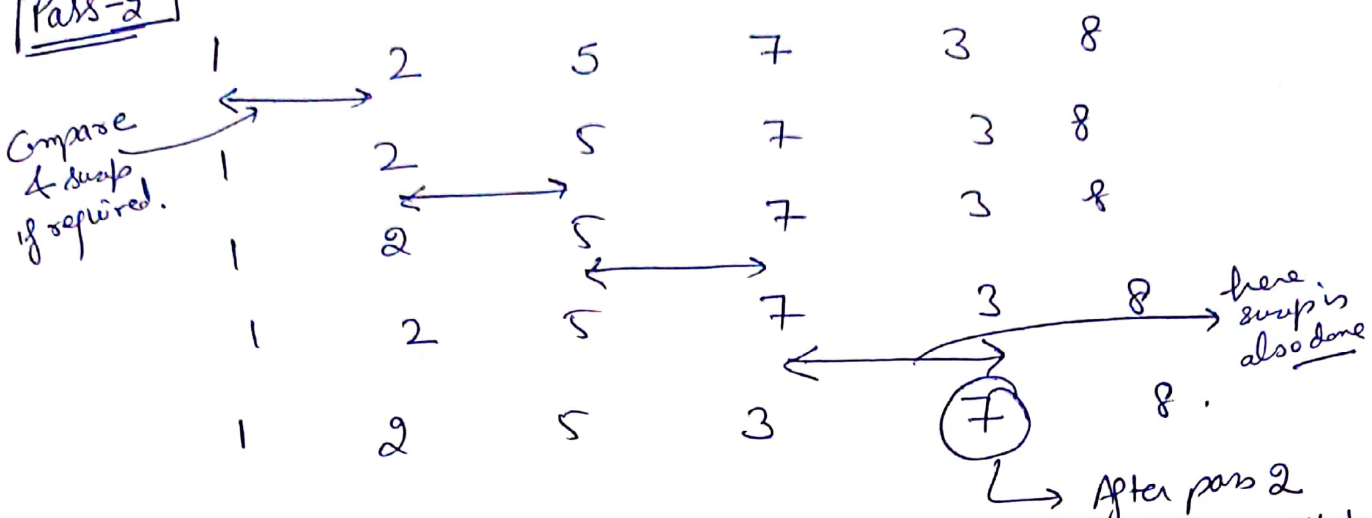


Since we have 6 elements then in Pass-1 we have to make 6-1 Comparison i.e 5 Comparison. to fit the largest element at its position.

i.e for n element we had made n-1 Comparison in pass-1

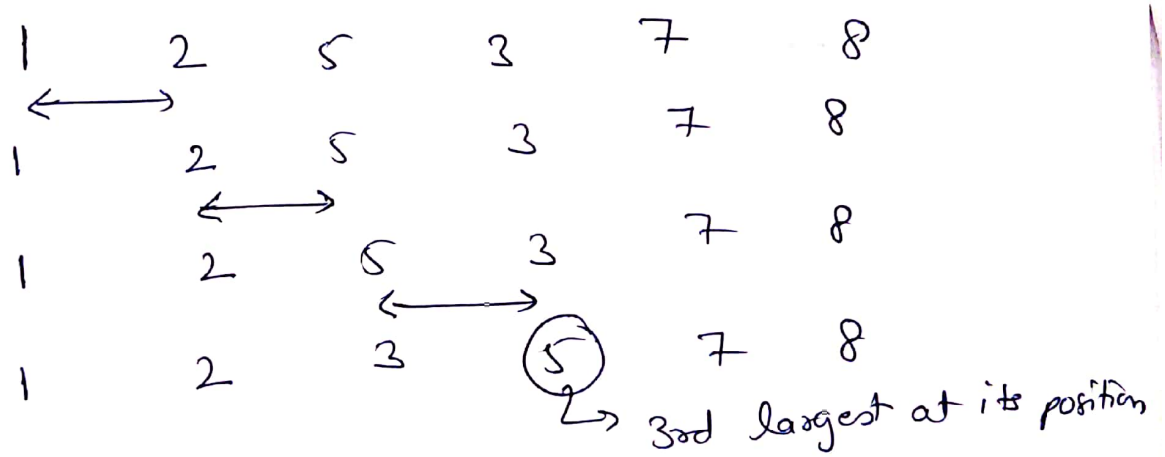
Scanned by CamScanner

Pass-2



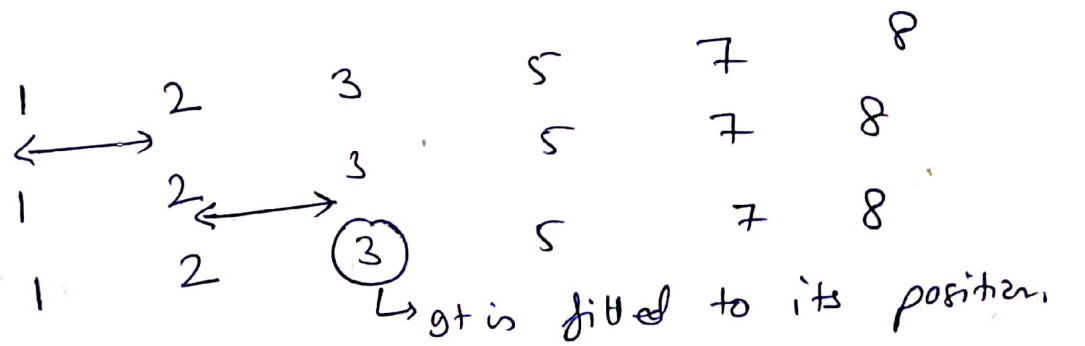
Since in Pass-2 we had made 4 comparison. So we can say that in Pass-2 we had made n-2 comparison.

Pass-3



ie We had made n-3 comparison.

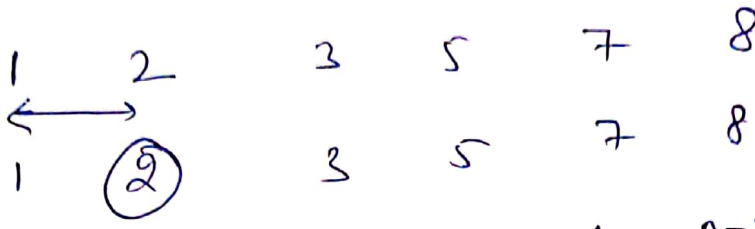
Pass-4



ie We had made n-4 comparison.

Scanned by CamScanner

Pass-5 :



ie in last pass we had made n-5 comparison.

ie only 1 comparison.

This is how Bubble sort work

Algorithm Bubble (Data, N) → No. of elements.
↳ Name of the array

1. Repeat step 2 & 3 for K = 1 to N-1 || outer loop for the no. of passes.

2. Set ptr = 1 || initialize the pass pointer ptr.

3. Repeat while ptr ≤ N-k || inner loop for executing the pass.

(a) if Data[ptr] > Data[ptr+1], then
interchange Data[ptr] and Data[ptr+1].
End of if.

(b) Set ptr = ptr+1

End of outer loop at step 1

4. Exit

Complexity :—

$$\underbrace{n-1}_{\text{no. of Comparison in 1st pass}} + \underbrace{n-2}_{\text{no. of comparison in 2nd pass}} + n-3 + \text{-----} + \underbrace{1}_{\text{no. of comparism in the lastpass}}$$

Scanned by CamScanner

Writing the previous eqn in reverse order.

$1 + 2 + \cdots\cdots n-3 + n-2 + n-1$

Sum of Natural no.

is $\frac{n(n+1)}{2}$. But here n is $n-1$.

So $\frac{(n-1)(n-1+1)}{2} \Rightarrow \frac{n^2}{2} - \frac{n}{2} \simeq O(n^2)$.

When list is already sorted still we are going for $n-1$ passes. So we can use flag (1-bit variable) or a counter to signal when no interchange take place during a pass then it means that the list is already sorted. So it will cut down the no. of passes.

Modified Bubble Sort (Data, N, flag = 0)

1. Repeat 2 & 3 (for k = 1 to N-1 && flag == 0)

2. set flag = 1 & ptr = 1

3. Repeat while ptr ≤ N-k
   (a) if Data[ptr] > Data[ptr+1]
   then interchange
   & set flag = 0;
   (b) ptr = ptr + 1

4. Exit.

new condition is added. to reduce the no. of passes if the list is already sorted

Scanned by CamScanner

Note :- Use of this 1-bit variable flag is efficient only when the list originally is almost in sorted order.

- It will not effect the time complexity of Bubble sort. Because no. of passes can reduce only when you consider an array which is almost in a sorted order otherwise we have to go upto n-1 passes to sort a given array while using Bubble sort.

So $\simeq O(n^2)$.

Self Assessment Question

1. Using Bubble sort find the no. of comparison & no. of interchanges which alphabetize the letters in PEOPLE.

Scanned by CamScanner