Data Structure and Algorithms

Session-19

Dr. Subhra Rani Patra
SCOPE, VIT Chennai

What will we learn in this Topic ?

- ✓ What is sorting
- ✓ Types of Sorting
- ✓ Sorting Terminologies
- ✓ Why learn so many sorting techniques ?
- ✓ Sorting Algorithms –
  - ✓ Bubble
  - ✓ Selection
  - ✓ Insertion
  - ✓ Bucket
  - ✓ Merge
  - ✓ Quick
  - ✓ Heap
- ✓ Comparison of all types of sorting techniques

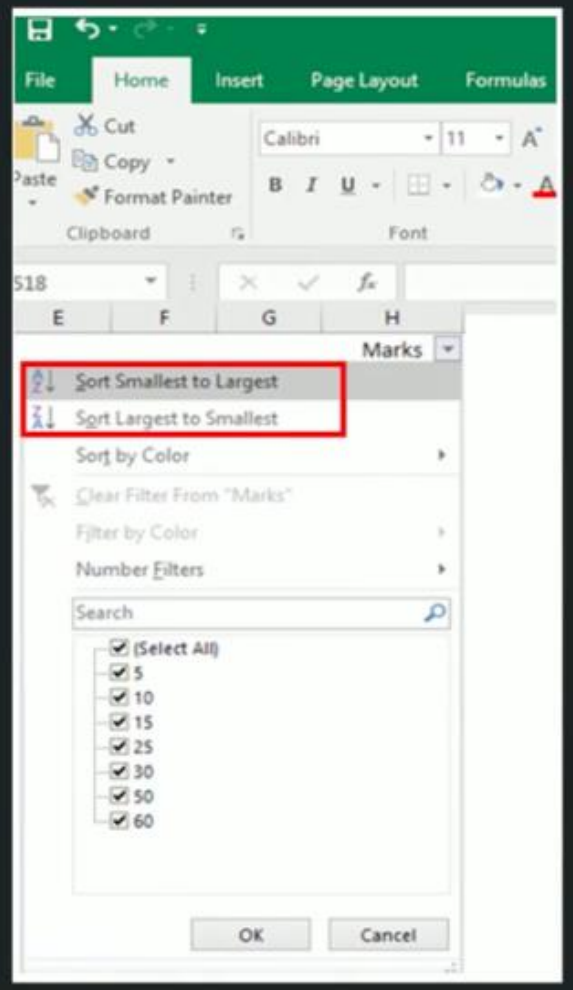
What is Sorting ?

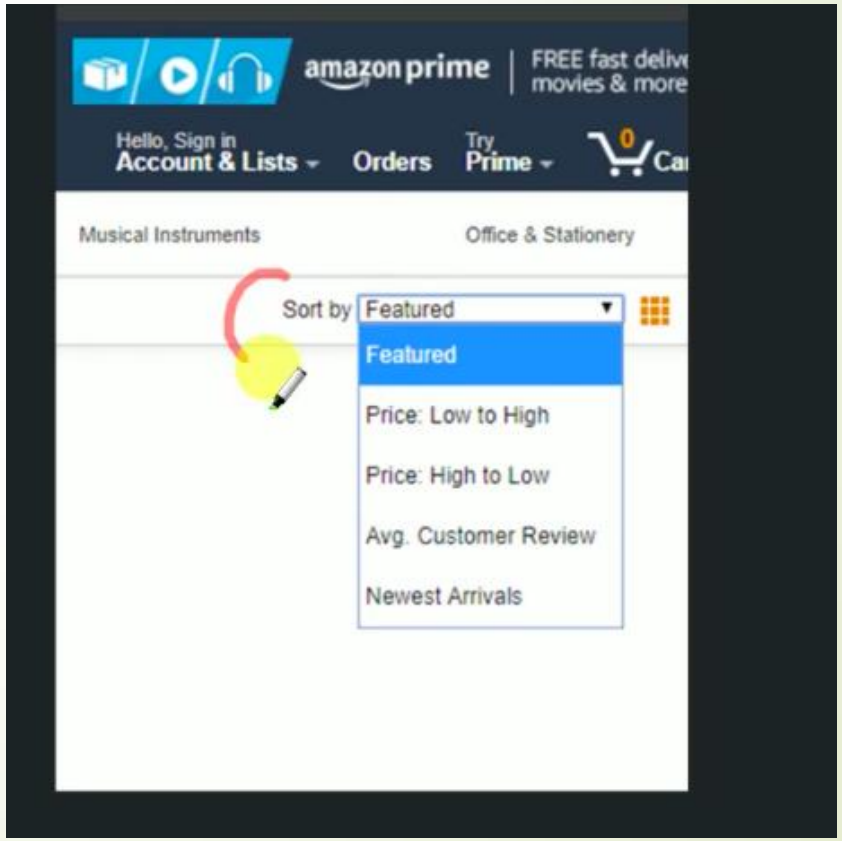
✓ **Definition:** *Sorting refers to arranging data in a particular format: either ascending or descending.*

Practical uses of Sorting:

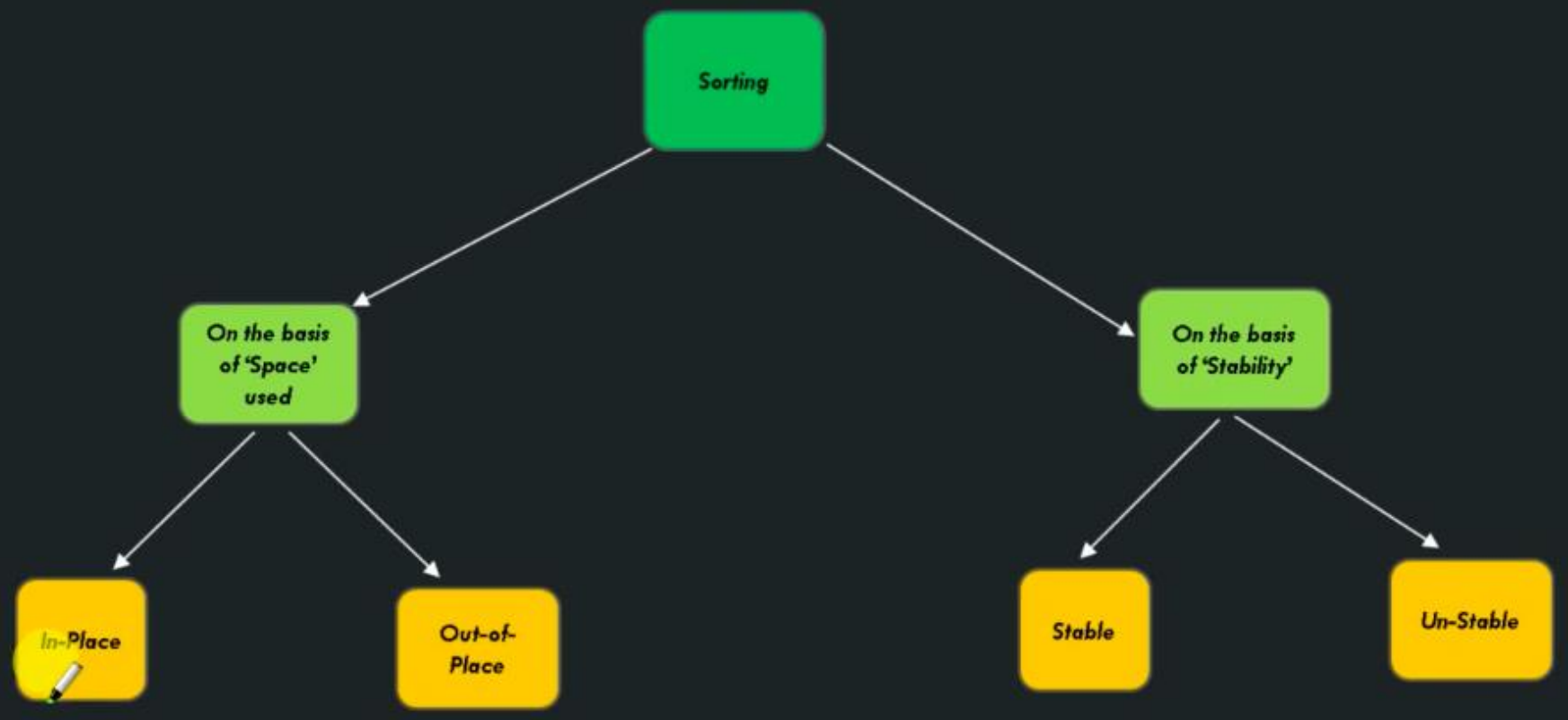
✓ **Microsoft excel** – Has an inbuilt functionality to sort data.

✓ **Online Stores** - online stores generally have option of sorting search results by Price, Review, Ratings, etc.





Types of Sorting:



In-Place vs Out-Place Sorting:

✓ **In-Place Sort:**

- ✓ Sorting algorithms which does not require any extra space for sorting.
- ✓ Example - Bubble sort

30	10	40	50	70	60	20	80

10	20	30	40	50	60	70	80

✓ **Out-Place Sort:**

- ✓ Sorting algorithms which requires extra space for sorting.
- ✓ Example - Merge sort

30	10	40	50	70	60	20	80

10	20	30	40	50	60	70	80

Stable vs Unstable Sorting:

- ✓ **Stable Sort:**
  - ✓ If a Sorting algorithm after sorting the contents does not change the sequence of similar content in which they appear, is called Stable sorting.
  - ✓ Example – Insertion sort

30	10	40	50	70	50	20	80

10	20	30	40	50	50	70	80

- ✓ **UnStable Sort:**
  - ✓ If a sorting algorithm after sorting the contents, changes the sequence of similar content in which they appear, it is called unstable sort
  - ✓ Example – Quick Sort

30	10	40	50	70	50	20	80

10	20	30	40	50	50	70	80

Why 'Stable Sort' is important ?

- ✓ Scenarios where 'sort key' is not the entire identity of the item.
- ✓ Consider a person object with a name and a Age. Let's say we sorted based on their name. If we were to then sort by age in a stable way, we'd guarantee that our original ordering would be preserved for people with the same age.
- ✓ 'group by' clause of Database uses this concept very heavily.

UnSorted Data	
Name	Age
Reena	1
Nalini	2
Reshma	2
Preeti	1
Sita	1

Sorted by Name	
Name	Age
Nalini	2
Preeti	1
Reena	1
Reshma	2
Sita	1

Sorted by Age (Stable)	
Name	Age
Preeti	1
Reena	1
Sita	1
Nalini	2
Reshma	2

Sorted by Age (UnStable)	
Name	Age
Preeti	1
Sita	1
Reena	1
Nalini	2
Reshma	2

Few Terminologies:

- ✓ **Increasing Order:**
  - ✓ If successive element is greater than the previous one.
  - ✓ Example: 1, 3, 4, 6, 8, 9.
- ✓ **Decreasing Order:**
  - ✓ If successive element is less than the current one.
  - ✓ Example, 9, 8, 6, 4, 3, 1.
- ✓ **Non-Increasing Order:**
  - ✓ If successive element is less than or equal to its previous element in the sequence. This order occurs when the sequence contains duplicate values.
  - ✓ Example, 9, 8, 6, 3, 3, 1.
- ✓ **Non-Decreasing Order:**
  - ✓ If the successive element is greater than or equal to its previous element in the sequence. This order occurs when the sequence contains duplicate values.
  - ✓ Example: 1, 3, 3, 6, 8, 9.

Why should we read so many Sorting techniques ?

- ✓ Every Sorting techniques comes with its set of Pros and Cons. So we need to use specific sorting technique as per the situation.
  - ✓ Do we have special requirement of 'Stability" ?
  - ✓ Is 'Space' Priority to us?
  - ✓ Is 'Time' priority to us ?

What is Bubble Sort:

- ✓Bubble sort, sometimes is also referred as Sinking sort
- ✓Repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order.

30	10	50	20	60	40

```
bubbleSort (int arr[])
    int n = arr.length;
    for (int i = 0; i < n - 1; i++) //run from first cell to last cell
        for (int j = 0; j < n - i - 1; j++) //run from first cell to "last cell - iteration"
            if (arr[j] > arr[j + 1]) {
                swap(arr[j], arr[j+1])
```

Time & Space Complexity of Bubble Sort Algorithm:

```
bubbleSort (int arr[])
    int n = arr.length ------------------------------------------------------ O(1)
    for (int i = 0; i < n - 1; i++) ------------------------------------------ O(n)
        for (int j = 0; j < n - i - 1; j++) ---------------------------------- O(n)
            if (arr[j] > arr[j + 1]) ------------------------------------------ O(1)
                swap(arr[j], arr[j+1]) -------------------------------------- O(1)
```

Time Complexity – $O(n^2)$

Space Complexity – $O(1)$

When to Use/Avoid Bubble Sort:

- ✓ **When to use:**
  - ✓ When input is already sorted
  - ✓ Space is a concern
  - ✓ Easy to implement

- ✓ **When not to use:**
  - ✓ Average case time complexity is poor

Selection Sort:

✓*The Selection sort algorithm is based on the idea of finding the minimum or maximum element in an unsorted array and then putting it in its correct position in a sorted array.*

30	10	50	20	60	40

Selection Sort Algorithm:

```
SelectionSort(A):
    loop: j = 0 to n-1
        int iMin = j;
        loop: i = j+1 to n-1
            if (a[i] < a[iMin])
                iMin = i
        if (iMin != j)
            swap(a[j], a[iMin]);
```

30	10	50	20	60	40

Time & Space Complexity of Selection Sort Algorithm:

```
SelectionSort(A):
    loop: j = 0 to n-1 ------------------------------------------------------ O(n)
        int iMin = j; ------------------------------ O(1)
        loop: i = j+1 to n-1 --------------------- O(n)
            if (a[i] < a[iMin]) ---------------- O(1)      ------------- O(n)      ---------------- O(n^2)
            iMin = i -------------------------- O(1)
        if (iMin != j) ----------------------------- O(1)
        swap(a[j], a[iMin]); --------------------- O(1)
```

Time Complexity – $O(n^2)$

Space Complexity – $O(1)$

When to Use/Avoid Selection Sort:

- ✓ **When to use:**
  - ✓ When we don't have additional memory
  - ✓ Want easy implementation

- ✓ **When not to use:**
  - ✓ When time complexity is a concern


Thank you