Data Structure
“Sorting”

Submitted by:
Richa Shukla

Sorting

- Sorting is nothing but arranging the data in ascending or descending order.
- There are so many things in our real life that we need to search for, like a particular record in database, roll numbers in merit list, a particular telephone number in telephone directory, a particular page in a book etc. All this would have been a mess if the data was kept unordered and unsorted, but fortunately the concept of **sorting** came into existence, making it easier for everyone to arrange data in an order, hence making it easier to search.
- **Sorting** arranges data in a sequence which makes searching easier.

- Sorting technique depends on the situation. It depends on two parameters.

1. Execution time of program that means time taken for execution of program.
2. Space that means space taken by the program.

- **Sorting can be performed using several methods :**

  1. Bubble Sort
  2. Selection Sort
  3. Insertion Sort
  4. Quick Sort
  5. Heap Sort

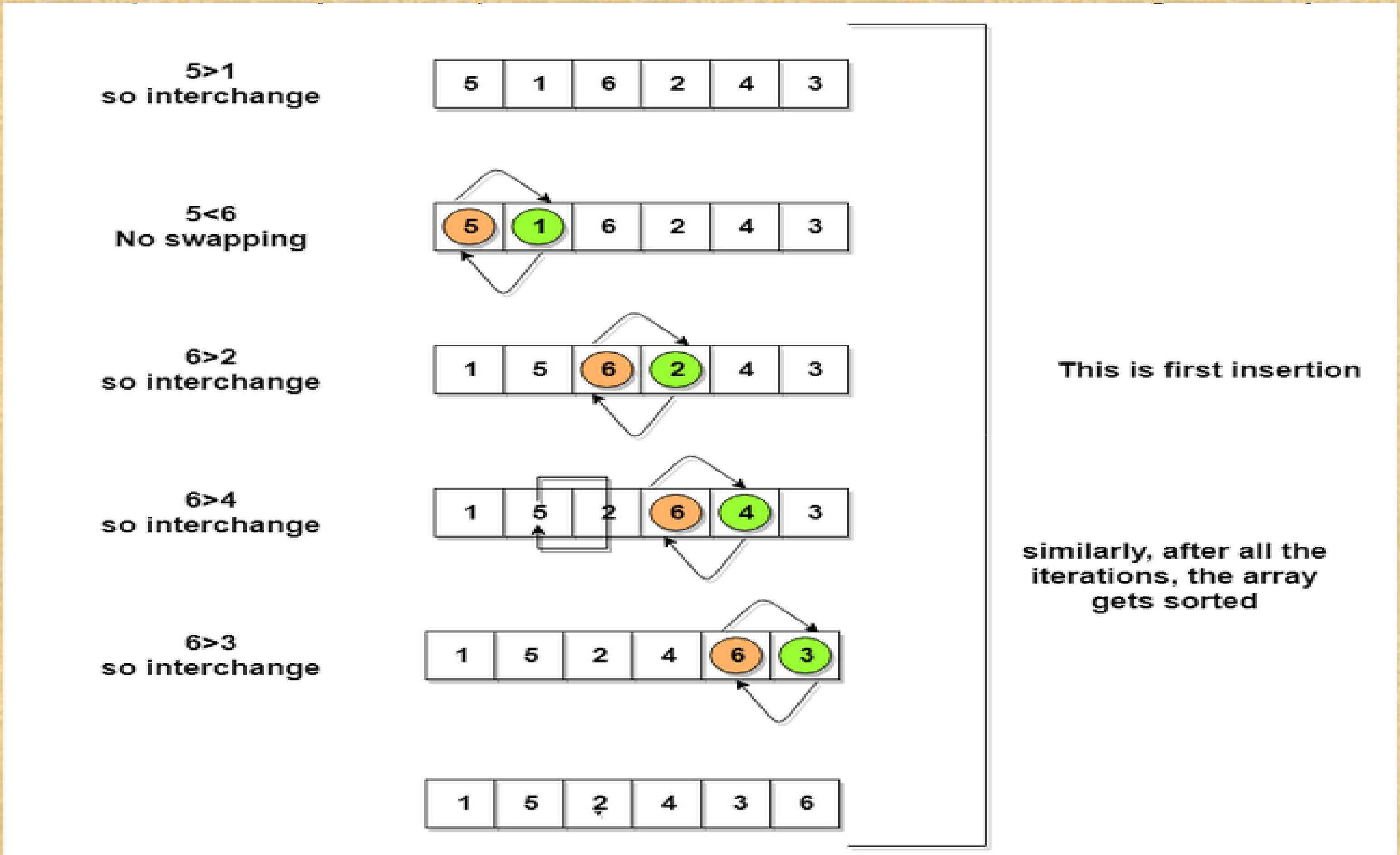
  Bubble sort and selection sort are explained in detail.

Bubble Sort Algorithm

- Bubble Sort is a simple algorithm which is used to sort a given set of n elements provided in form of an array with n number of elements.
- Bubble Sort compares all the element one by one and sort them based on their values.
- If the given array has to be sorted in ascending order, then bubble sort will start by comparing the first element of the array with the second element, if the first element is greater than the second element, it will swap both the elements, and then move on to compare the second and the third element, and so on.
- If we have total n elements, then we need to repeat this process for n-1 times.

- It is known as **bubble sort**, because with every complete iteration the largest element in the given array, bubbles up towards the last place or the highest index, just like a water bubble rises up to the water surface.
- Sorting takes place by stepping through all the elements one-by-one and comparing it with the adjacent element and swapping them if required.

Let's consider an array with values {5, 1, 6, 2, 4, 3}



So as we can see in the representation above, after the first iteration, `6` is placed at the last index, which is the correct position for it.

- Similarly after the second iteration, 5 will be at the second last index, and so on.
- The Time and Space complexity for the Bubble Sort algorithm:

➢ Worst Case Time Complexity [Big-O]: **O(n^2)**

➢ Best Case Time Complexity [Big-omega]: **O(n)**

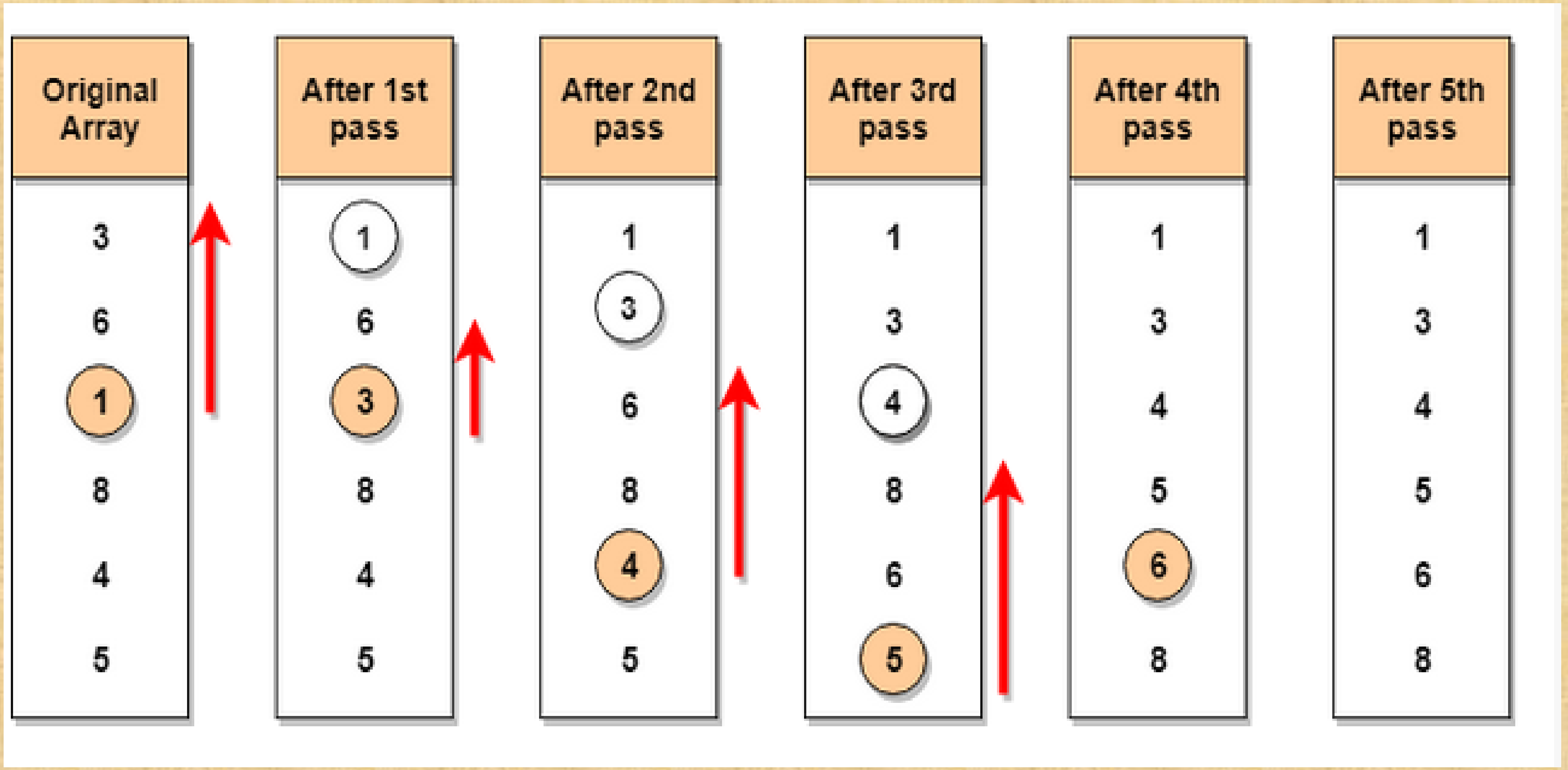
➢ Average Time Complexity [Big-theta]: **O(n^2)**

➢ Space Complexity: **O(1)**

Selection Sort Algorithm

- Selection sort is conceptually the most simplest sorting algorithm. This algorithm will first find the **smallest** element in the array and swap it with the element in the **first** position, then it will find the **second smallest** element and swap it with the element in the **second** position, and it will keep on doing this until the entire array is sorted.
- It is called selection sort because it repeatedly **selects** the next-smallest element and swaps it into the right place.

Let's consider an array with values {3, 6, 1, 8, 4, 5}



Selection Sort :

- Starting from the first element, we search the smallest element in the array, and replace it with the element in the first position.
- We then move on to the second position, and look for smallest element present in the subarray, starting from index 1, till the last index.
- We replace the element at the **second** position in the original array, or we can say at the first position in the subarray , with the second smallest element.
- This is repeated, until the array is completely sorted.

- The time and space complexity for selection sort algorithm:

- ➢ Worst Case Time Complexity [Big-O]: **O(n^2)**
- ➢ Best Case Time Complexity [Big-omega]: **O(n^2)**
- ➢ Average Time Complexity [Big-theta]: **O(n^2)**
- ➢ Space Complexity: **O(1)**

Thank You