Data Structure
Algorithms & Applications

CT-159

Prepared by
Muhammad Kamran

Sorting

- Sorting refers to arranging data in a particular format.
- Sorting algorithm specifies the way to arrange data in a particular order.
- The importance of sorting lies in the fact that data searching can be optimized to a very high level, if data is presented in a sorted manner.
- Sorting is also used to represent data in more readable formats.

In-place vs Not-In-place

- The sorting algorithms which do not require any extra space and sorting is said to happen in-place, or for example, within the array itself. This is called **in-place sorting**. Bubble sort is an example of in-place sorting.
- However in some sorting algorithms, the program requires space which is more than or equal to the elements being sorted. Sorting which uses equal or more space is called **not-in-place sorting**. Merge-sort is an example of not-in-place sorting.

Adaptive and Non-Adaptive

- A sorting algorithm is said to be adaptive, if it takes advantage of already 'sorted' elements in the list that is to be sorted. Example: Insertion Sort and Bubble Sort.
- A non-adaptive algorithm is one which does not take into account the elements which are already sorted. They try to force every single element to be re-ordered to confirm their sortedness. Example: Selection Sort and Merge Sort.

Important Terms

- Increasing Order 1, 3, 6, 12 . . .
- Decreasing Order 44, 33, 21, 9, 1.

Selection Sort

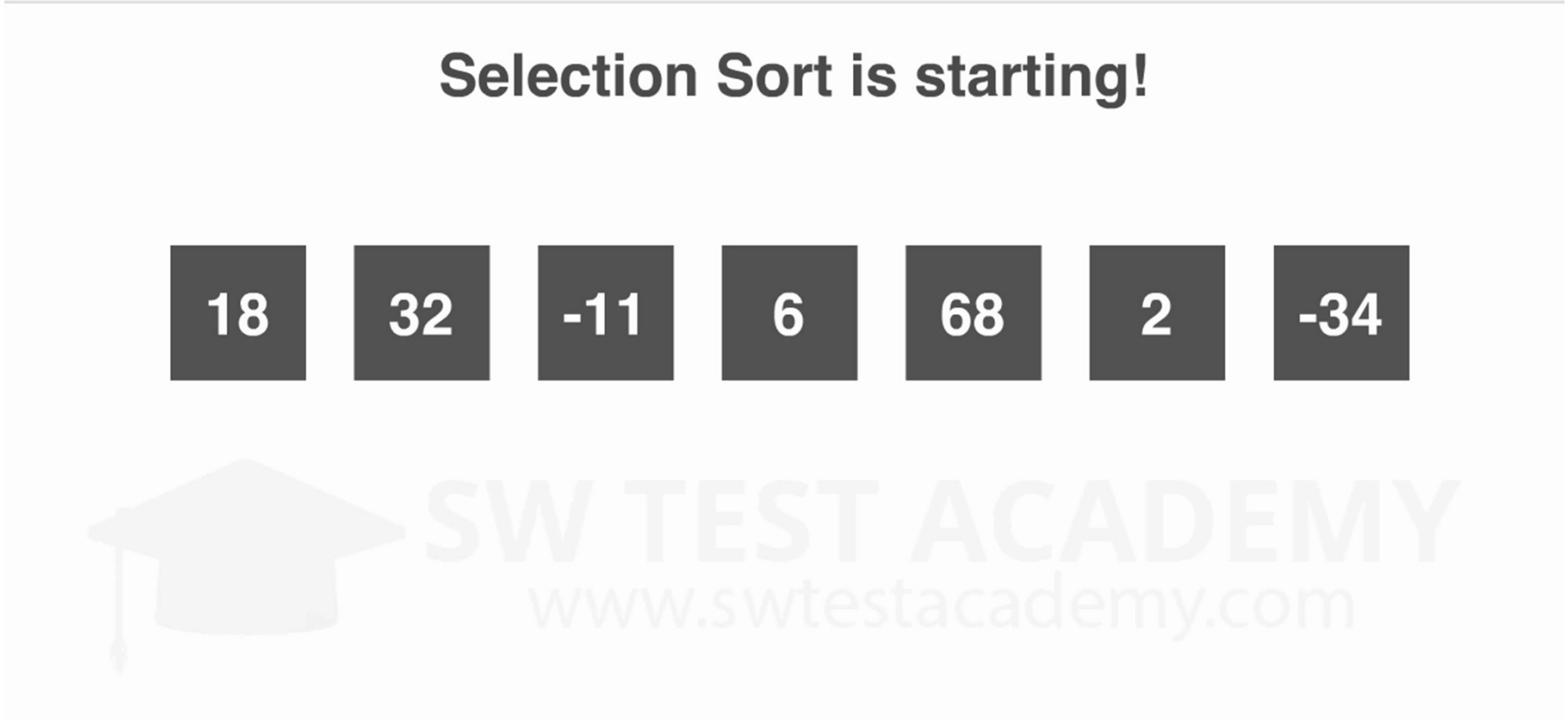
- Selection sort is a simple sorting algorithm.
- This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end.
- The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning.

Selection Sort

- **Step 1** – Set MIN to location 0
- **Step 2** – Search the minimum element in the list
- **Step 3** – Swap with value at location MIN
- **Step 4** – Increment MIN to point to next element
- **Step 5** – Repeat until list is sorted

Selection Sort

Selection Sort



Bubble Sort

- Bubble sort is another simple sorting algorithm.
- This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order.
- This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$ where **n** is the number of items.

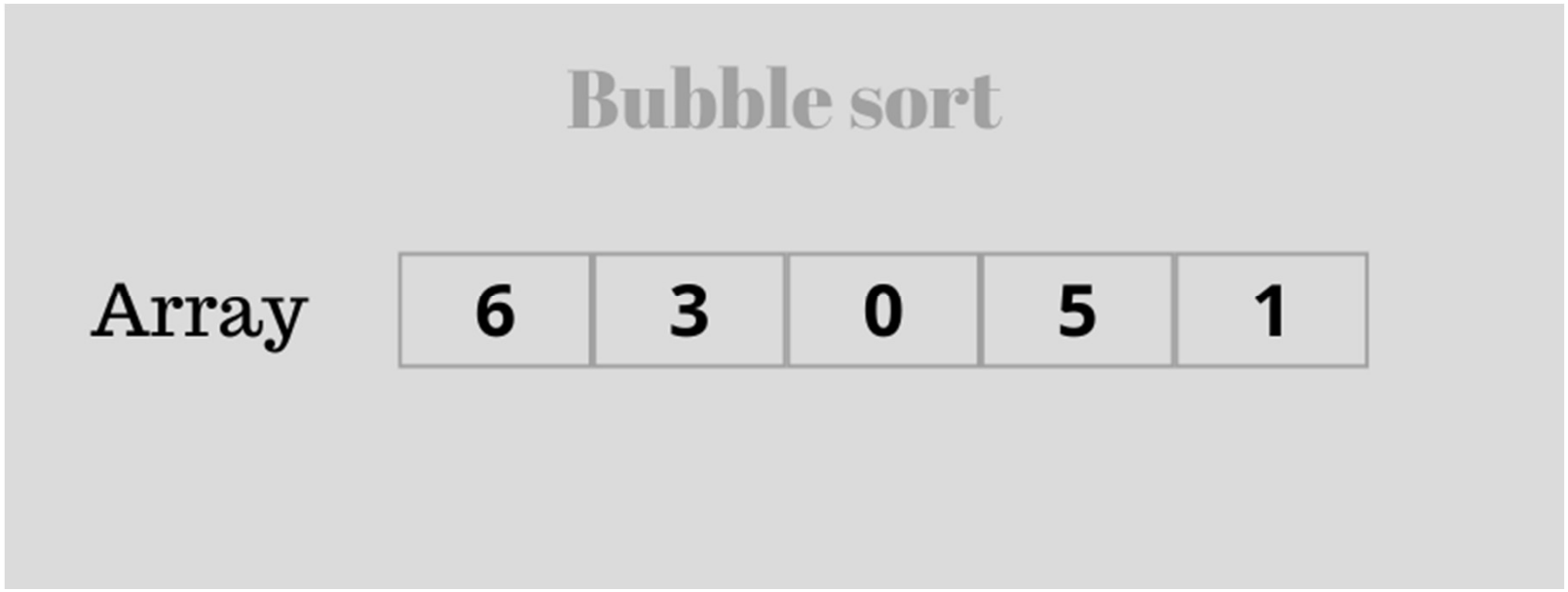
Bubble Sort

Start

Bubble Sort Algo


```
begin BubbleSort(list)
    for all elements of list
        if list[i] > list[i+1]
        swap(list[i], list[i+1])
        end if
    end for
     return list
end BubbleSort
```

Bubble Sort



Insertion Sort

- An in-place comparison-based sorting algorithm.
- An element which is to be inserted in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, **insertion sort**.
- This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$, where **n** is the number of items.

Insertion Sort

- **Step 1** – If it is the first element, it is already sorted. return 1;
- **Step 2** – Pick next element
- **Step 3** – Compare with all elements in the sorted sub-list
- **Step 4** – Shift all the elements in the sorted sub-list that is greater than the value to be sorted
- **Step 5** – Insert the value
- **Step 6** – Repeat until list is sorted

Insertion Sort

12, 11, 13, 5, 6

Let us loop for i = 1 (second element of the array) to 4 (last element of the array)

i = 1. Since 11 is smaller than 12, move 12 and insert 11 before 12

11, 12, 13, 5, 6

i = 2. 13 will remain at its position as all elements in A[0..I-1] are smaller than 13

11, 12, 13, 5, 6

i = 3. 5 will move to the beginning and all other elements from 11 to 13 will move one position ahead of their current position.

5, 11, 12, 13, 6

i = 4. 6 will move to position after 5, and elements from 11 to 13 will move one position ahead of their current position.

5, 6, 11, 12, 13

Thank You