ALGOMANIAX CTE - Intro to Competitive Coding

SORTING - I

What is Sorting ?

- Sorting refers to the process of arranging a list of items in a particular order.
  - Example : Sort a list of numbers in Ascending / Descending order
  - Sort the Time Table combinations in Preference order.
  - Sort the marks of all students in a course for Grade Allotment.

- It is estimated that 25~50% of all computing power is used for sorting activities, and hence we need optimized sorting algorithms, requiring less memory and consuming less time.

Some Types of Sorting Algorithms

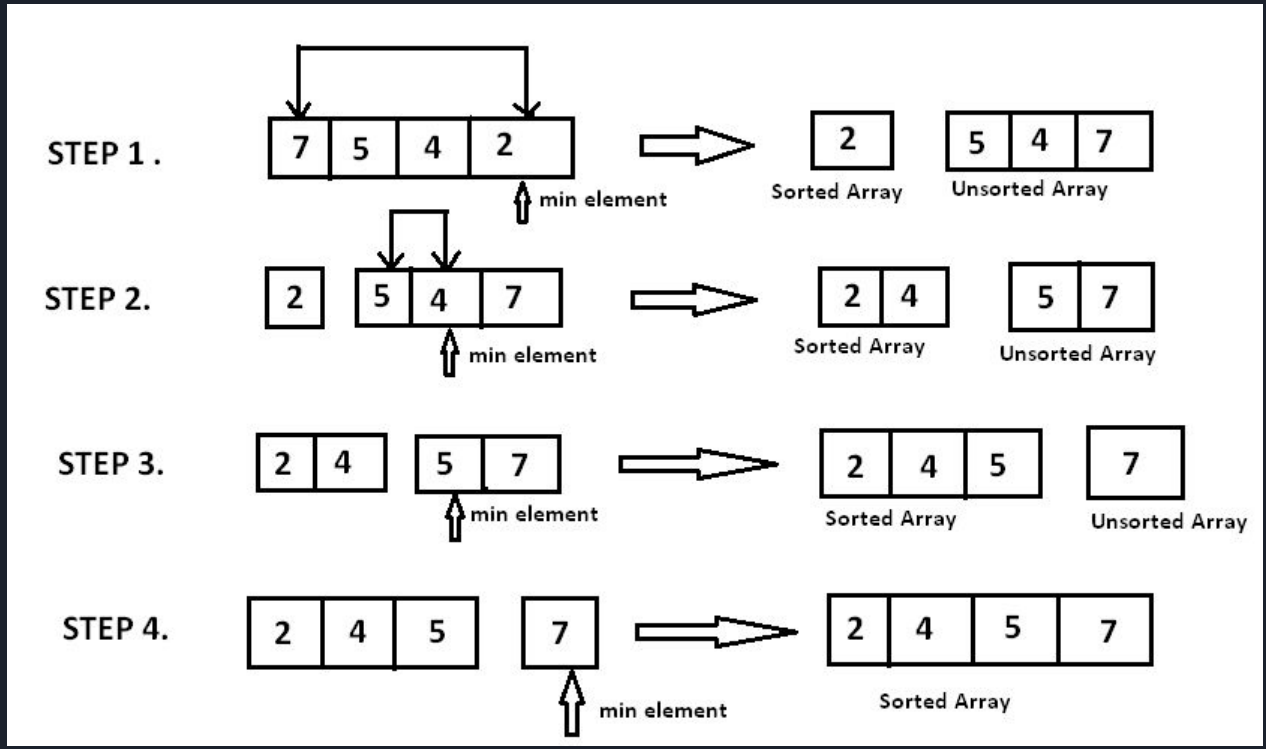
- Selection Sort
- Bubble Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort
- Radix Sort
- …. and the list goes on.

Selection Sort

- The idea is to find the smallest element in the list, and put it at the 1st position.
- Then find the second smallest element, and put it at the 2nd position.
- Then find the third smallest element, and put it at the 3rd position.
- And continue the process until the whole list is sorted.

This process is similar to the way how even our brain generally sorts a list.

Selection Sort - A visual representation



Selection Sort - The code

```
void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n-1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;

        for (j = i+1; j < n; j++)
          if (arr[j] < arr[min_idx])
            min_idx = j;

        // Swap the found minimum element with the first element
        swap(&arr[min_idx], &arr[i]);
        // Assuming we have defined a function to swap two elements of an array.
     }
}
```

Bubble Sort

- Bubble sort examines the array from start to finish, comparing elements as it goes.
- Any time it finds a larger element before a smaller element, it swaps the two.
- In this way, the larger elements are passed towards the end.
- The largest element of the array therefore "bubbles" to the end of the array.
- Then it repeats the process for the unsorted portion of the array until the whole array is sorted.

Bubble Sort - A visual representation

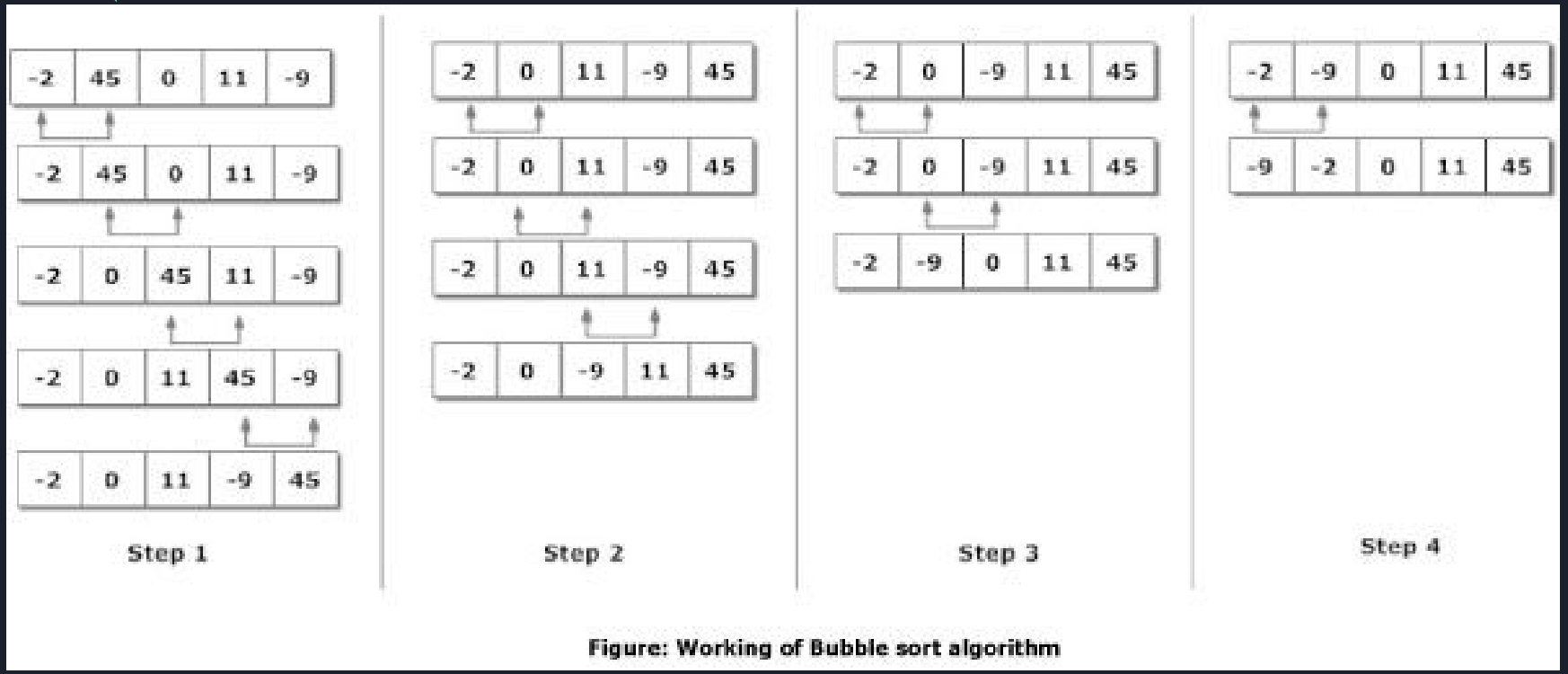


Figure: Working of Bubble sort algorithm

Bubble Sort - The code

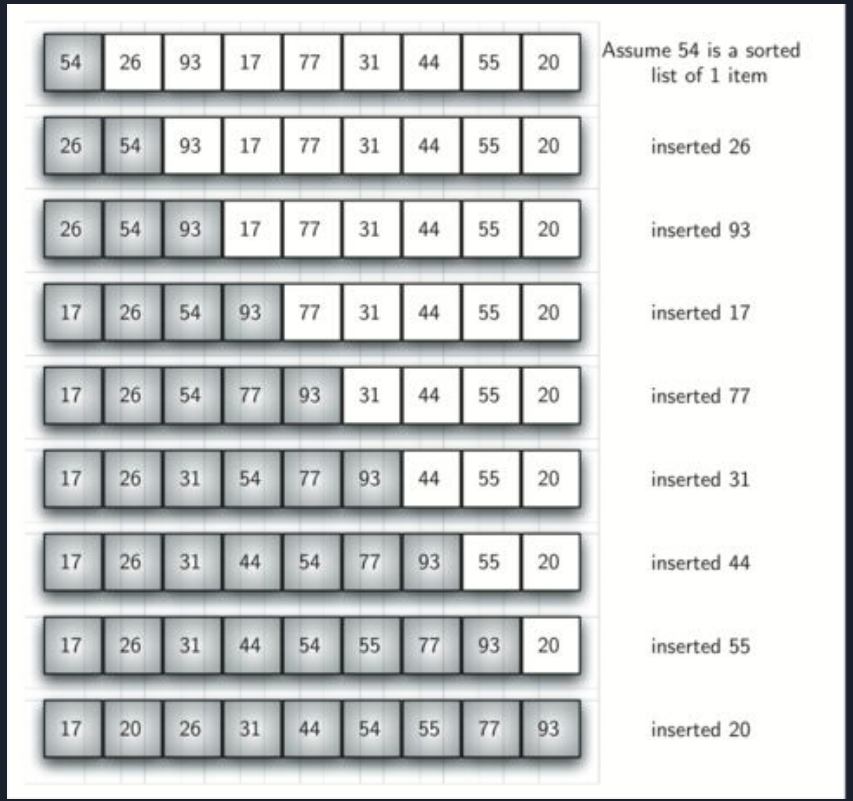
```
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

       // Last i elements are already in place
       for (j = 0; j < n-i-1; j++)
           if (arr[j] > arr[j+1])
              swap(&arr[j], &arr[j+1]);
}
```

Insertion Sort

- Insertion sort is a simple sorting algorithm that builds the final sorted array (or list) one item at a time.
- After the first iteration, the first 2 elements will be sorted.
- After the 2nd iteration, the first 3 elements will be sorted, i.e. the 3rd element will be “inserted” to its correct position.
- Similarly, after ‘n-1’ iterations, all the elements are sorted.

Insertion Sort - A visual representation



Insertion Sort - The Code

```
void insertionSort(int arr[], int n)
{
   int i, key, j;
   for (i = 1; i < n; i++)
   {
       key = arr[i];
       j = i-1;

       /* Move elements of arr[0..i-1], that are greater than key, to one
       position ahead of their current position */

       while (j >= 0 && arr[j] > key)
       {
           arr[j+1] = arr[j];
           j = j-1;
       }
       arr[j+1] = key;
   }
}
```

Efficiency ?

1. Selection Sort :
    a. Fixed number of comparisons = $O(n^2)$
2. Bubble Sort :
    a. Fixed number of comparisons = $O(n^2)$
3. Insertion Sort :
    a. Best-case = $O(n)$
    b. Worst-case = $O(n^2)$

Efficiency -

As we saw, all of these are $O(n^2)$ algorithms.

At $n > 10^4$, these algorithms will take a lot of time to compute, and hence we need better.

Hence, we have Merge-Sort and Quick-Sort, which are O(n log(n)) algorithms.

We'll discuss more about them in the next lecture.

Thank You !