Data Structure And Algorithms

Bubble Sort Algorithm

- Bubble sort is a 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.

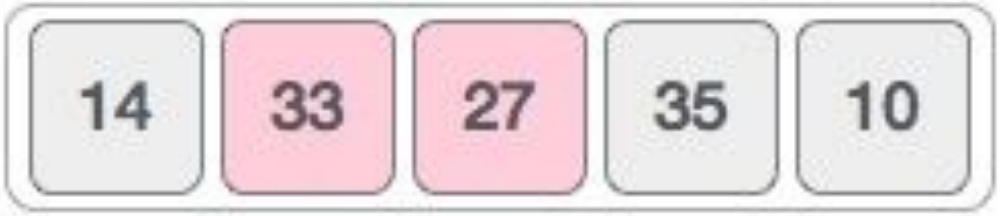
How Bubble Sort Works?

- We take an unsorted array for our example. Bubble sort takes $O(n^2)$ time so we're keeping it short and precise.

Bubble sort starts with very first two elements, comparing them to check which one is greater.

- In this case, value 33 is greater than 14, so it is already in sorted locations. Next, we compare 33 with 27.

- We find that 27 is smaller than 33 and these two values must be swapped.
-



The new array should look like this –

- Next we compare 33 and 35. We find that both are in already sorted positions.
-

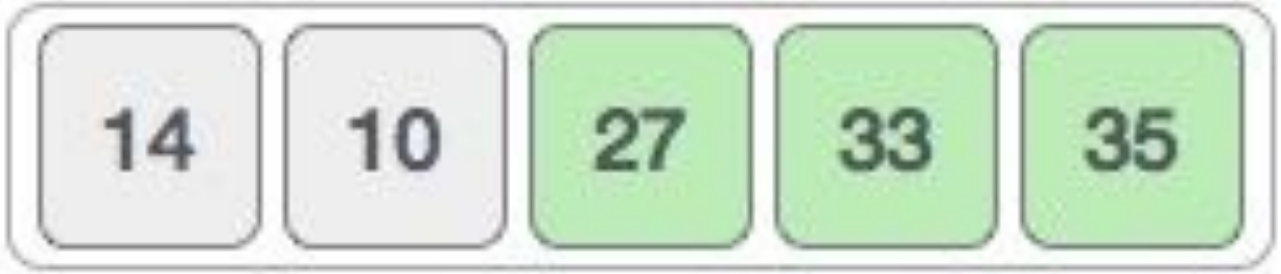
- Then we move to the next two values, 35 and 10.

- We know then that 10 is smaller 35. Hence they are not sorted.

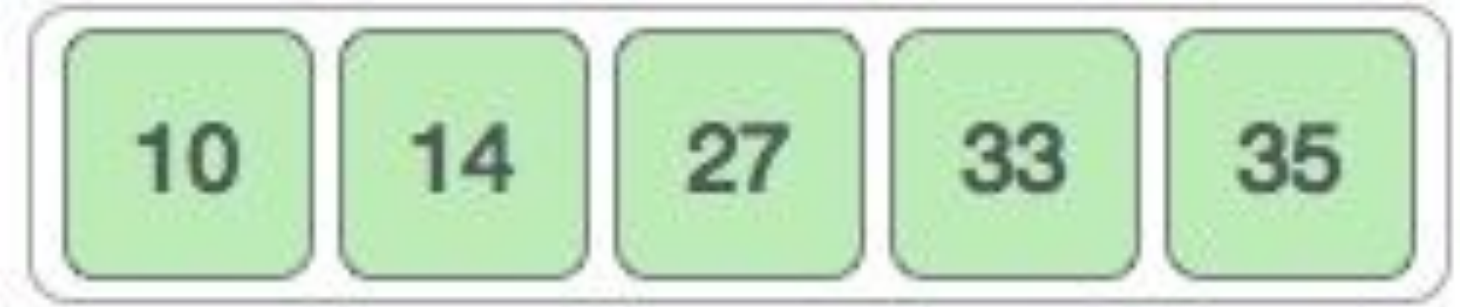
- We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this –

- To be precise, we are now showing how an array should look like after each iteration. After the second iteration, it should look like this –

- Notice that after each iteration, at least one value moves at the end.



- And when there's no swap required, bubble sorts learns that an array is completely sorted.
-



Algorithm

- 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

```java
public static void bubbleSort(int arr[],int n){
    n=arr.length;
    if (n==0||n==1){
        return;
    }

    int temp=0;
    for(int i=0;i<n-1;i++){
        for(int j=0;j<n-i-1;j++){
            if(arr[j]>arr[j+1]){
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;

            }
        }
    }
    for(int k:arr){
        System.out.print(k+" ");
    }
    }
```

Worst Case Analysis for Bubble Sort:

- ***At pass 1 :*** *Number of comparisons = (n-1)*
  Number of swaps = (n-1)
- ***At pass 2 :*** *Number of comparisons = (n-2)*
  Number of swaps = (n-2)
- ***At pass 3 :*** *Number of comparisons = (n-3)*
  Number of swaps = (n-3)

  .
  .
  .

  At pass n-1 : *Number of comparisons = 1*
  Number of swaps = 1
- *Now , calculating total number of comparison required to sort the array*
  = (n-1) + (n-2) + (n-3) + . . . 2 + 1
  = (n-1)(n-1+1)/2 { by using sum of N natural Number formula }*
  *= **n (n-1)/2***

- ***Total number of swaps = Total number of comparison***
  Total number of comparison (Worst case) = ***n(n-1)/2***
  Total number of swaps (Worst case) = ***n(n-1)/2***