# Bubble Sort

⊙ What is Bubble Sort ? Explain with java code step by step.

→

## Bubble Sort

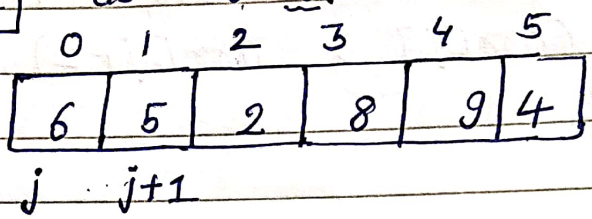
Def 1: " One of the simplest sorting algorithm used to sort a list of elements. It works by repeatedly stepping through the list, comparing adjacent elements, and swapping them if they are in the wrong order. This process is repeated until the list is sorted."

Def 2: " also known as exchange sort. Which arranges values by iterating over the list several times & in each iteration the large value gets bubble up to the end of the list. This algorithm uses multiple passes and in each pass the first and second data items are compared. If the first data item is bigger than the second, then the two items are swapped. Next the items in 2nd & 3rd position are compared and if the 1st one is larger than the 2nd, then they are swapped, otherwise no change in their order."

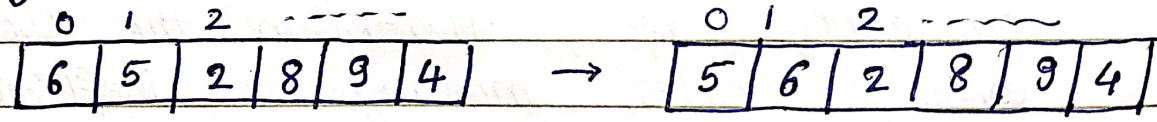
## Step-by-Step Explanation

Start at the begining of the list:
Compare 1st two elements.

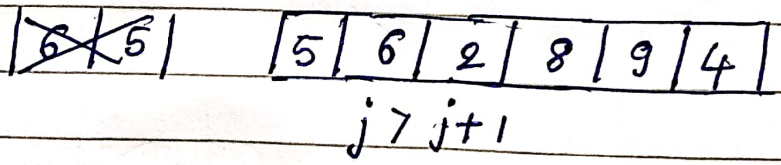
Consider first element as $j = 6$
& $j+1$ as 5.




Swap if needed :- If the first element is greater than the second element, swap them.

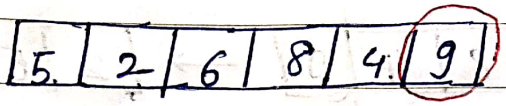



Move the next pair :- Compare the next two elements.

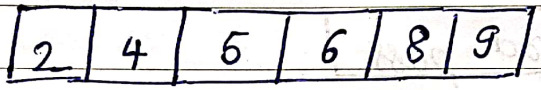



Repeat the process :- Comparing & Swapping adjacent elements until you reach the end of the list.

5] Pass through the list :- After each pass through the list, the next largest element will be in its corresponding position.

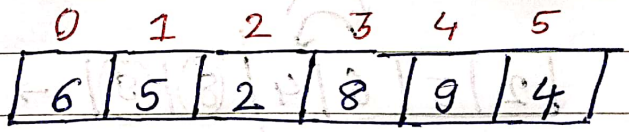



6) Repeat the passes :- Continue making passes through the list until no swaps are needed, indicating the list is sorted.

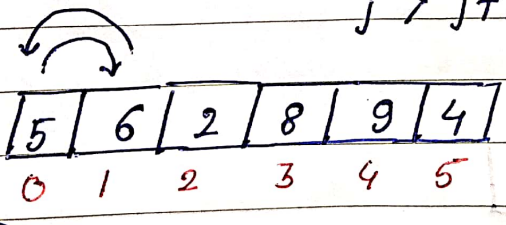



Example :-

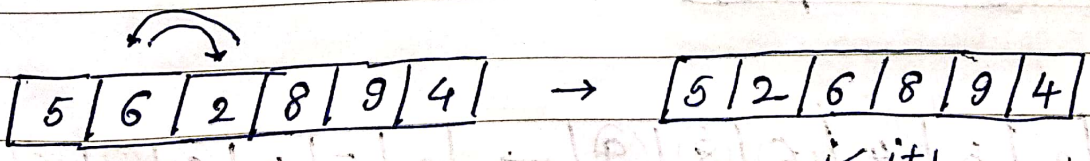
Initial array :

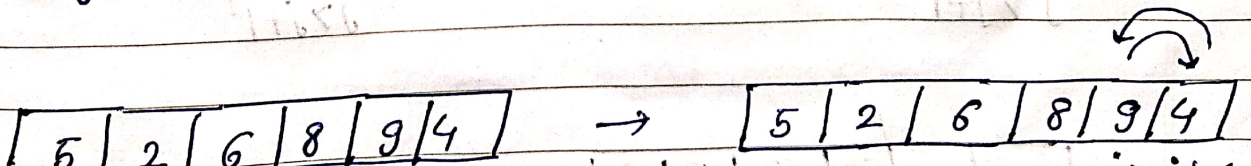


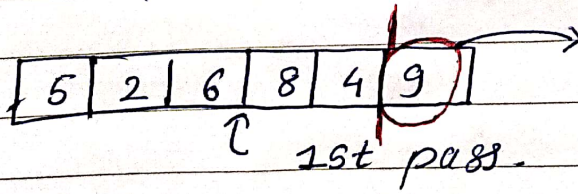

$j > j+1$

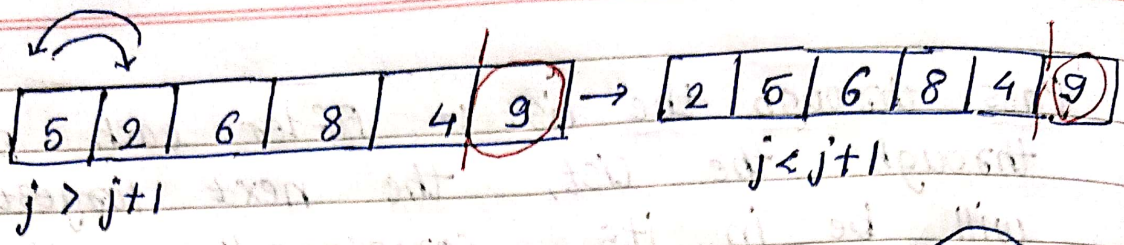


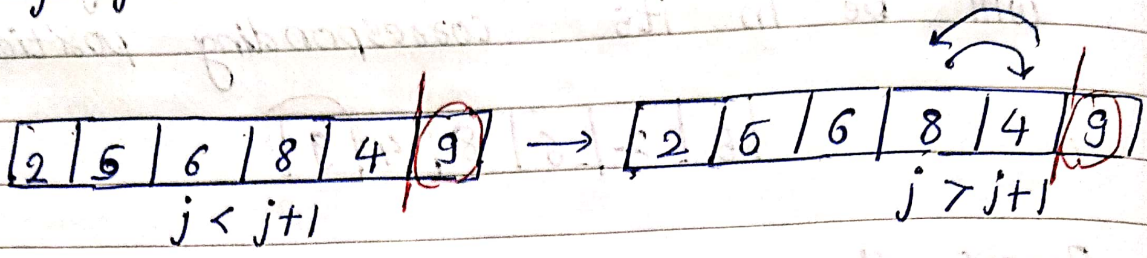

Swap 6 & 5.

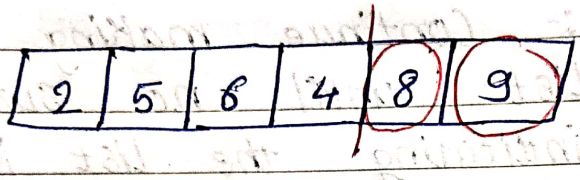


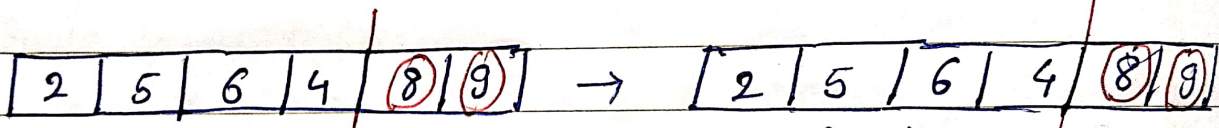


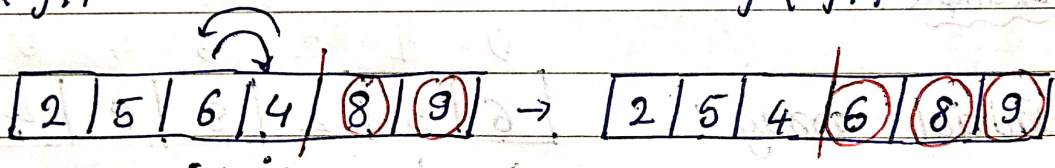



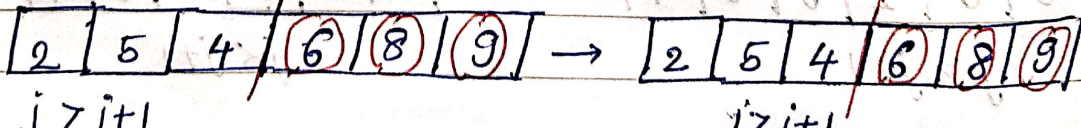

← 2nd pass.

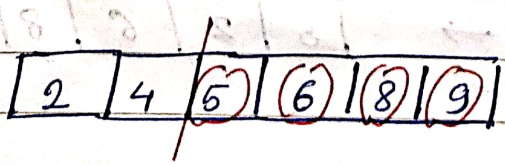
After 3rd pass



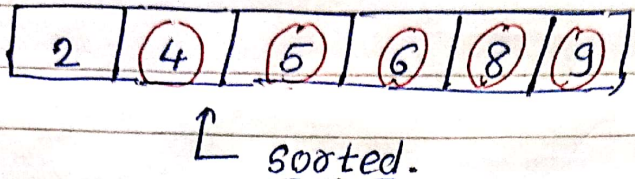




After 4 th pass



After 5 th pass




# Bubble Sort code :-

```
public class P2_BubbleSort {

    // method to perform bubble sort.
    public static void bubbleSort(int[] arr){
        int len = arr.length;
        int temp = 0;

    // Loop through all elements in the array.
    for(int i=0; i<len-1; i++){

        // Inner loop for each pass.
        for(int j=0; j<len-i-1; j++){
            // Compare adjacent elements.
            if(arr[j] > arr[j+1]){
                // Swap them if they are in
                // wrong order.
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;

    printArray(arr);
```

```
        }
    }
}
}

// method to print the array.
public static void printArray(int[] arr){
    // Using enhanced for loop.
    for( int value : arr){
        System.out.println( value + " ");
    }
    System.out.println();
}

// main method to test the bubble sort
public static void main(String[] args){

    int[] arr = { 64, 34, 25, 12, 22};


    System.out.println("original array: ");
    printArray(arr);
    System.out.println("--- --- ---");

    bubbleSort(arr);
    System.out.println("--- --- ---");
    System.out.println("Sorted array:");
    printArray(arr);
}
```

# Output.

Original array:

64 34 25 12 22

34 64 25 12 22

34 25 64 12 22

34 25 12 64 22

34 25 12 22 64

25 34 12 22 64

25 12 34 22 64

25 12 22 34 64

12 25 22 34 64

12 22 25 34 64

Sorted array:

12 22 25 34 64

# Explanation

## Bubble Sort method

- The outer loop from the beginning of the array to the 2nd last of the array i.e (i < len-1). This loop keeps track of how many elements are sorted.

- The inner loop runs from ~~j=i+1 i.e it runs~~ beginning of the array to the last unsorted element (j < len-i-1). This loop performs the comparison & swapping of adjacent elements.

- Inside the inner loop, if the current element (arr[j] > (arr[j+1]), then they are swapped.

- The printArray method is called after each loop to show the array's state

⊙ Efficiency :- Bubble Sort has a time complexity of $O(n^2)$ in the worst & average cases, making it inefficient on large lists.