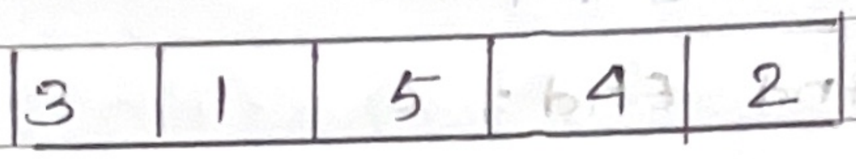
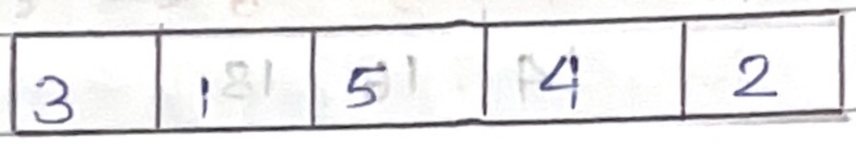
BUBBLE SORT

- Bubble sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order
- Not suitable for large data sets.

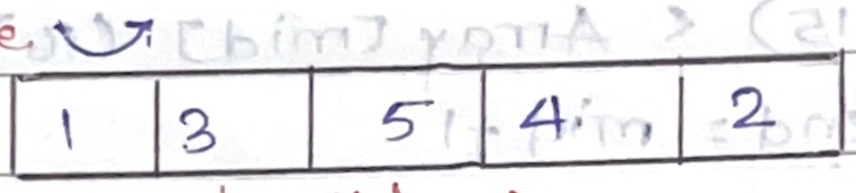
- WORKING -



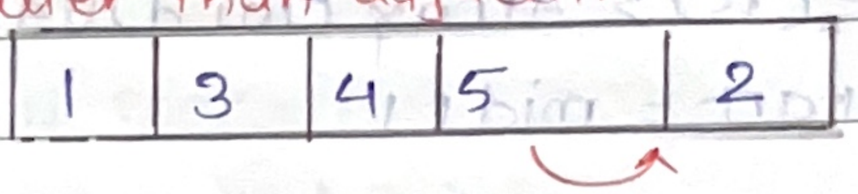
First pass -

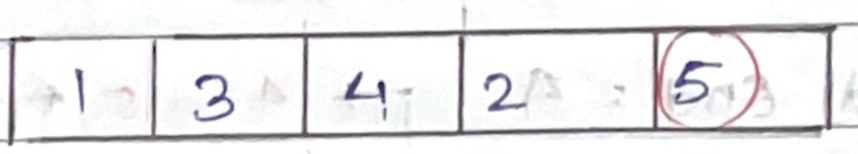


bigger than adjacent ele.



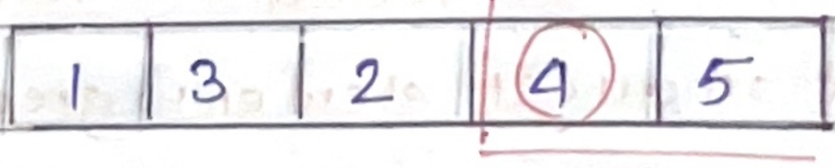
as the element is smaller than adjacent.





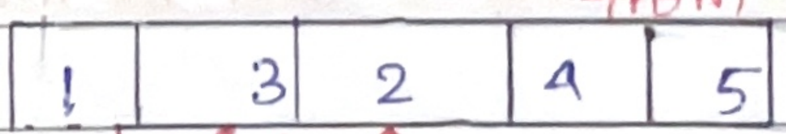
With first pass through the entire array, the largest element come to end.

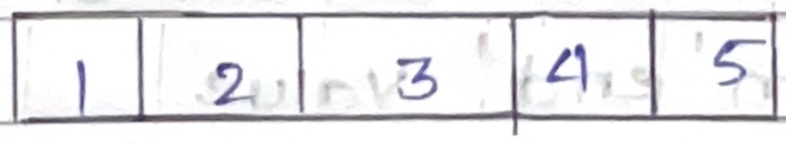
Second pass



with second pass second largest element comes at second from reverse index.

Third pass





sorted

Bubble Sort is also known as Sinking Sort or Exchange Sort

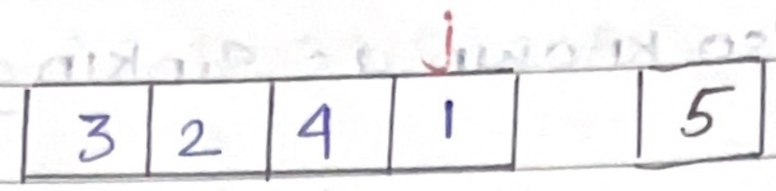
- Space complexity - O(1) (constant)
  since here no extra space is required i.e like copying the array, etc. is not required. also known as inplace sorting algorithm.

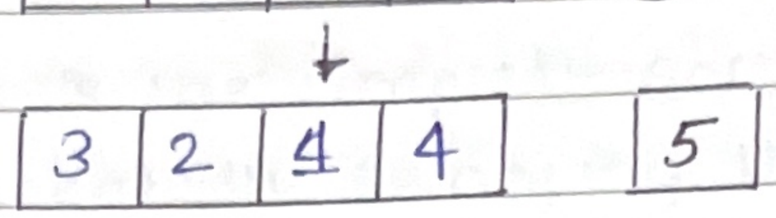
- Time complexity
  Best Case ⟶ O(N)
  The best case occurs when the array is sorted already. So the number of comparisons required is N-1 and number of swaps required = 0. Hence the best case complexity is O(N)

  Worst Case ⟶ $O(N^2)$
  The worst case occurs when array is sorted in descending order.

First pass:

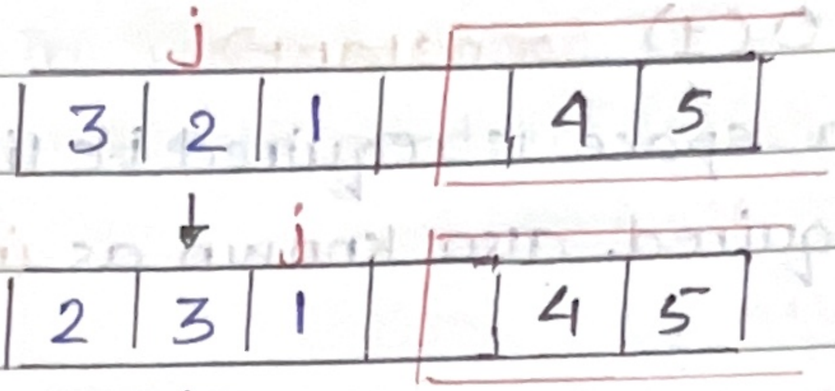
Second pass:





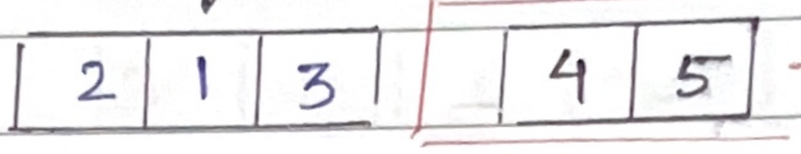
→ (N-2) swaps

Third pass:



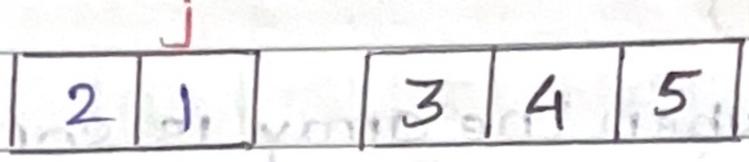
sorted

i = 2



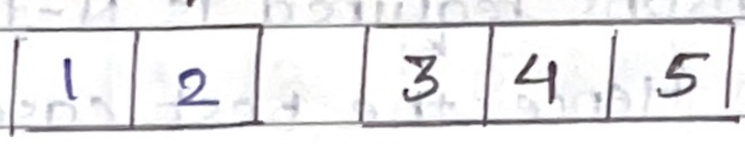
→ (N-3) swaps

fourth pass



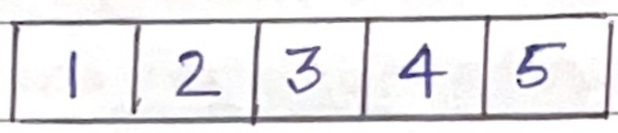
sorted

i = 3



→ (N-4) swaps

Final:



Sorted

Total comparisons: $N-1+N-2+N-3+N-4$

$4N-(1+2+3+4)$

$4N-\left[\frac{N\times(N-1)}{2}\right]$

$4N-\frac{N^2+N}{2}$

$O\left(\frac{7N-N^2}{2}\right)$ → In time complexity constant and less dominating items are ignored.

Total comparisons: $O(N^2)$

- Stable Algorithm: When you sort equal elements, their orignal order is maintained in sorted result
- Unstable Algorithm: When you sort equal element, their orignal order might not be preserved in sorted algorithm

Example:

```
public class Main {
    public static void main (String [] args) {
        int [] arr = {5, 4, 3, 2, 1};
        bubbleSort (arr);
        System.out.println (Arrays.toString (arr));
    }

    static void bubbleSort (int [] arr) {
        boolean swapped;
        for (int i=0; i<arr.length; i++)
        {
            // for each step, max item will come at last
            // respective index
            swapped = false;
            for (int j=1; j<arr.length-i; j++)
            {
                //swap if item is smaller than previous item
                if (arr[j] < arr[j-1])
                {
                    int temp = arr[j];
                    arr[j] = arr[j-1];
                    arr[j-1] = temp;
                    swapped = true;
                }
            }
        }
        //if you didnot swap for particular value of i
        // it means the array is sorted stop the program.
        if(!swapped) // swapped = false
            break;
        }
    }
}
```

O/P- [1, 2, 3, 4, 5]