M T W T F S S

Page No.: YOUVA

Date:

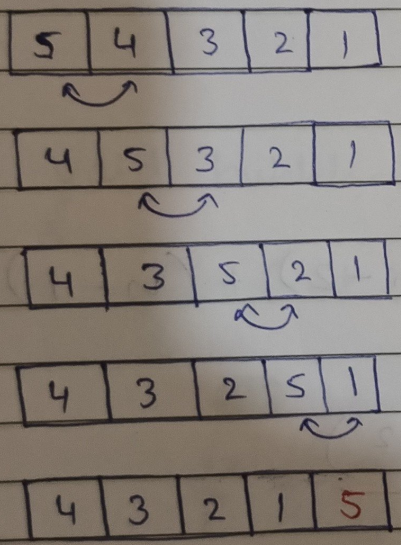
Bubble Sort

Bubble Sort compare ~~ajd~~ adjest adjacent element in a list and swaps them if they are in wrong order. It repeatedly goes through the list until no more swap are needed.

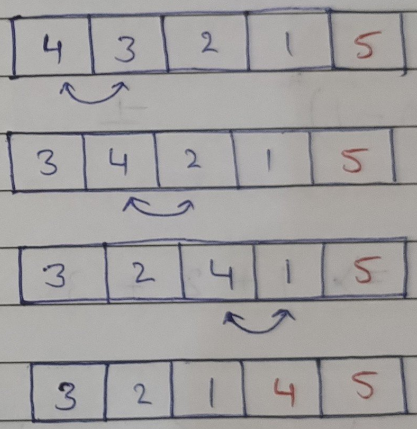
↳ Agar agge wala element pichu wale se bada hu to unko swap kara.

arr[] = {5, 4, 3, 2, 1}

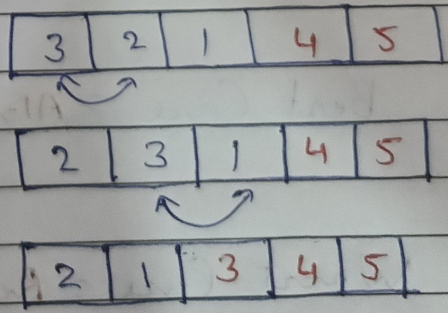
Round 1:



Round 2:

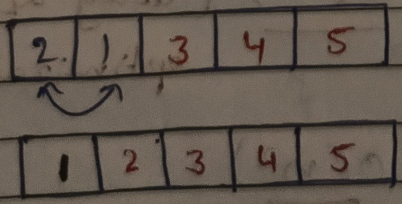


Round 3:



Note: We don't need to chuck for red part because it's already sorted.

Round 4:



Note: We don't have to chuck ① because it's already at it's correct position.

7 May 2023, 8:58 pm

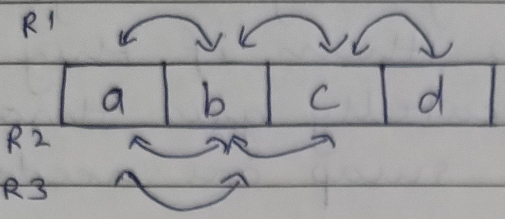
M T W T F S S

Page No.:

Date:

YOUVA

Time Complexity:



$n = 4$

$R_1 = (n-1)$ (3)

$R_2 = (n-2)$ (2)

$R_3 = (n-3)$ (1)

$(n-1)^{th}$ 1

$\Rightarrow 1 + 2 + 3 \ldots\ldots\ldots (n-2) + (n-1)$

$\Rightarrow \frac{n(n-1)}{2} = O(n^2)$

Best Case = Already Sorted = $O(n)$ (Only after optimization)

Worst Case = Reverse Sorted = $O(n^2)$

Space Complexity:

$\Rightarrow O(1)$ (No extra space required)

7 May 2023, 8:58 pm

Advantages:

1) Bubble Sort is stable
2) Easy to understand and implement
3) Low memory required
4) Usefull for small data set (<100)

Disadvantages:

1) Make more swapping than selection sort.
2) Not good for mostly sorted array.
3) Slower than selection sort.

Code:

```
void bubble_sort(int *arr, int n)
{
    for(int i=1; i<n; i++)
    {
        bool swapped = false;
        for(int j=0; j<n-i-1; j++)
                        ↓
            // Coz we don't need to check sorted part
            if(arr[j] > arr[j+1]){
                swap(arr[j], arr[j+1]);
                swapped = true;
            }
    }
    if(swapped == true)
        break;  // Means list is already sorted
    }

}
```