Bubble Sort:

* Bubble Sort is comparison sort method.

* In every step you are comparing adjacent elements.

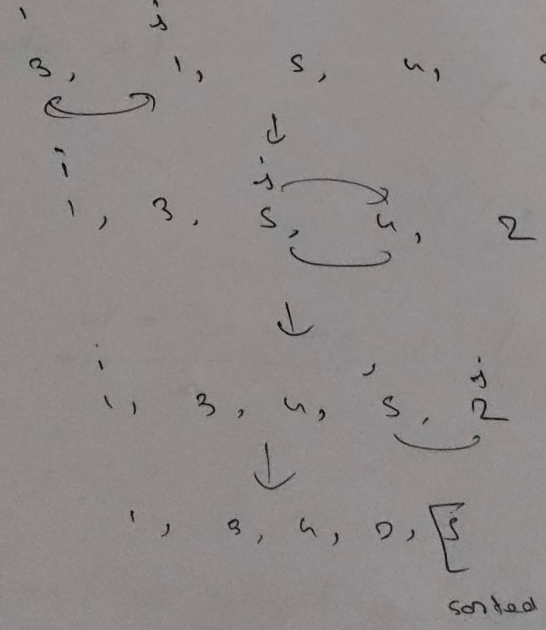
* What are we doing is comparing adjacent element if the element is greater than next element swap it.

* Why are we doing, on the first pass through the array, the largest element came in the end. like wise with pass n, n^{th} largest element in at the n^{th} from the last index.

* Bubble sort is also known as sinking sort or exchange sort.

* For every pass last part of the array sorted by the leagest numbers are moving that side.

Eg:- 3, 1, 5, 4, 2



Here

i is internal loop runs n-1 time for every pass.

i is counter here.

~~* Time complexity of Bubble sort O~~

* The space complexity of Bubble sort O(1). Because the array size is not changing for swapping or copping. This also known as Inplace sorting algorithm

* The time complexity:

Best case: O(N) ⇒ Sorted

Worst case: O(N²) ⇒ ~~Not~~ Sorted in descarding.

* When it never swaps for ith pass, then array is sorted, exit the loop or function.

worst case

$= (N-1) + (N-2) + (N-3) + (N-4) \ldots$

lets take until 4

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

$= 4N - \left(\frac{N * (N+1)}{2}\right)$

$= 4N - \left(\frac{N^2 + N}{2}\right) \Rightarrow \frac{8N - N^2 - N}{2}$

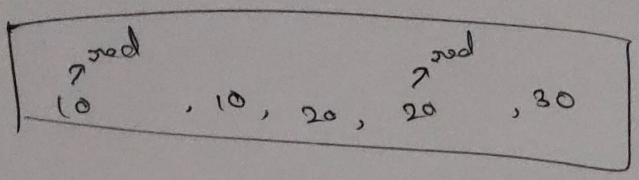
$= \frac{7N - N^2}{2}$

$= O(N^2)$ ∴ highest degree and ignore constant.

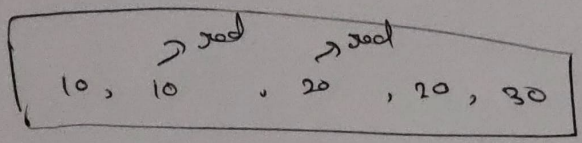
Stability

10, 20 (red), 20, 30, 10 (red) remain block.

In orginal array, block ball of 10 has before red ball of 10. And in the sorted one, this order is maintaed.



→ unsorted due to order change.



→ sorted.

Code:

```
static void bubbleSort(int[] arr){
    boolean swapped;
    for(int i=0; i < arr.length ; i++){
        swapped = false;
        for(int j=1 ; j < arr.length-i ; j++){

            if (arr[j] < arr[j-1])
            {
                int temp = arr[j];
                arr[j] = arr[j-1];
                arr[j-1] = temp;    swapped = true;
            }
        }
        if (swapped == false){
            break;              // if array is sorted.
    } }
```