-> 16 - Jun - 2024 [WeekDay]

- Sorted Array / Non-decending ✓
  - Accending ↑ $i < i+1^{th} < i+2^{nd}$
  - might have 2 same values ↑

- Bubble Sort: In every step, compare adjust elems, if not correctly placed, swap them.

Worst Case Complexity: $O(n^2)$ (longest)

- At each iteration, the rightmost unsorted elem gets correctly placed.

- Max passes / iterations = len - 1

- Keep track of var swapped at each inner iteration. if traversed through inner loop and still remain false, return and skip rest of iteration as it is already sorted.

- Also Sinking sort / Exchange sort

- Outer loop i [0 to $i < n-1$]
  Inner loop j [1 to $j < n-i$]
  (checks i to $(j-1)$ ✓)

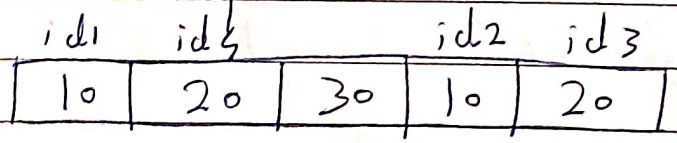
- Space Complexity: O(1)
  No requirement of new array space
  ↑
  Also inplace Sorting Algo.

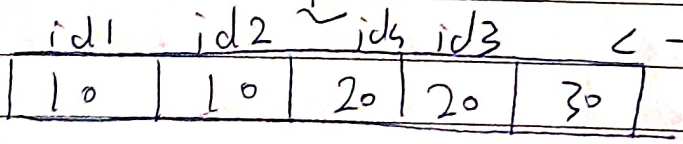
- Best Case Time Complexity O(n)
  ↑
  Already Sorted

- Worst Case Time Complexity $O(n^2)$
  ↑
  Decendig Array

- Complexity is how time/space grows as input increase.

- It is stable sorting algorithm





still id1 is before id2 ✓

Stable] Original order is maintain for the same values after sorting

Code for Bubble Sort]

def ~

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

-> Selection Sort

-> Select an elem and place it at its correct position in the array

-> Select the largest unsorted elem and swap it with its correct index position.

-> Again length-1 passes for worst case (descending array)

-> Can also take min instead max

- Space Complexity: Inplace ~ O(1)

- Best Case Time: $O(n^2)$ <- Has to find max every time

- Worst Case Time: $O(n^2)$

iter i -> n-1
iter i-1 -> n-2
...
iter last -> 0

Total Comparisions: $1 + 2 + \ldots + (n-1)$

$$= \frac{(n-1)(n-1+1)}{2}$$

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

$$= \frac{n^2 - n}{2}$$ <- less dominating term

<- const

$\approx (n^2)$ ~

- Not stable Sorting Algo

- Well performance on small lists