② Bubble, Selection and Insertion Sort.

i) Bubble Sort → $O(n^2)$

→ Bubble sorting algo.

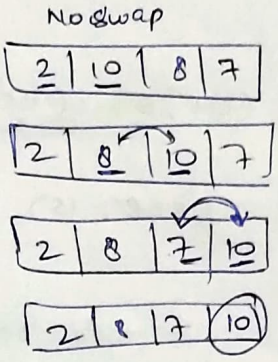
→ Swapping done when 2 elements compared are out of order.

→ In first pass, largest element moved to it's final position i.e last position.

→ In second pass, second largest etc.

→ etc / so on. (n − 1 passes).

⇒



1st pass

$(n-1) + (n-2) \cdots$
$\cdots + 2 + 1$
$= \frac{n*(n-1)}{2}$
$= O(n^2)$

Algorithm

⇒ Void BubbleSort (arr, n)

```
{
  for(i=0; i<n-1; i++) {
    for(j=0; j<n-1; j++) {        → (n-i-1) for optimize
      if(arr[j] > arr[j+1]) {
        swap(arr[j], arr[j+1]);
      }
    }
  }
}
```

11) Selection Sort

→ O(n²) Algorithm. → Same as Bubble

→ Does less memory writes than other popular sorting algo.
(But most optimal w.r.t memory writes is Cycle sort)

→ foundation for heap sort. → in-place.

→ Not Stable.

→ Algo.

→ find min elem put it in arr[0];

→ find 2ee min ele put it in arr[1];

:

so on.

Algo.

```
void SelectSort(arr, n)
{
  for(i=0; i<n; i++)
  {
    min_index = i;
    for (int j = i+1; j<n; j++)
    { if (arr[j] < arr[min_index])
      { min_index = j; }
    Swap(arr[min_index], arr[i]);
    }
  }
}
```

(iii) Insertion Sort

- $O(n^2)$ (O(n) in best case). | → In-place and stable.
- → Most efficient for small arrays. (Used in hybrid algo like TimSort and IntroSort).
- → 0 ---- i-1, i, ---------- n-1

Sorted.	key	Unsorted.

- → Eg:
  - [20, 5, 40, 60, 10, 30]
  - [5, 20, 40, 60, 10, 30]
  - [5, 20, 40, 60, 10, 30]
  - [5, 20, 40, 60, 10, 30]
  - [5, 10, 20, 40, 60, 30]
  - [5, 10, 20, 30, 40, 60]
- → Algo.

```
void InsertionSort(int arr[], int n).
{  for(int i=1; i<n; i++)
   { int key = arr[i];
     int j = i-1;
     while (j>=0 && arr[j] > key)
     {  arr[j+1] = arr[j];
        j--;
     }
     arr[j+1] = key;
   }
}
```