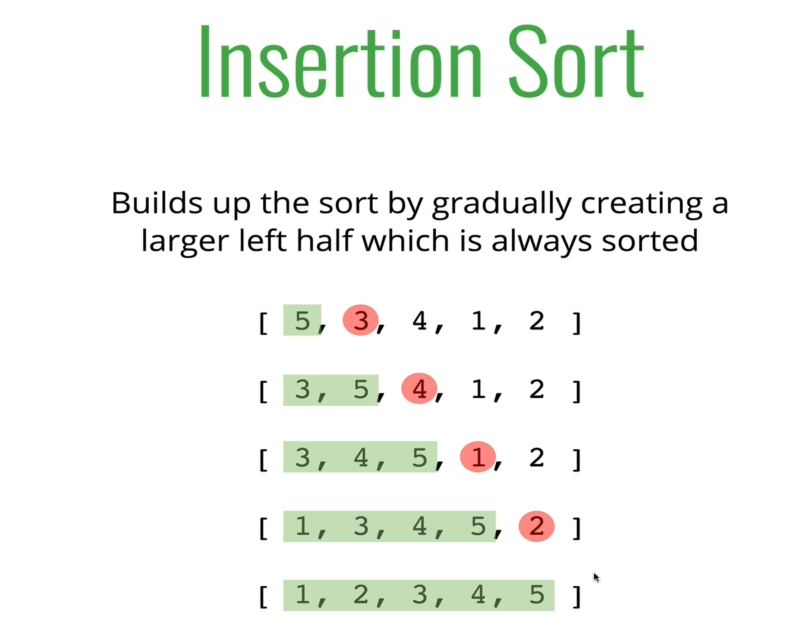
Chapter 6 : Part 3

**Sorting Algorithms: Insertion Sort**

**6.3.1 Insertion Sort**

Similar to bubble sort and selection sort, but in some cases it does a good job compared to previous tow sorting algorithms.

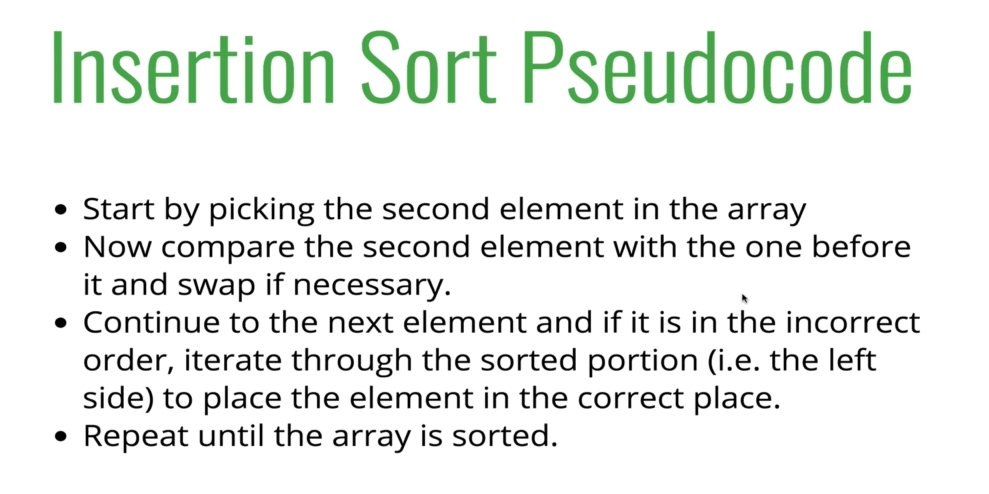
**6.3.2 How it works:**



|  |  |
| --- | --- |
|  |  |

* For an unsorted array, we take an element one at a time and put it into a right place on the sorted left side.

**6.3.3 Pseudocode**



**function** **insertSort**(arr){

**var** curRent;

**for**(**var** i=1; i**<** arr**.**length; i++){

        curRent = arr[i];

**for**(**var** j=i-1; (j**>=**0) **&&** (arr[j] **>** curRent); j--){

            // *reversed direction checking*

            // *(arr[j] > curRent); curRent is compared to left elements one by one "from right to left" or "big to samall"*

            // *(j>=0) && (arr[j] > curRent) condition is used :*

            // *To avoid unnecessary loops if the curRent is already in right place*

                // *And if the codition in the loop is true,  i.e curRent is less then the other values in the left*

                // *this loop runs and sift the elements by increasing their index*

            arr[j+1] = arr[j];

        }

        arr[j+1] = curRent;

        // *Note that: after the nested loop j holds the last value where the loop ends*

    }

**return** arr;

}

* Time complexity:
* The worst case is reverse sorted array: [9, 8, 7, 6, 5, 4, 3, 2, 1]. And the time complexity is **O(n2)**.
* The best case is nearly sorted arrays: [1, 2, 5, 3, 4, 6, 7, 8, 9].
* Used it in SERVER to store online live data: Consider a server where user put data and we can use this algorithm to put the current user data (live/streamed data) in right place where the previous data are sorted.