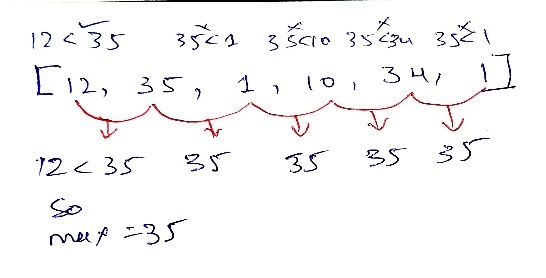
**Finding Largest Element in a Array**

arr = [12, 35, 1, 10, 34, 1]

Take the first element as ‘max’. Then loop the array to the length. In this example we are getting

**Example :-** max = arr[0] = 12

Loop from i = 1 to n-1;

* (true)12 is less than 35, So assign the max = arr[i] = 35, max value becomes 35.
* (false)35 is not less than 1, So no change.
* (false)35 is not less than 10. No change.
* (false)35 is not less than 34. No change.
* (false)35 is not less than 1. No change.
* So loops ends, the max becomes 35.

//Brute force approach { TC -> O(n log n) }

public int largest(int[] arr) {

Arrays.*sort*(arr);

int n = arr.length-1;

int max = arr[n];

return max;

}

//Optimal approach { TC -> O(n) }

public int largest1(int[] arr) {

int max = arr[0];

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

if (arr[i] > max) { //Or you can use the max < arr[i]

max = arr[i];

}

}

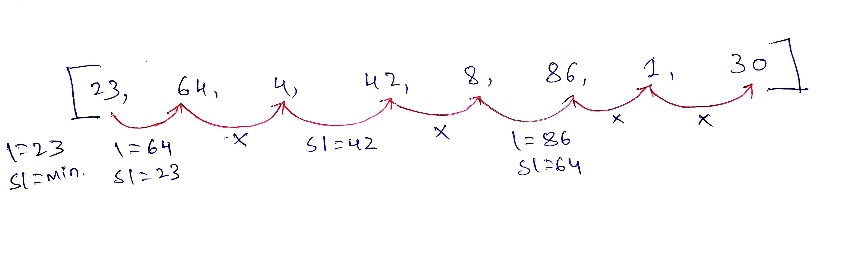
return max;

}

**Finding Second Largest Element in the Array**

arr = [23, 64, 4, 42, 8, 86, 1, 30]

Take the first element as maximum and store it to the ‘largest’ variable same as we did for finding the maximum element in the array. Then assign one variable called ‘slargest’ as ‘Integer.MIN\_VALUE’.



**Example :-** largest = arr[0] = 23, slargest = Integer.MIN\_VALUE;

Loop from i = 1 to n-1;

* (true)64 is more than 24 so, slargest = 23 and largest = 64.
* (false)4 is not more than 64. (false) 4 is not more than 23, No changes.
* (false)42 is not more than 64. (true) 42 is more than 23 so, slargest = 42.
* (false)8 is not more than 64. (false) 8 is not more than 42, No changes.
* (true)86 is more than 64 so, slargest = 64 and largest = 86.
* (false)1 is not more than 86. (false) 1 is not more than 64, No changes.
* (false)30 is not more than 86. (false) 30 is not more than 64, No changes.
* So loops ends, the largest becomes 86 and second largest is 64

//Optimal approach { TC -> O(n) }

public int getSecondLargest2(int[] arr) {

int largest = arr[0];

int slargest = Integer.***MIN\_VALUE***;

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

if(arr[i] > largest) {

slargest = largest;

largest = arr[i];

}

else if (arr[i] < largest && arr[i] > slargest) {

slargest = arr[i];

}

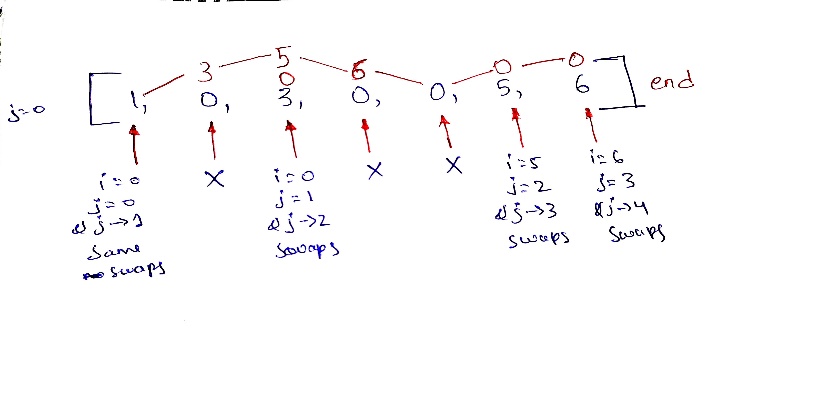
}

return slargest;

}

**Moving All Zeros to End**

arr = [1, 0, 3, 0, 0, 5, 6]

Take a the ‘j’ as 0 and run the loop through the array.

**Example :-**  j = 0

Loop from i = 0 to the n-1;

* (true) i = 0, j = 0, arr[i]/temp = 1, arr[j] = 1, so swapping the both i and j should become same {1, 0, 3, 0, 0, 5, 6} and j is updated to 1.
* (false) i = 1, arr[i] = 0, here is zero so skip the condition to move the all zeros to the end and j is remains same.
* (true) i = 2, j = 1, arr[i]/temp = 3, arr[j] = 0, so swapping the both i and j should become {1, 3, 0, 0, 0, 5, 6} (the index should considered on this changed elements)and j is updated to 2.
* (false) i = 3, j = 2, arr[i] = 0. No change.
* (false) i = 4, j = 2, arr[i] = 0. No change.
* (true) i = 5, j = 2, arr[i]/temp = 5, arr[j] = 0, so swapping should become the {1, 3, 5, 0, 0, 0, 6} and j is updated to 3.
* (true) i = 6, j = 3, arr[i]/temp = 6, arr[j] = 0, so swapping should become the

{1, 3, 5, 6, 0, 0, 0} and j is updated to 4. So Loops ends and the final array becomes all zeros are allocated to the end.

//TC -> O(n)

int j = 0;

for (int i = 0; i < arr.length; i++) {

if(arr[i] != 0) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

j++;

}

}