Searching and Sorting

- Binary search
  - → WORKS ONLY FOR SORTED ARRAY

6	10	15	32	35	40	45	60
0	1	2	3	4	5	6	7

Elem	start	End	Mid
32	0	7	$\frac{0+7}{2}=3$
35	0	7	$\frac{0+7}{2}=3$
	4	4	$\frac{4+7}{2}=5$
			4

$$mid = \frac{start + End}{2}$$

RIGHT → (mid + 1 → end)

left → (start → mid - 1)

Start > End

→ THEN STOPS

In BINARY SEARCH WE HAVE to do less work as compare to linear search

main function

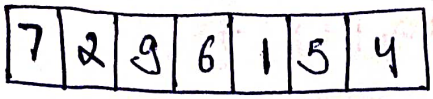
```
Public static void main(String[] args){
    int [] input = {2,4,5,8,9,15,21,28};
    int index = binarySearch (input,5);
    System.out.println(index);
}
```

#

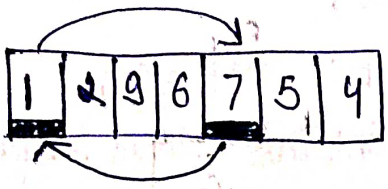
```
Public class solution{
    Public static int binarysearch (int[] arr, int x){
        int first = 0;
        int last = arr.length - 1;
        while (first <= last) {
            int mid = (first + last)/2;
            if (arr[mid] < x) {
                first = mid + 1;
            }
            else if (arr[mid] == x) {
                return mid;
            }
            else {
                last = mid - 1;
            }
        }
        return -1;
    }
}
```

```
{
    int start = 0;
    int last = input.length - 1;
while(start <= end)
{
    int mid = (start + last)/2;
    if(elem == input[mid]) {
        return mid;
    }
    else if (elem > input[mid]){
        start = mid + 1;
    }
    else { end = mid - 1;
    }
}
    return -1;
}
```

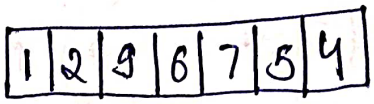
Selection SORT



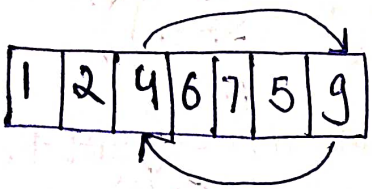
Round 0



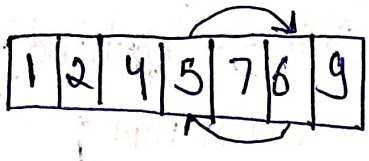
Round 1



Round 2



Round 3



Round 4

n=6 Round-6

max Round Possible

#

```
Public static void selectionSort(int[] input){
for(int i=0 ; i< input.length-1 ; i++)
{ int min = input[i] ;
  int minIndex = i ;
  for(int j= i+1 ; j<input.length ; j++)
  { if(input[j] < min)
    { min = input[j] ;
      minIndex = j ;
    }
  }if (minIndex != i ) {
      input[minIndex] = input[i] ;
      input[i] = min ;
  }
}
```

This algorithm maintains two subarrays in a given array.

1) The subarray which is already sorted.
2) Remaining subarray which is unsorted.

Selection Sort

```
Public static void main(        )
{
    int[] a = {38, 52, 9, 18, 6, 62, 13};
    int min, temp = 0;
    for(int i=0; i<a.length; i++)
    {
        min= i;
        for(int J= i+1; J<a.length; J++)
        {
            if(a[J] < a[min])
            {
                min = J;
            }
        }
        temp= a[i];
        a[i] = a[min];
        a[min] = temp;
    }
    for(int i=0; i< a.length; i++)
    {
        System.out.print(a[i] + " ");
    }
}
```

i=0
min=~~0~~ 2
J=~~1~~ ~~2~~ ~~3~~ 3 4
a[2] < a[0]
52 < 38 ×
9 < 38 ✓
~~18~~ < 38
6 < 38

Bubble SORT

ROUND-1

6	9	4	8	3	1

6 9 4 8 3 1	i=0
6 4 9 8 3 1	1
6 4 8 9 3 1	2
6 4 8 3 9 1	3
6 4 8 3 1 9	4

ROUND-2

6 4 8 3 1 9	
4 6 8 3 1 9	
4 6 8 3 1 9	
4 6 3 8 1 9	
4 6 3 1 8 9	

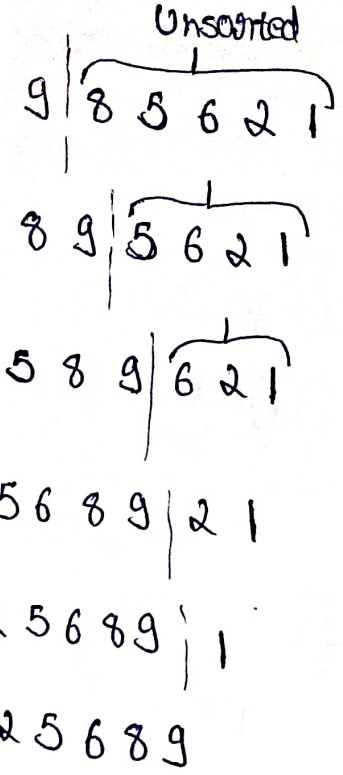
WORST TIME

$O(n^2)$

```
Public class BubbleSort {
  Public static void bubblesort(int input[]){
    for(int i=0; i< input.length-1 ; i++) {
      for(int j=0; j< input.length-i-1 ; j++){
        if(input[j] > input[j+1])
        {  int temp = input[j] ;
           input[j] = input[j+1] ;
           input[j+1] = temp ;
        }
      }
    }
  }
  Public static void main(String[] args){
     int input[] = {8,2,6,1,5};
     bubblesort(input);
     for(int i=0; i< input.length ; i++) {
        System.out.print(input[i] + " ");
     }
  }
```

30
45
30
105

Insertion SORT



9	8	5	6	2	1

Time complexity $O(n^2)$

```
# Public class InsertionSort{
    Public static void insertionSort(int arr[]){
        for(int i=1; i< arr.length ; i++){
        int j= i-1;
        int temp= arr[i];
        while(j>=0 && arr[j] > temp){
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = temp;
    }
}
    Public static void main(String[] args){
        int arr[]= {6,4,3,5,2,1,9};
        INSERTIONSORT(arr);
        for(int i=0; i< arr.length ; i++){
            System.out.print(arr[i] + " ");
        }
    }
}
```

i=1
j= 0 + 1
temp= 4 ;

arr[1] = 6
Arraysort( ) ;

i=1
j= 0
temp= 4
arr[0] = 6

6 4

6

MERGE TWO SORTED ARRAY

Arr 1

2	6	9	15

Arr 2

1	4	6	7

1	2	4	6	7	9	15

```
# Public class mergeTwoSortedArrays {
    Public static int[] merge(int[] arr1, int[] arr2) {
     int i=0;
     int j=0;
     int k=0;
     int ans[] = new int[arr1.length + arr2.length];
     while(i<arr1.length && j< arr2.length)
     { if(arr1[i] < arr2[j]) {
          ans[k] = arr1[i];
           i++;
           k++;
       }
       else {
          ans[k] = arr2[j];
           j++;
           k++;
       }
    }
```

```
while (i < arr1.length) {
    ans[k] = arr1[i];
    k++;
    i++;
}

while (j < arr2.length) {
    ans[k] = arr2[j];
    j++;
    k++;
}

    return ans;
}

public static void main(String[] args) {
    int arr1[] = {1,3,5,7};
    int arr2[] = {2,4,6};
    int ans[] = merge(arr1, arr2);
    for (int i=0; i < ans.length; i++){
        System.out.print(ans[i] + " ");
    }
    System.out.println();
  }
}
```

Push zero to END

9 | 8 | 2 | 4

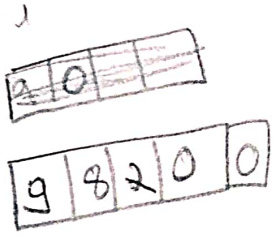
```
Public class solution {
Public static void pushzeroAtEnd(int[] arr)
{
int J=0;
int n = arr.length;
for(int i=0; i<n; i++){
if(arr[i]!=0) {
int temp = arr[J];
arr[J] = arr[i];
arr[i] = temp;
J++;
}
}
}
```

0	1	2	3	4
9	0	0	8	2

```
for(int j=0; j < arr.length; j++){
if(arr[j]==0) {
count++;
}
else {
system.out.print(arr[j]);
}
}
```

```
While(i ≤ COUNT)
{
system.out.print(0);
j++;
}
```

→ Time complexity JADA HAA.

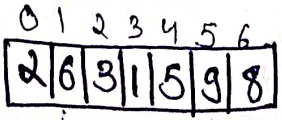


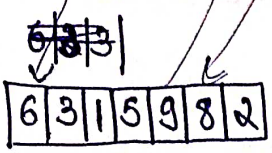
j=0 X X X 4
J=0 X 2 temp=9;

temp = 0;

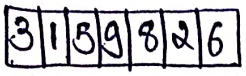
J=0 1
J=0 X X X 4

ROTATE ARRAY —— X ——





R=1



R=2

0 1 2 3 4 5 6

```
while(d<R)
{
  temp = a[0];  -> for(i=0; i<arr.leg-1,
  a[i] = a[i+1];
  d++;
}
a[arr.length-1] = temp;
```

2nd method

—— d to end elements

{ arr[i] = arr[i+d] }

i <= [arr.length - 1 - d]

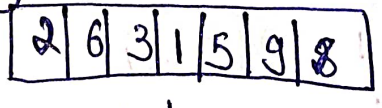
arr[0] = arr[0+2]

```
[] int arr1 = new int[] arr;
for(i=0; i<=d; i++)
{ arr[i] = arr[i];
}
```

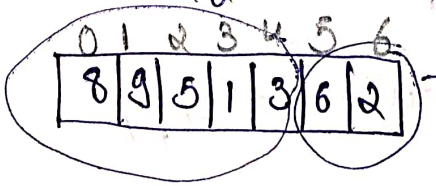
i = arr.leg - 1 - d ; i < arr.length - 1 d =

7 - 2

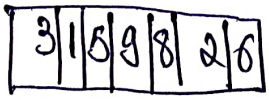
method - (3)



n-d



Reverse the array



2 method

```
{
    int [] arr1 = new int[d];
    for(int k=0; k<d ; k++) {
        arr1[k] = arr[k];
    }

    for(int i=0 ; i<=(arr.length -1-d) ; i++){
        arr[i] = arr[i+d];
    }

    int r=0 ;
    for(int i = (arr.length - d) ; i< arr.length ; i++){
        arr[i] = arr1[r];
        r++ ;
    }
}
```

#

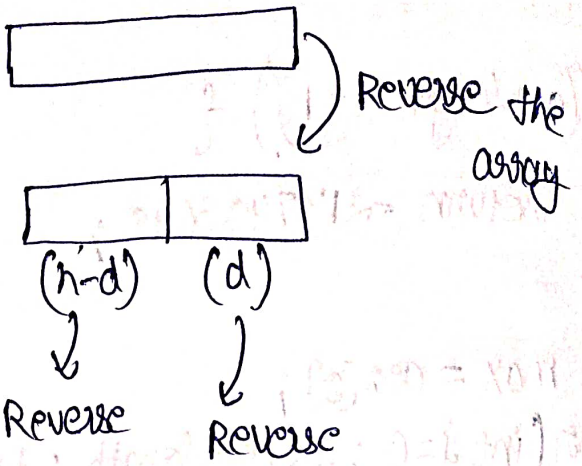
```
{ int r=0 ;
  while(r<d){
    int temp = arr[0];
    for(int i=0 ; i<arr.length-1 ; i++ ) {
        arr[i] = arr[i+1];
    } r++;

    arr[arr.length-1] = temp;
  }
}
```

Rev an array

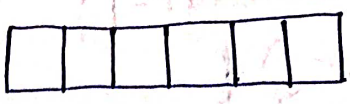
```
int i=0 ;
int j= arr.length-1 ;

while(i<j)
{ int temp = arr[i];
  arr[i] = arr[j] ;
  arr[j] = temp ;
  i++ ;
  j-- ;
}
```

#

```
{
    int i=0;
    int j= arr.length-1;
    while(i<j){
      int temp= arr[i];
      arr[i] = arr[j];
      arr[j] = temp;
       i++;
       j--;
    }

    i=0;
    j= arr.length-1-d;
    while(i<j){
       int temp= arr[i];
       arr[i]= arr[j];
       arr[j]= temp;
        i++;
        j--;
    }

    i= arr.length-d;
    j= arr.length-1;
    while(i<j){
        int temp= arr[i];
        arr[i]= arr[j];
        arr[j]= temp;
         i++;
         j--;
    }
}
```

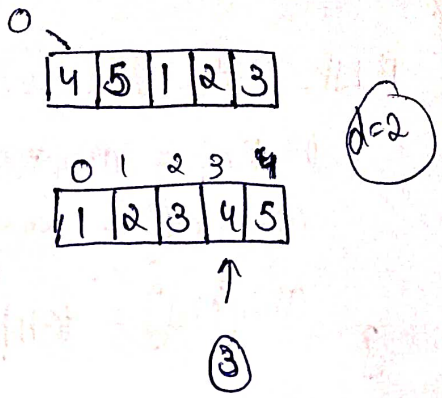


Second Largest In Array



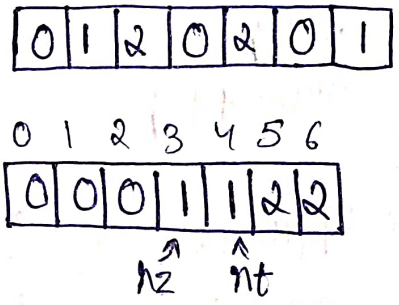
```
{
    if((arr.length <= 1)) {
        return -2147483648;
    }

    int max = arr[0];
    for(int i=0; i<arr.length; i++) {
        if(arr[i] > max) {
            max = arr[i];
        }
    }

    for(int i=0; i<arr.length; i++) {
        if(arr[i] == max) {
            arr[i] = Integer.MIN_VALUE;
        }
    }

    int second = arr[0];
    for(int i=0; i<arr.length; i++){
        if(arr[i] > second) {
            second = arr[i];
        }
    }
    return second;
}
```

Check Array Rotations




```
{
    int min = Integer.MAX_VALUE;
    int minIndex = 0;
    for(int j=0; j< arr.length ; j++){
      if(arr[j] < min){
        min = arr[j];
        minIndex = j;
      }
    return minIndex;
}
```

SORT 0 1 2




```
int nz = 0;
int nt = arr.length-1;
for(j=0; ---          )
{ if(arr[i]==0)
   { arr[nz] = 0;
     nz++;
   }
  else if(arr[i]==2)
   { arr[nt] = 2;
     nt--;
   }
}
for(j=nz; j<= nt ; j++).
```

#

```
{
    Public static void swap(int arr[], int start, int end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
    }

    Public static void sort012(int[] arr) {
        int start = 0;
        int end = arr.length - 1;
        int i = 0;
        while (i <= end) {
            if (arr[i] == 0) {
                swap(arr, i, start);
                i++;
                start++;
            }
            else if (arr[i] == 2) {
                swap(arr, i, end);
                end--;
            }
            else {
                i++;
            }
        }
    }
}
```