Array Notes

- To initialize an array we do like this
  int arr[] = new int[10];
- The indexing will start from 0 to n-1
- When we initialize the array, all the value by default initialize to 0.
- If we try to access element outside the indexing range then we will get array index out of bond error.
- boolean arr[] = new boolean[5]; , for Boolean array the by default all the value will always initialize to false.

1. **Taking input and print of an array(without using function)**

```
import java.util.Scanner;

public class InputAndOutput {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Scanner s = new Scanner(System.in);
		System.out.println("Enter the size of array: ");5
		int n = s.nextInt();

		int arr[] = new int[n];

		// For taking Input

		for(int i = 0 ; i<n ; i++)
		{
			System.out.println("Enter the "+i+"th element of the
array:");
			arr[i] = s.nextInt();
		}

		// For printing output

		for(int i = 0 ; i<n ; i++)
		{
			System.out.println("The "+i+"th element is "+arr[i]);
		}
	}
}
```

2. **Taking input and print the array(using function)**

```
import java.util.Scanner;

public class InputAndOutputUsingFunction {

	public static int[] TakeInput()
```

```
	{
		Scanner s = new Scanner(System.in);
		System.out.println("Enter the size of array: ");
		int n = s.nextInt();

		int arr[] = new int[n];

		// For taking Input

		for(int i = 0 ; i<n ; i++)
		{
			System.out.println("Enter the "+i+"th element of the
array:");
			arr[i] = s.nextInt();
		}

		return arr;
	}

	public static void PrintArray(int[] arr)
	{
		// For printing output

		int n = arr.length;

		for(int i = 0 ; i<n ; i++)
		{
		  System.out.println("The "+i+"th element is "+arr[i]);
		}
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		int arr[] = TakeInput();
		PrintArray(arr);

	}

}
```

3. **Finding the sum of the given array**

```
package array;

import java.util.Scanner;

public class ArraySum {
```

```
    public static int[] TakeInput()
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the size of
array: ");
        int n = s.nextInt();

        int arr[] = new int[n];

        // For taking Input

        for(int i = 0 ; i<n ; i++)
        {
            System.out.println("Enter the
"+i+"th element of the array:");
            arr[i] = s.nextInt();
        }

        return arr;
    }

    public static void SumOfArray(int arr[])
    {
        int sum = 0;
        for(int i = 0 ; i<arr.length; i++)
        {
            sum += arr[i];
        }

        System.out.print("The sum of arr is "+
sum);
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int arr[] = TakeInput();
        SumOfArray(arr);
```

```
    }

}
```

4. **Populate the number and arrange the array**

```
5.package array;
6.
7.import java.util.Scanner;
8.
9.public class ArrangeNumber {
10.
11.        public static int[] ArrangeArray()
12.        {
13.             Scanner s = new Scanner(System.in);
14.             System.out.println("Enter the size
  of the array: ");
15.             int size = s.nextInt();
16.             int arr[] = new int[size];
17.             int start = 0 , end = size-1;
18.             int num = 1;
19.
20.             while(start<=end)
21.             {
22.             if(num%2==1)
23.             {
24.                  arr[start] = num;
25.                  num++;
26.                  start++;
27.             }
28.             else
29.             {
30.                  arr[end]=num;
31.                  num++;
32.                  end--;
33.             }
34.
35.             }
36.
37.             return arr;
```

```
38.         }
39.
40.         public static void PrintArray(int arr[])
41.         {
42.             for(int i = 0 ; i<arr.length ; i++)
43.             {
44.                 System.out.println(arr[i]);
45.             }
46.
47.         }
48.
49.         public static void main(String[] args) {
50.             // TODO Auto-generated method stub
51.             Scanner s = new Scanner(System.in);
52.             System.out.println("Enter the number
  of test cases : ");
53.             int t = s.nextInt();
54.             while(t>0)
55.             {
56.                int arr[] = ArrangeArray();
57.                PrintArray(arr);
58.             }
59.
60.
61.
62.         }
63.
64.     }
65.
```

Output:

```
Enter the number of test cases :
1
Enter the size of the array:
5
1
3
5
4
2
```

Array Notes

- **Data types**

  - ❖ Primitive Data type (int, char, double , float etc.)
  - ❖ Non Primitive data type(string , Scanner , array etc.)

Note: The difference between both of them is that, primitive data type store the actual value of that variable but in case of non-primitive data type it store the reference/address of that location of that variable which it is storing. In case of array and string it also store the length of the array as well.

- If we try to print array reference

```
public static void PrintArray(int arr[])
    {
        for(int i = 0 ; i<arr.length ; i++)
        {
            System.out.println(arr[i]);
        }

        System.out.println(arr);
```

It will print the reference like this:

```
[I@31221be2
```

Here [denotes an 1D array
I denote it is of integer type and after @ we have the reference of the variable.

- ❖ In case of array and non-primitive data type we pass the reference of the array so whatever change we do in the passed array function will also make changes to main array. This is not the case with primitive data type variable.

```
import java.util.Scanner;
public class PrimitivesAndNonPrimitives {
    public static void increment(int i){
        i++;
    }
    public static void printArray(int[]arr){
        int n=arr.length;
        for(int i=0;i<n;i++){
            System.out.println(arr[i]);
```

```
        }
    }

   public static void incrementArray(int[] arr){
       for(int i=0;i<arr.length;i++){
           arr[i]=arr[i]+1;
       }
   }

public static void main(String args[]) {
     //int i=10;
     //increment(i);
     //System.out.println(i);
     int[] arr={1,2,3,4,5};
     incrementArray(arr);
     printArray(arr);
    }
}
```

Output will be 2 3 4 5 6

- Program to print all pair in an array

```
package array;

import java.util.Scanner;

public class ArrayPair {

    public static int[] TakeInput()
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the size of array:
");
        int n = s.nextInt();

        int arr[] = new int[n];

        // For taking Input

        for(int i = 0 ; i<n ; i++)
        {
            System.out.println("Enter the "+i+"th
element of the array:");
```

```
            arr[i] = s.nextInt();
        }

        return arr;
    }

    public static void PrintArray(int[] arr)
    {
        // For printing output

        int n = arr.length;

        for(int i = 0 ; i<n ; i++)
        {
          System.out.println("The "+i+"th element is
"+arr[i]);
        }
    }

    public static void PrintPair(int arr[])
    {
        for(int i = 0 ; i<arr.length; i++)
        {
            for(int j = i ; j<arr.length; j++)
            {
                System.out.println(arr[i] + " " +
arr[j]);
            }
        }
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int arr[] = TakeInput();
        PrintArray(arr);
        PrintPair(arr);

    }

}
```

Output :

```
Enter the size of array:
5
Enter the 0th element of the array:
1
Enter the 1th element of the array:
2
Enter the 2th element of the array:
3
Enter the 3th element of the array:
4
Enter the 4th element of the array:
5
The 0th element is 1
The 1th element is 2
The 2th element is 3
The 3th element is 4
The 4th element is 5
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
5 5
```

- To find the duplicate element in the array

```
package array;

import java.util.Scanner;

public class UniqueElements {
```

```
    public static int[] TakeInput()
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the size of array:
");
        int n = s.nextInt();

        int arr[] = new int[n];

        // For taking Input

        for(int i = 0 ; i<n ; i++)
        {
            System.out.println("Enter the "+i+"th
element of the array:");
            arr[i] = s.nextInt();
        }

        return arr;
    }

    public static void Unique(int arr[])
    {
        for(int i = 0 ; i<arr.length; i++)
        {
            int count = 0;
            for(int j = 0 ; j<arr.length ; j++)
            {
                if(arr[i] == arr[j])
                {
                    count++;
                }
            }

            if(count == 1)
            {
                System.out.print(arr[i] + " ");
            }
        }
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
      int arr[]=TakeInput();
```

```
        Unique(arr);
    }

}
```

Output :

```
Enter the size of array:
5
Enter the 0th element of the array:

10
Enter the 1th element of the array:
20
Enter the 2th element of the array:
30
Enter the 3th element of the array:
10
Enter the 4th element of the array:
40
20 30 40
```

- To find the duplicate element in the array

```
package array;

import java.util.Scanner;

public class Dublicate {

    public static int[] TakeInput()
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the size of array:
");
        int n = s.nextInt();

        int arr[] = new int[n];

        // For taking Input

        for(int i = 0 ; i<n ; i++)
        {
```

```
            System.out.println("Enter the "+i+"th
element of the array:");
            arr[i] = s.nextInt();
        }

        return arr;
    }

    public static void DuplicateElement(int arr[])
    {
        for(int i = 0 ; i<arr.length; i++)
        {
            int count = 0;
            for(int j = i ; j<arr.length ; j++)
            {
                if(arr[i] == arr[j])
                {
                    count++;
                }

            }

            if(count > 1)
            {
                System.out.print(arr[i] + " ");
            }
        }
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the test cases :
");
        int t = s.nextInt();
        while(t>0)
        {
         int arr[] = TakeInput();
         DuplicateElement(arr);
         t--;
        }
    }

}
```

Output:

```
Enter the test cases :
1
Enter the size of array:
5
Enter the 0th element of the array:
10
Enter the 1th element of the array:
20
Enter the 2th element of the array:
30
Enter the 3th element of the array:
40
Enter the 4th element of the array:
20
20
```

- Intersection in two array

```
package array;

import java.util.Scanner;
public class Intersection {

    public static int[] TakeInput()
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the size of array :
");
        int n = s.nextInt();
        int arr[]=new int[n];

        for(int i=0;i<n;i++)
        {
            System.out.print("Enter the "+i+"th
element: ");
            arr[i]=s.nextInt();
        }
        return arr;
    }
    public static void IntersectionElement(int
input1[],int input2[])
    {
```

```
        for(int i = 0 ; i<input1.length; i++)
        {
            for(int j = 0; j<input2.length ; j++)
            {
                if(input1[i]==input2[j])
                {
                    System.out.println(input1[i]);
                    input2[j]=Integer.MIN_VALUE;
                    break;
                }
            }
        }
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Scanner s = new Scanner(System.in);
        System.out.print("Enter the no of test cases
: ");
        int t = s.nextInt();

        while(t>0)
        {
            int arr1[]=TakeInput();
            int arr2[]=TakeInput();
            IntersectionElement(arr1,arr2);
            t--;
        }

    }

}
```

Output:

```
Enter the no of test cases : 1
Enter the size of array : 5
Enter the 0th element: 10
Enter the 1th element: 20
Enter the 2th element: 30
Enter the 3th element: 40
Enter the 4th element: 50
Enter the size of array : 4
Enter the 0th element: 1
```

```
Enter the 1th element: 5
Enter the 2th element: 6
Enter the 3th element: 10
10
```

- Pair sum program

```
package array;

import java.util.Scanner;

public class PairSum {

    public static int[] TakeInput()
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the size of
array : ");
        int n = s.nextInt();
        int arr[]=new int[n];

        for(int i=0;i<n;i++)
        {
            System.out.print("Enter the "+i+"th
element: ");
            arr[i]=s.nextInt();
        }
        return arr;
    }

    public static void SumPair(int arr[])
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the pair sum :
");
        int sum = s.nextInt();
        int pairs = 0;
        for(int i  = 0 ; i<arr.length; i++)
```

```
        {

            for(int j = i+1; j<arr.length; j++)
            {
                if(arr[i]+arr[j] == sum)
                    pairs++;
            }
        }
        System.out.print(pairs);
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
          int arr[] = TakeInput();
          SumPair(arr);
    }

}
```

Output:

```
Enter the size of array : 5
Enter the 0th element: 1
Enter the 1th element: 2
Enter the 2th element: 3
Enter the 3th element: 4
Enter the 4th element: 5
Enter the pair sum : 6
2
```

- Triplet Sum in an array

```
package array;

import java.util.Scanner;

public class TripletSum {

    public static int[] TakeInput()
```

```
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the size of
array : ");
        int n = s.nextInt();
        int arr[]=new int[n];

        for(int i=0;i<n;i++)
        {
            System.out.print("Enter the "+i+"th
element: ");
            arr[i]=s.nextInt();
        }
        return arr;
    }

    public static void Triplet(int arr[])
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the pair sum :
");
        int sum = s.nextInt();
        int Triplet = 0;
        for(int i  = 0 ; i<arr.length; i++)
        {

            for(int j = i+1; j<arr.length; j++)
            {
                for(int k=j+1; k<arr.length;
k++)
                {
                    if(arr[i]+arr[j]+arr[k] ==
sum)
                        Triplet++;
                }

            }
        }
```

```
        System.out.print(Triplet);
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
           int arr[] = TakeInput();
           Triplet(arr);
    }

}
```

Output:

```
Enter the size of array : 7
Enter the 0th element: 1
Enter the 1th element: 2
Enter the 2th element: 3
Enter the 3th element: 4
Enter the 4th element: 5
Enter the 5th element: 6
Enter the 6th element: 7
Enter the pair sum : 12
5
```

- Sort 0,1 in one scan

```
package array;

import java.util.Scanner;

public class Sort01 {

    public static int[] TakeInput()
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the size of
array : ");
```

```
        int n = s.nextInt();
        int arr[]=new int[n];

        for(int i=0;i<n;i++)
        {
            System.out.print("Enter the
"+i+"th element: ");
            arr[i]=s.nextInt();
        }
        return arr;
    }

    public static void Sort(int arr[])
    {
        int i = 0 ;
        int j = arr.length-1;
        while(i<j)
        {
            if(arr[i]==0)
            {
                i++;
            }
            else
            {
                if(arr[j]==1)
                {
                    j--;
                }
                else
                {
                    int temp = arr[i];
                    arr[i]=arr[j];
                    arr[j]=temp;
                }
```

```
            }
        }
    }

    public static void PrintArray(int[]
arr)
    {
        // For printing output

        int n = arr.length;

        for(int i = 0 ; i<n ; i++)
        {
          System.out.println("The "+i+"th
element is "+arr[i]);
        }
    }
    public static void main(String[] args)
{
        // TODO Auto-generated method stub
            int arr[] = TakeInput();
            Sort(arr);
            PrintArray(arr);
    }

}
```

Output:

```
Enter the size of array : 6
Enter the 0th element: 1
Enter the 1th element: 1
Enter the 2th element: 1
Enter the 3th element: 0
```

```
Enter the 4th element: 0
Enter the 5th element: 0
The 0th element is 0
The 1th element is 0
The 2th element is 0
The 3th element is 1
The 4th element is 1
The 5th element is 1
```

- Binary Search

Binary Search is defined as a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(log N).

Conditions for when to apply Binary Search in a Data Structure:
To apply Binary Search algorithm:
- The data structure must be sorted.
- Access to any element of the data structure takes constant time.

- *Step to approach BSA*

The general steps for both methods are discussed below.

1. The array in which searching is to be performed is:

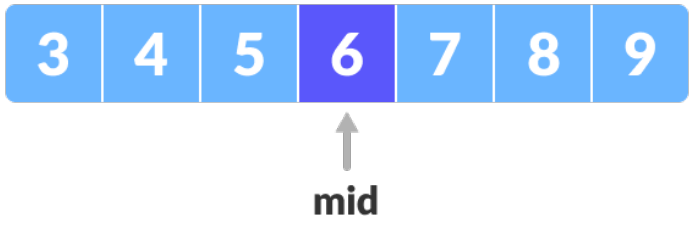
Initial array

Let `x = 4` be the element to be searched.

2. Set two pointers low and high at the lowest and the highest positions respectively.

Setting pointers

3. Find the middle element `mid` of the array ie. `arr[(low + high)/2] = 6`.



Mid element

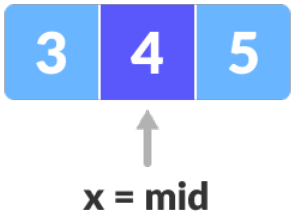
4. If x == mid, then return mid. Else, compare the element to be searched with m.
5. If `x > mid`, compare `x` with the middle element of the elements on the right side of `mid`. This is done by setting `low` to `low = mid + 1`.
6. Else, compare `x` with the middle element of the elements on the left side of `mid`. This is done by setting `high` to `high = mid - 1`.

Finding mid element

7. Repeat steps 3 to 6 until low meets high.

Mid element

8. `x = 4` is found.



Found

```
package array;

import java.util.Scanner;

public class BinarySearch {

    public static int[] TakeInput()
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the size of
array : ");
        int n = s.nextInt();
        int arr[]=new int[n];
```

```
        System.out.println("Enter the array
sorted element :");
        for(int i=0;i<n;i++)
        {
            arr[i]=s.nextInt();
        }
        return arr;
    }

    public static int BS(int arr[])
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the element
to be searched :");
        int target = s.nextInt();
        int start = 0;
        int mid;
        int end=arr.length-1;

        while(start<=end)
        {
            mid = (start+end)/2;
            if(target==arr[mid])
            {
                return mid;
            }
            else
            {
                if(target<arr[mid])
                {
                    end = mid-1;
                }
                else
                {
```

```
                    start=mid+1;
                }

            }
        }
        return -1;
    }

    public static void main(String[] args)
{
        // TODO Auto-generated method stub
        int arr[]=TakeInput();
        System.out.println(BS(arr));

    }

}
```

Output:

```
Enter the size of array : 5
Enter the array sorted element :
10
20
30
40
60
Enter the element to be searched :30
2
```

• Selection Sort Algorithm

Selection sort is a sorting algorithm that selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list.

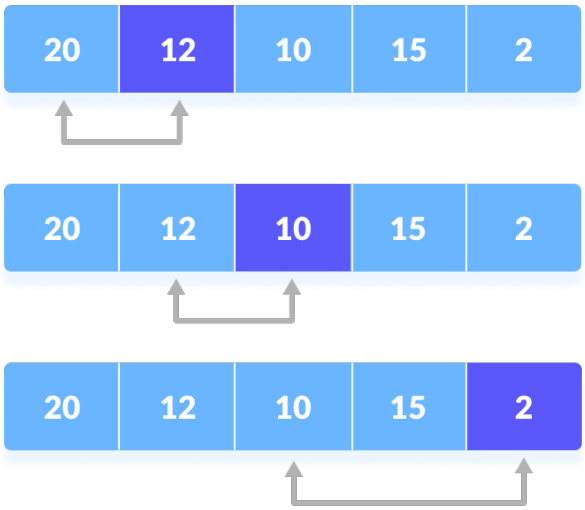
Working of Selection Sort

1. Set the first element as `minimum`.

Select first element as minimum

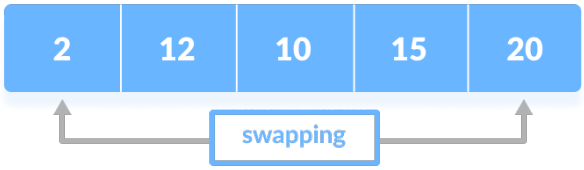
2. Compare `minimum` with the second element. If the second element is smaller than `minimum`, assign the second element as `minimum`.

   Compare `minimum` with the third element. Again, if the third element is smaller, then assign `minimum` to the third element otherwise do nothing. The process goes on until the last element.



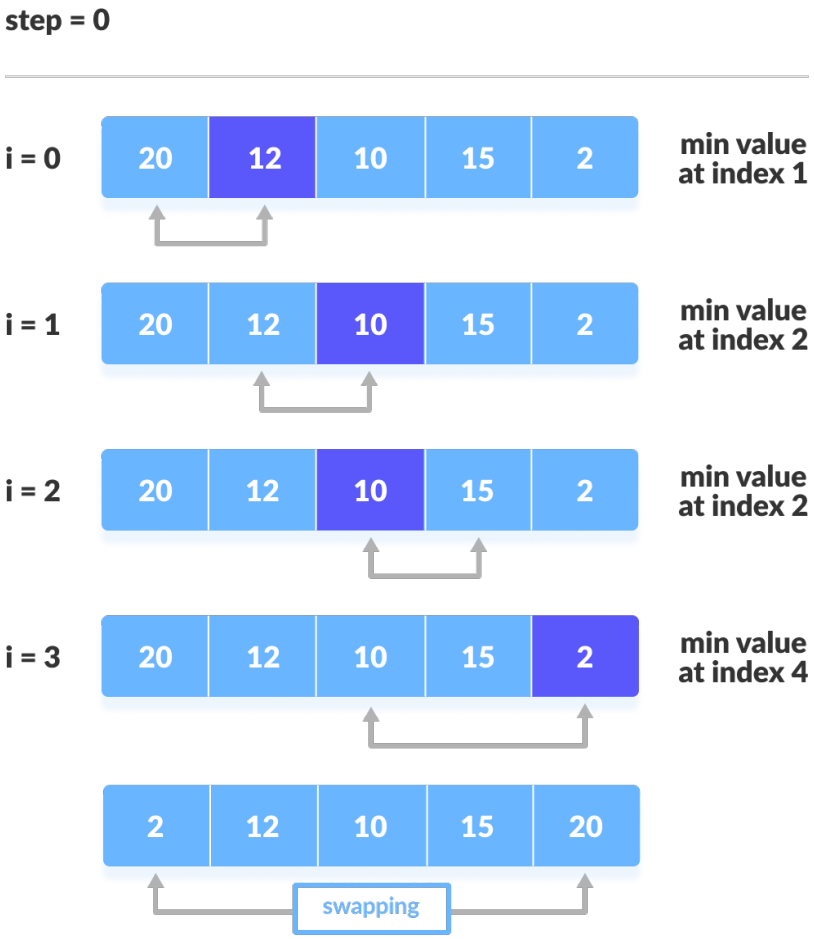
Compare minimum with the remaining elements

3. After each iteration, `minimum` is placed in the front of the unsorted list.

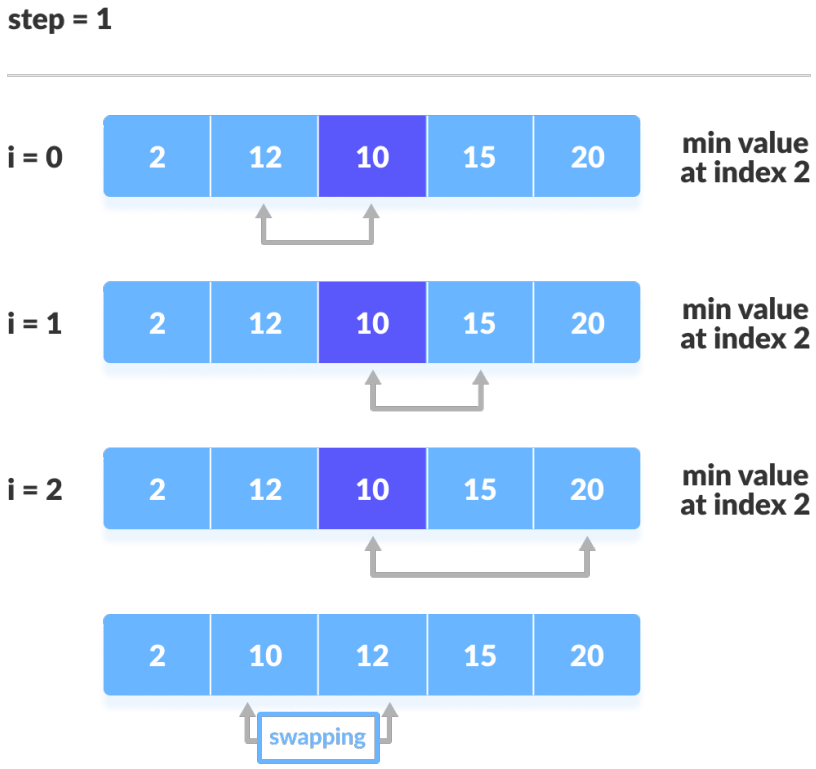


Swap the first with minimum

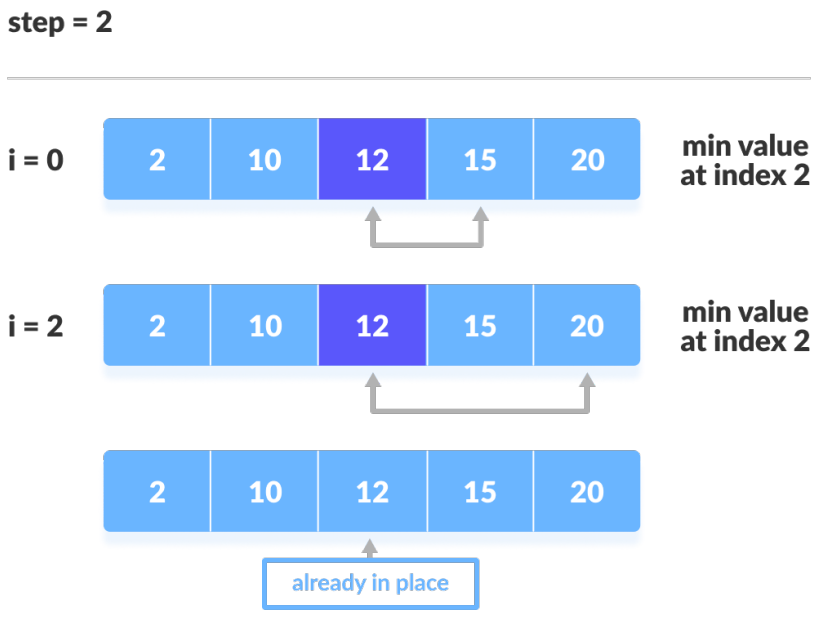
4. For each iteration, indexing starts from the first unsorted element. Step 1 to 3 are repeated until all the elements are placed at their correct positions.



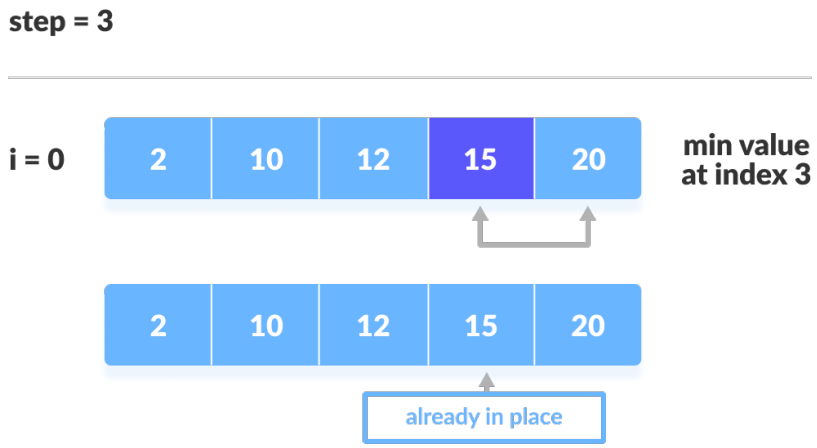
The first iteration



The second iteration



The third iteration



The fourth iteration

```
package array;

import java.util.Scanner;

public class Practice {

    public static void SelectionSort(int arr[])
    {
        int n = arr.length;
        for(int i = 0 ; i<n ; i++)
        {
            int min=arr[i];
            int minIndex= i;
            for(int j = i; j<n;j++)
            {
                if(arr[j]<min)
                {
                    min = arr[j];
                    minIndex=j;
                }

            }
            //Swapping of element
            int temp = arr[i];
            arr[i] = arr[minIndex];
```

```
                arr[minIndex]=temp;
            }
        }

        public static void PrintArray(int arr[])
        {
            for(int i = 0 ; i<arr.length; i++)
            {
                System.out.print(arr[i]+" ");
            }
        }

        public static void main(String[] args)
        {
            // TODO Auto-generated method stub
          Scanner s = new Scanner(System.in);
            int arr[]= {50,60,80,20,39,40,70};
            SelectionSort(arr);
            PrintArray(arr);
        }
}
```

Output:

```
20 39 40 50 60 70 80
```

Bubble Sort

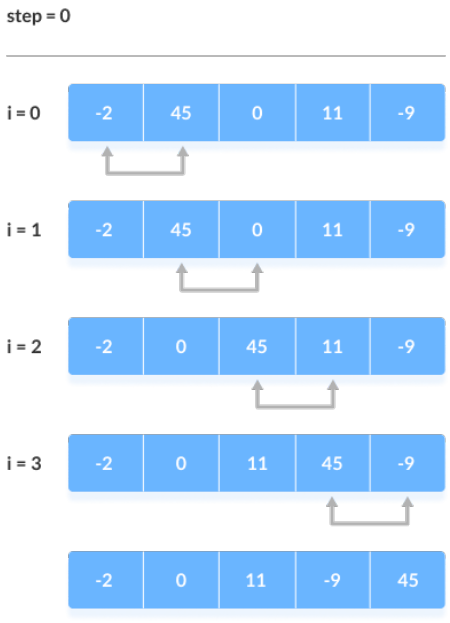
Bubble sort is a sorting algorithm that compares two adjacent elements and swaps them until they are in the intended order.
Just like the movement of air bubbles in the water that rise up to the surface, each element of the array move to the end in each iteration. Therefore, it is called a bubble sort.

Working of Bubble Sort

Suppose we are trying to sort the elements in **ascending order**.

1. First Iteration (Compare and Swap)

1. Starting from the first index, compare the first and the second elements.
2. If the first element is greater than the second element, they are swapped.
3. Now, compare the second and the third elements. Swap them if they are not in order.
4. The above process goes on until the last element.

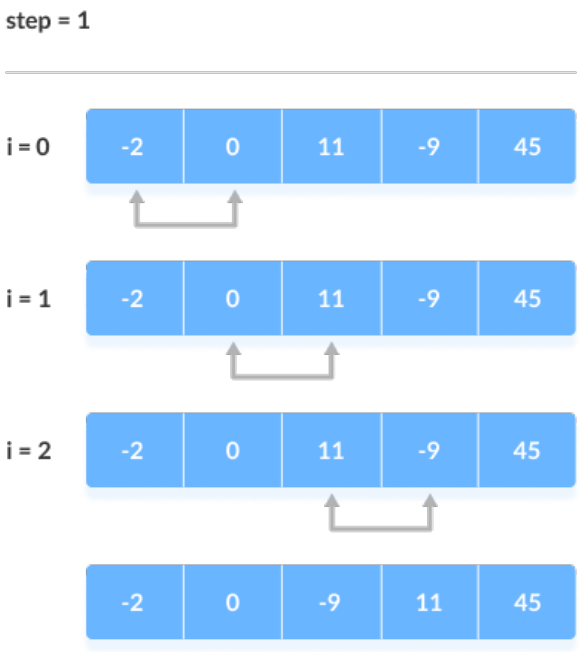


Compare the Adjacent Elements

2. Remaining Iteration

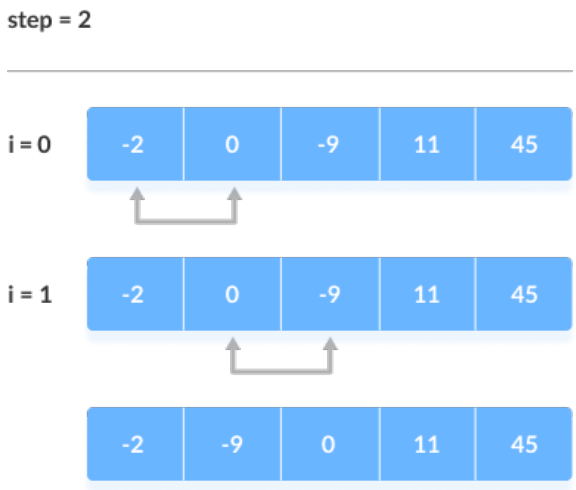
The same process goes on for the remaining iterations.

After each iteration, the largest element among the unsorted elements is placed at the end.



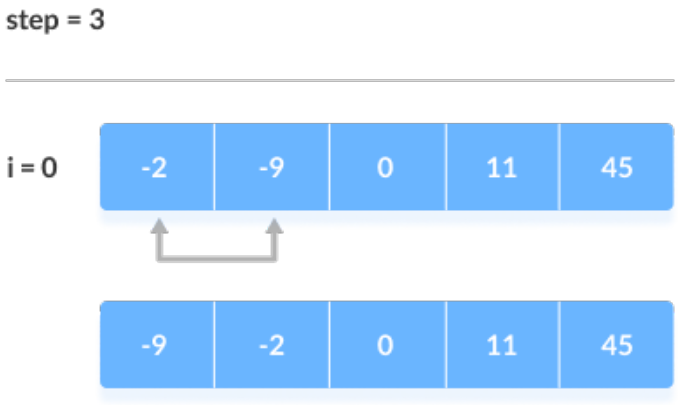
Put the largest element at the end

In each iteration, the comparison takes place up to the last unsorted element.



Compare the adjacent elements

The array is sorted when all the unsorted elements are placed at their correct positions.



The array is sorted if all elements are kept in the right order

```
package array;

import java.util.Scanner;

public class Practice {

    public static void BubbleSort(int arr[])
    {
        int n = arr.length;
        for(int i = 0 ; i<n-1 ; i++)
        {
            for(int j = 0 ; j<n-1-i;j++)
            {
                if(arr[j]>arr[j+1])
                {
                    int temp = arr[j];
                    arr[j]= arr[j+1];
                    arr[j+1]=temp;
                }
            }
        }
    }
```

```
    public static void PrintArray(int arr[])
    {
        for(int i = 0 ; i<arr.length; i++)
        {
            System.out.print(arr[i]+" ");
        }
    }

    public static void main(String[] args)
    {
        // TODO Auto-generated method stub
       Scanner s = new Scanner(System.in);
        int arr[]= {50,60,80,20,39,40,70};
        BubbleSort(arr);
        PrintArray(arr);
    }


}
```

Output:

```
20 39 40 50 60 70 80
```

Insertion Sort Algorithm

Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration.
Insertion sort works similarly as we sort cards in our hand in a card game.

We assume that the first card is already sorted then, we select an unsorted card. If the unsorted card is greater than the card in hand, it is placed on the right otherwise, to the left. In the same way, other unsorted cards are taken and put in their right place.

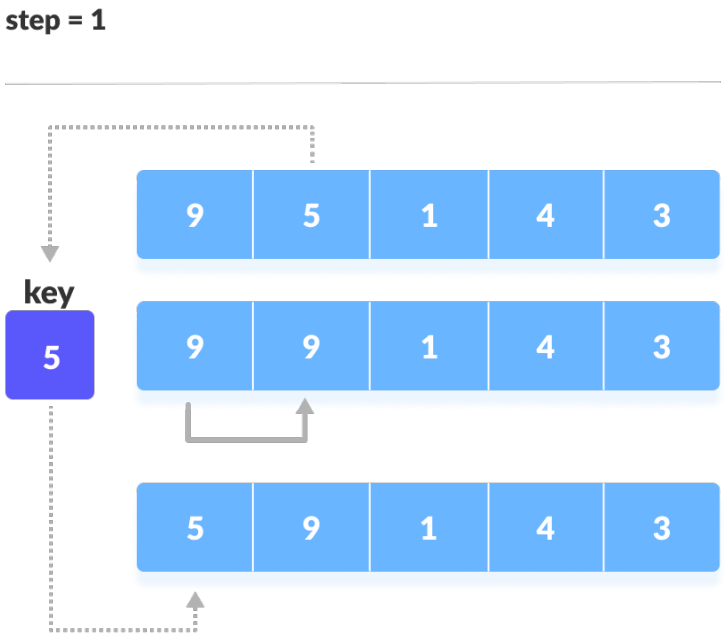
A similar approach is used by insertion sort.

Working of Insertion Sort

Suppose we need to sort the following array.

Initial array

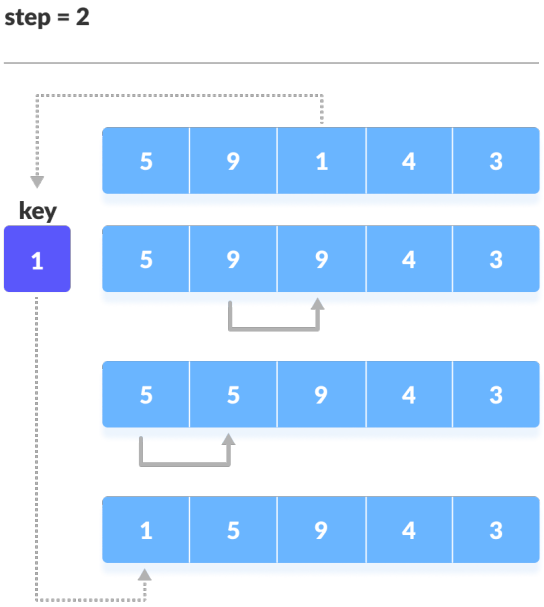
1. The first element in the array is assumed to be sorted. Take the second element and store it separately in `key`.
   Compare `key` with the first element. If the first element is greater than `key`, then `key` is placed in front of the first element.



If the first element is greater than key, then key is placed in front of the first element.

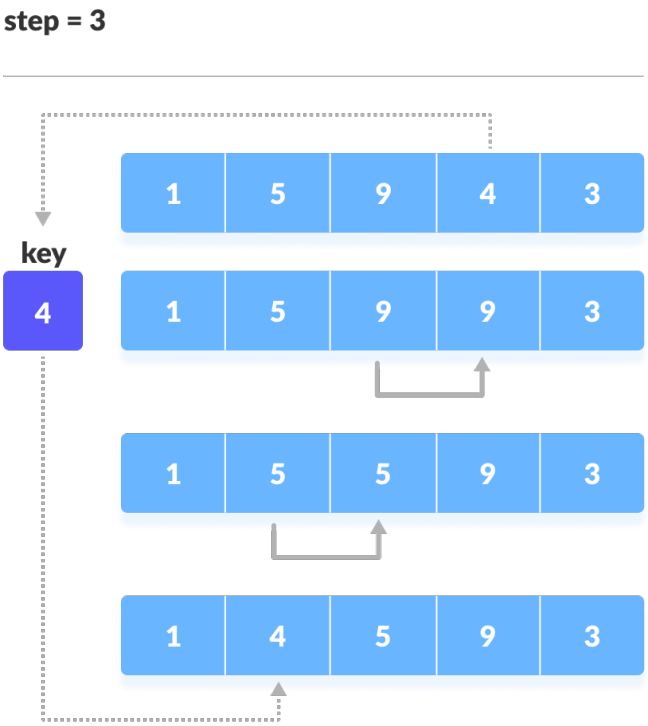
2. Now, the first two elements are sorted.
   Take the third element and compare it with the elements on the left of it.

Placed it just behind the element smaller than it. If there is no element smaller than it, then place it at the beginning of the array.

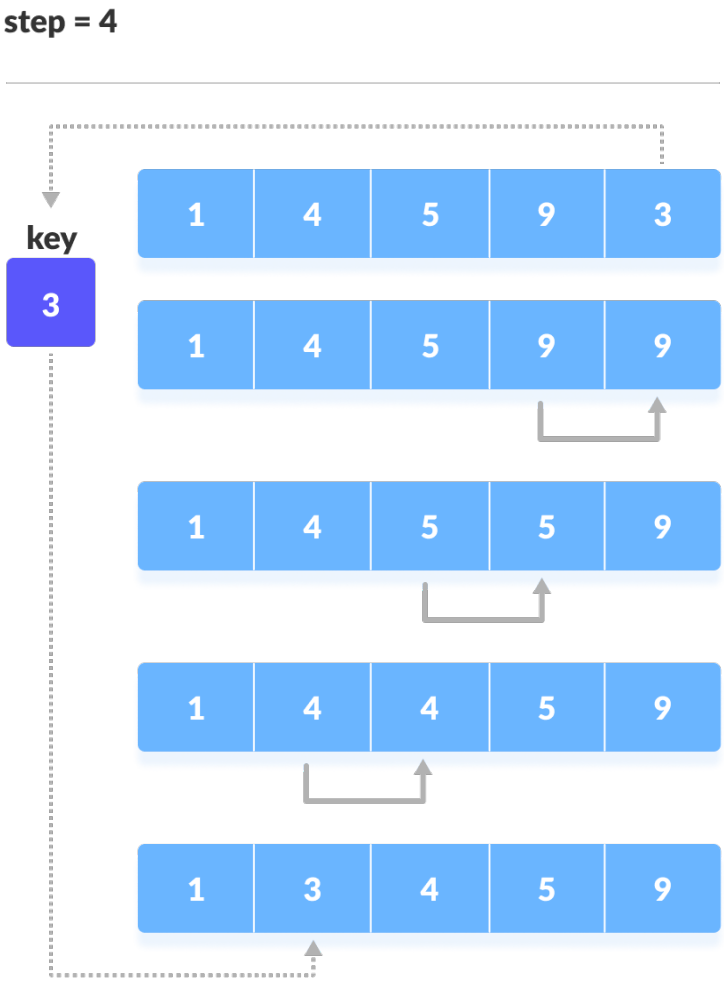


Place 1 at the beginning

Similarly, place every unsorted element at its correct position.



Place 4 behind 1



Place 3 behind 1 and the array is sorted

```
package array;

import java.util.Scanner;

public class Practice {

    public static void InsertionSort(int arr[])
    {
        int n = arr.length;

        for(int i=1;i<n;i++)
```

```
        {
            int j = i-1;
            int value = arr[i];
            while(j>=0 && arr[j]>value)
            {
                arr[j+1]=arr[j];
                j--;
            }
            arr[j+1] = value;
        }

    }

    public static void PrintArray(int arr[])
    {
        for(int i = 0 ; i<arr.length; i++)
        {
            System.out.print(arr[i]+" ");
        }
    }

    public static void main(String[] args)
    {
        // TODO Auto-generated method stub
      Scanner s = new Scanner(System.in);
        int arr[]= {50,60,80,20,39,40,70};
        InsertionSort(arr);
        PrintArray(arr);
    }
}
```

Output:

```
20 39 40 50 60 70 80
```

Merge Sort:

```
package array;

import java.util.Scanner;

public class Practice {

    public static int[] MergeSort(int arr1[] , int arr2[])
    {
        int m = arr1.length;
        int n = arr2.length;

        int arr[] = new int[m+n];

        int i = 0 ;
        int j = 0 ;
        int k = 0 ;

        while(i<m && j<n)
        {
            if(arr1[i]<=arr2[j])
            {
                arr[k] = arr1[i];
                i++;
                k++;
            }
            else
            {
                arr[k] = arr2[j];
                j++;
                k++;
            }
        }

        while(i<m)
```

```
        {
            arr[k] = arr1[i];
            i++;
            k++;
        }
        while(j<n)
        {
            arr[k] = arr1[j];
            j++;
            k++;
        }

        return arr;
    }

    public static void PrintArray(int arr[])
    {
        for(int i = 0 ; i<arr.length; i++)
        {
            System.out.print(arr[i]+" ");
        }
    }

    public static void main(String[] args)
    {
        // TODO Auto-generated method stub
      Scanner s = new Scanner(System.in);
        int arr1[]= {10,30,50,70,90};
        int arr2[] = {20,40,60,80};
        int arr[]=MergeSort(arr1,arr2);
        PrintArray(arr);
    }
}
```

Output:

```
10 20 30 40 50 60 70 80 90
```

Array Notes

I. Push Zeros to end

You have been given a random integer array/list(ARR) of size N. You have been required to push all the zeros that are present in the array/list to the end of it. Also, make sure to maintain the relative order of the non-zero elements.

Note:

Change in the input array/list itself. You don't need to return or print the elements.

You need to do this in one scan of array only. Don't use extra space.

Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains an integer 'N' representing the size of the array/list.

Second line contains 'N' single space separated integers representing the elements in the array/list.

Output Format :

For each test case, print the elements of the array/list in the desired order separated by a single space.

Output for every test case will be printed in a separate line.

Constraints :

1 <= t <= 10^2
0 <= N <= 10^5
Time Limit: 1 sec

Sample Input 1:

1
7
2 0 0 1 3 0 0

Sample Output 1:

2 1 3 0 0 0 0

Explanation for the Sample Input 1 :

All the zeros have been pushed towards the end of the array/list. Another important fact is that the order of the non-zero elements have been maintained as they appear in the input array/list.

```
package array;

public class PushZerosToEnd {

    public static void PushZeros(int arr[])
    {
        int n = arr.length;
         int k = 0;

         for(int i = 0 ; i<n ; i++)
         {
         if(arr[i] != 0)
         {
```

```
                int temp = arr[i];
                arr[i]=arr[k];
                arr[k]=temp;
                k++;
            }
            }
    }

    public static void PrintArray(int arr[])
    {
        for(int i = 0 ; i<arr.length; i++)
        {
            System.out.print(arr[i]+" ");
        }
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int arr[] = {2,3,0,0,1,0,4,0,0};
        PushZeros(arr);
        PrintArray(arr);
    }

}
```

Output:

```
2 3 1 4 0 0 0 0 0
```