Think and Tell

Which feature of an online shopping app makes it easier and quicker to shop online?

Tweets on Twitter

How do tweets appear on Twitter?

Medal Tally

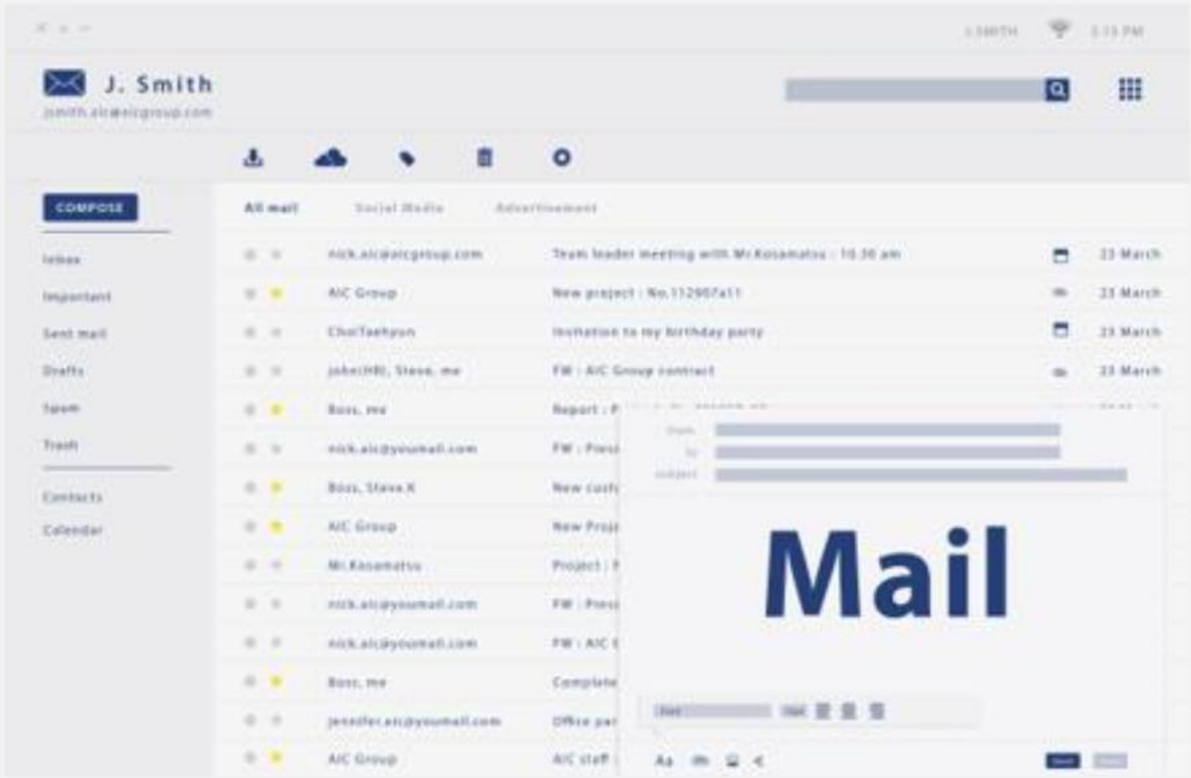
Can you identify the top three countries winning the highest number of medals?

Contact Details

If the contact list of your phone can be arranged in the order you like, how will you organize it?

E-mails

- How do emails appear on your mailbox?
- Which feature makes it easier to locate a particular mail?



Implement Sorting on Arrays

Learning Objectives

- Explore nested for Loops
- Explain sorting
- Implement Bubble sort on an Array
- Implement Insertion sort on an Array

Explore Nested for Loops

Nested for loops

- Loops that are enclosed within another loop and have a nest-like structure are known as nested loops or loop within a loop.
- The outer loop monitors the number of executions of the inner loop.
- A nested loop is useful when an action needs to be repeated on the data in the inner loop for each pass through the outer loop.
- For example, let's assume reading a file line by line and the occurrence of the word “bird” needs to be counted. In this case,
  - The outer loop will read the lines.
  - The inner loop will search each line for the word “bird” by looping through the words in the line.

Working of a Nested for loop

- In the code shown:
  - The outer `for` loop will execute once and print the first element of an `array`. The inner `for` loop will then complete the 3 iterations and print the `strArray` elements.
  - Control will then move to the outer `for` loop for the second time and print the second element of `array`. Again, the inner `for` loop will complete the 3 iterations and print the `strArray` elements.
  - Here, the control will move to the outer `for` loop for a third time and print the third element of `array`. Again, the inner `for` loop will complete the 3 iterations and print the `strArray` elements.

```
String[] array = {"Hello","Welcome","World"};
String[] strArray = {"World","Home","Class"};

for (int i = 0; i < array.length; i++)
{
    for (int j = 0; j < strArray.length; j++)
    {
        System.out.print(array[i]+" : "+strArray[j]+"\t");
    }
    System.out.println();
}
```

Inner loop

Outer loop

Output

```
Hello : World   Hello : Home    Hello : Class
Welcome : World Welcome : Home  Welcome : Class
World : World   World : Home    World : Class
```

Quick Check

What do you think the below code will print?

```
for (int i = 1; i < 7; i++)
{
    for (int j = 1; j <= 5; j++)
    {
        System.out.print("*");
    }
    System.out.println();
}
```

1. A rectangle of 7 rows with 5 stars per row.
2. A rectangle of 7 rows with 4 stars per row.
3. A rectangle of 6 rows with 5 stars per row.
4. A rectangle of 6 rows with 4 stars per row.

Quick Check: Solution

What do you think the below code will print?

```
for (int i = 1; i < 7; i++)
{
    for (int j = 1; j <= 5; j++)
    {
        System.out.print("*");
    }
    System.out.println();
}
```

1. A rectangle of 7 rows with 5 stars per row.
2. A rectangle of 7 rows with 4 stars per row.
3. **A rectangle of 6 rows with 5 stars per row.**
4. A rectangle of 6 rows with 4 stars per row.

Explain Sorting

Sorting an Array

- Sorting an array means to arrange its elements in ascending or descending order.
- Sorting can be used in multiple situations when dealing with data in the array.
  - If the roll numbers of students are stored in an array, they need to be sorted in ascending order whenever the teacher takes the roll call.
  - To determine the student who topped this year, the ar ray that holds the marks of the students of a class must be in the order of the highest performers.

Sorting Algorithms

- Sorting can be performed on array elements by using multiple sorting algorithms.
- Few of the sorting algorithms are as follows,
  - Bubble sort
  - Selection sort
  - Insertion sort
  - Merge sort
  - Quick sort
  - Heap sort
- The simplest sorting algorithms are bubble sort and insertion sort.

Implement Bubble Sort on an Array

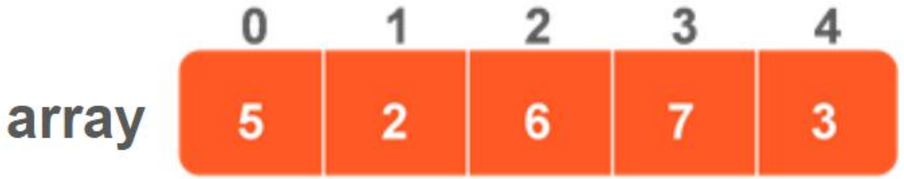
Bubble Sort

- Bubble Sort is the simplest sorting algorithm. It works by repeatedly swapping adjacent elements of an array when they are placed in the wrong order.
- In Bubble Sort, an array must be iterated multiple times until the entire array is sorted.
- One complete iteration until the end of the array is reached is called a pass.
- Multiple pass is required to sort an array.

Pass 1

Implementing Bubble Sort

- The unsorted array



- Compare the first two adjacent elements and swap if array[0]>array[1].

- Compare the next two adjacent elements, if array[1]>array[2] then swap else, there is no change.

Implementing Bubble Sort (contd.)

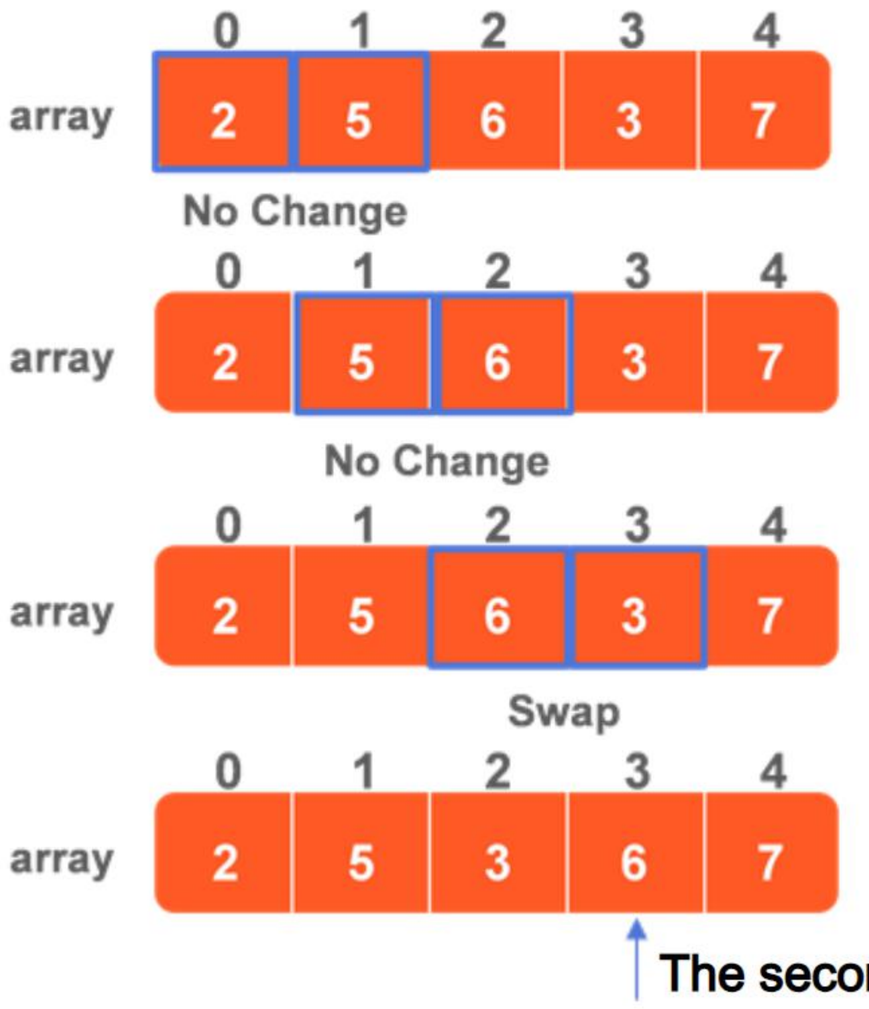
- Continue the comparison until the end of the array is reached.

The largest element is placed at the correct position after Pass 1.

Pass 2

Implementing Bubble Sort (contd.)

- Start from the beginning of the array and perform a comparison of adjacent elements until the end of the array is reached.



The second largest element is placed at the correct position after Pass 2.

Pass 3

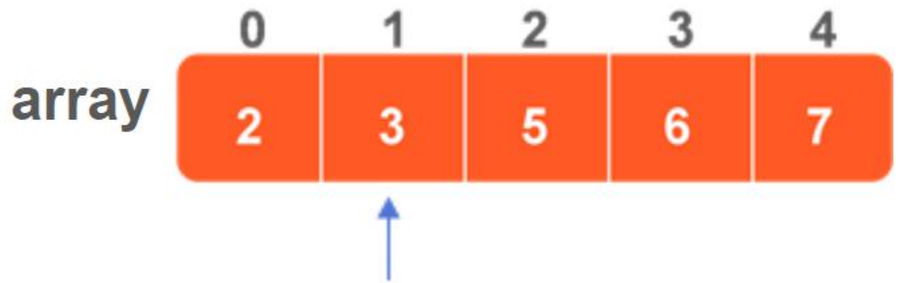
Implementing Bubble Sort (contd.)

- Compare adjacent elements until the array is completely sorted.

The third largest element is placed at the correct position after Pass 3.

Pass 4

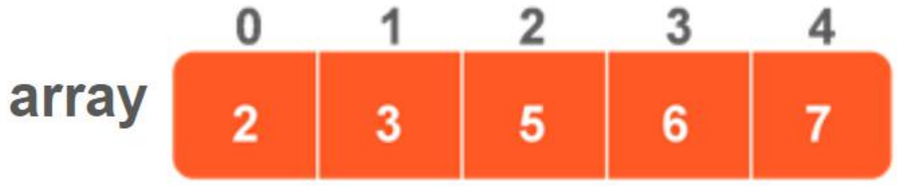
Implementing Bubble Sort (contd.)



The fourth largest element is placed at the correct position after Pass 4.

Implementing Bubble Sort (contd.)

- At the end of Pass 4, all the elements of the array are sorted.



Bubble Sort – Pseudocode Explained

- Begin
- Get the integer array `array = 5,2,6,7,3`
- Get the length of the array `len = array.length`
- Loop through the entire array – `for i = 0 to len`
- Maintain an inner loop to loop through adjacent elements – `for j = 0 to len-i-1`
- Compare adjacent elements–

```
if array[j] > array[j+1]
```

- Swap if the first element is greater than the second element – Set a temp variable and swap

```
temp = array[j]
array[j] = array[j+1]
array[j+1] = temp
```

- Continue until all the array elements are sorted.

Dry Run of Bubble Sort – Pass 1

- The outer loop will run through entire array.
- Iteration 1 – i = 0, j = 0, len=5
- Inner loop for the 4 comparisons in the pass j = 0 to 5-0-1 = 4
- 5 > 2 – swap the elements 2,5,6,7,3
- Iteration 2 – i = 1, j = 1, len= 5
- Inner loop for the 3 comparisons in the pass j = 1 to 5-1-1 = 3
- 5 > 6 – false so no swap happens 2,5,6,7,3
- Iteration 3 – i = 2, j = 2, len= 5
- Inner loop for the 2 comparisons in the pass j = 1 to 5-2-1 = 2
- 6 > 7 – false so no swap happens 2,5,6,7,3
- Iteration 4 – i = 3, j = 3, len= 5
- Inner loop for the 1 comparisons in the pass j = 1 to 5-3-1 = 1
- 7 > 3 – swap the elements 2,5,6,3,7
- Iteration 5 – i = 4, j = 5, len= 5
- Inner loop for the 0 comparisons in the pass j = 1 to 5-4-1 = 0, loop exits.
- Pass 1 is complete. The array at the end of the pass 2,5,6,7,3.
- This continues until all the elements are sorted.

Bubble Sort – Pseudocode

- The pseudocode to implement Bubble Sort is shown in the image.
- Iterate through the array elements using the two for loops.
- Two for loops are used so that adjacent elements can be compared.
- Swap elements, if the first element is greater than the second element.
- Iterate through the array multiple times until all the elements are sorted.

```
1. BEGIN
2. GET intArray
3. SET temp = 0
4. SET n = length(intArray)
5. BEGIN FOR i = 0 to n
          BEGIN FOR j = 0 to n-i-1
           IF intArray[j] > intArray[j+1]
           BEGIN
             temp = intArray[j]
             intArray[j] = intArray[j+1]
             intArray[j+1] = temp
           END
          END FOR
6. END FOR
7. END
```

Bubble Sort Implementation Using Java

- A method that performs bubble sort is shown in the first image.
- Two `for` loops are used to iterate through the array for each pass.
- The second image shows the main method, where the `bubbleSort` method is called and the integer array that holds the `gradesOfStudents` is passed to the method.

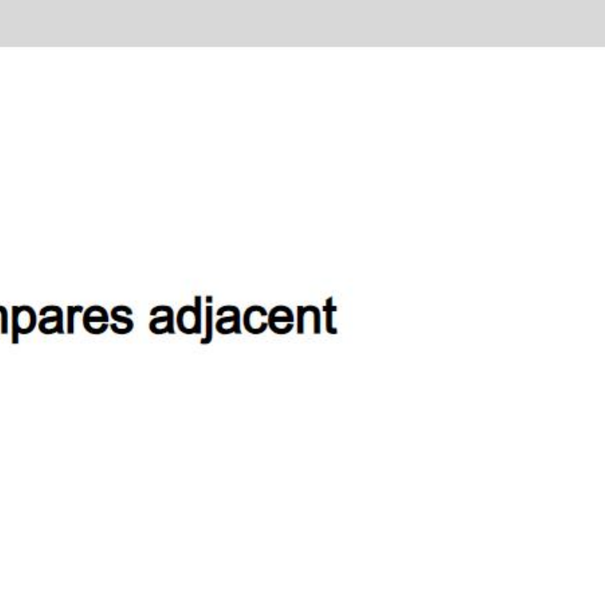
```java
int[] bubbleSort(int[] arr) {
    int temp = 0;
    for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr.length - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    return arr;
}
```

```java
public static void main(String[] args) {
    int[] gradesOfStudents = {5, 10, 1, 2, 23, 12, 6, 9, 3, 7};
    BubbleSort bubbleSort = new BubbleSort();
    int[] sortedArray = bubbleSort.bubbleSort(gradesOfStudents);
    System.out.println("\n" + "The Sorted Grades of the Students are : ");
    bubbleSort.printArray(sortedArray);
}
```

Quick Check

_____ is a sorting algorithm that repeatedly iterates through the list, compares adjacent elements, and swaps them if they are placed incorrectly.

1. Selection sort
2. Bubble sort
3. Merge sort
4. Insertion sort



Quick Check : Solution

_____ is a sorting algorithm that repeatedly iterates through the list, compares adjacent elements, and swaps them if they are placed incorrectly.

1. Selection sort
2. Bubble sort
3. Merge sort
4. Insertion sort

Student Roll Numbers – Bubble Sort

Write a program to sort the roll numbers of 10 students stored in an array in ascending order. Implement bubble sort and display the roll numbers.

- Click here for the solution.
- The workbench must be used for the demonstration.
- Execute the test cases provided in the test folder.

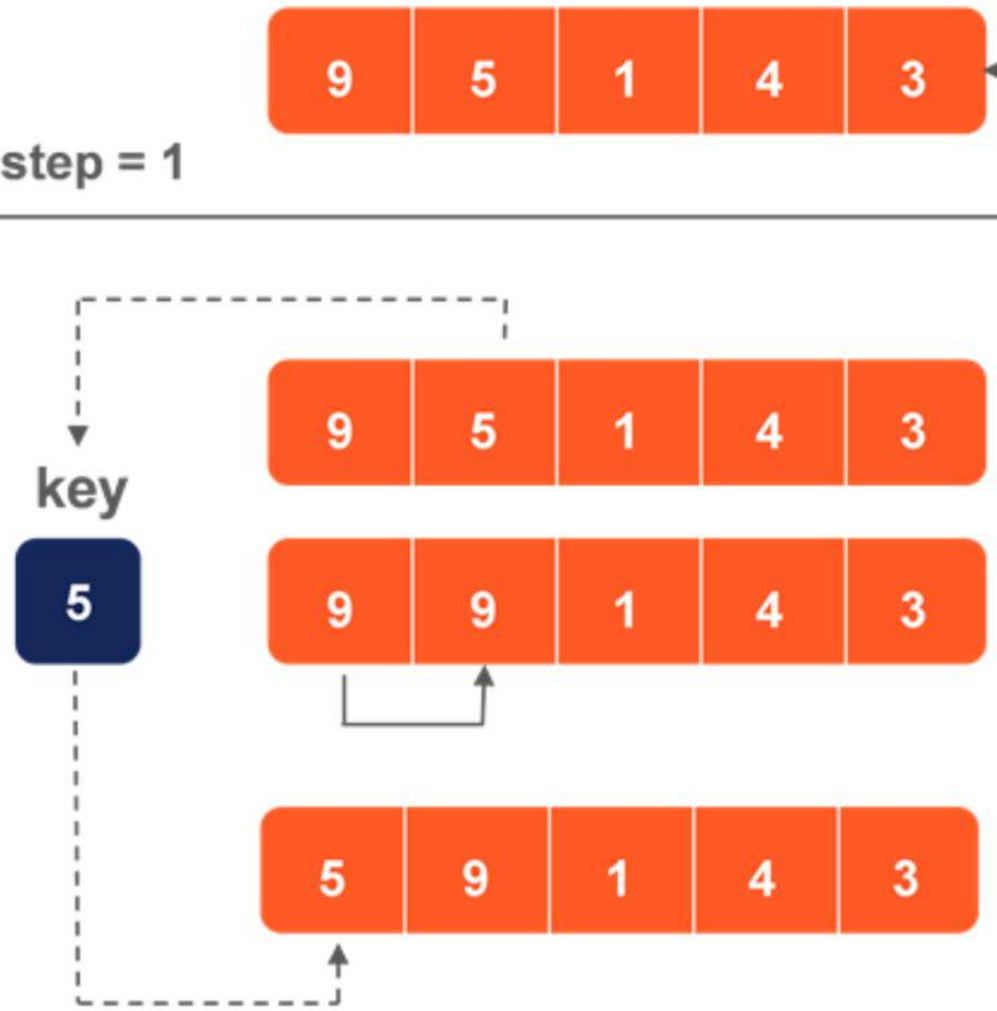
DEMO

Implement Insertion Sort on an Array

Insertion Sort

- Insertion sort is a simple sorting algorithm that builds the final sorted array one element at a time.
- The below steps explain how insertion sort is done on an array.
  - The first element in the array is assumed to be sorted.
  - The next element is selected and stored separately as a key.
  - The key is compared with all elements in the sorted array.
  - If the element in the sorted array is smaller than the key, the next element is selected.
  - Else, greater elements in the array are shifted towards the right of the array.
  - The key is inserted at the appropriate position.
  - The steps are repeated until the array is sorted.

Implement Insertion Sort

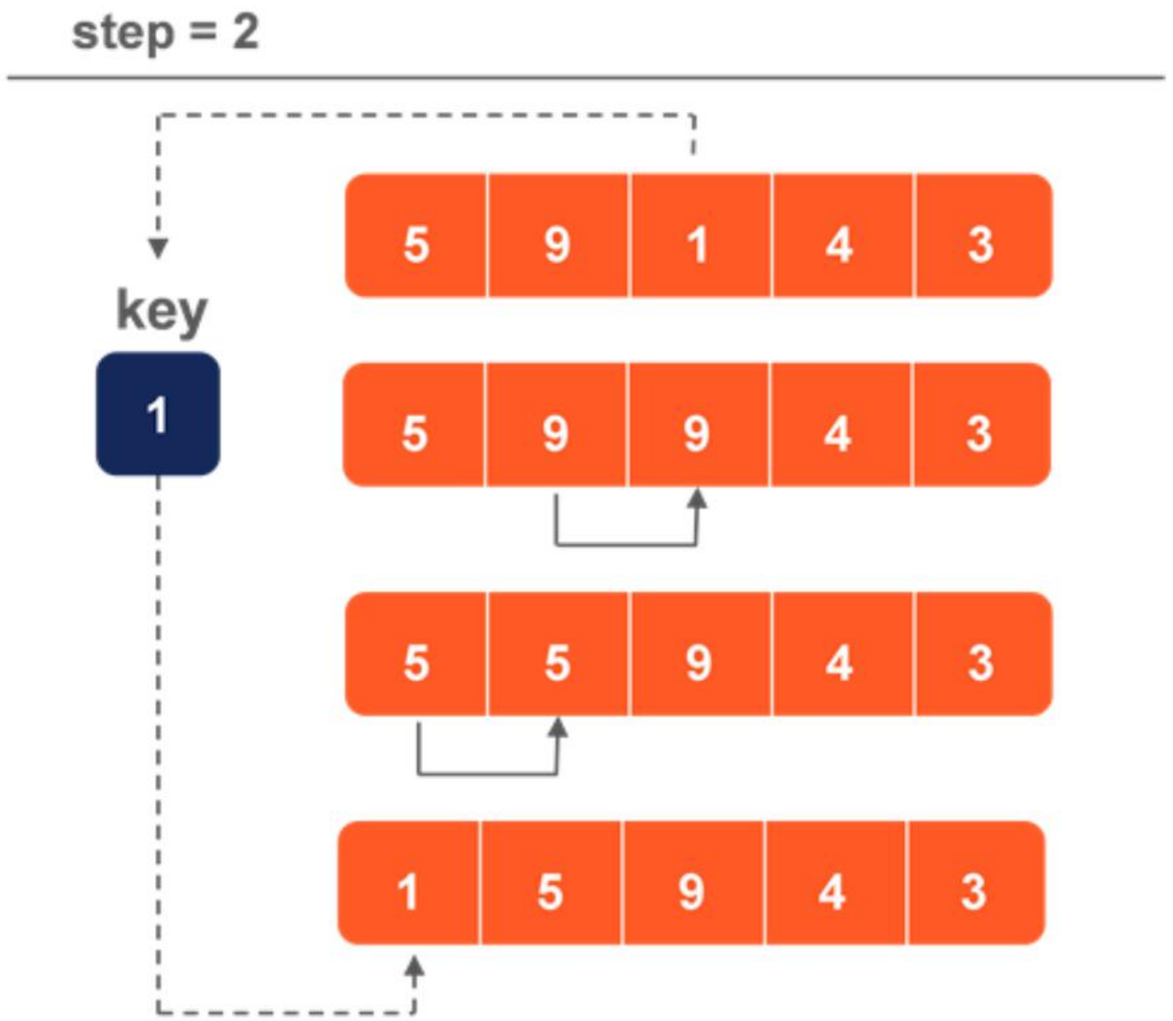


The Initial Array

- The first element 9 in the array is assumed to be sorted.
- The second element 5 is stored in a key.
- If the first element 9 is greater than key 5.
- Then 9 is shifted one position up in the array i.e., 9 is moved from array[0] to array[1].
- Then the key 5 is placed in front of the first element 9 since 5 is less than 9 i.e., array[0] = 5.
- The first 2 elements are sorted.

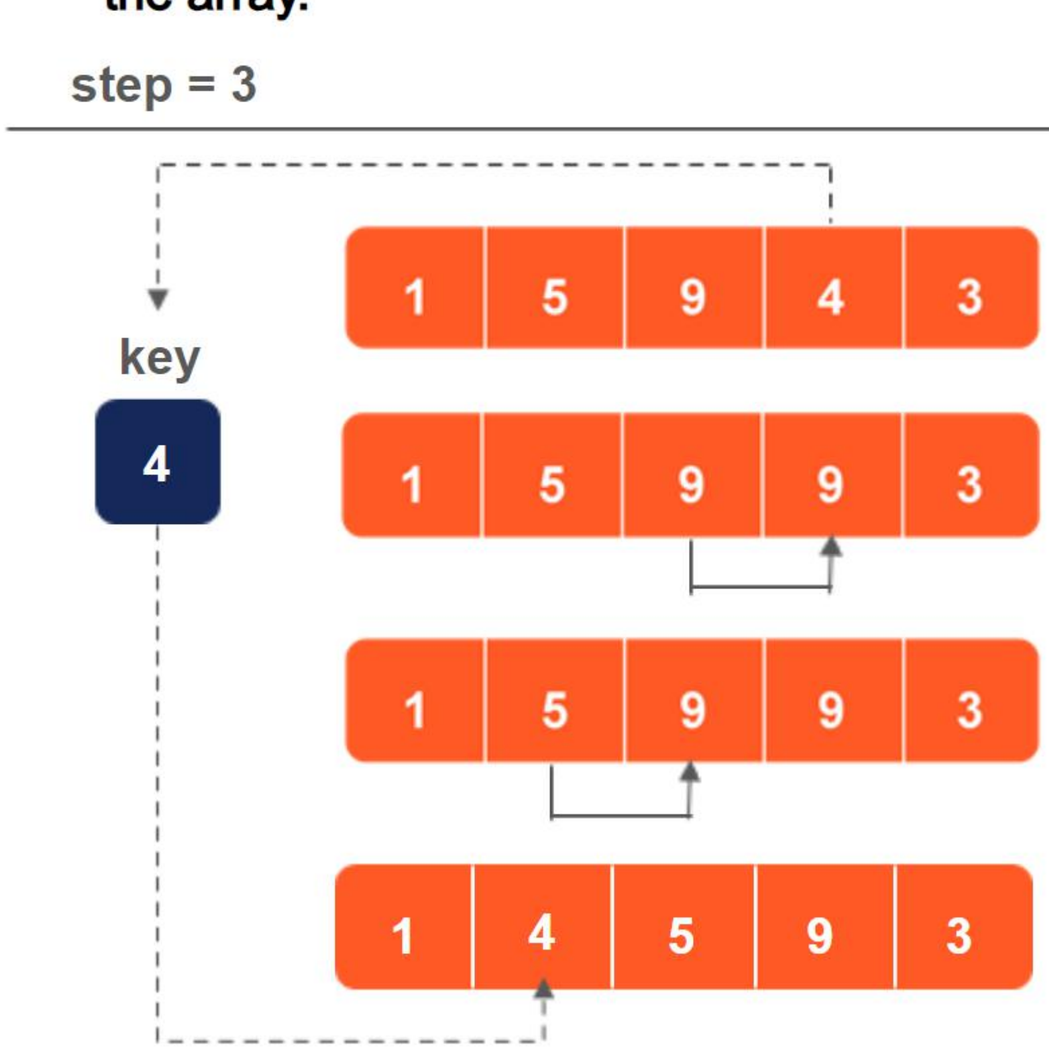
Implementing Insertion Sort

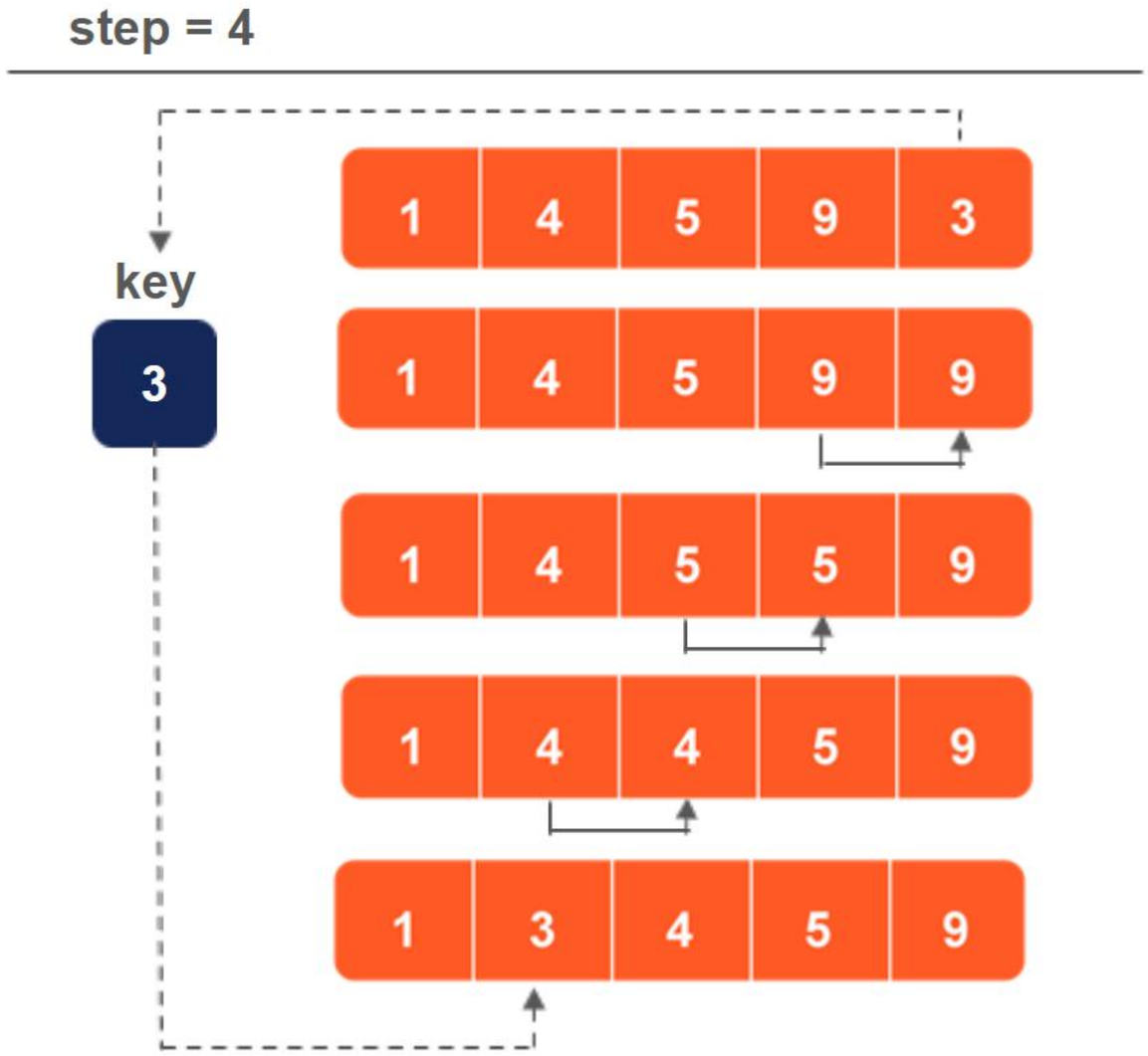
- Now the third element (1) is the key, since 1 is less than 5 and 9, it is placed at the beginning.



Implementing Insertion Sort

- Similarly, place all the unsorted elements at the correct position by iterating through the elements of the array.





Insertion Sort – Pseudocode

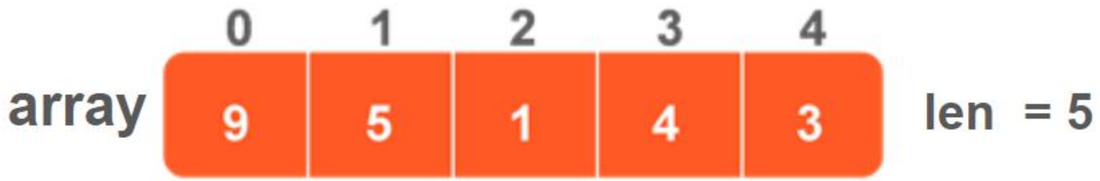
- The pseudocode to implement insertion sort is shown in the image.
- The first element of the array is assumed to be sorted.
- The key is the second element of the array.
- Use a while loop to loop through the array elements.
- Compare the elements with the key.
- Shift the elements that are greater than key, to one position ahead of their current position.

```
1. BEGIN
2. GET intArray
3. SET n = length(intArray)
4. BEGIN FOR i = 1 to n
     SET key = intArray[i]
     SET j = i - 1
    BEGIN WHILE j>=0 and intArray[j]>key
         intArray[j+1] = intArray[j]
         j = j - 1
     END WHILE
     intArray[j+1] = key
5. END FOR
6. END
```

Insertion Sort – Pseudocode Explained

- Begin
- Get the integer array `array = 9,5,1,4,3`
- Get the length of the array `len = array.length`
- Loop through the entire array. Begin from index 1 `for i = 1 to n`
- Set key as the second element in the array `key = array[i]`
- Set a temporary variable `j` to build the array of sorted elements
- Use a while loop to check if the next element in the array is greater than key and `j>=0`
- Keep changing the key value to the next element and compare it with the other array elements and then build the array.

Dry Run of Insertion Sort



- One iteration of the Insertion sort is as follows:
- Loop through the array starting at the second index 1

Iteration 1

- `key = array[1] = 5`
- `i = 1`
- `j = i - 1 = 0`
- **Is** `j >= 0 and array[0] > key` **– condition is true** `since j = 0 and 9 > 5`
- Adjust the index values in the array and build the sorted array.

- `array[j+1] = array[j] => array[j+1] = 9`
- **Decrement** `j = j - 1 => j = -1`
- **Assign the key value to the previous index.**

  `array[j+1] = key => array[0] = 5`
- **Hence the array is 5,9,1,4,3 after the first iteration.**
- **The same is continued for all iterations until we get a sorted array.**

```java
int[] insertionSort(int[] arr)
{
    for (int i = 1; i < arr.length; ++i) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
    return arr;
}
```

```java
public static void main(String[] args)
{
    int[] gradesOfStudents = {5, 10, 1, 2, 23, 12, 6, 9, 3, 7};
    InsertionSort insertionSort = new InsertionSort();
    System.out.println("The Unsorted Grades : ");
    insertionSort.printArray((gradesOfStudents));
    System.out.println("\n"+"The Sorted Grades : ");
    int[] sortedArray = insertionSort.insertionSort(gradesOfStudents);
    insertionSort.printArray(sortedArray);

}
```

Insertion Sort Implementation Using Java

- A method that performs insertion sort on an array is shown in the image.
- Iterate through the array.
- Select a key element.
- Check if the array element is greater than the key.
- Move the elements in the array accordingly.
- The `insertionSort` method is called from the main method.

Quick Check

In an insertion sort, the array elements from the sorted part are picked and placed at the correct position in the unsorted part.

1. True

2. False

Quick Check: Solution

In an insertion sort, the array elements from the sorted part are picked and placed at the correct position in the unsorted part.

1. True

2. False

Student Roll Numbers – Insertion Sort

Write a program to sort the roll numbers of 10 students stored in an array in ascending order. Implement insertion sort and display the roll numbers.

- Click here for the solution.
- The workbench must be used for the demonstration.
- Execute the test cases provided in the test folder.

DEMO