Bubble Sort

Bubble sort is a sorting algorithm that compares two adjacent elements and swaps them until they are in the intended order.

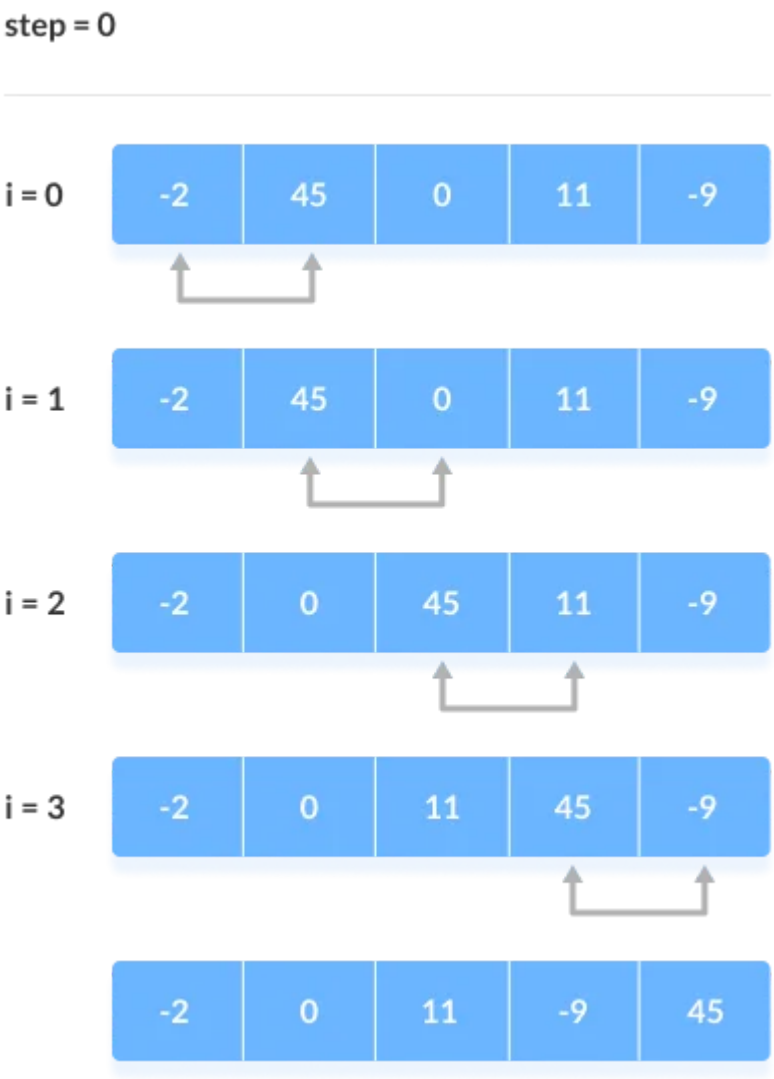
Just like the movement of air bubbles in the water that rise up to the surface, each element of the array move to the end in each iteration. Therefore, it is called a bubble sort.

Working of Bubble Sort

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

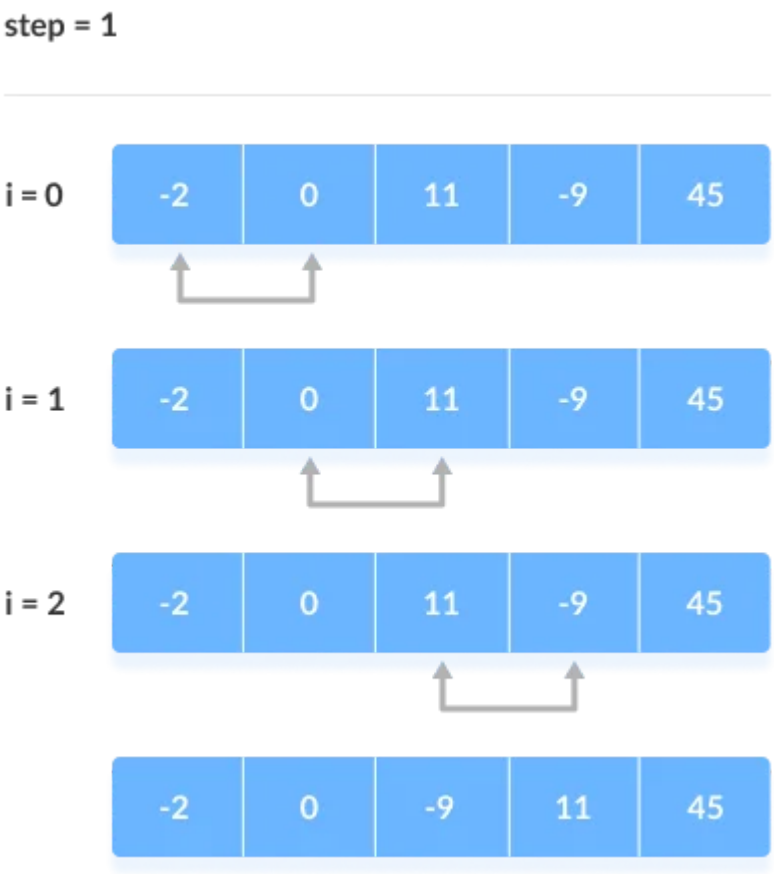
1. First Iteration (Compare and Swap)

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

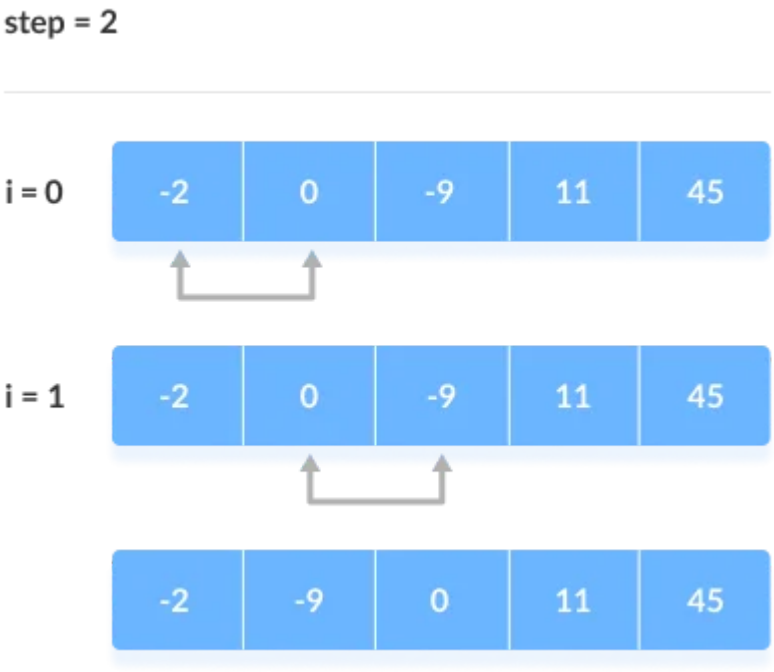


2. Remaining Iteration

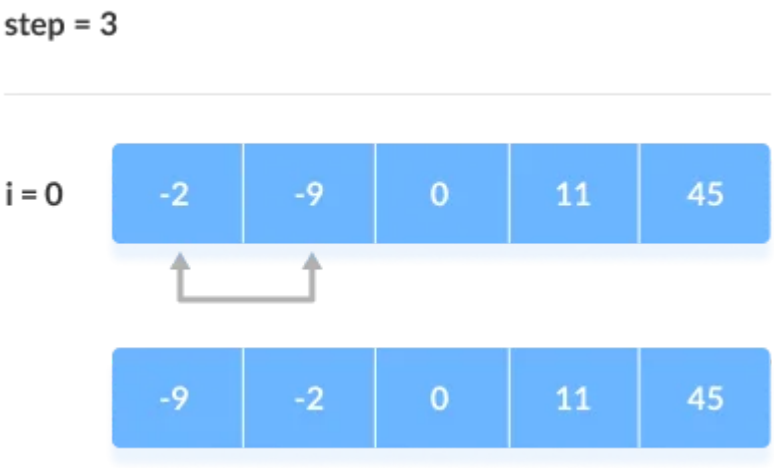
- The same process goes on for the remaining iterations.
- After each iteration, the largest element among the unsorted elements is placed at the end.



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



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



Bubble Sort Algorithm

```
bubbleSort(array)
  for i <- 1 to indexOfLastUnsortedElement-1
    if leftElement > rightElement
      swap leftElement and rightElement
end bubbleSort
```

Java Syntax for Bubble Sort

```
// Bubble sort in Java

import java.util.Arrays;

class Main {

  // perform the bubble sort
  static void bubbleSort(int array[]) {
    int size = array.length;

    // loop to access each array element
    for (int i = 0; i < size - 1; i++)

      // loop to compare array elements
      for (int j = 0; j < size - i - 1; j++)

        // compare two adjacent elements
        // change > to < to sort in descending order
        if (array[j] > array[j + 1]) {

          // swapping occurs if elements
          // are not in the intended order
```

```
            int temp = array[j];
            array[j] = array[j + 1];
            array[j + 1] = temp;
          }
    }

    public static void main(String args[]) {

      int[] data = { -2, 45, 0, 11, -9 };

      // call method using class name
      Main.bubbleSort(data);

      System.out.println("Sorted Array in Ascending Order:");
      System.out.println(Arrays.toString(data));
    }
  }
```

Optimized Bubble Sort Algorithm

- In the above algorithm, all the comparisons are made even if the array is already sorted.
- This increases the execution time.
- To solve this, we can introduce an extra variable swapped. The value of swapped is set true if there occurs swapping of elements. Otherwise, it is set false.
- After an iteration, if there is no swapping, the value of swapped will be false. This means elements are already sorted and there is no need to perform further iterations.
- This will reduce the execution time and helps to optimize the bubble sort.

Algorithm for optimized bubble sort is

```
bubbleSort(array)
  swapped <- false
  for i <- 1 to indexOfLastUnsortedElement-1
    if leftElement > rightElement
      swap leftElement and rightElement
      swapped <- true
end bubbleSort
```

Java syntax for optimized bubble sort

```
// Optimized Bubble sort in Java

import java.util.Arrays;

class Main {

  // perform the bubble sort
  static void bubbleSort(int array[]) {
```

```
    int size = array.length;

    // loop to access each array element
    for (int i = 0; i < (size-1); i++) {

      // check if swapping occurs
      boolean swapped = false;

      // loop to compare adjacent elements
      for (int j = 0; j < (size-i-1); j++) {

        // compare two array elements
        // change > to < to sort in descending order
        if (array[j] > array[j + 1]) {

          // swapping occurs if elements
          // are not in the intended order
          int temp = array[j];
          array[j] = array[j + 1];
          array[j + 1] = temp;

          swapped = true;
        }
      }
      // no swapping means the array is already sorted
      // so no need for further comparison
      if (!swapped)
        break;

    }
  }

  public static void main(String args[]) {

    int[] data = { -2, 45, 0, 11, -9 };

    // call method using the class name
    Main.bubbleSort(data);

    System.out.println("Sorted Array in Ascending Order:");
    System.out.println(Arrays.toString(data));
  }
}
```

Bubble Sort Complexity

Time Complexity

- Best => O(n)
- Worst => O(n^2)
- Average => O(n^2)

Space Complexity

- O(1)

Bubble Sort Applications

Bubble sort is used if

- complexity does not matter
- short and simple code is preferred