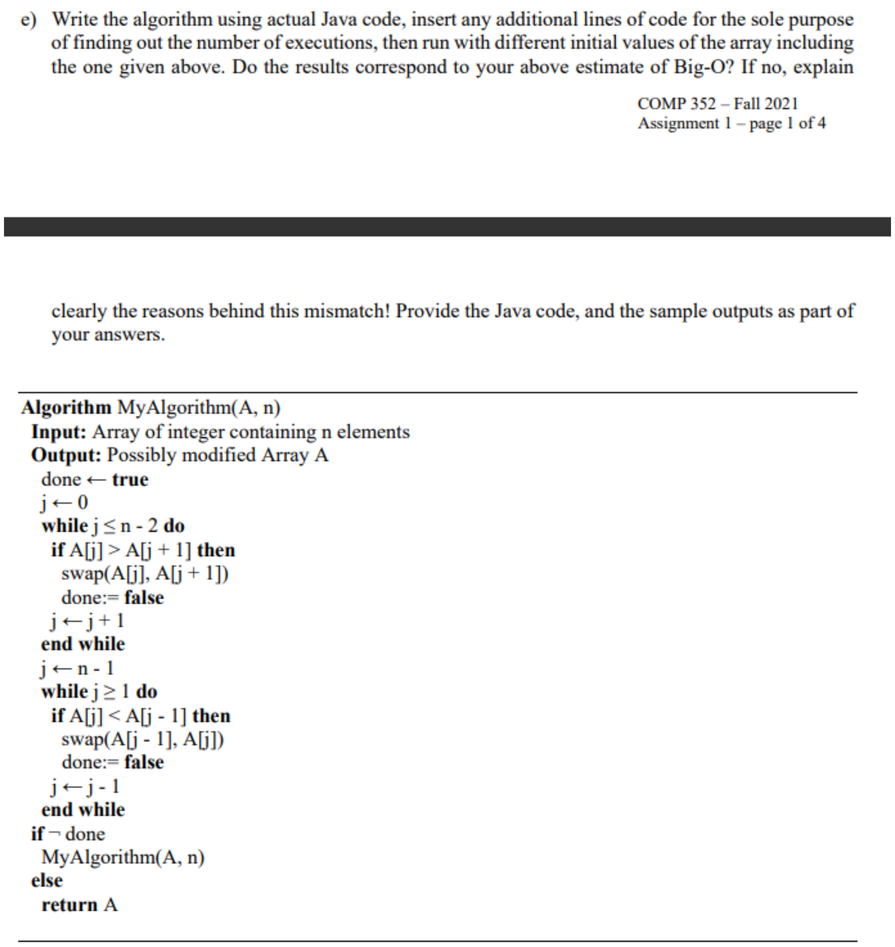
Question: E) Write the algorithm using actual Java code, insert any addit...

(1 bookmark)


e) Write the algorithm using actual Java code, insert any additional lines of code for the sole purpose of finding out the number of executions, then run with different initial values of the array including the one given above. Do the results correspond to your above estimate of Big-O? If no, explain

COMP 352 – Fall 2021
Assignment 1 – page 1 of 4

clearly the reasons behind this mismatch! Provide the Java code, and the sample outputs as part of your answers.

```
Algorithm MyAlgorithm(A, n)
 Input: Array of integer containing n elements
 Output: Possibly modified Array A
  done ← true
  j ← 0
  while j ≤ n - 2 do
    if A[j] > A[j + 1] then
      swap(A[j], A[j + 1])
      done:= false
    j ← j + 1
  end while
  j ← n - 1
  while j ≥ 1 do
    if A[j] < A[j - 1] then
      swap(A[j - 1], A[j])
      done:= false
    j ← j - 1
  end while
 if ¬ done
  MyAlgorithm(A, n)
 else
  return A
```

Show transcribed image text

Expert Answer

Anonymous answered this
116 answers

Was this answer helpful? 4 0

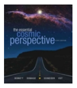
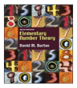
The algorithm is sorting the array in ascending order. There are two while loops whose running time is proportional to the size of the array. The while loops time complexity is:

O(N)+O(N)=O(N)

The function is calling itself if any time the if condition gets executed. So, the function is calling itself when **done = false**.

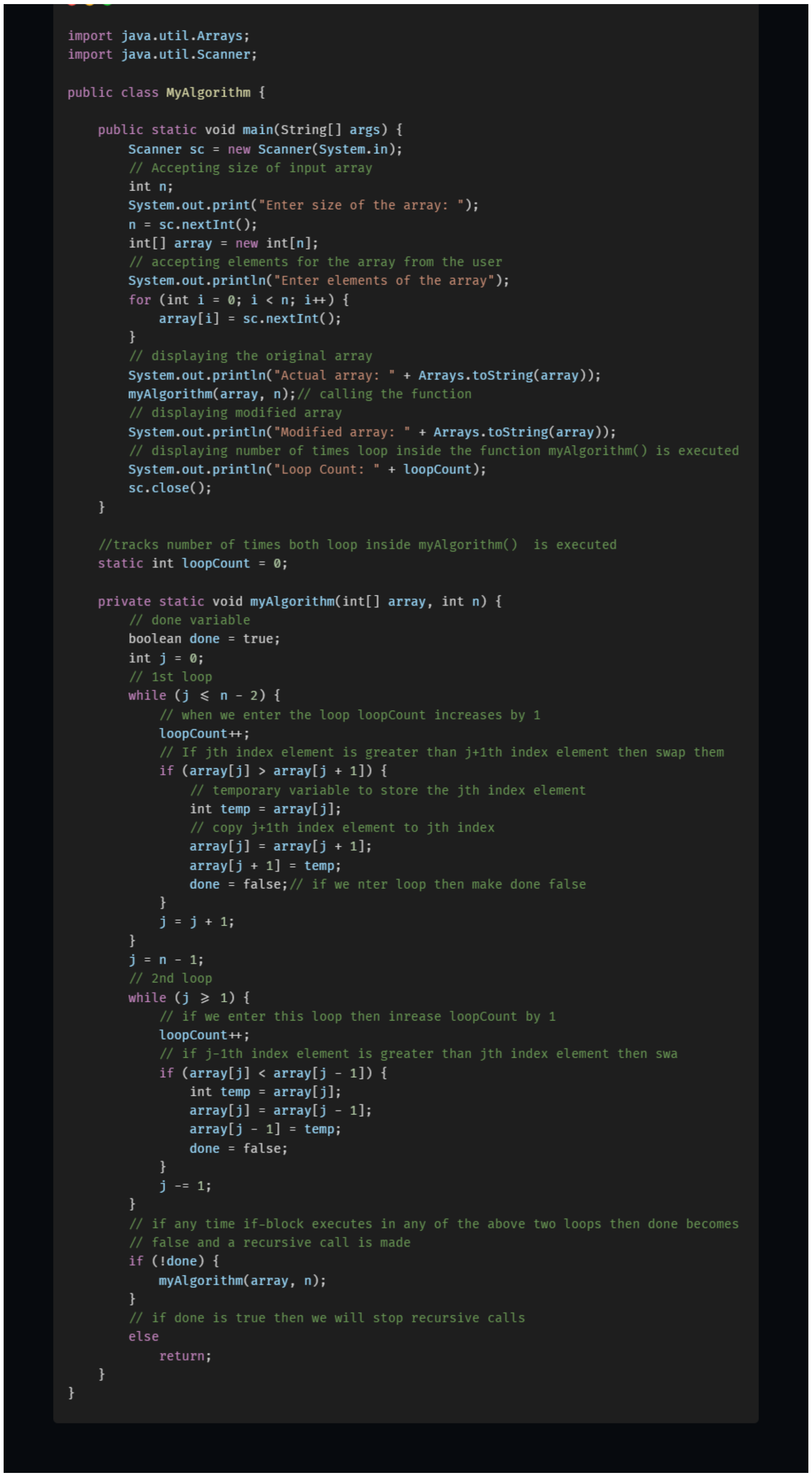
I have written the java program below and kept a variable loopCount to count the total number of times the loops inside the function are executed. This variable counts the number of times both while loop is executed.

```
import java.util.Arrays;
import java.util.Scanner;

public class MyAlgorithm {

    public static void main(String[] args) {
```

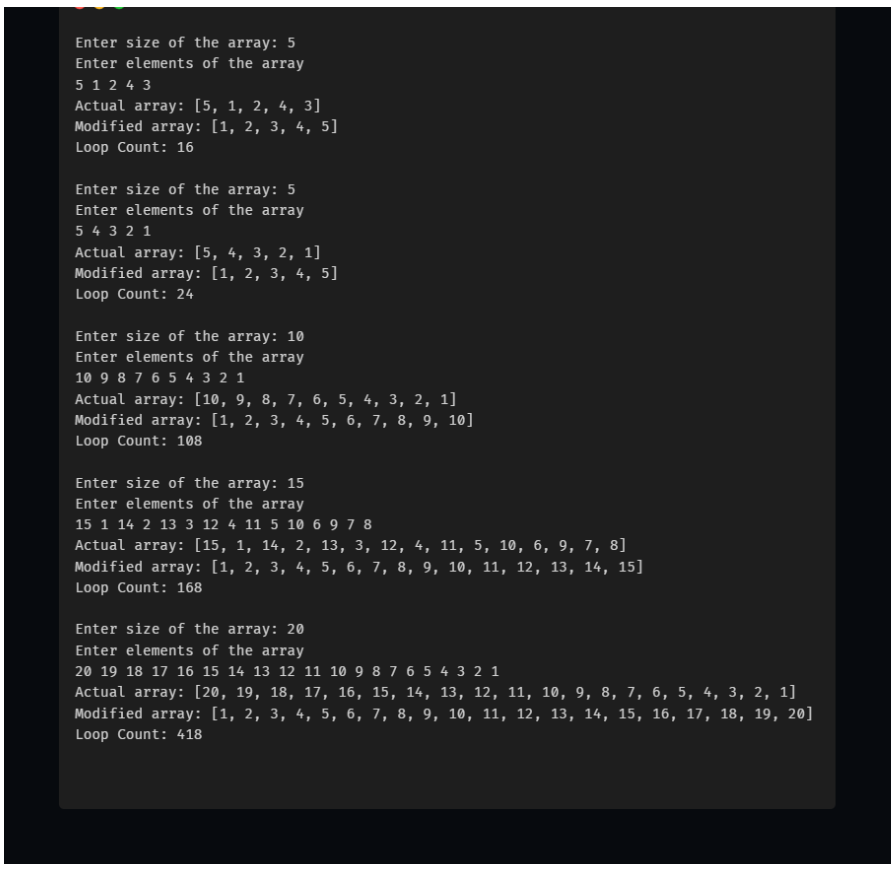



```
        System.out.print("Enter size of the array: ");
        n = sc.nextInt();
        int[] array = new int[n];
        // accepting elements for the array from the user
        System.out.println("Enter elements of the array");
        for (int i = 0; i < n; i++) {
            array[i] = sc.nextInt();
        }
        // displaying the original array
        System.out.println("Actual array: " + Arrays.toString(array));
        myAlgorithm(array, n);// calling the function
        // displaying modified array
        System.out.println("Modified array: " + Arrays.toString(array));
        // displaying number of times loop inside the function myAlgorithm() is executed
        System.out.println("Loop Count: " + loopCount);
        sc.close();
    }

    //tracks number of times both loop inside myAlgorithm()  is executed
    static int loopCount = 0;

    private static void myAlgorithm(int[] array, int n) {
        // done variable
        boolean done = true;
        int j = 0;
        // 1st loop
        while (j <= n - 2) {
            // when we enter the loop loopCount increases by 1
            loopCount++;
            // If jth index element is greater than j+1th index element then swap them
            if (array[j] > array[j + 1]) {
                // temporary variable to store the jth index element
                int temp = array[j];
                // copy j+1th index element to jth index
                array[j] = array[j + 1];
                array[j + 1] = temp;
                done = false;// if we nter loop then make done false
            }
            j = j + 1;
        }
        j = n - 1;
        // 2nd loop
        while (j >= 1) {
            // if we enter this loop then inrease loopCount by 1
            loopCount++;
            // if j-1th index element is greater than jth index element then swa
            if (array[j] < array[j - 1]) {
                int temp = array[j];
                array[j] = array[j - 1];
                array[j - 1] = temp;
                done = false;
            }
            j -= 1;
        }
        // if any time if-block executes in any of the above two loops then done becomes
        // false and a recursive call is made
        if (!done) {
            myAlgorithm(array, n);
        }
        // if done is true then we will stop recursive calls
        else
            return;
    }
}
```

Chegg Study Textbook Solutions Expert Q&A Study Pack Practice

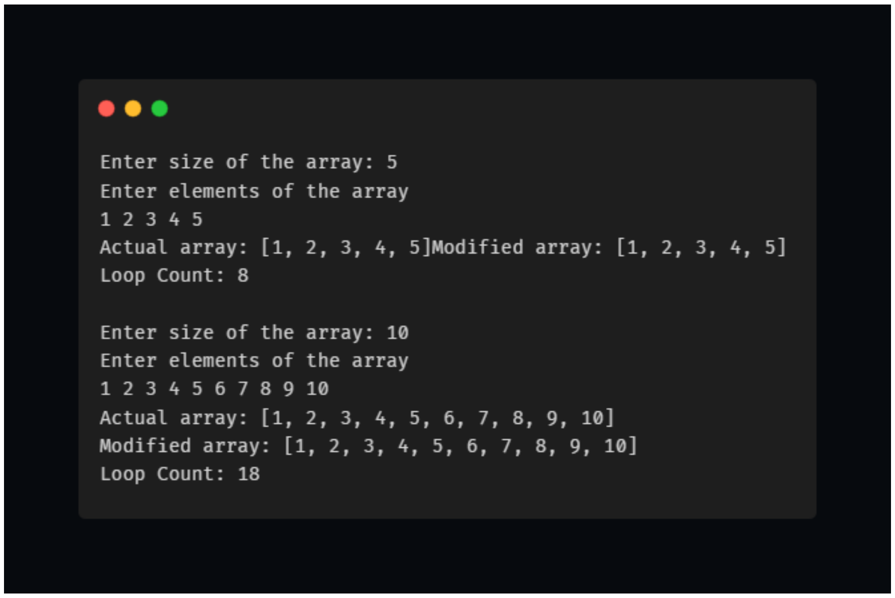


```
import java.util.Arrays;
import java.util.Scanner;

public class MyAlgorithm {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Accepting size of input array
        int n;
        System.out.print("Enter size of the array: ");
        n = sc.nextInt();
        int[] array = new int[n];
        // accepting elements for the array from the user
        System.out.println("Enter elements of the array");
        for (int i = 0; i < n; i++) {
            array[i] = sc.nextInt();
        }
        // displaying the original array
        System.out.println("Actual array: " + Arrays.toString(array));
        myAlgorithm(array, n);// calling the function
        // displaying modified array
        System.out.println("Modified array: " + Arrays.toString(array));
        // displaying number of times loop inside the function myAlgorithm() is executed
        System.out.println("Loop Count: " + loopCount);
        sc.close();
    }

    //tracks number of times both loop inside myAlgorithm()  is executed
    static int loopCount = 0;

    private static void myAlgorithm(int[] array, int n) {
        // done variable
        boolean done = true;
        int j = 0;
        // 1st loop
        while (j ⩽ n - 2) {
            // when we enter the loop loopCount increases by 1
            loopCount++;
            // If jth index element is greater than j+1th index element then swap them
            if (array[j] > array[j + 1]) {
                // temporary variable to store the jth index element
                int temp = array[j];
                // copy j+1th index element to jth index
                array[j] = array[j + 1];
                array[j + 1] = temp;
                done = false;// if we nter loop then make done false
            }
            j = j + 1;
        }
        j = n - 1;
        // 2nd loop
        while (j ⩾ 1) {
            // if we enter this loop then inrease loopCount by 1
            loopCount++;
            // if j-1th index element is greater than jth index element then swa
            if (array[j] < array[j - 1]) {
                int temp = array[j];
                array[j] = array[j - 1];
                array[j - 1] = temp;
                done = false;
            }
            j -= 1;
        }
        // if any time if-block executes in any of the above two loops then done becomes
        // false and a recursive call is made
        if (!done) {
            myAlgorithm(array, n);
        }
        // if done is true then we will stop recursive calls
        else
            return;
    }
}
```



```
Enter size of the array: 5
Enter elements of the array
5 1 2 4 3
Actual array: [5, 1, 2, 4, 3]
Modified array: [1, 2, 3, 4, 5]
Loop Count: 16

Enter size of the array: 5
Enter elements of the array
5 4 3 2 1
Actual array: [5, 4, 3, 2, 1]
Modified array: [1, 2, 3, 4, 5]
Loop Count: 24

Enter size of the array: 10
Enter elements of the array
10 9 8 7 6 5 4 3 2 1
Actual array: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
Modified array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Loop Count: 108

Enter size of the array: 15
Enter elements of the array
15 1 14 2 13 3 12 4 11 5 10 6 9 7 8
Actual array: [15, 1, 14, 2, 13, 3, 12, 4, 11, 5, 10, 6, 9, 7, 8]
Modified array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Loop Count: 168

Enter size of the array: 20
Enter elements of the array
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Actual array: [20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
Modified array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
Loop Count: 418
```

Even after array is sorted in ascending order the loop will get executed for (N-1)*2 times. Lets see few examples:



```
Enter size of the array: 5
Enter elements of the array
1 2 3 4 5
Actual array: [1, 2, 3, 4, 5]Modified array: [1, 2, 3, 4, 5]
Loop Count: 8

Enter size of the array: 10
Enter elements of the array
1 2 3 4 5 6 7 8 9 10
Actual array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Modified array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Loop Count: 18
```

So, in worst case that is when array is in descending order we see that the loop executes in N^2 times.

So, we conclude from the above examples that worst case or Big-O is $O(N^2)$

Hide comments (1)

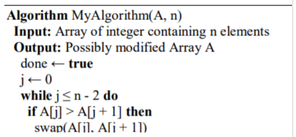
Comments

Anonymous posted 5 days ago

If you found this helpful then if possible please upvote this.

Up next for you in Computer Science

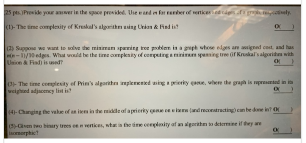
Can somebody put this sorting pseudo code in java?



```
Algorithm MyAlgorithm(A, n)
  Input: Array of integer containing n elements
  Output: Possibly modified Array A
    done ← true
    j ← 0
    while j ≤ n - 2 do
      if A[j] > A[j + 1] then
        swap(A[j], A[j + 1])
```

See answer

please answer these time complexity questions



(25 pts.)Provide your answer in the space provided. Use n and m for number of vertices and edges of a graph, respectively.

(1)- The time complexity of Kruskal's algorithm using Union & Find is? O(___)

(2) Suppose we want to solve the minimum spanning tree problem in a graph whose edges are assigned cost, and has $n(n-1)/10$ edges. What would be the time complexity of computing a minimum spanning tree (if Kruskal's algorithm with Union & Find) is used? O(___)

(3)- The time complexity of Prim's algorithm implemented using a priority queue, where the graph is represented in its weighted adjacency list is? O(___)

(4)- Changing the value of an item in the middle of a priority queue on n items (and reconstructing) can be done in? O(___)

(5)-Given two binary trees on n vertices, what is the time complexity of an algorithm to determine if they are isomorphic? O(___)

See answer

See more questions for subjects you study

Questions viewed by other students

Q: Write the algorithm using actual Java code, insert any additional lines of code for the sole purpose of finding out the number of executions, then run with different initial values of the array including the one given above. Do the results correspond to your estimate of Big-O? If no, explain ***my Big-O estimate is O(n^2) 01 Algorithm MyAlgorithm(A, n) 02 Input: Array of integers A...

A: See answer

Q: a) What is the big-O (O(n)) and big-Omega (Ω(n)) time complexity for algorithm M. Algorithm below in terms of n? Show all necessary steps. b) Trace (hand-run) MyAlgorithm for an array A = (4,105, 1,3). What is the resulting A? c What does MyAlgorithm do? What can be asserted about its result given any arbitrary array A of n integers? d) Can the runtime of MyAlgorithm be improved...

A: See answer 100% (1 rating)

Show more

COMPANY

LEGAL & POLICIES

CHEGG PRODUCTS AND SERVICES

CHEGG NETWORK

CUSTOMER SERVICE