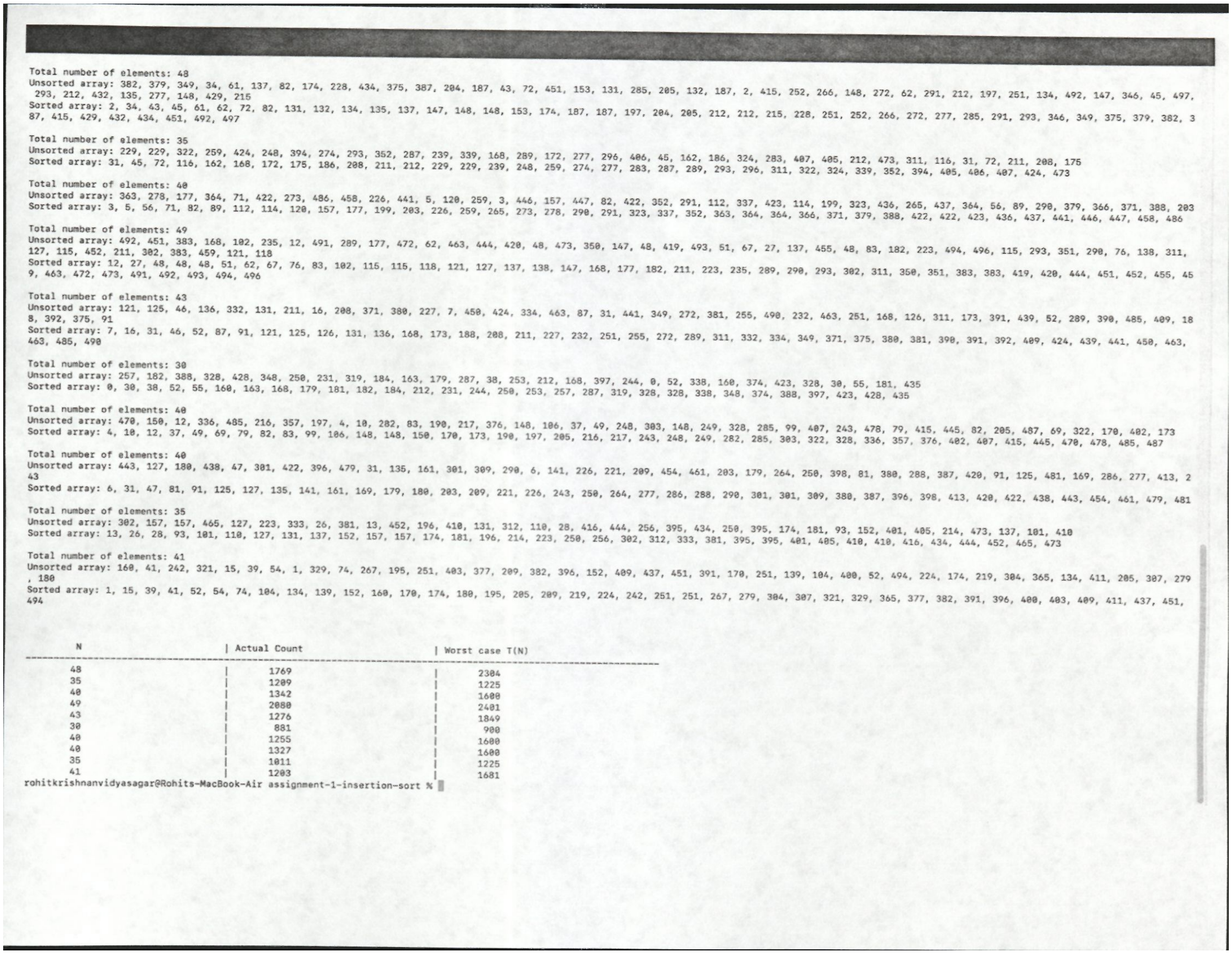
```
import java.lang.Math;
import java.util.HashMap;
import java.util.Map;


public class InsertionSort {

    int getRandomNumber(int min, int max) {
        // Get a random number
        return (int) (Math.random() * (max - min)) + min;
    }

    int[] getInitializedRandomArray(int[] arr, int size) {
        // Inserting random elements into the array
        for(int i=0; i<size; i++) {
            arr[i] = this.getRandomNumber(0, 500);
        }

        return arr;
    }

    void printArray(int[] arr, int size) {
        for(int j=0; j<size; j++) {
            System.out.print(arr[j]);

            // To not print ',' after the final element
            if (j!=size-1) {
                System.out.print(", ");
            }
        }
    }

    public static void main(String[] args) {
        // Initializing the insertion sortObj to use the class methods
        InsertionSort insertionSortObj = new InsertionSort();

        // Setting the min and max number of sets/arrays
        int minNumberOfSets = 10, maxNumberOfSets = 13;

        // Determining the total number of sets or arrays to be sorted
        int totalNumberOfSets =
insertionSortObj.getRandomNumber(minNumberOfSets, maxNumberOfSets);
        System.out.println("Total number of sets: " + totalNumberOfSets + "\n");

        // Initializing an empty array
        int[] arr = new int[60];

        // Setting the min and max number of elements for each array and
declaring total number of elements
        int minNumberOfElements=30, maxNumberOfElements=60,
totalNumberOfElements;

        // Declaring the variables to be used for sorting
        int key, j;

        // Declaring the counter to maintain the actual count of instructions being
run
        int counter;

        // Initializing a map to maintain a record of the number of inputs and the
number of instructions taken to sort the array
        Map<Integer, Integer> inputToActualCountMap = new HashMap<>();

        while(totalNumberOfSets > 0) {
            // Determining the total number of elements for an array using a
randomizer
            totalNumberOfElements =
insertionSortObj.getRandomNumber(minNumberOfElements,
maxNumberOfElements);
            System.out.println("Total number of elements: " +
totalNumberOfElements);

            // Get the initialized randomizer array
            arr = insertionSortObj.getInitializedRandomArray(arr,
totalNumberOfElements);

            System.out.print("Unsorted array: ");
            insertionSortObj.printArray(arr, totalNumberOfElements);

            // Sorting the array
            counter = 0;
        for(int i=1; i<totalNumberOfElements; i++) {
            // Incrementing the counter for the first time the 'i' counter is
initialized and everytime it is incremented
            counter++;

            key = arr[i];
            // Incrementing the counter for everytime the 'key' variable is
assigned
            counter++;


            j = i-1;
            // Incrementing the counter for everytime the 'j' counter is assigned
            counter++;

            while(j>=0 && arr[j] > key) {
               // Incrementing the counter everytime the comparison occurs
               counter++;

               arr[j+1] = arr[j];
               // Incrementing the counter everytime a number is moved/
swapped to the right
               counter++;

               --j;
               // Incrementing the counter everytime the counter 'j' is
decremented
               counter++;
            }
            // Incrementing the counter here because before exiting the loop the
comparison will be carried out once
            counter++;

            arr[j+1] = key;
            // Incrementing the counter everytime 'key' is assigned to the array
index 'j+1'
            counter++;
        }
        // Incrementing the counter here because before exiting the loop the
comparison will be carried out once
```

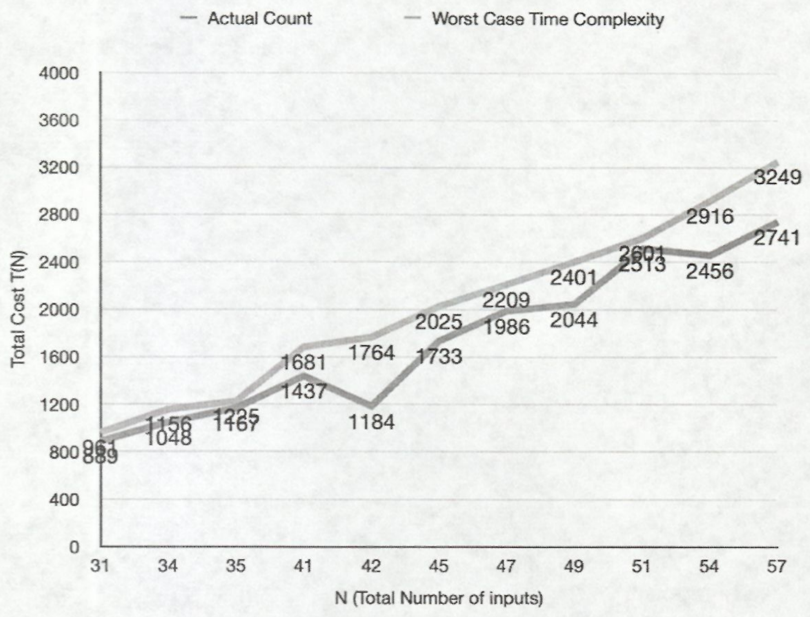
```
            counter++;

            System.out.println();
            System.out.print("Sorted array: ");
            insertionSortObj.printArray(arr, totalNumberOfElements);

            inputToActualCountMap.put(totalNumberOfElements, counter);

            System.out.println("\n");
            totalNumberOfSets--;
        }

        System.out.println("\n");

        System.out.println(String.format("%10s %25s %10s %23s %10s", "N",
"|", "Actual Count", "|", "Worst case T(N)"));
        System.out.println(String.format("%s",
"--------------------------------------------------------------------------------------------
------------------------"));

        inputToActualCountMap.forEach((input, count) -> {
            System.out.println(String.format("%10d %25s %10d %25s %10d",
input, "|", count, "|", (input * input)));
        });
    }
}
```



```
Total number of elements: 48
Unsorted array: 382, 379, 349, 34, 61, 137, 82, 174, 228, 434, 375, 387, 204, 187, 43, 72, 451, 153, 131, 285, 205, 132, 187, 2, 415, 252, 266, 148, 272, 62, 291, 212, 197, 251, 134, 492, 147, 346, 45, 497,
 293, 212, 432, 135, 277, 148, 429, 215
Sorted array: 2, 34, 43, 45, 61, 62, 72, 82, 131, 132, 134, 135, 137, 147, 148, 148, 153, 174, 187, 187, 197, 204, 205, 212, 212, 215, 228, 251, 252, 266, 272, 277, 285, 291, 293, 346, 349, 375, 379, 382, 3
87, 415, 429, 432, 434, 451, 492, 497

Total number of elements: 35
Unsorted array: 229, 229, 322, 259, 424, 248, 394, 274, 293, 352, 287, 239, 339, 168, 289, 172, 277, 296, 406, 45, 162, 186, 324, 283, 407, 405, 212, 473, 311, 116, 31, 72, 211, 208, 175
Sorted array: 31, 45, 72, 116, 162, 168, 172, 175, 186, 208, 211, 212, 229, 229, 239, 248, 259, 274, 277, 283, 287, 289, 293, 296, 311, 322, 324, 339, 352, 394, 405, 406, 407, 424, 473

Total number of elements: 40
Unsorted array: 363, 278, 177, 364, 71, 422, 273, 486, 458, 226, 441, 5, 120, 259, 3, 446, 157, 447, 82, 422, 352, 291, 112, 337, 423, 114, 199, 323, 436, 265, 437, 364, 56, 89, 290, 379, 366, 371, 388, 203
Sorted array: 3, 5, 56, 71, 82, 89, 112, 114, 120, 157, 177, 199, 203, 226, 259, 265, 273, 278, 290, 291, 323, 337, 352, 363, 364, 364, 366, 371, 379, 388, 422, 422, 423, 436, 437, 441, 446, 447, 458, 486

Total number of elements: 49
Unsorted array: 492, 451, 383, 168, 102, 235, 12, 491, 289, 177, 472, 62, 463, 444, 420, 48, 473, 350, 147, 48, 419, 493, 51, 67, 27, 137, 455, 48, 83, 182, 223, 494, 496, 115, 293, 351, 290, 76, 138, 311,
127, 115, 452, 211, 302, 383, 459, 121, 118
Sorted array: 12, 27, 48, 48, 48, 51, 62, 67, 76, 83, 102, 115, 115, 118, 121, 127, 137, 138, 147, 168, 177, 182, 211, 223, 235, 289, 290, 293, 302, 311, 350, 351, 383, 383, 419, 420, 444, 451, 452, 455, 45
9, 463, 472, 473, 491, 492, 493, 494, 496

Total number of elements: 43
Unsorted array: 121, 125, 46, 136, 332, 131, 211, 16, 208, 371, 380, 227, 7, 450, 424, 334, 463, 87, 31, 441, 349, 272, 381, 255, 490, 232, 463, 251, 168, 126, 311, 173, 391, 439, 52, 289, 390, 485, 409, 18
8, 392, 375, 91
Sorted array: 7, 16, 31, 46, 52, 87, 91, 121, 125, 126, 131, 136, 168, 173, 188, 208, 211, 227, 232, 251, 255, 272, 289, 311, 332, 334, 349, 371, 375, 380, 381, 390, 391, 392, 409, 424, 439, 441, 450, 463,
463, 485, 490

Total number of elements: 30
Unsorted array: 257, 182, 388, 328, 428, 348, 250, 231, 319, 184, 163, 179, 287, 38, 253, 212, 168, 397, 244, 0, 52, 338, 160, 374, 423, 328, 30, 55, 181, 435
Sorted array: 0, 30, 38, 52, 55, 160, 163, 168, 179, 181, 182, 184, 212, 231, 244, 250, 253, 257, 287, 319, 328, 328, 338, 348, 374, 388, 397, 423, 428, 435

Total number of elements: 40
Unsorted array: 470, 150, 12, 336, 485, 216, 357, 197, 4, 10, 282, 83, 190, 217, 376, 148, 106, 37, 49, 248, 303, 148, 249, 328, 285, 99, 407, 243, 478, 79, 415, 445, 82, 205, 487, 69, 322, 170, 402, 173
Sorted array: 4, 10, 12, 37, 49, 69, 79, 82, 83, 99, 106, 148, 148, 150, 170, 173, 190, 197, 205, 216, 217, 243, 248, 249, 282, 285, 303, 322, 328, 336, 357, 376, 402, 407, 415, 445, 470, 478, 485, 487

Total number of elements: 40
Unsorted array: 443, 127, 180, 438, 47, 301, 422, 396, 479, 31, 135, 161, 301, 309, 290, 6, 141, 226, 221, 209, 454, 461, 203, 179, 264, 250, 398, 81, 380, 288, 387, 420, 91, 125, 481, 169, 286, 277, 413, 2
43
Sorted array: 6, 31, 47, 81, 91, 125, 127, 135, 141, 161, 169, 179, 180, 203, 209, 221, 226, 243, 250, 264, 277, 286, 288, 290, 301, 301, 309, 380, 387, 396, 398, 413, 420, 422, 438, 443, 454, 461, 479, 481

Total number of elements: 35
Unsorted array: 302, 157, 157, 465, 127, 223, 333, 26, 381, 13, 452, 196, 410, 131, 312, 110, 28, 416, 444, 256, 395, 434, 250, 395, 174, 181, 93, 152, 401, 405, 214, 473, 137, 101, 410
Sorted array: 13, 26, 28, 93, 101, 110, 127, 131, 137, 152, 157, 157, 174, 181, 196, 214, 223, 250, 256, 302, 312, 333, 381, 395, 395, 401, 405, 410, 410, 416, 434, 444, 452, 465, 473

Total number of elements: 41
Unsorted array: 160, 41, 242, 321, 15, 39, 54, 1, 329, 74, 267, 195, 251, 403, 377, 209, 382, 396, 152, 409, 437, 451, 391, 170, 251, 139, 104, 400, 52, 494, 224, 174, 219, 304, 365, 134, 411, 205, 307, 279
, 180
Sorted array: 1, 15, 39, 41, 52, 54, 74, 104, 134, 139, 152, 160, 170, 174, 180, 195, 205, 209, 219, 224, 242, 251, 251, 267, 279, 304, 307, 321, 329, 365, 377, 382, 391, 396, 400, 403, 409, 411, 437, 451,
494


     N                              | Actual Count                  | Worst case T(N)
-----------------------------------------------------------------------------------------------
    48                              |      1769                     |     2304
    35                              |      1209                     |     1225
    40                              |      1342                     |     1600
    49                              |      2080                     |     2401
    43                              |      1276                     |     1849
    30                              |       881                     |      900
    40                              |      1255                     |     1600
    40                              |      1327                     |     1600
    35                              |      1011                     |     1225
    41                              |      1203                     |     1681
rohitkrishnanvidyasagar@Rohits-MacBook-Air assignment-1-insertion-sort %
```

Insertion Sort Time Complexity

	Actual Count	Worst Case Time Complexity
31	889	961
34	1048	1156
35	1167	1225
41	1437	1681
42	1184	1764
45	1733	2025
47	1986	2209
49	2044	2401
51	2513	2601
54	2456	2916
57	2741	3249


Actual Count
Worst Case Time Complexity
Total Cost T(N)
N (Total Number of inputs)