

**Q2**

INSERTION SORT ON SMALL ARRAY INSTEAD OF MERGE SORT

Dr. Maryam Bashir

Design and Analysis of Algorithms (CS-B)

FAST National University

**By**

Ayesha Tahreem Aamir (14L-4473)

15th March 2017

Contents

[Runtime Environment 3](#_Toc476608668)

[Experiment 3](#_Toc476608669)

[Time Complexity of Sorting Algorithms 3](#_Toc476608670)

[Results 4](#_Toc476608671)

[For standard merge sort 4](#_Toc476608672)

[For modified merge sort 4](#_Toc476608673)

[ For n = 1000 4](#_Toc476608674)

[ For n = 10000 5](#_Toc476608675)

[ For n = 50000 5](#_Toc476608676)

[ For n = 100,000 5](#_Toc476608677)

[Overall Analysis 6](#_Toc476608678)

# Runtime Environment

|  |  |
| --- | --- |
| System | 64-bit Operating System, x64-based processor |
| Processor | Intel® Core™ i3-4500U CPU @ 1.70GHz 1.70 GHz |
| RAM | 8.00 GB |
| Language | C++ |
| IDE | Microsoft Visual Studio 2015 |
| Compiler | Visual C++ 14.0 |

# Experiment

1. For each experiment, data set of **n** integers (for 1000, 10000, 50000 and 100,000) were generated using random generator algorithm (Mersenne Twister algorithm).
2. Standard merge sort was executed 5 times on the **n** sized unsorted data to get an average running time. Keeping n same, modified merge sort was executed 5 times with different values of k to get average running time.
3. Different values of **k** were computed. **k** having least avg. running time in modified merge sort has been chosen as optimal value of **k** in order to improve time complexity of modified merge sort.

# Time Complexity of Sorting Algorithms

For worst-case;

* Time complexity for insertion sort of n sized array = O(n^2)
* Time complexity for insertion sort of k sized array = O(k^2)
* Time complexity for insertion sort of n sized n/kth array = O((n/k) \* (k^2)) = O(n\*k) 🡪 A
* Since levels = log(n/k) and complexity of each level = O(n) hence

running time of all levels up to n/k sub-lists = O(n\*log(n/k)) 🡪 B

From A and B:

* Total running time complexity of modified merge sort = O((n\*k) + n\*log(n/k))

# Results

Running time (in milliseconds) for both standard and modified merge sort is given as:

## For standard merge sort

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | T1 | T2 | T3 | T4 | T5 | Average |
| 1000 | 2 | 1 | 2 | 2 | 1 | 1.6 |
| 10,000 | 17 | 17 | 16 | 16 | 16 | 16.4 |
| 50,000 | 88 | 85 | 84 | 85 | 84 | 85.2 |
| 100,000 | 170 | 168 | 168 | 168 | 168 | 168.4 |

## For modified merge sort

### For n = 1000

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K | T1 | T2 | T3 | T4 | T5 | Average |
| 10 | 1 | 2 | 2 | 2 | 2 | 1.8 |
| 50 | **2** | **1** | **1** | **1** | **2** | **1.4** |
| 100 | 2 | 2 | 2 | 2 | 2 | 2 |
| 200 | 2 | 1 | 2 | 2 | 2 | 1.8 |
| 500 | 2 | 2 | 1 | 1 | 2 | 1.6 |

### For n = 10000

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K | T1 | T2 | T3 | T4 | T5 | Average |
| 250 | 17 | 16 | 17 | 16 | 17 | 16.6 |
| 500 | 16 | 16 | 17 | 17 | 17 | 16.6 |
| 1000 | **16** | **17** | **17** | **16** | **16** | **16.4** |
| 5000 | 16 | 17 | 16 | 17 | 16 | 16.4 |
| 8000 | 16 | 17 | 17 | 16 | 16 | 16.4 |

### For n = 50000

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K | T1 | T2 | T3 | T4 | T5 | Average |
| 1000 | **83** | **83** | **84** | **83** | **83** | **83.2** |
| 5,000 | 83 | 84 | 83 | 83 | 83 | 83.2 |
| 10,000 | 83 | 84 | 83 | 98 | 100 | 89.6 |
| 13,000 | 84 | 84 | 83 | 84 | 83 | 83.6 |
| 25,000 | 83 | 83 | 83 | 83 | 83 | 83 |

### For n = 100,000

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K | T1 | T2 | T3 | T4 | T5 | Average |
| 20,000 | 168 | 167 | 167 | 169 | 168 | 167.8 |
| 40,000 | 167 | 167 | 167 | 167 | 168 | 167.2 |
| 50,000 | **168** | **168** | **167** | **167** | **167** | **167.4** |
| 60,000 | 168 | 194 | 198 | 180 | 169 | 181.8 |
| 90,000 | 167 | 196 | 186 | 170 | 167 | 177.2 |

# Overall Analysis

Since,

O (n\*k + n\*log (n/k)) < O (n\*log (n)) --holds true for 1 < k < 10

Also, avg. k value equals (50+1000+1000+50000)/4=13012.5 depicting that for arrays of size <= 10 elements, insertion sort performs better in rather than merge sort.

Thus, as mathematically justified, insertion sort performs better than merge sort for 1 < k < 10.