Programming Assignment

Design/Analysis of Algorithms

Professor. Cardei

Florida Atlantic University
By: Mackenzie Falla
2023

Table of Contents:

Problem Definition

Insertion Sort:

Problem: Given an array of elements, the goal is to sort the elements in ascending (or descending) order using the insertion sort algorithm.

Input: An array or list of elements, typically integers or real numbers. The length of the array, denoted as “n”.

Output: The same array with its elements rearranged in ascending order.

Objective: The objective of insertion sort is to rearrange the elements in the input array in a way that they form a sorted sequence, either in ascending order or descending. Insertion sort achieves this by iteratively taking one element at a time and inserting it into its correct position within the already sorted part of the array.

Merge Sort:

Problem: Given the array or list of elements, the goal is to sort the elements in ascending order using the merge sort algorithm.

Input: An array of list of elements , typically integers or real numbers. The length of the array is denoted by “n”.

Output: The same array with its elements rearranged in ascending order.

Objective: The objective of merge sort is to divide the input array into smaller subarrays, recursively sort these arrays, and then merge them back together to form a single sorted array. Merge sort is known for its stability and ability to handle large datasets efficiently.

Randomized Select:

Problem: Given an array of distinct elements and an integer k, where k is between 1 and the length of the array, the goal is to find the kth smallest element in the array.

Input: An array or list of distinct elements, typically integers or real numbers. An integer k, where 1 <= k < = n(the length of the array)

Output: The kth smallest element from the input array.

Objective: The objective of the Randomized Select algorithm is to efficiently find the kth smallest element from the input array without sorting the entire array. It achieves this by repeatedly partitioning the array into two subarrays based on a randomly selected pivot element and then narrowing down the search to the subarray containing the kth smallest element. The algorithm is often used when you don't need to sort the entire array but only need to find a specific order statistic.

The Selection Problem:

Problem: Given a collection of elements, the goal is to find the ith smallest element within the collection.

Input: A collection of elements (array, list, set, etc.), the size of the collection (*n*), and an integer *i* where 1 <= *i* <= n.

Input Size: The size of the input, *n*, ranges from *10,000* to *100,000* and includes every *10,000*th increment between the two values.

Output: The smallest element within the collection.

Objective: The selection problem, sometimes referred to as the "*i*th element search" problem, is a computational challenge centered on identifying the smallest or largest element within a collection of data, such as a list or array. In its typical form, and as we explore it, this problem involves finding the smallest element within an unsorted list.

Real-World Applications: The selection problem finds practical applications in network routing, computer graphics, and statistics. In network routing, it aids in locating the shortest path between nodes. In computer graphics, it helps identify the closest object to a specified point. In statistics, it is instrumental in extracting various statistics, including the median, from datasets.

Algorithms and RT Analysis

INSERTION_SORT

```
for j = 2 to A.length
      key = A[j]
      //insert A[j] into the sorted sequence A[1…j-1]
     i = j - 1
    while i >0 and A[i] > key
          A[i+1] = A[i]
          i = i -1
  A[i+1] = key
```

RT = O (n^2)

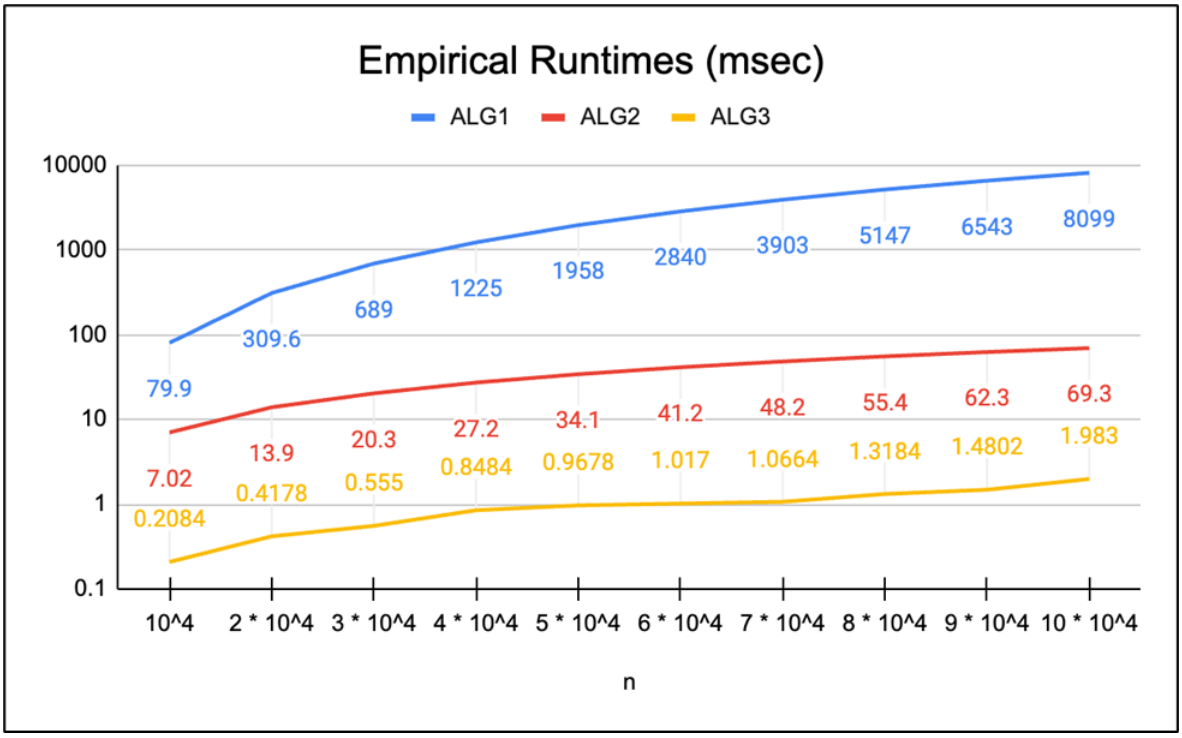
```
MERGE_SORT(A. P, q, r)
n1 = q-p+1
n2 = r-q
for i = 1 to n1
    L[i]=A[p+i-1]
for j=1 to n2
    R[j] = A[q+j]
L[n1 + 1] = infinity
R[n2 + 1] = infinity
i =1
j = 1
for k = p to r
     if L[i] ≤ R[j]
          A[k] = L[i]
          i = i + 1
     else
          A[k] = R[j]
          j = j + 1
```

RT = θ (n)

```
RANDOMIZED_SELECT(A, p, r, i)
if p == r
    return A[p]
q= RANDOMIZED_PARTITION(A, p, r)
k=q-p+1
if i = k
    return A[q]
else if i < k
     return RADOMIZED_SELECT(A, p, q-1, i)
else         //i > k
     return RANDOMIZED_SELECT(A, q=1, r, i-k)
```

RT = O(n)

Experimental Results



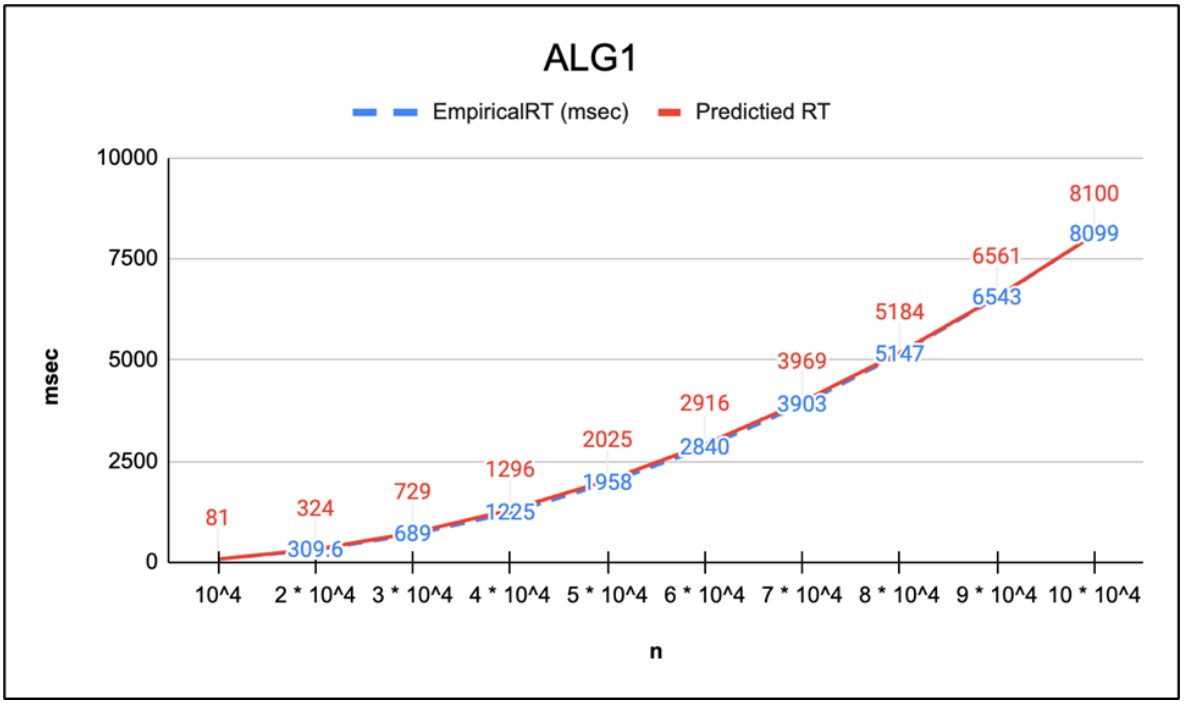

ALG1
EmpiricalRT (msec)
Predictied RT
msec
n
10000
7500
5000
2500
0
10^4
2 * 10^4
3 * 10^4
4 * 10^4
5 * 10^4
6 * 10^4
7 * 10^4
8 * 10^4
9 * 10^4
10 * 10^4
81
324
729
1296
2025
2916
3969
5184
6561
8100
309.6
689
1225
1958
2840
3903
5147
6543
8099

Table for ALG 1:

n	Theoretical RT n^2	Empirical RT (msec)	Ratio = (Empirical RT)/(Theoretical RT)	Predicted RT
10^4	10^8	79.9	$7.99 * 10^{-7}$	81
$2 * 10^4$	$4 * 10^8$	309.6	$7.74 * 10^{-7}$	324
$3 * 10^4$	$9 * 10^8$	689	$7.65 * 10^{-7}$	729
$4 * 10^4$	$16 * 10^8$	1225	$7.66 * 10^{-7}$	1296
$5 * 10^4$	$25 * 10^8$	1958	$7.83 * 10^{-7}$	2025
$6 * 10^4$	$36 * 10^8$	2840	$7.89 * 10^{-7}$	2916
$7 * 10^4$	$49 * 10^8$	3903	$7.97 * 10^{-7}$	3969
$8 * 10^4$	$64 * 10^8$	5147	$8.04 * 10^{-7}$	5184
$9 * 10^4$	$81 * 10^8$	6543	$8.08 * 10^{-7}$	6561
$10 * 10^4$	$100 * 10^8$	8099	$8.1 * 10^{-7}$	8100

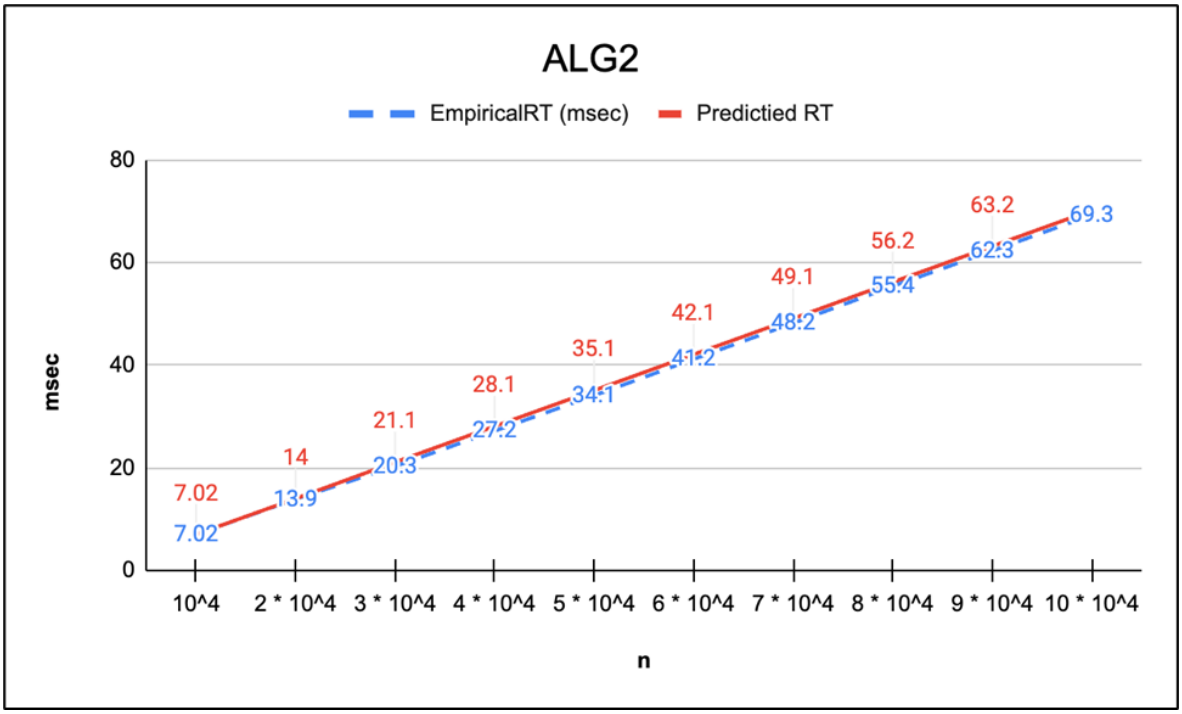

ALG2
EmpiricalRT (msec)
Predictied RT
msec
n
0
20
40
60
80
10^4
2 * 10^4
3 * 10^4
4 * 10^4
5 * 10^4
6 * 10^4
7 * 10^4
8 * 10^4
9 * 10^4
10 * 10^4
7.02
7.02
14
13.9
21.1
20.3
28.1
27.2
35.1
34.1
42.1
41.2
49.1
48.2
56.2
55.4
63.2
62.3
69.3

Table for ALG 2:

n	**Theoretical RT (n lgn)**	**Empirical RT (msec)**	**Ratio = (EmpiricalRT)/(TheoreticalRT)**	**Predicted RT**
10^4	132877	7.02	$5.283 * 10^{-5}$	7.02
$2 * 10^4$	265754	13.9	$5.230 * 10^{-5}$	14
$3 * 10^4$	398631	20.3	$5.092 * 10^{-5}$	21.1
$4 * 10^4$	531508	27.2	$5.118 * 10^{-5}$	28.1
$5 * 10^4$	664386	34.1	$5.133 * 10^{-5}$	35.1
$6 * 10^4$	797263	41.2	$5.168 * 10^{-5}$	42.1
$7 * 10^4$	930140	48.2	$5.182 * 10^{-5}$	49.1
$8 * 10^4$	1063017	55.4	$5.212 * 10^{-5}$	56.2
$9 * 10^4$	1195894	62.3	$5.209 * 10^{-5}$	63.2
$10 * 10^4$	1328771	69.3	$5.215 * 10^{-5}$	70.2

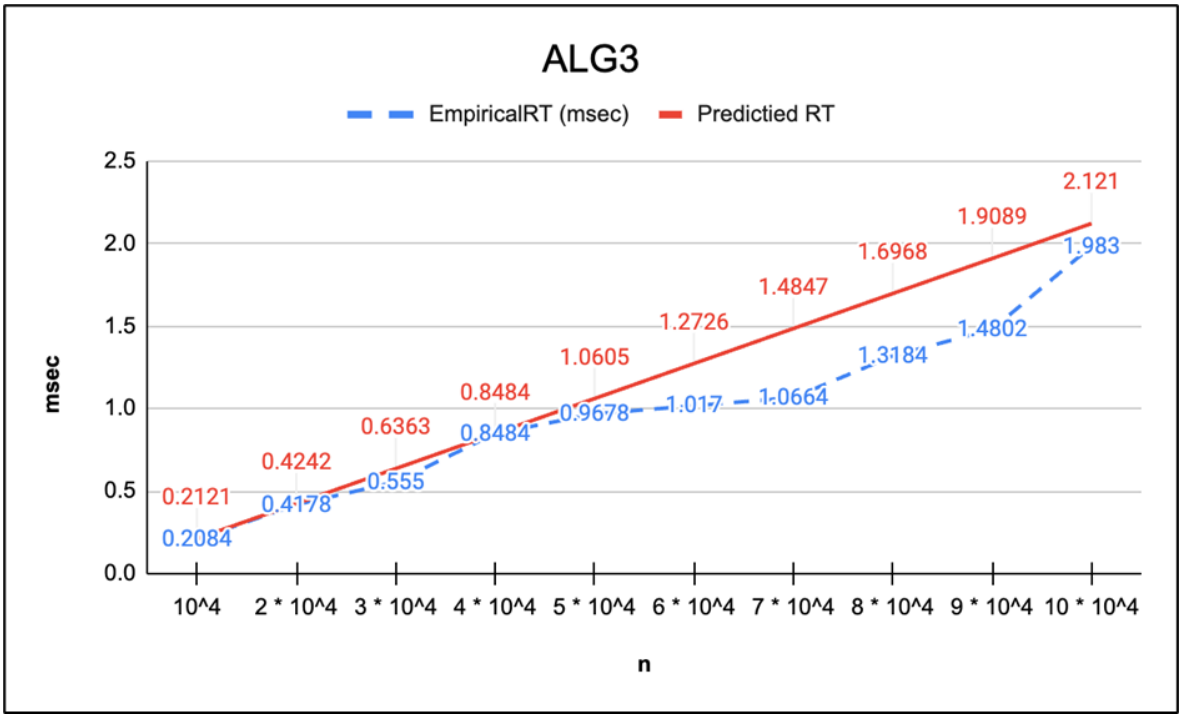

ALG3
EmpiricalRT (msec)
Predictied RT
msec
n
2.5
2.0
1.5
1.0
0.5
0.0
10^4
2 * 10^4
3 * 10^4
4 * 10^4
5 * 10^4
6 * 10^4
7 * 10^4
8 * 10^4
9 * 10^4
10 * 10^4
0.2121
0.4242
0.6363
0.8484
1.0605
1.2726
1.4847
1.6968
1.9089
2.121
0.2084
0.4178
0.555
0.8484
0.9678
1.017
1.0664
1.3184
1.4802
1.983

Table for ALG 3:

n	Theoretical RT n	EmpiricalRT (msec)	Ratio = (EmpiricalRT)/(TheoreticalRT)	Predicted RT
10^4	10^4	0.2084	$2.084 * 10^{-5}$	0.2121
$2 * 10^4$	$2 * 10^4$	0.4178	$2.089 * 10^{-5}$	0.4242
$3 * 10^4$	$3 * 10^4$	0.5550	$1.850 * 10^{-5}$	0.6363
$4 * 10^4$	$4 * 10^4$	0.8484	$2.121 * 10^{-5}$	0.8484
$5 * 10^4$	$5 * 10^4$	0.9678	$1.936 * 10^{-5}$	1.0605
$6 * 10^4$	$6 * 10^4$	1.0170	$1.695 * 10^{-5}$	1.2726
$7 * 10^4$	$7 * 10^4$	1.0664	$1.523 * 10^{-5}$	1.4847
$8 * 10^4$	$8 * 10^4$	1.3184	$1.648 * 10^{-5}$	1.6968
$9 * 10^4$	$9 * 10^4$	1.4802	$1.645 * 10^{-5}$	1.9089
$10 * 10^4$	$10 * 10^4$	1.9830	$1.983 * 10^{-5}$	2.1210

Conclusions

In this programming assignment, we successfully implemented three distinct algorithms. We began with insertion sort, followed by merge sort, and concluded with randomized select. Each algorithm employs unique strategies and possesses its own distinctive time complexity. Insertion sort is a straightforward sorting algorithm that constructs the final sorted array (or list) incrementally through comparisons, one item at a time. However, its efficiency significantly diminishes when applied to large lists compared to more sophisticated algorithms like quicksort, heapsort, or merge sort. Mergesort stands out as an efficient, versatile sorting algorithm grounded in comparisons. The majority of its implementations yield a stable sort, ensuring that the relative order of equivalent elements remains consistent between the input and output. Mergesort operates on the divide-and-conquer principle, contributing to its effectiveness. The randomized selection algorithm is as follows: Input: An array A of n distinct numbers and a number $k \in [n]$ Output: The "kth ranked element" of A (the element in position k if A were sorted).

References

"Insertion Sort (Article) | Algorithms." *Khan Academy*, www.khanacademy.org/computing/computer-science/algorithms/insertion-sort/a/insertion-sort.

GeeksforGeeks. "Merge Sort - GeeksforGeeks." *GeeksforGeeks*, 31 Oct. 2018, www.geeksforgeeks.org/merge-sort/.

"Understanding Randomized Select Algorithms." *Computer Science Stack Exchange*, cs.stackexchange.com/questions/124013/understanding-randomized-select-algorithm. Accessed 2 Oct. 2023.

"Randomized Algorithms." *GeeksforGeeks*, www.geeksforgeeks.org/randomized-algorithms/.