Comparison of Sorting Algorithms on Linux Platform

Hiteshri Patel[1] Ravi K Sheth[2]
[1]Student (M.Tech-Cyber Security) [2]Assistant Professor
[2]Department of Information Technology
[1,2]Raksha Shakti University Ahmedabad, Gujarat

Abstract— In this paper, we have implemented different types of sorting algorithms like Insertion sort, Selection sort and Merge sort on Linux platform and find the time complexity of these algorithm for Best case, Average case and Worst case. As the number of data increases then it affects the efficiency of the algorithms.

Key words: Complexity, Insertion sort, Selection sort, Merge sort, Best case, Average case and Worst case

I. INTRODUCTION

These three algorithms Insertion sort, Selection sort and Merge sort are used for sorting the array. But all algorithms follow different method for sorting. Time complexity is how much time required for executing the sorting of array. In these we had find three cases. The Best case running time of an algorithm gives us a lower bound for any input. The Worst case running time of an algorithm gives us upper bound for any input. Here, we have measured running time for different cases using 500 numbers of data as input.

II. INSERTION SORT

The Insertion sort is an efficient algorithm for sorting a small number of elements. In this sorting algorithm when we insert

The new value in array then first it will compare with first value of existing array if that value is smaller than existing one then it will came on first position. Otherwise one by one it will compare and take right position.

Here time complexity for best case in $\Theta(n)$ and for Worst case time complexity is $\Theta(n^2)$ and average case time complexity is $\Theta(n^2)$.

A. Pseudo code for Insertion Sort [1]:

INSERTION-SORT (A)

1) for j= 2 to A. *length*
2) *key* = A[j]
3) // Insert A[j] into the sorted sequence A[1...j -1].
4) i = j _ 1
5) while i > 0 and A[i] > *key*
6) A[i+1] = A[i]
7) i =i - 1
8) A[i+1] = *key*

B. Practical Analysis: (Time is measured in second):

1) Best Case:

```
[rsu@dhcppc12 ~]$ gcc 1.c
[rsu@dhcppc12 ~]$ gcc 1.c -o a
[rsu@dhcppc12 ~]$ ./a
BEST CASE
Required time is=3
```

Fig. 1: Best Case

2) Average Case:

```
[rsu@dhcppc12 ~]$ gcc 1.c
[rsu@dhcppc12 ~]$ gcc 1.c -o a
[rsu@dhcppc12 ~]$ ./a
AVERAGE CASE
Required time is=434
```

Fig. 2: Average Case

3) Worst Case:

```
[rsu@dhcppc12 ~]$ gcc 1.c
[rsu@dhcppc12 ~]$ gcc 1.c -o a
[rsu@dhcppc12 ~]$ ./a
WORST CASE
Required time is=124
```

Fig. 3: Worst Case

III. SELECTION SORT

The selection sort algorithm is also used for small amount of data for sorting elements. In this sorting algorithm first value of an array will compare with all the remaining values and then the value is replaced by higher value. In this one value was compared with every remaining value of array so more time required for execution. Here, for worst case, best case and for average case time complexity remains same.

A. Pseudo Code for Selection Sort [2]:

SELECTION_SORT (A)

1) for $i \leftarrow 1$ to n-1 do
2) min $j \leftarrow i$;
3) min $x \leftarrow$ A[i]
4) for $j \leftarrow i + 1$ to n do
5) f A[j] < min x then
6) min $j \leftarrow j$
7) min $x \leftarrow$ A[j]
8) A[min j] $\leftarrow$ A [i]
9) A[i] $\leftarrow$ min x

B. Practical Analysis (Time is measured in second):

1) Best case:

```
[rsu@dhcppc12 ~]$ gcc 1.c
[rsu@dhcppc12 ~]$ gcc 1.c -o a
[rsu@dhcppc12 ~]$ ./a
BEST CASE
Required time is=356
```

Fig. 4: Best Case

2) Average Case:

```
[rsu@dhcppc12 ~]$ gcc 1.c
[rsu@dhcppc12 ~]$ gcc 1.c -o a
[rsu@dhcppc12 ~]$ ./a
WORST CASE
Required time is=512
```

Fig. 5: Average Case

3) Worst case:

```
[rsu@dhcppc12 ~]$ gcc 1.c
[rsu@dhcppc12 ~]$ gcc 1.c -o a
[rsu@dhcppc12 ~]$ ./a
AVERAGE CASE
Required time is=720
```

Fig. 6: Worst Case

IV. MERGE SORT

Merge sort algorithm is generally used for large size of data. Merge sort algorithm used Divide and Conquer method. Divide and Conquer method break the problems into similar sub problems and solve these sub problems recursively, and then combine these solutions to create a solution to the original problem.

For Merge sort algorithm Divide and Conquer method is applied as follows:

- Divide: Divide the n elements into n/2 subparts.
- Conquer: Sort the two parts recursively using merge sort algorithm.
- Combine: Merge the two sorted parts.

Here, time complexity for Best case, Average case and Worst case is same as Θ(nlogn).

A. Pseudo Code for Merge Sort Algorithm [1]:

Here, In MERGE (A,p,q,r) A is an array, p, q and rare indices into the array such that p <= q < r. The procedure assumes that the subarrays A [p…q] and p[q+1….r] are in sorted array.

MERGE (*A, p, q, r*)

1) $n_1 \leftarrow q - p + 1$
2) $n_2 \leftarrow r - q$
3) Create arrays L[1 . . n_1 + 1] and R[1 . n_2 + 1]
4) FOR $i \leftarrow 1$ TO n_1
5) DO L[i] ← A[$p + i - 1$]
6) FOR $j \leftarrow 1$ TO n_2
7) DO R[j] ← A[$q + j$]
8) L[$n_1 + 1$] ← ∞
9) R[$n_2 + 1$] ← ∞
10) $i \leftarrow 1$
11) $j \leftarrow 1$
12) FOR $k \leftarrow p$ TO r
13) DO IF L[i] ≤ R[j]
14) THEN A[k] ← L[i]
15) $i \leftarrow i + 1$
16) ELSE A[k] ← R[j]
17) $j \leftarrow j + 1$ MERGE-

SORT (*A, p, r*)

1) IF p < r
2) THEN q ← [(p + r)/2]
3) MERGE (A, p, q)
4) MERGE (A, $q + 1$, r)
5) MERGE (A, p, q, r)

B. Practical Analysis (Time is measured in second):

1) Best case:

```
[rsu@dhcppc12 ~]$ gcc 1.c
[rsu@dhcppc12 ~]$ gcc 1.c -o a
[rsu@dhcppc12 ~]$ ./a
BEST CASE
Required time is=53
```

Fig. 7: Best Case

2) Average Case:

```
[rsu@dhcppc12 ~]$ gcc 1.c
[rsu@dhcppc12 ~]$ gcc 1.c -o a
[rsu@dhcppc12 ~]$ ./a
AVERAGE CASE
Required time is=53
```

Fig. 8: Average Case

3) Worst Case:

```
[rsu@dhcppc12 ~]$ gcc 1.c
[rsu@dhcppc12 ~]$ gcc 1.c -o a
[rsu@dhcppc12 ~]$ ./a
WORST CASE
Required time is=67
```

Fig. 9: Worst Case

V. ANALYSIS OF ALL ALGORITHMS

For the experimental purpose we have used 500 elements as a sample data and performed algorithms on Linux platform. For the three different cases we have arranged data in following manner 1) Best case- data is already sorted in ascending order 2) Average case- random sequence is used 3) Worst case- data is sorted in descending order. We have measured worst, average and best case running time with the help of *gatetimeofday ()* function. In this function we have used inbuilt two structures timeval and timezone of the said function. With the use of structure variable tv and tz we have fetched running time. This function use in this manner:

struct timeval tv; struct timezone tz;
gettimeofday(&tv,&tz);

Here, we have compared three algorithms Insertion sort, Selection sort and Merge sort based on time complexity.

Algorithms	Best case	Average case	Worst case
Insertion sort	3sec	434 sec	124 sec
Selection sort	356 sec	720 sec	512 sec
Merge sort	53 sec	53 sec	67 sec

Table 1: Analysis of All Algorithms

VI. CONCLUSION

This paper illustrated three basic types of sorting algorithms. After performing all the algorithms based on time complexity we have conclude that merge sort is best in all kind of data because in merge sort array is divided into two parts so this is faster than Selection sort. Insertion sort and Selection Sort are also used for small amount of data.

REFERENCES

[1] T. Cormen, C. Leiserson, and R. Rivest, Introduction to Algorithms, McGraw-Hill,New York,2008
[2] http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting /selectionSort.htm
[3] A. D. Mishra and D. Garg, "Selection of the best sorting algorithm", International Journal of Intelligent Information Processing, vol. 2, no. 2, (2008)

