Comparison of Sorting Algorithm

Merge Sort Compare with Radix Sort

Muhammad As'ad Muyassir
Departement of Electrical Engineering
University of Indonesia
Jakarta, Indonesia
muhammad.asad@ui.ac.id

***Abstract*—This paper will be explain about comparison from merge sort algorithm with radix sort algorithm. The comparison will be illustrate with the calculation of big O notation and with the experiment with python language. From the data obtained from the experiment, we will find which algorithm is better and more efficient from merge sort and radix sort.**

Keywords—algorithm, sorting, recursive, iterative, merge sort, radix sort, pseudocode, python, running time, efficient, big O

I. INTRODUCTION

Sorting algorithm is a basic algorithm that is usually used to compile a list into a sequence from the lowest or highest order. In its application, sorting algorithms are often used in various types of database processing applications, logical operators, data filters, pattern search, etc. With the importance of using sorting algorithms in everyday applications, sorting algorithm must be have high efficiency performance to increase the performance of main algorithm that require sorting algorithm. The algorithm is divided into two methods, recursive and iterative. Recursive sorting is a algorithm who will call himself to complete the sorting of list meanwhile, iteratively uses the looping in function to sorting the list.

In this paper we will discuss the comparison between recursive and iterative using one example algorithm in each method, the recursive method uses the merge sort algorithm while the iterative method uses the radix sort algorithm. The comparison is done to find the most effective algorithm in sorting a list with a stable time when given little input even when given a very large number of inputs. Testing with random input for any amount of data make input variations and from that we can also know the advantages and disadvantages of each algorithm.

Each chapter will be discussed in detail about the comparison between merge sort and radix sort. In the discussion at the beginning of the paper will begin by giving the reader knowledge about the things that underlie the experiments carried out, how the experiment can be done, analysis using big O notation. Then the test results will be given data with ideal conditions when the experiments are conducted. The next chapter will also explain in detail how these results can be obtained and why they occur. Next in the final chapter will be given a conclusion of all the tests that have been done.

II. BASIC THEORY

A. Merge Sort

Merge sort is one of the recursive type sorting algorithms. That is because in merge sort it calls itself repeatedly in the same way in each given list. List that can be sorted in the form of numbers in the array or a combination of letters (strings).

Merge sort is a sorting algorithm used to sort a list of data by dividing it into two parts which then the part will be divided into two parts until only one piece of data remains in one section. After each section has only one piece of data, each section will be recombined with other parts by observing the value of which part is greater so that the results of the merger of that part will be sorted and it will be repeated until all parts are combined into a unified whole and has been sorted.

The following is an example of a merge sort operation with step by step process with illustration from the example given array is [38, 27, 43, 3, 9, 82, 10].

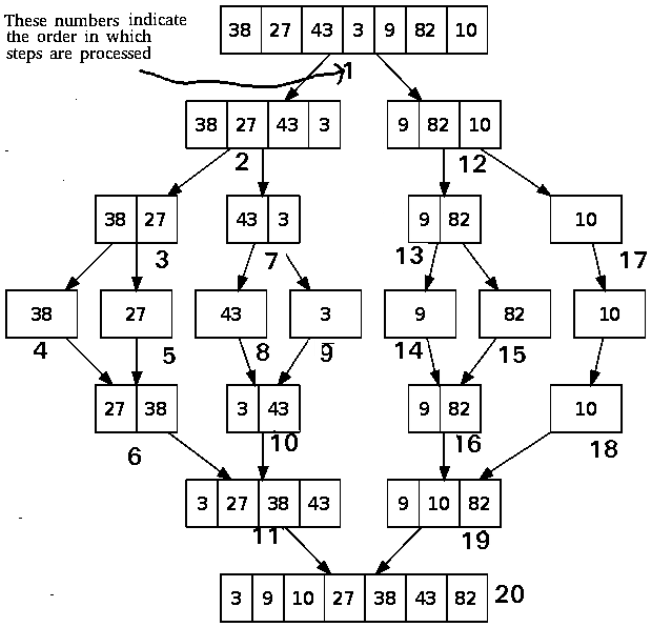

Fig. 1. Step of Merge Sort Process

It can be seen from the illustration that the data that originally consisted of 7 data was divided into 2 parts first, with a total of 4 data and 3 data, then the data was further divided up to leave only 1 piece of my data in each part. After that, the data will be combined by observing the value to each other parts so that the results of the merger will produce a list of new data with the values that have been arranged.

B. Radix Sort

Radix sort is one of the iterative type sorting algorithms. That is because the merge sorting is done repeatedly until the list is sorted according to the number of lists provided. Lists that can be sorted are usually in the form of integers or numbers in a list or array with a large amount of data.

Radix sort is a sorting algorithm used to sort a list of data by dividing it into the last 1 digit number which will then be sorted by 1 digit number. After 1 digit number has been sorted, the next digit will also be a sorted digit and the array will be sorted again from the beginning, and so on until all digit

numbers have been sorted. After all the digits are sorted, all the numbers in the array have been sorted.

The following is an example of a radix sort operation with step by step process with illustration from the example given array with 3 digit numbers.

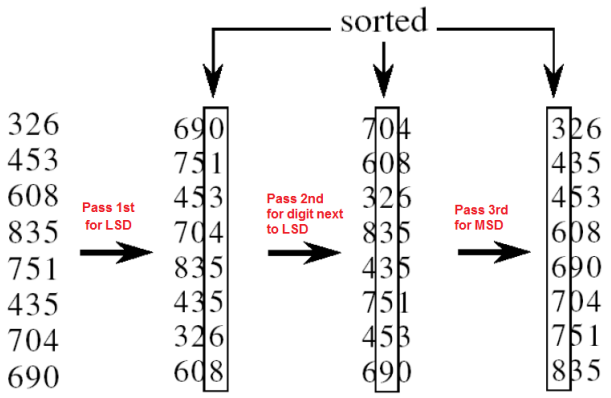

Fig. 2. Step of Radix Sort Process

It can be seen from the illustration that the data will be sorted by the rightmost digit first, after the rightmost digit has been sorted then the next digit will be sorted and so on. In the sorting process for each digit number, radix sort needs help from other sorting algorithms to sort them, because radix can only separate each digit number to then be sorted with other algorithms. So the time complexity of the radix sort algorithm is also affected by the algorithm that helps it, because the n input in the radix algorithm will be the n input of the algorithm that helps it and makes the radix algorithm a bit more complex in its work.

C. Pseudocode

Pseudocode from merge sort algorithm

Require : array x

```
n ← Panjang x
function MergeSort (x, p, n)
    if p < n do
        m ← (p+n)/2
        MergeSort (x, p, m)
        MergeSort (x, m+1, r)
        Merge (x, p, m, n)
    end if
end function
```

Pseudocode from radix sort algorithm

Require : array x

```
function RadixSort(x)
    max ← nilai maksimum pada array x
    index ← 1
    while max/index > 0:
        countingSort(arr, index)
        index *= 10
    end while
end function
```

D. Worst Case Analysis

For the pseudocode of the merge sort algorithm, a merge function can be defined with f (n) = 2 * f (n / 2) + n -1 where the function can be simplified to O (nlogn) to get the value of running time on the merge sort algorithm.

The pseudocode of the radix sort algorithm takes time with time complexity O (nk), but radix sort cannot stand alone because it has to sort each number on each digit, in this case counting sort is used to sorted it by time complexity O (n + k).

III. Experiment and Analysis

A. Computer Spesification for Test

In this paper, all experiments were performed on a computer with Intel Core I7-7700HQ processor, NVIDIA geforce GTX1050 2GB graphics card, 8GB DDR4-2400MHz memory, and 1TB 7200RPM harrdisk with Windows 10 operating system. The experiment written using python language and run using jupyter notebook that is based from anaconda3.

B. Result from Experiment

TABLE I. Table of Actual Running Time

Amount of Data	Actual Running Time (Microsesonds)	
	Merge Sort	Radix Sort
100	0.0	31253.34
1.000	15614.03	437427.28
10.000	46748.64	4041156.05
100.000	593496.80	42751903.30
1.000.000	7411774.87	439991469.62

Fig. 3. Table actual running time from experiment

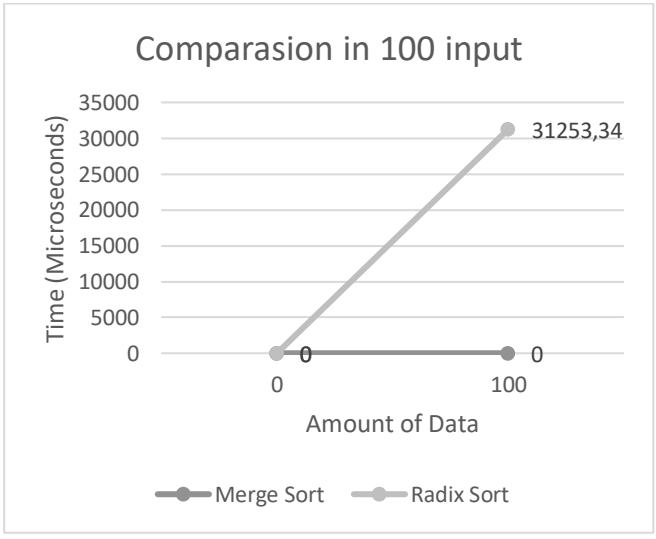

Fig. 4. Graphic comparison in 100 random input

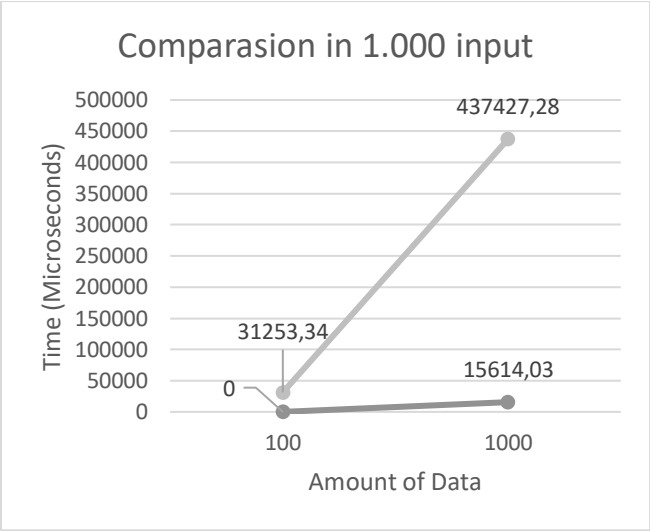

Fig. 5. Graphic comparison in 1.000 random input

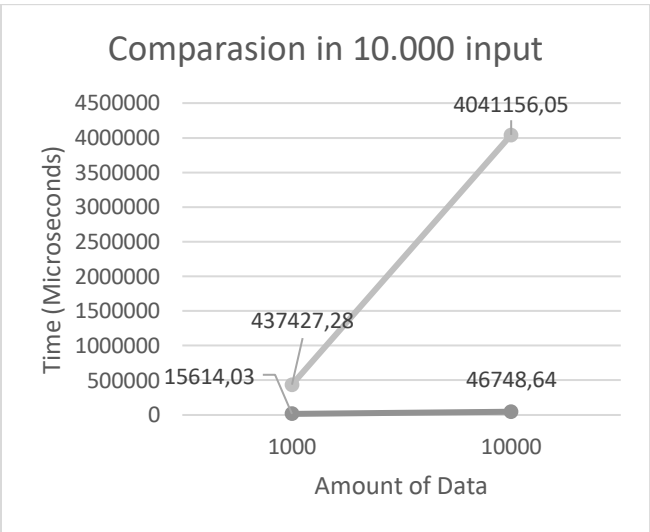

Fig. 6. Graphic comparison in 10.000 random input

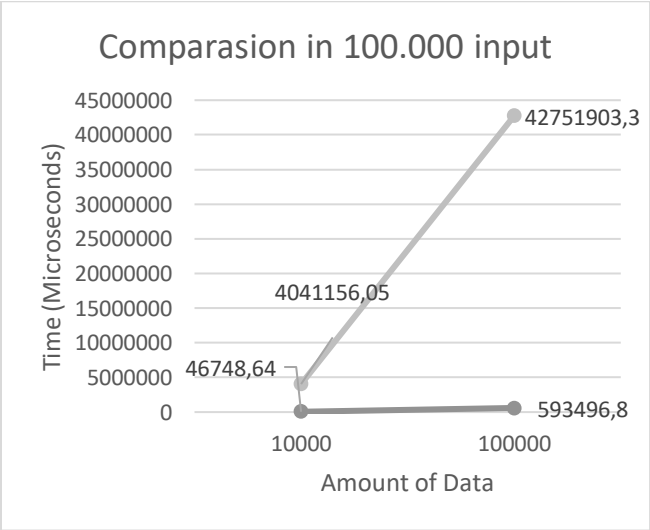

Fig. 7. Graphic comparison in 100.000 random input

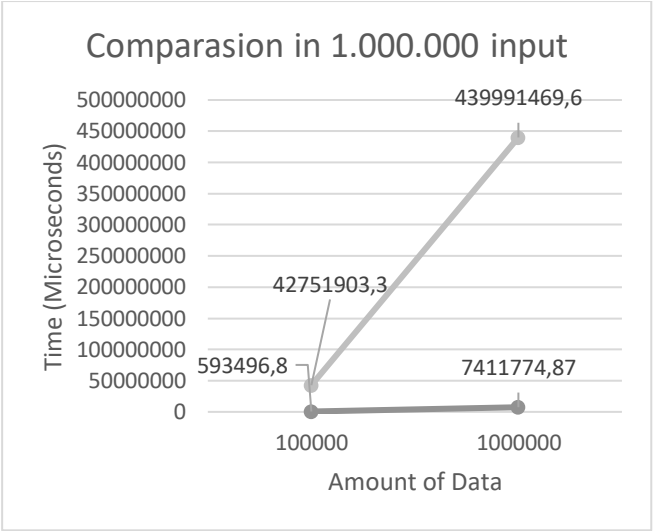

Fig. 8. Graphic comparison in 1.000.000 random input

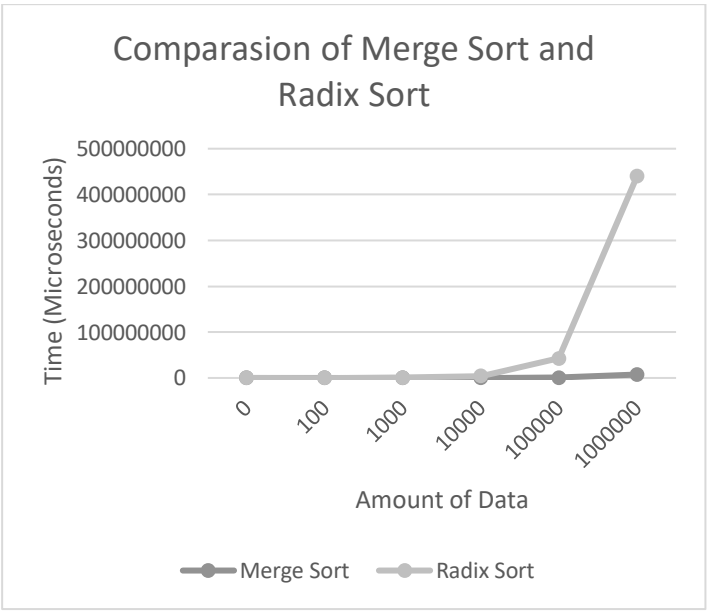

Fig. 9. Graphic actual running time from experiment

C. Analysis from Experiment

From the experiments, the results show that the performance shown by radix sort is very slow, especially if the constant factor *k* on the big O notation of radix sort gets bigger. It can be seen from the experimental results that the graph that occurs shows that radix is not very suitable if it is used to sort very large amounts of data, because radix sort also requires another type of sorting algorithm to sort the numbers on each digit of a number. Radix sort speed should be constant because the data used is in the form of numbers from 1 to 100, meaning that the time complexity O (nk) maximum value of *k* is only 3, but because of the use of counting sort to sort data on each digit, the algorithm radix sort will decrease its performance is directly proportional to the amount of data processed. In radix sort also sorting the entire data is done every digit number, then if there are 3 digit numbers then all data will be sorted 3 times, therefore the time complexity obtained by O (nk) because it will do the sorting repeat every digit number.

Meanwhile in the merge sort algorithm, they have their own time complexity with O (nlogn) without the need for other sorting assistance, where the graph that will be formed from time complexity is linear with a steady increase, it can also be seen from the results of experiments that show an increase in running time when the program is run it does not show high changes and has a low running time compared to radix sort. In merge sort, the time needed is only to divide the array into smaller parts, then the array will be rearranged in sorted order, then the merge sort only takes 2 times to solve it and complete the sorting process so that the running time will be constant increasing according to the number data inputted or sorted.

IV. Conclusion

In conclusion, the second sorting can be used to sort the data with a number that is not so much, although in this case merge sort is superior compared to radix sort. However, in case of data with a total of more than 100,000 the use of radix sort is not recommended because in radix sort the time required is very long because it is very dependent on other sorting algorithms, whereas the merge sort has a low running time and with a constant increase in the amount of data varies. Maybe the radix sort algorithm can improve its performance by replacing its supporting algorithm with an algorithm that has faster performance compared to counting sort, but still with a performance that cannot beat merge sort because if the constant value k in time complexity or number of digits is more than one then Radix sort performance will decrease significantly.

References

[1] Anonim. Sorting Algorithms. https://www.geeksforgeeks.org/sorting-algorithms/. accessed on 22 March 2020.

[2] Anonim. Merge Sort. https://www.geeksforgeeks.org/merge-sort/. accessed on 22 March 2020.

[3] Anonim. Radix Sort. https://www.geeksforgeeks.org/radix-sort/. accessed on 22 March 2020.

[4] Anonim. Counting Sort. https://www.geeksforgeeks.org/counting-sort/. accessed on 22 March 2020.

[5] Victor S.Adamchik. 2009. “Sorting” pittsburgh: Carniege Mellon University.

[6] Anonim. Time Complexities. https://www.hackerearth.com/practice/notes/sorting-and-searching-algorithms-time-complexities-cheat-sheet/. accessed on 22 March 2020

[7] Anonim. Radix Sort Pseudocode. https://www.codingeek.com/algorithms/radix-sort-explanation-pseudocode-and-implementation/. accessed on 23 March 2020

[8] A. Andersson, and S. Nillson. “A new efficient radix sort”, Lund. International Journal of Computer Science and Information Security (IJCSIS), December 2016.

[9] N. Akhter, M. Idrees, and F. Rehman, “Sorting Algorithms – A Comparative Study”, Lund. Sweden, December 1994.

[10] Th. H. Cormen, Ch. E. Leiserson, and R. L. Rivest. “Introduction to algorithms”, McGraw-Hill, 1990.