Available online at http://www.iorajournal.org/index.php/ijgor/index

International Journal of Global Operations Research

e-ISSN: 2722-1016
p-ISSN: 2723-1739

Vol. 3, No. 3, pp. 108-115, 2022

Comparative Analysis of the Speed of the Sorting Method on Google Translate Indonesian-English Using Binary Search

Maria Atik Sunarti Ekowati[1,*], Zefanya Permata Nindyatama[2], Widianto[3], Kristyanan Dananti4

[1]*Informatics Engineering Study Program, Faculty of Engineering, Surakarta Christian University, Central Java, Indonesia*
[2]*Communication Studies Program, Social and Political Sciences, Sebelas Maret University, Surakarta, Central Java, Indonesia*
[3]*Environmental Engineering Study Program, Faculty of Engineering, Surakarta Christian University, Central Java, Indonesia*
[4]*Management Study Program, Faculty of Economics, Surakarta Christian University, Central Java, Indonesia*

**Corresponding author email: maria.atik@uks.ac.id; maria.atik@gmail.com*

Abstract

In Indonesia, English is a compulsory subject for students. This course is an uninteresting subject and tends to be difficult for students to understand. Meanwhile, on the other hand, international issues are often discussed, namely English which is Students must know and be fluent in reading, writing and communication. The basic idea of research objectives, comparing the sorting process using two different algorithms, namely bubble sort and Merge Sort. The basic model of the research method Comparative Analysis of the Speed of the Shorting Method on Google Translate Indonesian-English Using Binary Search. Sorting, Sorted (ordered according to certain rules/rules), and the data is presented in sorted form, as said in dictionaries, and files in a directory. The algorithm used for sorting is bubble sort, which is an element comparison operation that is exchanged for other elements until the end of the data series is reached, until no more elements are swapped. Results for find out how well the performance speed of the bubble sort algorithm is in sorting data.

Keywords: Algorithm, Binary Search, Sorting, Bubble Sort, Merge Sort.

1. Introduction

Dictionaries play an important role in language learning because they can increase knowledge of vocabulary. The use of a dictionary may be minimal when there is homework to translate, but it is important when reading, writing and communicating. In this study, the author presents the title: "Comparative Analysis of the Speed of the Shorting Method on Google Translate Indonesian-English Using Binary Search". Technological advances have had an impact on the development of data. Data becomes bigger and more varied. As the amount of data increases, data processing becomes more complex. Before processing the data, there are processes that are carried out including sorting. Comparative Analysis of the Speed of the Shorting Method on Google Translate Indonesian-English Using Binary Search has its own advantages and disadvantages depending on the amount of data. Binary Search is a data search technique and must be sorted first using sorting techniques such as Bubble Sort, Quick Sort, Merge Sort, Insertion Sort, and Selection Sort. So that in the Comparative Analysis of the Speed of the Shorting Method on Google Translate Indonesian-English Using Binary Search, it will always relate to the Quick Sort, Merge Sort, Insertion Sort, Bubble Sort and Selection Sor algorithms. An algorithm that has the advantage of sorting large amounts of data. The Insertion Sort, Bubble Sort and Selection Sort algorithms have the advantage of sorting small amounts of data. Quick Sort and Merge Sort use the divide and conquer concept, namely by dividing the data into sub-sections and then dividing it again until the smallest parts are then sorted by small sub-sections. In the process of sorting these small subsections, they will be replaced by using the Bubble Sort, Insertion Sort, and Selection Sort algorithms. In this study, comparisons were made to the Merge Sort algorithm with Insertion Sort. The result of this research is that for data with less than 100 data, Insertion Sort is a faster algorithm than Merge Sort. However, for the amount of data more than 100, the Merge Sort algorithm is faster. In this study, comparisons were also made to the Merge Sort and Insertion Sort algorithms, but the difference was the amount of data used in the experiment. The results showed that Merge Sort was faster than Insertion Sort for large amounts of data. Comparisons were made to the Quick Sort and Insertion Sort algorithms, with the amount of data less than 100. Insertion Sort is a faster algorithm than Quick Sort. However, for the amount of data more than 100, the Quick Sort algorithm is faster. compared five algorithms, namely Quick Sort, Merge Sort, Selection Sort, Bubble Sort and Insertion Sort. The results of the research Quick Sort is a fast algorithm if the data used is 1000 but if the data is only 100 then Insertion Sort is a fast algorithm. Comparison made to five algorithms Bubble Sort, Selection Sort, Insertion Sort, Merge Sort and Quick Sort. The results of the Quick Sort research is the fastest algorithm. For small-scale data, Insertion Sort and Selection Sort are used. For data that has certain patterns and rules, Bubble Sort and Insertion Sort are used. For large-scale Quick Sort and Merge Sort data used, six algorithms are compared, namely Selection Sort, Insertion Sort, Merge Sort, Quick Sort, Bubble Sort and Comparison Sort. Bubble Sort and Quick Sort are the fastest algorithms (Hibbler, 2015; Sonita and Nurtaneo, 2015; Karve, 2016).

Various studies were conducted to increase the speed of the sorting algorithm, such as by optimizing Bubble Sort and Selection Sort. Optimization is done by reducing the number of data exchanges that occur during the sorting process. To increase the speed of the Bubble Sort and Selection Sort algorithms, optimization is done by avoiding the process of comparing data with other data during the sorting process, then optimization 1 on the Insertion Sort algorithm. Optimization is done by reducing the search for the right data position and reducing the process of exchanging data to the right position (Kumalasari, 2017; Tambunan, 2018). Quick Sort and Merge Sort algorithms are combined with Insertion Sort. The results of this study indicate that the results of the merger have a faster time. Merge-Insertion Sort is faster than Merge Sort. Quick-Insertion Sort is also faster than Quick Sort (Sitepu, 2017; Rachmat, 2018). The research carried out aims to obtain Comparative Analysis of the Speed of the Shorting Method on Google Translate Indonesian-English Using Binary Search which has a faster time in sorting data. The resulting algorithm has a faster time than with the usual algorithm (Horman, et al, 2016; Drodek, 2017; Rheinadi, 2019). This study uses another algorithm, namely Bubble Sort and Selection Sort for sorting data on a small scale can use Insertion Sort, Bubble Sort and Selection Sort so that Bubble Sort and Selection Sort can be a comparison for random data types so they need to be measured against other types of data such as reverse ordered and almost ordered. By doing this research, it is possible to add an algorithm that can be an alternative choice that can be adapted to the type of data such as random data, reverse ordered data and almost ordered data.

2. Methodology

The method that became the basic idea of the research is Comparative Analysis of the Speed of the Shorting Method on Google Translate Indonesian-English using Binary Search, which is to compare the sorting of algorithms and the process of combining algorithms that are often used in binary seats. The merging process is done by entering additional algorithms into the main algorithm. The main algorithm data is divided into small parts. It is in this small part that additional algorithms are used to sort the data. In order to distinguish when to use the main algorithm and when to use an additional algorithm, a limit value is needed.

If these small parts have a lot of data that is still above the limit value, then the main algorithm is used, but if it is smaller than the limit value, an additional algorithm is used. The main algorithms used are Merge Sort and Quick Sort. Additional algorithms are Insertion Sort, Bubble Sort and Selection Sort. By combining the main and additional algorithms, 6 candidates will be tested, namely, Merge-Insertion Sort (MIS), Merge-Bubble Sort (MBS), Merge-Selection Sort (MSS), Quick-Insertion Sort (QIS), Quick-Insertion Sort (QIS), Quick-Insertion Sort (QIS), and Quick-Insertion Sort. Bubble Sort (QSS) and Quick-Selection Sort (QSS). Six samples will be tested with 5 variants of the amount of data, namely 100, 1000, 10000, 100000 and 1000000 data words in the dictionary / google translate. The boundary value as a determinant between the main algorithm and additional algorithms is also used in the test. The limit value is the value used to determine when to use the main algorithm and additional algorithms. When the size of the data shared by the main algorithm is more than the limit value, the main algorithm is used, but if the data size is less than the limit, an additional algorithm is used. The limit values used are 8, 16, 32 and 64. There are 3 types of data tested, namely random, reverse-ordered and almost ordered. When testing, there will be changes to the main algorithm as well as additional algorithms, namely changes in determining the size and value of the data limit (Bingheng, 2018). Algorithm 1 below is an additional form of algorithm that has changed.

2.1. Insertion Sort

```
public void insertionSort(int A[],
int p, int r){
int key;
int i;
for(int j = p + 1 ; j <= r; j++){
key = A[j];
i = j;
while(i>p&& A[i-1] >= key){
A[i] = A[i-1];
--i;}
A[i] = key;}}
```

Algorithm 1. Modified form of Insertion Sort Algorithm

In this algorithm 1, the Insertion Sort algorithm has been modified. The form of changing the algorithm with the previous one is to add the values of p and r, each of which is an index that shows the beginning and end of a data. The p value is used to indicate the initial index of the data to be sorted. The value of r is used to designate the final index of the sorted data. The Insertion Sort algorithm is used when the size of the data resulting from the Quick Sort or Merge Sort division is below the limit value.

2.2. Bubble Sort

```
public void bubbleSort(int A[], int
p, int r){
        int temp = 0;
        for(int i=p;i<r;i++){
                for(int j=p+1;j<(r+1);j++){
                    if(A[j-1] > A[j]){
                        temp = A[j-1];
                        A[j-1] = A[j];
                        A[j] = temp;
                    }
                }
        }
}
```

Algorithm 2. Modified form of Bubble Sort Algorithm

In algorithm 2, the above is a modified Bubble Sort algorithm, changes to the Bubble Sort algorithm by adding p and r values, where p and r values are the initial and final indexes of the data section, the division results obtained at the time of Merge Sort division and Quick Sort. If the length of the data section is less than the limit value, the Bubble Sort algorithm is used to sort the data section.

2.3. Selection Sort

```
public void selectionSort(int A[],int
p,int r){
        for(int i=p;i<r;i++){
            int index = i;
            for(int j= i+1;j<r+1;j++){
                    if(A[j] < A[index]){
                            index = j;
                    }
            }
        int smallerNumber = A[index];
        A[index] = A[i];
        A[i] = smallerNumber;
    }
}
```

Algorithm 3. The modified form of the Selection Sort Algorithm

We know that in algorithm 3. is a modified Selection Sort algorithm. The form of changing the Selection Sort algorithm is to add p and r values. While the values of p and r are the initial and final indices of the data obtained from the results of the division of Merge Sort and Quick Sort. If the length of the data section is less than the limit value, the Selection Sort algorithm is used to sort the data section

4.4. Merge Sort

Below is the main Merge Sort algorithm that changes when it is used to sort parts of the data.

```
public void mergeSort(int A[], int
p, int r){
    if (p < r){
        if((r - p + 1) > batas){
            int q = (p+r)/2;
            mergeSort(A, p, q);
            mergeSort(A,q+1, r);
            merge(A, p, q, r);
        }
        Else {
            //insertionSort(A,p,r);
            //bubbleSort(A, p, r);
            //selectionSort(A, p,r);
        }
    }
}
```

Algorithm 4. Modified Merge Sort Algorithm

Merge Sort Algorithm form that has been modified, the algorithm 4 is done by adding conditions. If the data size is greater than the limit, then Merge Sort is carried out as usual. If the data size is less than the limit, then sorting is done using Insertion Sort, Bubble Sort or Selection Sort. The values of p and r are used to indicate the start and end of the data

2.5. Quick Sort

The Quick Sort algorithm is the same as the Merge Sort algorithm, changes are made by adding conditions. If the data size is greater than the limit, then sorting is done using the usual Quick Sort. However, if the data size is smaller then the sorting is done by Insertion Sort, Bubble Sort or Selection Sort. Algorithm 10 shows the modified Quick Sort algorithm. All data used in this experiment were obtained from randomization. In this experiment, the randomized numbers are numbers with one occurrence. Data randomization activities are carried out using a generator program.

The thing to do before measuring the sequencing time is to run the generator program first. The randomized data is stored in a file so that the comparisons are fair. In order to fulfill the type of data in reverse order, the random data stored will be sorted in reverse and then stored in a file using a program. In order to meet the types of data that are almost sorted, we also use the program, but not all data is sorted by the process. As with other types of data, quick sort is the same, that is, the types are almost sorted and stored in files. For example, the experiment was carried out with the IDE used, namely Netbeans 8 with JDK 8. And using an Intel Core i7-10700T processor, LGA 1151 socket, 4C/8T, 16 MB cache, 4.50 GHz Turbo Max frequency, 8 GB RAM and T- Power-Optimized lifestyle, 10th Generation. Below is the main Quick Sort algorithm that changes/modifies when it is used to sort parts of the data**.**

```
public void quickSort(int A[],
int p, int r) {
        if (p < r){
            if((r - p + 1) > batas){
            int q = partition(A, p, r);
            quickSort(A, p, q - 1);
            quickSort(A, q+1, r);
        }
        Else {
            //insertionSort(A,p,r);
            //bubbleSort(A, p, r);
            //selectionSort(A,p,r);
        }
    }
}
```

Algorithm 5. Modified Quick Sort Algorithm

3. Results and Discussion

3.1. Results

In this analysis and discussion, the results of the comparison of the algorithm used in the binary search method carried out in the data sorting process in the form of words in a dictionary / google translate can be seen in Table 1 below:

Table 1. Comparison Of Speed Algorithm with Number 100 Words Data and Random Data Types (In Nanoseconds)

Algorithm used	Limit			
	8	16	32	64
Merge Sort	127624	207342	131617	98362
Merge Insertion Sort	83292	49879	58752	82865
Merge Bubble Sort	73846	48695	78631	76628
Merge Selection Sort	68576	63884	62438	83820
Quick Sort	166194	166258	121190	116756
Quick Insertion Sort	176194	176521	117341	87641
Quick Bubble Sort	169779	243827	150535	164056
Quick Selection Sort	188923	241717	129908	176760

The time comparison of the main algorithm and the combined algorithm is shown in Table 1. What was done during the comparison experiment of the algorithm used in the binary search method was to compare the algorithm and the limit value to 100 random word data in the dictionary / google translate. The yellow-red part is the algorithm that has the fastest time. From the table it can be concluded that the fastest algorithm in this experiment is Merge Silection Sort (MSS) with a limit value of 16. From the table above, a graph is then made so that comparisons can be seen easily. The graph is made according to the table that is comparing the candidate algorithms and limit values against time.

Figure 1 shows the speed comparison for data size 100 and random data types. The algorithm that has the fastest overall time is Merge-Insertion Sort with a limit value that gives the fastest time at 16 bits.

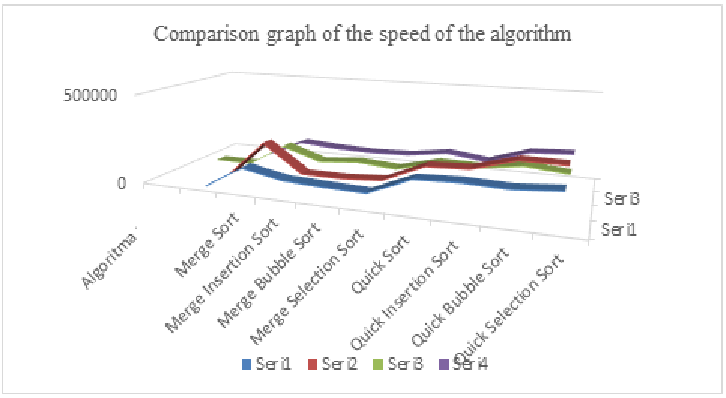

Figure 1. Comparison graph of the speed of the algorithm with the amount of data 100 words in the dictionary / google translate with random data types.

Furthermore, the comparison experiment of the algorithm used in the binary search method is carried out by comparing the algorithm and the limit value of 1000 random word data in the dictionary / google translate as Table 2 as follows:

Table 2. Comparison Of Speed Algorithm With Number 1000 Words Data And Random Data Types (In Nanoseconds)

Algorithm used	Limit			
	8	16	32	64
Merge Sort (MS)	960583	986535	944589	4974585
Merge Insertion Sort (MIS)	635923	568988	623948	2770774
Merge Bubble Sort (MBS)	762081	758087	1457019	3901065
Merge Selection Sort (MSS)	664575	639772	750533	1690518
Quick Sort (QS)	881396	869849	759087	1119172

Quick Insertion Sort (QIS)	911760	1044760	753955	1280398
Quick Bubble Sort (QBS)	883535	1170491	1247040	3319882
Quick Selection Sort (QSS)	771916	800142	972059	1558373

In Table 2. The above shows the comparison of the time of the main algorithm and the combined algorithm used in the binary search method. The comparison is done by comparing the algorithm and the limit value of 1000 random word data in the dictionary / google translate. The yellow-red part is the algorithm that has the fastest time. The fastest algorithm is Merge Insertion Sort (MIS) with a limit value of 16. From the table above, a graph is then made so that comparisons can be seen easily. The graph is made according to the table that is comparing the candidate algorithms and limit values against time.

Figure 2 shows a comparison of speeds for 1000 data sizes and random data types. The algorithm that has the fastest overall time is Merge-Insertion Sort with a limit value that gives the fastest time at 16 bits.

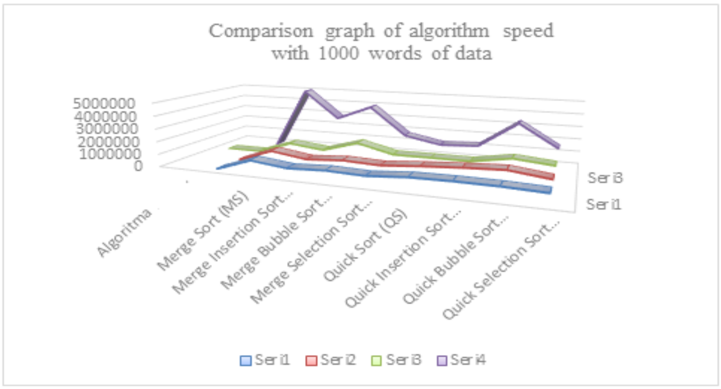

Figure 2. The graph of the comparison of the speed of the algorithm with the amount of data in 1000 words in the dictionary / google translate with random data types.

The next experiment to compare the speed of the algorithm carried out on the binary search method is to compare the algorithm for a data size of 100000 random word data in a dictionary / google translate. as table 3 as follows:

Table 3. Comparison Of Speed Algorithm with Number 10000 Data and Random Data Types (In nanoseconds)

Algorithm used	Limit			
	8	16	32	64
Merge Sort (MS)	35678221	47607358	33434032	35165178
Merge Insertion Sort (MMIS)	31684425	44154903	41642433	33410938
Merge Bubble Sort (MBS)	34745006	4283755	53623175	39808223
Merge Selection Sort (MSS)	36057622	46392392	46733232	35829325
Quick Sort (QS)	37991119	37430891	32415786	20340105
Quick Insertion Sort (QIS)	29842589	30326266	27960909	23644164
Quick Bubble Sort (QBS)	25276949	34466818	39351916	37199958
Quick Selection Sort (QSS)	23405105	31983000	25785431	34086205

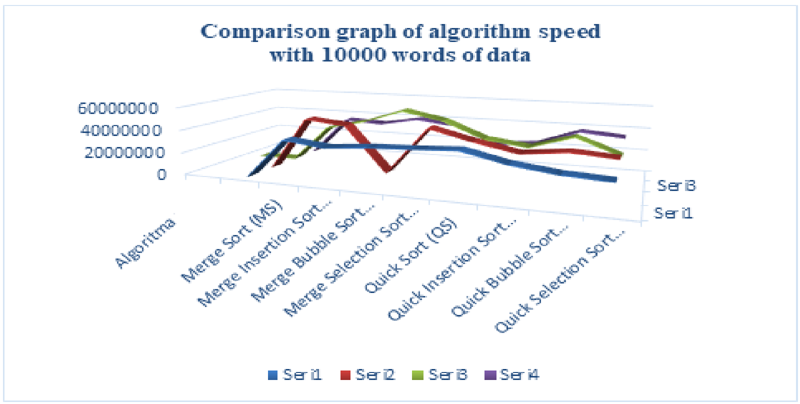

Figure 3. Comparison graph of the speed of the algorithm with the amount of data 10000 words in the dictionary / google translate with random data types.

The next algorithm speed comparison experiment was carried out on the binary search method up to 15 times. By comparing the algorithm for a data size of 1000000 types of words in a dictionary / google translate which are almost in order, as shown in table 4. below:

Table 4. Comparison Of Speed Algorithm with Number 1000000 Data and Almost Ordered Data Types (In nanoseconds)

	Limit			
Algorithm used	8	16	32	64
Merge Sort (MS)	1.69E+08	2.12E+08	1.73E+08	1.73E+08
Merge Insertion Sort (MMIS)	1.06E+08	1.16E+08 8.	1.16E+08	9.13E+07
Merge Bubble Sort (MBS)	8.13E+07	1.39E+08	1.37E+08	1.61E+08
Merge Selection Sort (MSS)	1.11E+08	1.33E+11	1.14E+08	1.04E+08
Quick Sort (QS)	9.27E+10	1.31E+11	3.20E+11	2.33E+11
Quick Insertion Sort (QIS)	5.23E+11	3.20E+11	4.33E+11	5.33E+11
Quick Bubble Sort (QBS)	1.20E+11	2.33E+11	6.33E+11	8.31E+10
Quick Selection Sort (QSS)	7.31E+10	9.20E+11	6.31E+11	8.53E+11

In Table 4. shows the comparison of the time of the main algorithm and the combined algorithm. The comparison is done by comparing the algorithm and the limit value of 1000000 types of words in the dictionary / google translate which are almost ordered. The yellow-red part is the algorithm that has the fastest time. The fastest algorithm is Merge-Bubble Sort (MBS) with a limit value of 8 bits. From the table above, a graph is then made so that the comparison can be seen easily. The graph is made according to the table that is comparing the algorithm and the limit value against time. Figure 4. shows the speed comparison for data size 1000000 and data type in reverse order. The algorithm that has the fastest overall time is Merge-Bubble Sort (MBS), for the limit value that gives the fastest time is 8 bits.

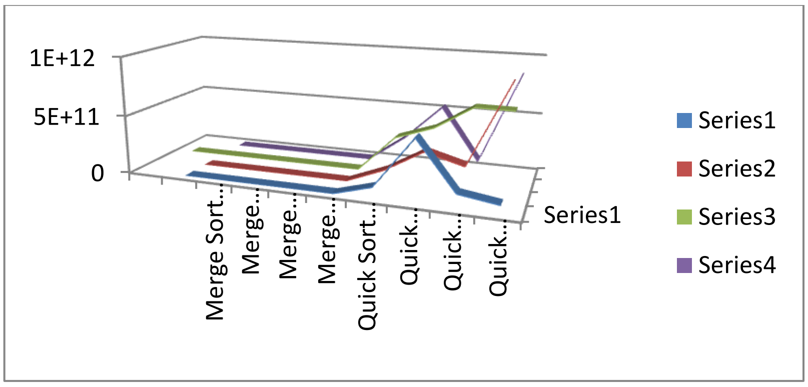

Figure 4. Comparison of the speed of the algorithm with the amount of data 1000000 types of words in the dictionary / google translate which are almost sorted

After the above experiments have been carried out, the comparison of the speed of the algorithm in the binary search method can be concluded that in Table 5. it can be seen which algorithms have the fastest time for each data size and data type, as shown in table 5. below this.

Table 5. Fastest Algorithm on Data Types and Data Size

	Data Type		
	Random	**Ordered Reverse**	**Almost Ordered**
100	Merge Insertion Sort (MIS, limit 16 bit	Merge Selection Sort (MSS), limit 16 bit	Merge Bubble Sort (MBS), limit 64 bit
1000	Merge Insertion Sort (MIS), Limit 16	Merge Selection Sort (MSS), limit 16 bit	Merge Insertion Sort (MMIS), limit 16 bit
10000	Merge Bubble Sort (MBS), limit 16 bit	Merge Sort (MS), limit 32 bit	Merge Selection Sort (MSS) limit 8 bit
100000	Quick Sort (QS), limit 64 bit	Merge Sort (MS), limit 32 bit	Merge Selection Sort (MSS), limit 8 bit
1000000	Quick Insertion Sort (QIS), 64 bit	Merge Selection Sort (MSS), limit 8 bit	Merge Bubble Sort (MBS), limit 8 bit

The combination of the main algorithm and additional algorithms has an effect when compared to the main algorithm alone. The combination of the main and additional algorithms can provide a faster time. The combination of the main and additional algorithms can also provide a longer time, as shown in table 6, as follows:

Table 6. Fastest Data Size

Size	**Data Type**		
	Random	**Ordered Reverse**	**Almost Ordered**
100	49879	5894	3,90E+05
1000	568988	5,80E+05	3,13E+06
10000	4283755	9,70E+05	2,13E+07
100000	20340105	5,99E+06	8,13E+07
1000000	2,25E+08	1,20E+05	8,13E+07

Based on experimental results, although Merge Sort and Quick Sort have the same way of working, Merge Sort has a faster time than Quick Sort. The biggest time difference between Merge Sort and Quick Sort occurs when the data types used are reverse ordered and almost ordered. If you look at Table 6, the algorithm that dominates in the experiment is Merge-Insertion Sort followed by Merge-Selection Sort. Merge Bubble Sort, Merger Sort, then Quick Sort. This can still be said to be better in the case of inversely ordered data types with data sizes from 10000 to 100000. Quick Sort has advantages in cases of random data types and 100000 data sizes. Quick-Insertion Sort can be used for cases of random data types and 1000000 data sizes.

4. Conclussion

From the research results, it can be concluded:

1. Combining the main algorithms Quick Sort and Merge Sort with additional algorithms Insertion Sort, Bubble Sort and Selection Sort can have an effect on time.
2. The effect of this merger can speed up as well as slow down. In addition to combining algorithms, limit values, the amount of data also affects the time.
3. Merge algorithms that dominate in the experiments carried out are Merge-Insertion Sort and Merge-Selection Sort. The results of combining algorithms can provide a faster time than Merge Sort. Merge Sort algorithm results in a faster time than Quick Sort.

References

Hibbler, R. (2015). *Merge Sort*. Dept. of Computer Science. Florida Institute of Technology. Florida, USA.

Sonita, A., Nurtaneo, F. (2015). Comparative Analysis of Bubble Sort, Merge Sort, and Quick Sort Algorithms in the Process of Sorting Combinations of Numbers and Letters. *Journal of Pseudecode, Vol 2*, No. 2, pp. 75-80.

Horman, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2016). *Introduction to Algorithms*. The MIT Press. Cambridge. Massachusetts London. England

Karve, S. (2016). *Insertion Sort Example*. http://www.dreamincode.net/code/snipp et279.htm. Retrieved 18 May 2016.

Kumalasari, D. (2017). Comparative Analysis of the Complexity of Bubble Sort, Cocktail Sort and Comb Sort Algorithms with C++ Programming Language. *Journal of Engineering and Education Research Centers (SPEED), Vol. 9*, No. 2, pp. 1-7.

Drozdek, A. (2017). *Data Structures and Algorithms in C++*. Brooks/Cole Thomson Learning. California. USA.

Sitepu, R. R. (2017). Implementation of Bubble Sort and Selection Sort Algorithm Using Multidimensional Arraylist in Multi-Priority Data Sorting. *Journal of Computing, Vol. 5*, No. 1, 2017, pp. 81-87.

Bingheng, W. (2018). *Merge Sort*. Dept. of Computer Science. Florida Institute of Technology. Florida, USA.

Tambunan, H. S. (2018). Optimization of Shell Sort Algorithm in Sorting Letters and Numbers Data. *Prima Computer Science Information System Journal (JUSIKOM PRIMA), Vol. 2*, No. 1, pp. 23-27 .

Rachmat, N. (2018). Comparison of Bubble Sort, Shell Sort, and Combination of Bubble Sort with Shell Sort. *JUSIKOM Journal, Vol. 3*, No. 1, pp. 59-64.

Rheinadi, R. (2019). Algorithmic Strategy. *Bandung Institute of Technology Paper*