ANALYSIS OF BUBBLE SORT AND INSERTION SORT ALGORITHM ON MEMORY EFFICIENCY USING DATA MINING APPROACH

Iqbal Dzulfiqar Iskandar[1*]; Imam Amirulloh[2]; Melisa Winda Pertiwi[3]; Mira Kusmira[4]; Agung Baitul Hikmah[5]; Deddy Supriadi[6]

[1,2,5,6]Information System
[1,2,5,6] Faculty of Engineering and Informatics, Universitas Bina Sarana Informatika, Tasikmalaya Campus
www.bsi.ac.id
[1]iqbal.iql@bsi.ac.id, [2]imam.iau@bsi.ac.id, [5]agung.abl@bsi.ac.id, [6]deddy.dys@bsi.ac.id

[3,4] Information System
[3,4]Sekolah Tinggi Manajemen Informatika dan Komputer Nusa Mandiri
www.nusamandiri.ac.id
[3]melisa.mwp@nusamandiri.ac.id, [4]mira.mik@nusamandiri.ac.id
*Corresponding Author

Abstract— Sorting algorithm in the computational process makes it easy for users when the data sorting process because the data is sorted by the process quickly and automatically. In addition to speed in sorting data, memory efficiency must also be considered. In this research, a retesting of two sorting methods is conducted, namely the bubble sort method and the insertion sort method based on the comparison of two programming languages, Java with Visual Basic 2010 using the decision tree method. This research aims to find out which algorithm has lower memory consumption in the sorting process using Java or Visual Basic 2010. The results of the comparison show, in Visual Basic 2010. insertion sort algorithm which has the lowest average memory consumption of 4.3243KB for .vb extensions and 2.0145KB for .exe extensions. while the bubble sort method with a consumption amount of 4.4358KB for the .vb extension and 2.0352 for extension.exe. Furthermore, if you use the Java programming language. So the bubble sort method still consumes the highest average memory, which is 546,242KB for the .jar extension and 4,337KB for the .exe extension, whereas from the insertion sort method, which has a low average memory consumption of 543,578 KB for extension .jar, and 4,381KB for extension .exe.

Keywords: Bubble sort, Decision Tree, Efficiency memory, Java, Visual Studio 2010.

Intisari— Algoritma *Sorting* dalam proses komputasi memberikan kemudahan bagi pengguna pada saat proses pengurutan data, karena data diurutkan dengan proses secara cepat dan otomatis. Selain kecepatan dalam pengurutan data, efisiensi *memory* juga harus diperhatikan. Pada riset ini dilakukan pengujian kembali terhadap perbandingan dua metode *sorting*, yaitu metode *bubble sort* dan metode *insertion sort* berdasarkan perbandingan dua bahasa pemograman, *Java* dengan *Visual basic* 2010 menggunakan metode *decision tree*. Riset ini bertujuan untuk mengetahui algoritma mana yang memiliki konsumsi *memory* yang lebih rendah dalam proses pengurutan menggunakan java atau *visual basic* 2010. Hasilkan hasil perbandingan menunjukan, pada *Visual basic 2010*, algoritma *insertion sort* yang memiliki rata-rata konsumsi *memory* yang paling rendah yaitu sebesar 4.3243KB untuk ekstensi .vb dan 2.0145KB untuk ekstensi .exe. sedangkan metode *bubble sort* dengan besaran konsumsi 4.4358KB untuk ekstensi .vb dan 2.0352 untuk ekstensi.exe. Selanjutnya jika menggunakan bahasa pemograman *Java.* Maka metode *bubble sort* tetap mengkonsumsi rata-rata *memory* yang paling besar yaitu sebesar 546,242KB untuk ekstensi .jar dan 4,337KB untuk ekstensi .exe, sedangkan dari metode *insertion sort,* yaitu yang memiliki konsumsi rata-rata *memory* yang rendah yaitu sebesar 543,578 KB untuk ekstensi .jar , dan 4,381KB untuk ekstensi exe

Kata Kunci*: Bubble sort, Decision Tree, Efisiensi memory, Java, Visual Studio 2010.*

PENDAHULUAN

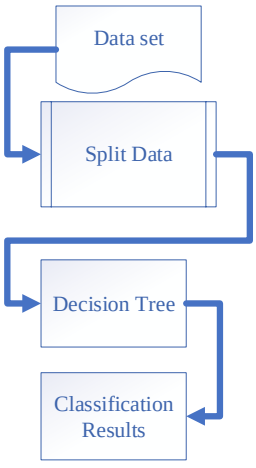
Sorting is the process of sorting computerized data using an algorithm that has been implemented into a programming language. Besides the speed in sorting data (Tjaru, 2009), Memory efficiency must also be considered. The results of previous studies, namely comparing two sorting methods, Bubble Sort with Insertion Sort on memory efficiency with implementation using C ++, (Saptadi & Sari, 2012) Bubble Sort sorting method which has lower

memory consumption while insertion sort is 40% faster in the sorting process compared to the selection sort method. (Suryani, 2013). Research related to decision trees that have been done before (Sutoyo, 2018) namely implementing a decision tree algorithm to classify students' classes based on the size of the report card value obtained, other studies related to the decision tree method, which is the classification process in legislative election data to predict election results and assess the accuracy of the decision tree algorithm for processing data accuracy, the accuracy value obtained is equal to 98.50% (Badrul, 2014). Based on previous research that has been done. Then do the re-testing of the comparison of two sorting methods, namely the bubble sort method and the insertion sort method based on the comparison of two programming languages, Java with Visual basic 2010 with a data mining approach using the decision tree method (Iskandar, 2019).

The purpose of this research is to identify the memory consumption of the shorting algorithm when compiling the data sorting process, by comparing two shorting algorithms that are different from the data mining approach, so it can be seen which algorithm has lower memory space consumption in the data sorting process using java or visual basic 2010.

MATERIALS AND METHODS

The proposed method for comparison analysis of the insertion sort algorithm with bubble sort is as follows:



Source:(Iskandar, 2019)

Figure 1. The framework classification of the memory efficiency of the proposed insertion sort and bubble sort algorithm.

Figure 1 is a proposed framework for analyzing the comparative efficiency of memory insertion sort algorithm with bubble sort, the explanation is as follows: first prepare the data set amount of memory consumption algorithm test results sorting process performed by the insertion sort algorithm with bubble sort, the white box testing process is done by sorting random number data from 100 numbers up to 1500 numbers. After preparing the data set. Then the next stage is the distribution of testing data and training data using Split data. The data used can be seen in table 1, table 2, table 3, and table 4

Table 1. Data set of memory size results from the sorting process of the Visual Studio 2010 bubble sort program algorithm

Number of digits	.vb (KiloBytes)	.exe (KiloBytes)	Average
100	4,224	1,944	3,084
200	4,28	1,944	3,112
300	4,268	2	3,134
400	4,224	2,004	3,114
500	4,44	2,044	3,242
600	4,428	2,024	3,226
700	4,28	2,032	3,156
800	4,296	2,064	3,18
900	4,364	2,044	3,204
1000	4,309	2,032	3,1705
1100	4,632	2,056	3,344
1200	4,656	2,036	3,346
1300	4,632	2,06	3,346
1400	4,828	2,184	3,506
1500	4,676	2,06	3,368

Source:(Iskandar et al., 2020)

Table2. Data set of the amount of memory from the Visual Studio 2010 Insertion Sort algorithm

Number of digits	.vb (KiloBytes)	.exe (KiloBytes)	Average
100	4,204	1,916	3,06
200	4,212	1,938	3,075
300	4,208	1,988	3,098
400	4,264	2	3,132
500	4,228	2,022	3,125
600	4,464	2,048	3,256
700	4,304	2,025	3,1645
800	4,348	2,036	3,192
900	4,276	2,024	3,15
1000	4,264	2,024	3,144
1100	4,688	2,024	3,356
1200	4,316	2,056	3,186
1300	4,372	2,056	3,214
1400	4,372	2,04	3,206
1500	4,344	2,036	3,19

Source: (Iskandar et al., 2020)

Table3. Data set of memory size results from the Java program's Insertion sort algorithm

Number of digits	.jar (KiloBytes)	.exe (KiloBytes)	Average
100	4,32	4,276	4,298
200	522,058	4,324	263,191
300	558,525	4,284	281,4045
400	574,114	4,348	289,231
500	565,568	4,324	284,946
600	587,544	4,36	295,952
700	606,92	4,356	305,638
800	637,153	4,38	320,7665
900	645,796	4,38	325,088
1000	668,576	4,38	336,478
1100	674,104	4,356	339,23
1200	410,22	4,388	207,304
1300	522,196	4,492	263,344
1400	584,42	4,564	294,492
1500	592,152	4,496	298,324

Source:(Iskandar et al., 2020)

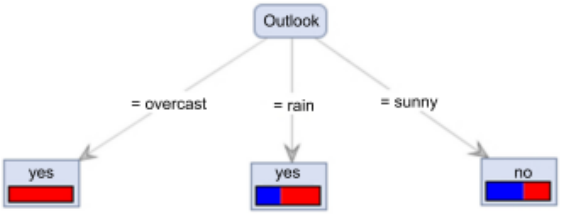
Table 4. Data set of memory size results from the sorting process of the Java bubble program sort algorithm

Number of digits	.jar (KiloBytes)	.exe (KiloBytes)	Average
100	8,012	4,296	6,154
200	480,552	4,304	242,428
300	567,54	4,304	285,922
400	552,888	4,304	278,596
500	559,86	4,336	282,098
600	585,352	4,396	294,874
700	589,224	4,356	296,79
800	624,432	4,14	314,286
900	636,522	4,32	320,421
1000	661,3	4,34	332,82
1100	656,596	4,412	330,504
1200	628,508	4,348	316,428
1300	505,303	4,38	254,8415
1400	551,02	4,364	277,692
1500	586,526	4,46	295,493

Source:(Iskandar et al., 2020)

Before data processing is done by a decision tree algorithm, the dataset is divided into two parts automatically by split data, namely training data and testing data with a ratio of 10% testing data and 90% training data (Rahayu et al., 2015). Testing data is used as a comparison between the results of the classification of training data in the past with the testing data in the future if the resulting classification results are the same or close. Then the classification results of processing algorithms for data have good accuracy.

After separating the testing data and training data the third stage is the classification process using the decision tree algorithm. *The decision tree is included in classification techniques in data mining. The perspective of an analyst, the classification tree, is used to separate a data set into classes on each response from a variable*. Usually, the response variable has two classes: Yes or no (1 or 0). If the response variable has more than two categories, a variant of the decision tree algorithm has been developed. In both cases, tree classification is used when the response or target variable is categorical.



Source:(Iskandar, 2019)

Figure 2. *Decision Tree weather prediction*

Figure 2 is an example of the Decision Tree Model making decisions by drawing a flowchart like an inverted tree, then the attributes will be tested at each node. At the end of a decision tree on a node or leaf is a prediction result that is generated based on the target variable. (Iskandar, 2019)

In the Decision Tree to construct a decision tree an Entropy or Gain index is needed because different criteria will build different trees through different biases. The formula for getting Entropy is as follows :

$$H = -\sum P_k Log_2(P_k) \quad \text{.....(1)}$$

Where k = 1,2,3, ... m represents the class for the target variable. Seedangakn (Pk) represents the proportion of the sample that belongs to class k. while the Gain Index (G) has similar characteristics to entropy measurements, for (G) can be defined by the following formula:

$$G = \sum(1 - {P_k}^2) \quad \text{.....(2)}$$

G values range between 0 and a maximum value of 0.5, but instead have identical properties to H, and one of these formulations can be used to create partitions on data. (Vijay Kotu & Deshpande, 2015).

The decision tree can be used for estimation in determining decisions, measuring and selecting decisions on node attributes. The decision tree algorithm used is the C4.5 decision tree algorithm based on ID3, which can handle continuous attributes, processing sample sets with missing values, resulting in rules and other new features. Therefore, the C4.5 algorithm is used to build a decision tree. (Jiang & Wang, 2016).

The C4.5 algorithm uses the information to obtain ratios as criteria for attribute selection and sample classification, which overcomes the lack of information in attribute selection. The calculation method is as follows:

Set S as the training sample set, | S | representation of the total number of training samples. Assume to be a property of S, with m non-repeat separate values. marked as V = {vl, v2, ..., vm} ... The training sample value in A is based on S divided into m subsets as {S1, S2, ..., Sm}.

The values on Si by all in training sample A are vi. U is a collection class, set freg (ui, S) shows the number of samples belonging to the ui class on S. *Enthalpy information, as a measure of uncertainty about the source of information of events that are likely to occur. The Enthalpy information in the sample set S divided according to the decision attribute U uses the following formula*

$$Info\ (U) = -\sum_i P(u_i) log_2 P(u_i) \quad \text{.....(3)}$$

P (Ui) represents the percentage of UI in the total number of samples. Conditional enthalpy, which indicates uncertainty when starting the original variable randomize after certain conditions. The conditioning formula for the conditional Enthalphy sample set S divided by domain partition attribute V is:

$$Info(U|V) = \sum_j P(v_j) \sum_i P(u_i|v_j) log_2 \left(\frac{1}{P(u_i|v_j)}\right) \quad \text{.....(4)}$$

P (ui | vj) shows the conditional probability condition in the ui category when the value of A is vj. Gain information, also known as mutual information, represents the uncertainty of the average output set for the input set. Gain information as a sample training set S based on the attribute partition A is using the following formula:

$$Gain(A) = Info(U) - (A|V) \quad \text{.....(5)}$$

V is all the output region sets in A. represent the status information of the output set V about A. Enthalpy has a difference by eliminating uncertain data, namely by the quality of information on the amount of information obtained. The training sample values for S sets with different values of m in attribute A is {S1, S2, ... Sm}, the calculation formula for partition information:

$$\text{SplitInfo(A)} = -\sum_{i=1}^{m} \frac{|S_i|}{|S|} \log_2 \left(\frac{|S_i|}{|S|}\right) \quad \text{.....(6)}$$

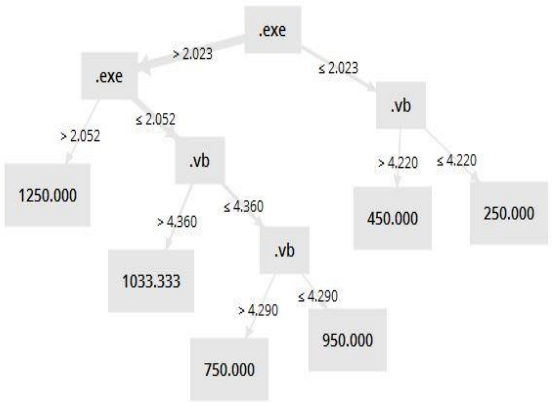
The gain ratio is the ratio of the gain rate information to the amount of the gain information. The formula for calculating Gain Ratios for the level of gain information in the training sample set S is classified by attribute A as follows:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)} \quad \text{.....(7)}$$

C4.5 algorithm, the test attribute selected for each node in the decision tree using the gain ratio information. Select the attribute with the highest gain ratio information as the test node attribute used. (Jiang & Wang, 2016). For a faster and more accurate process. Then the process of classifying the amount of memory data from the classification of bubble sort and insertion sort algorithm is done using rapid miner tools. The final stage is the analysis of the results of data classification using memory insertion sort and bubble sort algorithms that are processed by the decision tree method. So that it can be seen the level of efficiency of memory space consumption based on comparison of different programming languages namely Java and Visual Studio 2010.

RESULTS AND DISCUSSION

The final results of the classification process of memory size data from the processing of the sorting insertion sort and bubble sort algorithm using the decision tree method will be explained as follows:

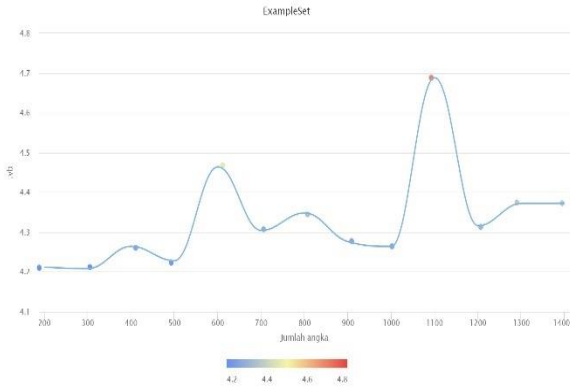


Source:(Iskandar et al., 2020)

Figure 3. Decision tree data is the amount of memory consumption by insertion sort algorithm with Visual Studio 2010

Figure 3 is a decision tree of data on the amount of memory consumption of the insertion sort algorithm implemented into the visual studio programming language 2010, From this process, we obtained data on the amount of memory that is classified based on the number sorting process by the Insertion Sort algorithm. When the insertion sort algorithm has become an executable (exe) program the sorting process totaling 1250,000

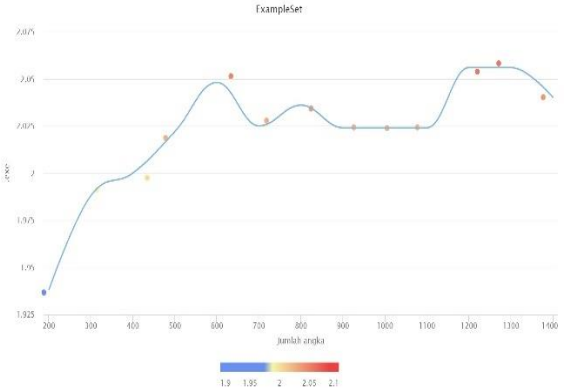
numbers consumes a minimum of 2,052 KB of memory. Whereas the average vb extension requires 4000KB of memory space to sort more than 950,000 random numbers.



Source:(Iskandar et al., 2020)

Figure 4. Scatter chart data amount of memory consumption algorithm insertion sort with visual studio 2010 extension vb program.

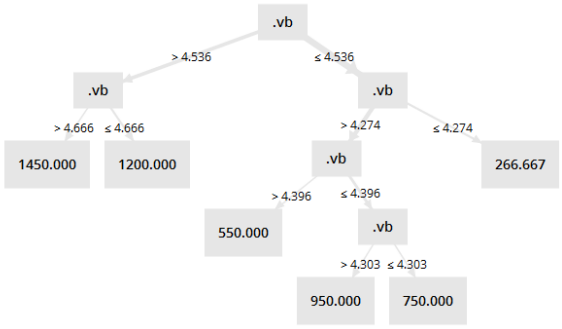
In Figure 4. scatter chart data the amount of memory consumption with the vb extension looks more and more random numbers are sorted. So the consumption graphic is getting better but not stable, between 4000KB-4700KB. While the ordering process with the EXE extension there is a decrease in memory consumption by 50% which will be explained in Figure 5.



Source:(Iskandar et al., 2020)

Figure 5. Scatter chart data amount of memory consumption algorithm insertion sort with Visual Studio 2010 extension EXE program.

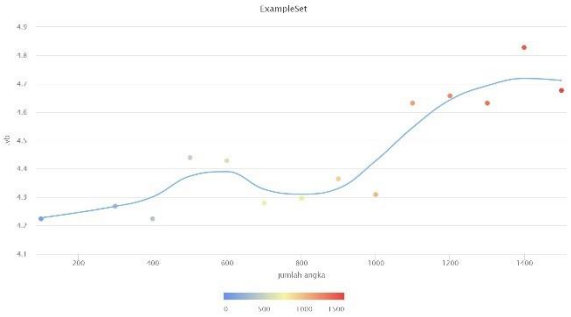
After the sorting program, the insertion sort is changed into an exe extension. In graph Figure 5 there was a significant reduction of around 50%, to 2000KB which used an average of 4000KB of memory space to be used for sorting random numbers.



Source:(Iskandar et al., 2020)

Figure 6. Decision tree data is the amount of memory consumption bubble sort algorithm with Visual Studio 2010 program

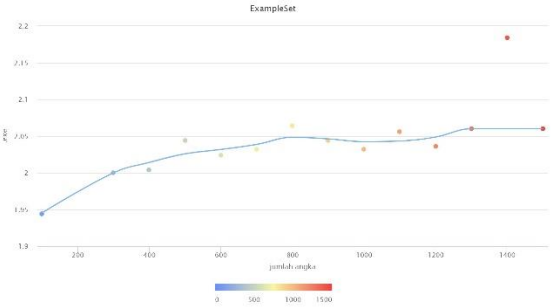
Figure 6 is the Decision Tree data of bubble sort memory consumption algorithm using Visual Studio 2010, the process of sorting random numbers in bubble sort with vb extension, consumes more memory than insertion sort, which is 4000KB-4666KB to sort 750.000 random numbers.



Source:(Iskandar et al., 2020)

Figure 7. Scatter chart data is the amount of memory consumption bubble sort algorithm with visual studio 2010 extension vb.

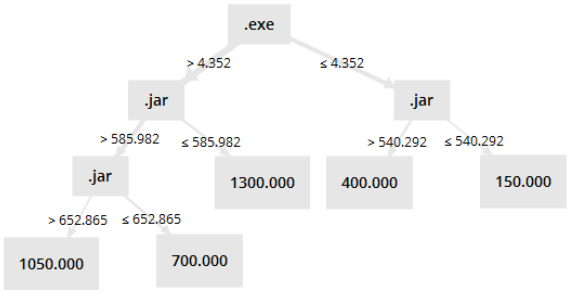
Seen in the scatter chart graph Figure 7. When the random number sorting process is 16000 digits. Then the memory consumption graph will continue to increase from 4000KB to 4666KB.



Source:(Iskandar et al., 2020)

Figure 8. Scatter chart data is the amount of memory consumption bubble sort algorithm with Visual Studio 2010 extension EXE program.

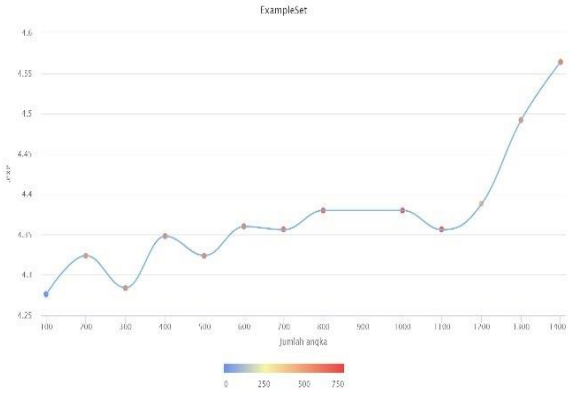
Figure 8 is a scatter chart graph illustrating memory consumption in the process of sorting random numbers using the bubble sort algorithm, the graph looks more stable and lower than the ordering of numbers with extensions vb. Namely memory consumption around 1944KB to 2184KB to sort 1500 random numbers.



Source:(Iskandar et al., 2020)

Figure9. Decision tree data is the amount of memory consumption algorithm insertion sort with the java extension .exe and .jar programs

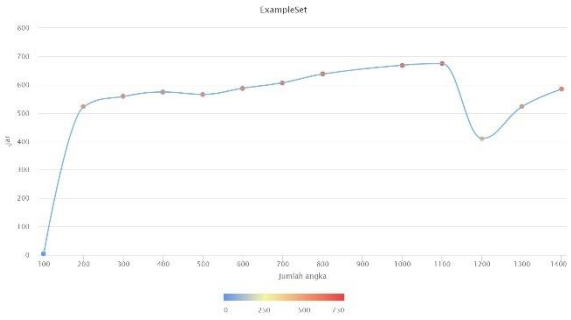
Figure 9 is a decision tree of the amount of Insertion Sort memory consumption with jar and ex. The memory consumption of the number sequencing process performed by the exe extension is 4.352KB, by sorting as many as 700,000 random digits. Whereas the sorting is done by the jar extension program only requires memory consumption of 652,865KB to sort 700,000 random numbers.



Source:(Iskandar et al., 2020)

Figure 10. Scatter Chart data is the amount of memory consumption by insertion sort algorithm with the .java extension .exe program

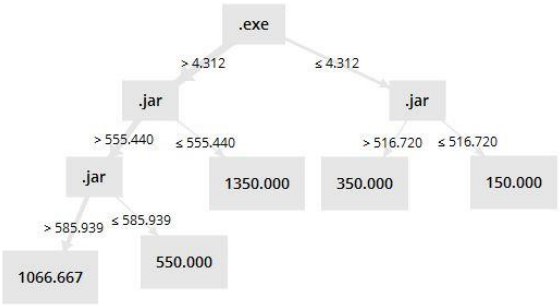
Figure 10 is a graph of the insertion sort sorting process with the java programming language with the extension exe, the graph shows the increasing number of numbers sorted. Then the greater memory consumption, which is as much as 4,564KB to sort 1400 random digit numbers.



Source:(Iskandar et al., 2020)

Figure11. Scatter Chart data is the amount of memory consumption by insertion sort algorithm with the java extension jar program

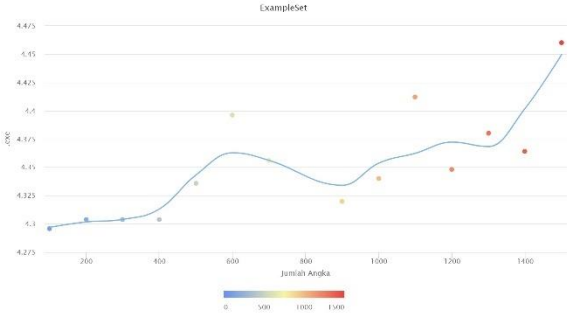
Figure 11 Scatter Chart data is the amount of memory consumption by the insertion sort algorithm with the java extension jar program.



Source:(Iskandar et al., 2020)

Figure 12. Decision tree data is the amount of memory consumption bubble sort algorithm with the java extension .exe and .jar programs.

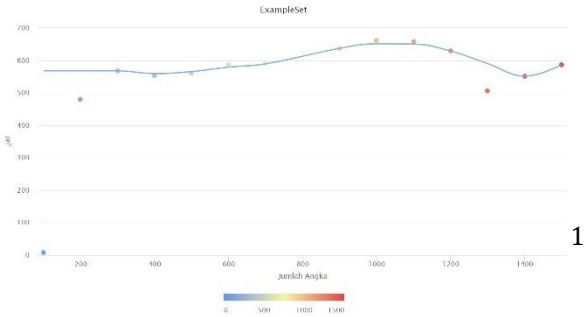
Figure 12 is a classification of memory consumption from the bubble sorting algorithm process using the Java programming language with the exe and jar status. The sorting process performed by the bubble sort algorithm with the jar extension that has been changed to exe requires a larger memory consumption of around 4,312KB to sort as many as 13500 random numbers. While the bubble sort program which still has the jar extension, to do the process of sorting as many as 13500 random numbers only require memory consumption of 555,540KB.



Source:(Iskandar et al., 2020)

Figure 13 Scatter Chart data is the amount of memory consumption bubble sort algorithm with the java extension exe program.

The Scatter chart in Figure 13 shows the graph of memory consumption increases when the program is sorting by the number of numbers that are more and more. To sort 1500 random numbers, the memory required is 4460KB.



1.

Source:(Iskandar et al., 2020)

Figure 14. Scatter Chart data is the amount of memory consumption bubble sort algorithm with the java extension jar program.

Whereas when compared to the bubble sort algorithm the Java programming language with the jar extension is seen in Figure 14, Memory consumption in the sorting process on the graph tends to be stable at 200KB-585KB to sort 100-1500 random numbers.

CONCLUSION

The process of comparing bubble sort and insertion sort methods in the use of memory efficiency with the programming language, Java and visual studio 2010 can be done. So that the results obtained as follows: The visual studio program insertion sort algorithm with executable (exe) sorting process totaling 1250,000 numbers uses at least 2,052 KB of memory. Whereas the average vb extension requires as much as 4.3243KB KB of memory space to sort more than 950,000 random numbers. The bubble sort method algorithm for memory efficiency in the Visual Studio 2010 programming language has the lowest average memory space consumption of 4.4358KB to sort 1450,000 random numbers in the .vb extension and 2.0145KB for the .exe extension. Then the most efficient algorithm is insertion sort in the sorting process. Algorithm The insertion sort method requires memory consumption from the number sequencing process performed by the 4.352KB exe extension with 700,000 random number sequences. Whereas the sorting is done by the jar extension program only requires memory consumption of 652,865KB to sort 700,000 random numbers. The sorting process performed by the bubble sort algorithm with the jar extension that has been converted into an exe(executable) extension requires a larger memory consumption of around 4,312KB to sort as many as 13500 random numbers. While the bubble sort program which still has the jar extension, to do the process of sorting as many as 13500 random numbers only require memory consumption of 555,540KB. so the most efficient algorithm is an insertion sort. Suggestions to complete the shortcomings in this research are the testing process can be developed by analyzing efficient memory sorting of descending numbers and sorting testing on various letters of the alphabet.

REFERENCES

Badrul, M. (2014). Perbandingan Algoritma C4.5 Dan Neural Network Untuk Memprediksi Hasil Pemilu Legislatif DKI Jakarta. *Juurnal Pilar Nusa Mandiri*, *10*(2), 127–138. https://doi.org/10.33480/pilar.v10i2.470

Iskandar, I. D. (2019). Parents' Sum of Salaries Analyses towards School Tuition Fee Arrears Potential with Decision Tree Method. *Indonesian Journal of Information Systems*, *2*(1), 45. https://doi.org/10.24002/ijis.v2i1.2168

Iskandar, I. D., Amirulloh, I., Pertiwi, M. W., Kusmira, M., Hikmah, A. B., & Supriadi, D. (2020). *Laporan Akhir Penelitian Mandiri: Analisis Algoritma Bubble Sort Dan Insertion Sort Terhadap Efisiensi Memori Menggunakan Pendekatan Data Mining*.

Jiang, H., & Wang, R. (2016). Fault mode prediction based on decision tree. *IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 1729–1733. https://doi.org/10.1109/IMCEC.2016.7867514

Rahayu, E. S., Wahono, R. S., & Supriyanto, C. (2015). Penerapan Metode Average Gain, Threshold Pruning dan Cost. *Journal of Intelligent Systems*, *1*(2), 91–97. http://www.journal.ilmukomputer.org/index.php?journal=jis&page=article&op=view&path%5B%5D=80

Saptadi, A. H., & Sari, D. W. (2012). Analisis Algoritma Insertion Sort, Merge Sort Dan Implementasinya Dalam Bahasa Pemrograman C++. *JURNAL INFOTEL* -

Informatika Telekomunikasi Elektronika, *4*(2), 10. https://doi.org/10.20895/infotel.v4i2.103

Suryani, D. (2013). Perbandingan Metode Bubble Sort dan Insertion Sort Terhadap Efisiensi Memori. *Jurnal Teknologi Informasi & Pendidikan*, *6*(1), 146–162.

Sutoyo, I. (2018). Implementasi Algoritma Decision Tree Untuk Klasifikasi Data Peserta Didik. *Jurnal Pilar Nusa Mandiri*, *14*(2), 217. https://doi.org/10.33480/pilar.v14i2.926

Tjaru, S. N. B. (2009). Kompleksitas Algoritma Pengurutan Selection Sort dan Insertion Sort. *Makalah IF2091 Strategi Algoritmik Tahun 2009*. http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2009-2010/Makalah0910/MakalahStrukdis0910-074.pdf

Vijay Kotu, & Deshpande, B. (2015). *Predictive Analytics and Data Mining: Concepts and Practice with Rapidminer*. Elsevier Inc. https://doi.org/10.1016/C2014-0-00329-2