Performance Analysis of Hashing Mathods on the Employment of App

Anton Yudhana[1], Abdul Fadlil[2], Eko Prianto[3]
[1,2]Departement of Electrical Engineering, Ahmad Dahlan University Indonesia
[3]Master of Informatics Engineering, Ahmad Dahlan University, Indonesia

Article Info

Article history:

Received Jan 30, 2018
Revised Apr 12, 2018
Accepted Apr 19, 2018

Keyword:

Collision
Hashing
Hashing function
Time

ABSTRACT

The administrative process carried out continuously produces large data. So the search process takes a long time. The search process by hashing methods can save time faster. Hashing is methods that directly access data in a table by making references to the key that hashing becomes the address in the table. The performance analysis of the hashing method is done by the number of 18 digit character values. The process of analysis is done on applications that have been implemented in the application. The algorithm of hashing method analyzed is progressive overflow (PO) and linear quotient (LQ). The main purpose of performance analysis of hashing method is to know how gig the performance of each method. The results analyzed showed the average value of collision with 15 keys in the analysis of 53.3% yield the same value, while 46.7% showed the linear quotient has better performance.

Corresponding Author:

Eko Prianto,
Master of Electrical Engineering,
Ahmad Dahlan University,
Jl. Prof. Soepomo, Janturan-55164, Yogyakarta, Indonesia.
Email: ekoprianto05@gmail.com

1. INTRODUCTION

Indonesia is an archipelago consisting of thousands of large and small islands with an area of about 2 million km2 and the fouth most populous population in the world after china, India, and America. The growing occupation occupies more and more data.

The presence of Big Data in Indonesia increasingly popular dya almost in all circles. In terms of Government institutions, Big Data is a huge data set that will then be analyzed or processed again for specific purposes such as dicision making, prediction, and others. Because, by entering the digital age then, the digital pollution we call “data” will overwhelm in all the digital products that exist today. With a very large awareness then, the development of Big Data in Indonesia has increased. Because Big Data can be implemented in all types of fields in Indonesia.

Data searching is a process that exists and is needed in the construction of an application. Many algorithms are applicable, but not all algorithmts have good efficiency as long as the algorithm runs. The small data may not be the too Big difference, but now the development of data is also growing the so-called Big Data. Big Data is a very large set of data can consist of text, numeric or multimedia data stored in a particular database.

In the process of managing information technology database such as storing and searching data that tahes a lot of time and effort. Database design is needed to facilitate the management. Hundreds of thousands of data, take a long time. So a lot of searching using hashing methods to complete the research [1]-[4].

Search application interaction requires consistency of database structure wirh the search engine to search quicky, easily, and efficiently [5]. Surely with hashing algorithm can be very influential to improve search quality in a large number of database.

The hashing technique was introduced in the 60s [6]. Hashing is an externally resettable data addressing technique [7]. This technique allows us to insert, delete, and search records based an key search values. When applied correctly, this operation can be done in a constant time. In fact, properly-tuned hash systems usually only see one or two records for earch search, insert, or delete operation. This is much beter than the average cost of 0 (log n) required to perform a binary search on a sequenced n record vhain, or the average cost n (log n) needed to perform operations on a binary search tree. However, while hashing is based on a very simple idea, it is very difficult to implements correctly. The designer should pay attention to all the details associated with implementing the hash system.

The search process requires a key, making it faster and more accurate. The key used is ID Employees in Indonesia. The key consists of 18 numbers of units. In Indonesia ID Employee is a symbol of employee data. So from the ID Employees can immediately recognize who the owner ID Employees quickly. The above problems can be solved by a good staffing administration plan. The authors schose the method of progressive overflow (PO) and linear quotient (LQ) hashing, because of each method will recult in the average value of the collision and the speed of the search process. Collision, ie the occurrence of collisions in the placement of key values on the index number of the same memory address table. At the average value of the collision does not use units and for the length of time used in the search process using the stopwatch function with unit of time (millisecond). Both methods were analyzed and determined the most effective in case studies of the use of memory management.

The data used amounted to 2000 records to determine the performance of both methods. If the 2000 record data has not produced the performance of the two methods, then the data will be added until the method looks good performance. So the future can be a reference in the process of searching data with a large amount. The author conducted a study aimed at the implementation og memory management, preventing the same data repetition, good database management, and provide easy access to data. The analysis process by entering predetermined keywords can search the data and display with the required accuravy of time quickly.

2. BASIC THEORY

2.1. Hashing

The search process is used for various activities performed both online and ofline, many algorithms can be used to perform the search process which is a hashing algorithm [8]. Hashing is a widely used technique for accessing files on data storage. Hashing can be found in applications in other fields such as fuzzy matching, error checking, authentication, cryptography, and networking. The hashing technique has found the application to provide faster access to the routing table, with an increase in the size of the routing table [6].

2.2. Hash function

The hash function is the mapping of the function of integers in $\{0, 1, \ldots, m - 1\}$ to integers in $\{0, 1, \ldots, m' - 1\}$ with $m' < m$. when applying the hash function to n two integers can be mapped to the same value. This is called a collision. The perfect hash function at n integers is a hash function that has no collisions for n integers [9].

The value returned from the hash function is called the hash code. It is known hash algorithm works in one way and can not be reversed. A one-way hash algorithm is designed using two steps. First, convert the input data into the matrix system by using all the conversion needed to generate the initial hash value. Second, use the output from the first step to create a summary of the data and ultimately generate a safe hash value [10]. A good hash function must meet three properties, namely preimage resistance, second-order preimage, and collision resistance. The collision resistance means it is difficult to compute get two different inputs that have the same hash value [11].

The main idea of the hash value function approach, h(k), as the array bucket index, U, is the storage of the k key. This means that lock storage (k) in bucket U [h (k)] is a hash function to reduce the index array handling. The division method (mod function $h(k) = \text{mod } m$) is used to create a hash function h(k) in the hash table [12].

2.3. Progressive overflow

Linear Probing or Progressive Overflow is a classic implementation of the hash table. It uses the hash h function to map a set of n keys into an m size array [13]. In the Po method, if the kay is crashd then the kay value wil be placed on the next index that is still empaty. The next location search is done from the beginning of the table and forms a circular index structure [14]. Linear Probing using hashing function $h(k,i) = (h'(k) + i) \bmod m$ for $i=0, 1, \ldots, m-1$. This is a major difficulty of grouping even though it is the empaty slot state if the full slot is done the next probability with $(i + 1) / m$ [15].

2.4. Linear quotient

Linear Quotient or Double Hashing is a technique used in hash table to solve collisions when two values produce two identical keys [16]. Double hashing becomes another alternative to predict frequently-defined items in a large number of datasets. While the square is to investigate the loss of primary clustering problems, thus limiting the number of keys that can be placed on the full table. Double hashing is another method for geerating probling sequences [17].

The second hashing function is a step in the event of multiple collisions the idea is if two hash values to the same calculated from the initial value using the second hashing function. So it can be used to change the order of locations in the table, but still, have access to the entire table [15].

This technique uses the second idea to apply the hash function h' (key) to the key in the event of a collision. The result of the second hash function is to enter the number of positions from the collision point. There are several requirements for the second function:

a. Do not have to evaluate to zero
b. Must make sure that all can be investigated

The probing sequence is then calculated as follows [18]: Hi (x) = (h (x) + i h' (x)) mod m. where h(x) is the original function h'(x) the second function, i the number of collisions and m size of the table.

Figure 1 assumes where each R record is uniquely identified by the key K. in addition, K note R contains some useful information not specified in the INFO field. Steps to organize notes so you can quikly find records the table collection. All retrieval requests and updates are specified exclusively with the recording key. An easy way to implement an organization is to keep records in a table. The table entry is empty or contains one of the steps. Search for records with a given key by examining all table entries in depth. Similarly, a new record can be inserted into the table by searching for empty positions [19].

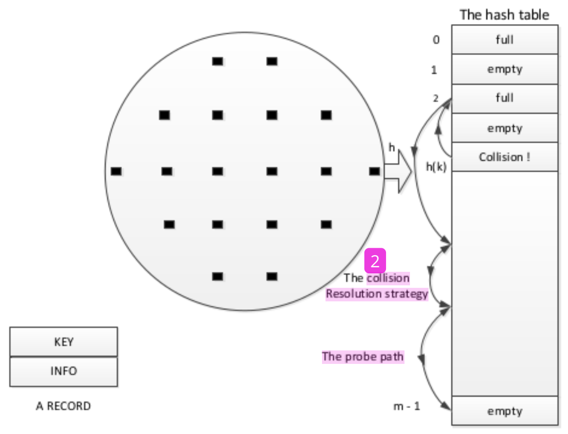

Figure 1. H hash function as a mapping [19]

Figure 1 of hashing using the transformation on key K gives ti the place where the record containing the K key is located. Suppose that the table has entries or positions, numbers 0, 1, …, m-1. Then m maps all keys, which are assumed to be very large, into the set (0, 1, …, m-1) by calling the function hash h, and describes it as a mapping. If j (K) = s, it can be said that the key K hash to position s. sure, soe keys may hash to the same position. So if you try to insert a new key K into the table, it may happen that the table h (K) entry is already occupied by another key. In this case, we need a mechanism to investigate the rest of the table until there is an empaty entry.

3. RESEARCH METHODLOGY

3.1. Key experiment

2000 data record data already in the input, 15 data taken randomly to test, among other: 196104051983041003, 196405151983121003, 197106041993071002, 195704161981031004, 197007091999031005, 198012302009022005, 198104032006042012, 196411301992061001, 198206012006042010, 198208282011012005, 198303042009022005, 198307232009022005, 196201031984031016, 195701011979031012.

Furthermore, the key is done an analysis of the average value of collision and the length of time required to access the data. Collision analysis process is done by manual calculation and application algorithm analysis result in application. Whie the calculation of the time required in a single search, using the stopwatch function that has been implemented in the application system.

3.2. Methodology

The flow of research through the stages that start from the identification, input, process, and output. The flow diagram in this study can be seen in Figure 2.

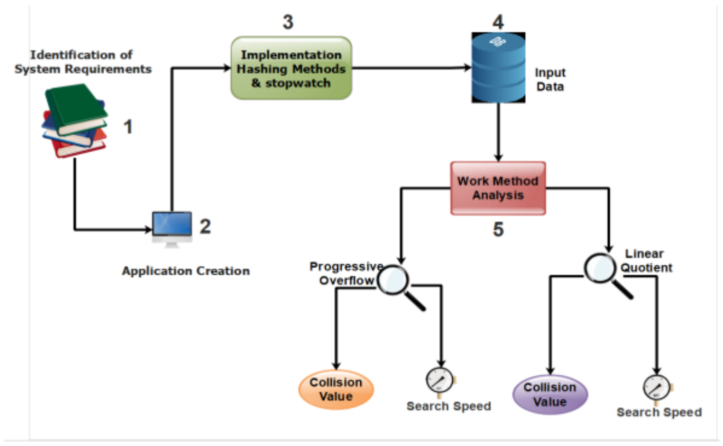

Figure 2. Diagram of the research flow

Figure 2 diagram of the research flow describes several stages as follows:

a. In the identified stage the problem related to the research is like determining what method is appropriate for use in the case study. Needs data and how much will be used in the application. Next, determine the system needs to be run on the application, so it can run well as needed.
   Figure 3 describes the flow of the analysis process, where the officer from the login by entering the username and password. After that officers can input employee data until completion and continued with system testing. Testing is done by 2 ways, namely by determining the average value of collision and hashing method analysis with the first hashing method is implemented into the application which later with the method will be seen the accuracy of time. The last stage is the output in the form of a report in which this report will be made every table of employee data.

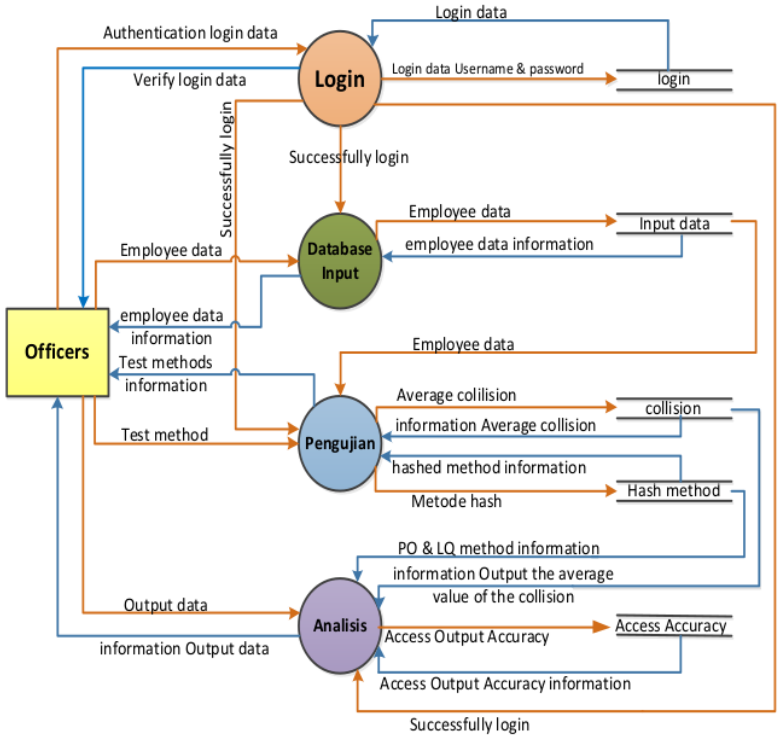

Figure 3. DAD Level 1 process of hashing method

b. Is making application such as making GUI according to user needs, so that the will not have trouble in understanding. After the creation of the application is complete then continue the creation of a database in accordance with the existing system as shown in Figure 4.

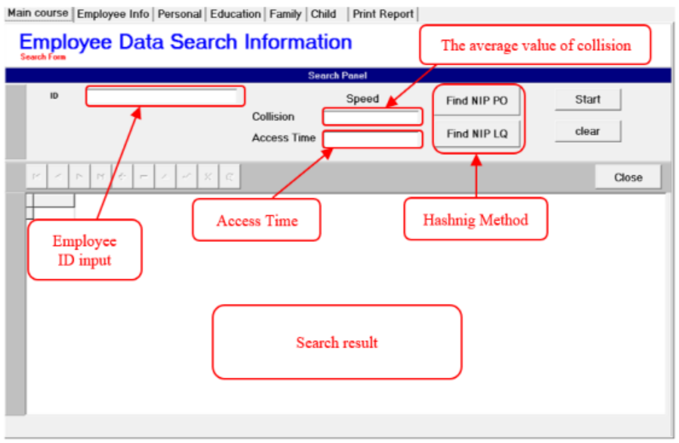

Figure 4. GUI method analysis

c. Hashing process implementation process, applied to the main home to facilitate the process of performance analysis methods. The process of implementing the algorithm is placed on the button of each method and stopwatch function.
d. The data input process is devided into two parts, the first database that the number of 1000 records the second amounted to 2000 records.
e. In the analysis, the phase is done by determining the key taken at random, after it determined the hashing method to be analysed first. Each method yields two values, the first being the average value of the collision and the time value in searching the data.

The last stage to make the results of analysis of each method in the form of tables comparison of the average value of collision, the length of time in the search process for comparison using 1000 record data and 2000 records so easy to analyze.

4. RESULTS AND DISCUSSION

4.1. Algorithm hashing method

The process of collision analysis is done by calculation of algorithm and manual system analysis. Calculation of system analysis is implemented in the application. In the application system yields the average value of the collision and the second time accuracy of the hashing method algorithm. As the comparative material of the analysis on the application below is a collision algorithm and manual calculation of the average value of collision.

```
i1:=StrToInt(Copy(B, 1, 2));
i2:=StrToInt(Copy(B, 3, 2));
i3:=StrToInt(Copy(B, 5, 2));
i4:=StrToInt(Copy(B, 7, 2));
i5:=StrToInt(Copy(B, 9, 2));
i6:=StrToInt(Copy(B, 11, 2));
i7:=StrToInt(Copy(B, 13, 2));
i8:=StrToInt(Copy(B, 15, 2));
i9:=StrToInt(Copy(B, 17, 2));
```

Source code 1. Prime number

Source code 1 explains to determine primes. In this study, the key used 18 digits, where the lift is grouped into 9 group. The process of reading the number starts with the keys and the numbers 1 to 2 and so on.

```
i1:=i1 mod 11;
i2:=i2 mod 11;
i3:=i3 mod 11;
i4:=i4 mod 11;
i5:=i5 mod 11;
i6:=i6 mod 11;
i7:=i7 mod 11;
i8:=i8 mod 11;
i9:=i9 mod 11;
```

Source code 2. Function H1

Source code 2 shows the function H1 is the first numbers in the prim modulus to determine the address of the index. The next step reads 3 and 4 numbers to determine the address of the index, and so on.

```
for i:=1 to 9 do
begin
if i2=i1 then
  begin
    i2:=i2+1;
    c1:=c1+1;
end;
```

Source code 3. Address placement of collision case index

In source code 3 is the settlement of the PO and LQ method the solution is done by adding 1, to get the blank index address. If at the address of the index is still going collision, then step is done until 9. i2 is a variable of number 3 and 4 of the 18 number digits, as c1 the variable of the collision.

```
b1:=Trunc(a1/11);
b2:=Trunc(a2/11);
b3:=Trunc(a3/11);
b4:=trunc(a4/11);
b5:=Trunc(a5/11);
b6:=Trunc(a6/11);
b7:=Trunc(a7/11);
b8:=Trunc(a8/11);
b9:=Trunc(a9/11);
```

Source code 4. Rounding modulus primes on LQ

Source code 4 describes the function of H2 on the LQ method. b1 is a variable for determining trunc results (rounding results). Next b1 adds a1 (variable from i1) and in modulus, returns to get the empty index address. The step is done up to b9.

```
if i1>10 then
i1:=0;
if i2>10 then
i2:=0;
if i3>10 then
i3:=0;
if i4>10 then
i4:=0;
if i5>10 then
i5:=0;
if i6>10 then
i6:=0;
if i7>10 then
i7:=0;
if i8>10 then
i8:=0;
if i9>10 then
i9:=0;
c9:=c1+c2+c3+c4+c5+c6+c7+c8;
c9:=c9/9;
```

Source code 5. The recurrence function to return to i1 if it has reached i10

Source code 5 is a step from the PO dan LQ methods. The process is done if the calculation of the collision has reached i9 still not get the index address then return i1. At the address index if it has reached index 10 feed back to index 0, until the process of the last two numbers of numbers 17 and 18. The next step is c1 to c8 summed divided by c9 to get the average value of collision. C1 is a variable of the settlement of two-digit numbers (numbers 3 and 4) in case of collisions. C9 is the number of groups of the cultivators, namely 9 groups.

4.2. Collision analysis

Experiment PO
If known key value as follows:
19 61 04 05 19 83 04 10 03
The placement of these key values with the progressive overflow method is as follow.
Resolution:
N = 9, P = 11, Address index= 0 to 10
Calculation:
H (19) = 19 mod 11 => 8
H (61) = 61 mod 11 => 6
H (04) = 04 mod 11 => 4
H (05) = 05 mod 11 => 5
H (19) = 19 mod 11 => 8 (collision)
    Placed on the next empty location: 9
H (83) = 83 mod 11 => 6 (collision)
    Placed on the next empty location: 7
H (04) = 04 mod 11 => 4 (collision)
    Placed on the next location: 5 (collision)
    Placed on the next location: 6 (collision)
    Placed on the next location: 7 (collision)
    Placed on the next location: 8 (collision)
    Placed on the next location: 9 (collision)

Placed on the next empty location: 10
H (10) = 10 mod 11 => 10 (collision)
Placed on the next empty location: 0
H (03) = 03 mod 11 => 3
The results of these calculations are shown in figure 5. The average for accessing a key value is:
Average collision = (1 + 1 + 6 + 1) / 9 = 1

Experiment LQ
If known key value as follows:
19 61 04 05 19 83 04 10 03
So the placement of these key values with the linear quotient method is as follows.
Resolution:
N = 9, P = 11, Address index = 0 to 10
Calculation:
H (19) = 19 mod => 8
H (61) = 61 mod => 6
H (04) = 04 mod => 4
H (05) = 05 mod => 5
H (19) = 19 mod => 8 (collision)
Placed on a new location
Increment => (19 div 11) mod 11 => 1
New location => (8 + 1) mod 11 => 9
H (83) = 83 mod 11 = 6 (collision)
Placed on a new location
Increment => (19 div 11) mod 11 => 7
New location => (6 + 7) mod 11 => 2
H (04) = 04 mod 11 => 4 (collision)
Placed on a new location
Increment => (04 div 11) mod 11 => 0 set => 1
New location => (4 + 1) mod => 5 (collision)
Placed on a new location
Increment => 1
New location => (5 + 1) mod => 6 (collision)
Placed on a new location
Increment => 1
New location => (6 + 1) mod => 7
H (10) = 10 mod 11 => 10
H (03) = 03 mod 11 => 3
The calculation results are shown in figure 5. Average for access to a value keys are:
Average collision = (1 + 1 + 3) / 9 = 0.5556

4.3. Analysis results

The author presents the results of previous research. This process is done as a comparative material from the findings of the author. Research [18] has two main methods on the collision resolution of hash tables that are chaining (close addressing) and open addressing. The study considers on the performance of open deleivery techniques of collision resolution, is Linear probing, Quadratic probing and double hashing. The algorithm is implemented in C^{++}.

The case study considers entering the student registration number (alphanumeric data type) in the hash table implemented using the C^{++} programming language, to monitor performance as shown in the Table 1. This study only records the number of probes required to complete the collisions occurring each time an insertion.

Table 1. Result of Number of Probes by each Algorithm in Sample Data

REGISTRATION NUMBER	Linear Probing (probes)	Quadratic Probing (probes)	Double Hashing (probes)
KUST / SCI / 05 / 356	0	0	0
KUST / SCI / 05 / 214	4	2	2
KUST / SCI / 05 / 117	0	0	0
KUST / SCI / 05 / 714	0	0	0
KUST / SCI / 05 / 735	1	1	3

REGISTRATION NUMBER	Linear Probing (probes)	Quadratic Probing (probes)	Double Hashing (probes)
KUST / SCI / 05 / 821	0	0	0
KUST / SCI / 05 / 434	2	3	0
KUST / SCI / 05 / 578	1	1	0

In this research, the difference of search process is to analyze the average value of collision and time. Current research algorithms are implemented in Delphi. The search process in performed to view the performance of the method by using the stopwatch function. Data used 1000 records and 2000 records for comparison of search duration on a large amount of data.

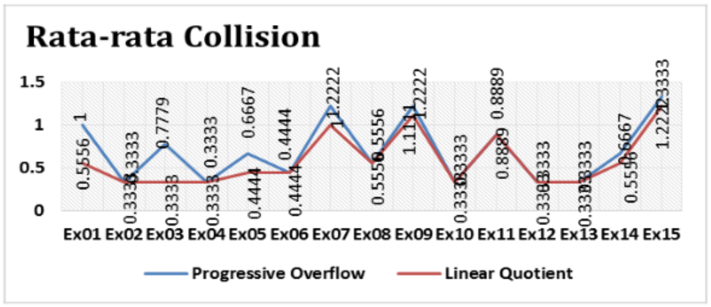

Figure 5. Graph comparison the mean values of PO and LQ collision methods

Figure 5 illustrates the test of 15 key values of each method, there are several differences in the values of the collision in Ex01, Ex03, Ex05, Ex07, Ex09, Ex14, and Ex15. There are some of the same ie Ex02, Ex04, Ex06, Ex08, Ex10, Ex11, Ex12 and Ex13. Testing is done t determine the level of error when determining the address of the index. From the graph can be seen the lowest collision average value of 0.3333 obtained from the testing of PO and LQ methods, while the highest value of 1.3333 obtained from testing the PO method.

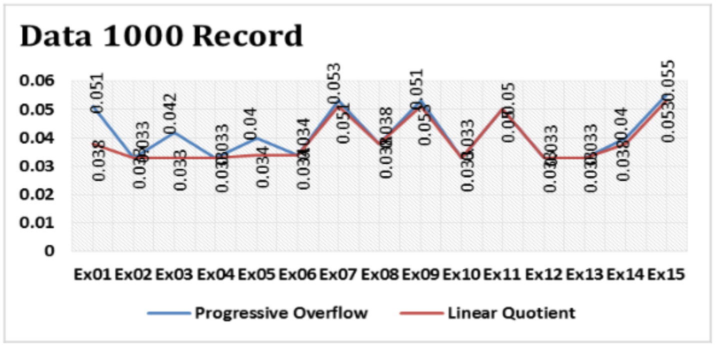

Figure 6. The time comparison graph of search speed using 1000 record data

In Figure 6 illustrates from the test of 15 key values of each method. the lowest search speed og 0.33 millisecond is found in Ex02, Ex03, Ex04, Ex10, Ex12 and Ex13 from the PO and LQ methods, whereas the highest search rate of 0.55 millisecond is found in Ex15 from PO mode. If we associate between Figure 5 the average collision graph and Figure 6 of the search speed graph we can conclude that the greater the average value of the collision, the longer it takes to search for a data. Vice versa, the smaller the value of the average collison, the less time it takes to search the data.

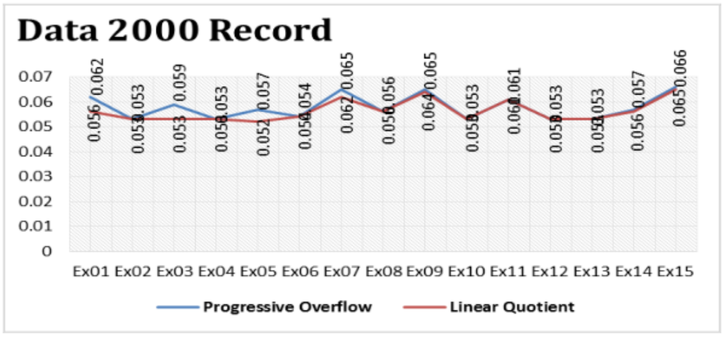

Figure 7. Graph of time comparison of search speed using 2000 record data

Figure 7 illustrates that from the test of 15 key values using 2000 records of each method, there is a difference in the values of data search speed. The lowest values of the search speed of 0.52 millisecond can be seen in Ex05 on the LQ method, whereas the highest values of 0.066 milliseconds is the test of Ex15 on the PO method. Between Figure 6 1000 records and Figure 7 2000 records can be concluded that the number of records has an influence in testing the speed of data search. The large the number of records in a database, the longer it takes to search the data.

5. CONCLUSION

The research that has been carried out to produce the conclusion that is:

a. The 15 tests conducted by taking the Employee ID as the keyword to analyse 8 keys that the average value of collision yields the same value and 7 different. The magnitude of the average value of collision is influenced by the number of times the number of collisions at the time of determining the address of the index. The test results show that linear quotient method is better than the progressive overflow method.
b. The testing process to determine how long it takes to perfom a data search 1000 records and 2000 records on PO and LQ methods. From the test that has been done shows the difference between search time between 1000 and 2000 records on keys number 2, 4, 6, 10, 11, 12, and 13 that is 0.02 millisecond, whie key number 8 is difference 0.018 millisecond. The rest of the key numbers 1, 3, 5, 9, 14 and 15 have different search times, both between the method and the amount of data used, it is influenced from the algorithm rules LQ method. So it can be concluded that hashing method can run well on desktop applications, and become a reference for the development of the next method and certainly can be a reference.

ACKNOWLEDGEMENTS

I am grateful to Ahmad Dahlan University Yogyakarta for giving me the opportunity to do research.

REFERENCES

[1] L. Jianwei and C. Huijie, "A Dynamic Hashing Algorithm Suitable for Embedded System", *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, vol. 11, no. 6, pp. 3220-3227, 2013.
[2] Y. Huang, Q. Zhang, and Z. Yuan, "Perceptual Speech Hashing Authentication Algorithm Based on Linear Prediction Analysis", *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, vol. 12, no. 4, pp. 3214-3223, 2014.
[3] G. Indrawan, *et al*., "A Multi-Threaded Fingerprint Direct-Access Strategy Using Local-Star-Structure-based Discriminator Features", *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, vol. 12, no. 5, pp. 4079-4090, 2014.
[4] N. Qiu, *et al*., "Research on Optimization Strategy to Data Clustered Storage of Consistent Hashing Algorithm", *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, vol. 14, no. 3, pp. 824-830, 2016.

[5] K. Kowsari *et al*., "Construction of FuzzyFind Dictionary using Golay Coding Transformation for Searching Applications", *Int. J. Adv. Comput. Sci. Appl*., vol. 6, no. 3, pp. 81-87, 2015.
[6] G. M. Sridevi and M. V Ramakrishna, "A Survey of Hashing Techniques for High Performance Computing", *Int. J. Recent Innov. Trends Comuting Comun*., vol. 4, no. 6, pp. 619-623, 2016.
[7] B. W. A. Martini, *et al*., "Double Hashing with Multiple Passbits", *Int. J. Found. Comput. Sci*., vol. 14, no. 6, pp. 1165-1182, 2003.
[8] R. Rahim, *et al*., "Double Hashing Technique in Closed Hashing Search Process", *IOP Conf. Ser. Mater. Sci. Eng*., vol. 237, p. 12027, 2017.
[9] Y. Han, "Construct a Perfect Word hash Function in Time Independent of the Size of Integers", *Inf. Process. Lett*., vol. 128, no. July, pp. 5-10, 2017.
[10] M. Abutaha, "New One Way Hash Algorithm Using Non- Invertible Matrix", in *Computer Medical Applications (ICCMA)*, 2013, pp. 1-5.
[11] F. Hariyanto and B. H. Susanti, "Collision Attack Against Tav-128 Hash Function", *J. Phys. Conf. Ser*., vol. 893, no. 1, 2017.
[12] N. Negm and A. B. M. Salem, "Clustering Web Documents based on Efficient Multi- Tire Hashing Algorithm for Mining Frequent Termsets", *Int. J. Adv. Res. Artif. Intell*., vol. 2, no. 6, pp. 6-14, 2013.
[13] M. Thorup, "Fast and Powerful Hashing using Tabulation", *Commun. ACM*, vol. 60, no. 7, pp. 94-101, 2017.
[14] P. Reviriego, *et al*., "Efficient Implementation of Error Correction Codes in Hash Tables", *Int. J. Innov. Res. Comput. Commun. Eng*., vol. 2, no. 12, pp. 338-340, 2014.
[15] P. Nimbe, *et al*., "An Efficient Strategy for Collision Resolution in Hash Tables", vol. 9, no. 2, pp. 258-267, 2014.
[16] J. Jayalakshmi, *et al*., "Frequent Itemset Generation using Double Hashing Technique", *Procedia Eng*., vol. 38, pp. 1467-1478, 2012.
[17] D. R. Vadiyu and Senthil P, "Image Content with Double Hashing Techniques", *Int. J. Adv. Trends Comput. Sci. Eng*., vol. 1, no. 3, pp. 98–101, 2012.
[18] S. A. Bello, *et al*., "Comparative Analysis of Linear Probing, Quadratic Probing and Double Hashing Techniques for Resolving Collusion in a Hash Table", *Int. J. Sci. Eng. Res*., vol. 5, no. 4, pp. 685-687, 2014.
[19] L. J. Guibas and E. Szemeredi, "The Analysis of Double Hashing", *J. Comput. Syst. Sci*., vol. 16, no. 2, pp. 226-274, 1978.

21. HASIL CEK_60960140

ORIGINALITY REPORT

7%	4%	3%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	garuda.kemdikbud.go.id Internet Source	4%
2	Leo J. Guibas, Endre Szemeredi. "The analysis of double hashing", Journal of Computer and System Sciences, 1978 Publication	3%

Exclude quotes	On	Exclude matches	< 3%
Exclude bibliography	On		