COMPUTATIONAL SCIENCE AND TECHNIQUES
Volume 7, 2020, 605-615
© Klaipėda University, 2020
Publisher: Klaipėda University
http://journals.ku.lt/index.php/CST
Online ISSN: 2029-9966
DOI: 10.15181/csat.v7i1.2091

EFFICIENCY ANALYSIS OF BINARY SEARCH AND QUADRATIC SEARCH IN BIG AND SMALL DATA

Kevin Hendy, Wirawan Istiono
Universitas Multimedia Nusantara
wirawan.istiono@umn.ac.id

Abstract. When doing a searching process, Binary Search is one of the classic algorithm used in sorted data. The characteristic of this algorithm is to make a comparison of the keywords you want to find with the start, middle, and end values of a data series. Keyword search is done by reducing the range of start and end points to finally find the keyword you want to search. The time complexity of the binary search algorithm is $O(\log_2 n)$ while the memory capacity needed is O(1) for iterative implementation and $O(\log_2 n)$ for recursive implementation. This research will develop a comparison level in binary search with quadratic search algorithm in order to get optimal performance according to using small amounts of data and big data. Based on the results of research conducted using 6 different amounts of data, where each algorithm from 6 cases was repeated five times to get the average execution time, and the results obtained that Quadratic Search get faster time than Binary Search in the amount of data under 100,000 data. But after using more than 100,000 data, Quadratic Search takes longer than Binary Seach to find the data. In addition to time efficiency.

Key words: Binary Search, Performance Analysis, Binary Search Rankings

1. Introduction

Searching is a common process carried out by many people in various circles. Searching is an activity carried out to search for objects, people, or even data that will be used to carry out another activity. Searching can be easily done if the available data is in small amounts. If there is a large amount of data available, the search process will become more complicated and need more time and effort to obtain the data.

Responding the need for large amounts of data searching with the fastest possible time, Searching algorithms was created, one of those algorithms is the Binary Search. Binary Search is a searching algorithm that uses start, middle, and end points, and the data that will be used must be sorted (ascending or descending). The time efficiency that generated using the Binary Search algorithm is dependent on the amount of data that will be used, and data position that you want to find. The more dataset, it means the more iterations that will be performed to find the data which makes it more inefficient to use this algorithm. Therefore, the Binary Search algorithm was developed which originally had 3 critical points (start, middle, end) into Quadratic Search which has 5 critical points.

2. Searching Algorithms

Searching is a process to find out the exact location of a data. Many algorithms are used to facilitate the searching process, such as the linear search method, the binary search method, and the interpolation search method. Each search algorithms has an implementation method and the requirements that must be met before searching data, and each algorithm has different efficiency or time complexity. For using this search method, it depends on how the data structure is used, the amount of data owned, and also the type of data that will be searched (Mehta, 2015).

3. Binary Search

Binary Search is a fast and efficient searching technique that requires sorted data either ascending or descending (w3schools, 2019). In data search, Binary Search uses the midpoint as a critical point to compare the value you want to find with the value at the current midpoint. Where the midpoint value is obtained from the addition of the start point and the end point then divided by two. If the value being searched is different from the value at the midpoint, there will be a displacement at 2 other critical points in one iteration (Winarno, 2018).

If the value that being searched is less than the midpoint value, then the endpoint value will be shift into the midpoint value minus one. If the value that being searched is greater than the midpoint value, then the start point value will be shift into the midpoint value plus one. Shifting these two points (midpoint with start point or midpoint with ending point) will be cause the data that needs to be searched need to have half the range of the initial search range.

4. Binary Search Algorithm

Steps used in binary search:

- Sorting the data set that will be searched (either ascending or descending sort).
- Initiate the start point value (L) with 0, and the end point value (R) with the length of data minus one.
- Do the iteration as long as the start point (L) is smaller than the end point (R). If the start point exceeds the end point, stop the iteration and the data being searched is not found.
- If the midpoint value is the same as the value being searched, then return the midpoint position value as the answer.
- If the midpoint value is smaller than the value being searched, then change the startpoint value to the midpoint value plus one.
- If the midpoint value is greater than the value being searched, then change the endpoint value to the midpoint value minus one (Balogun, 2013).

The picture below that shown in Fig1 is a flowchart of the Binary Search algorithm (cssimplified, 2019):

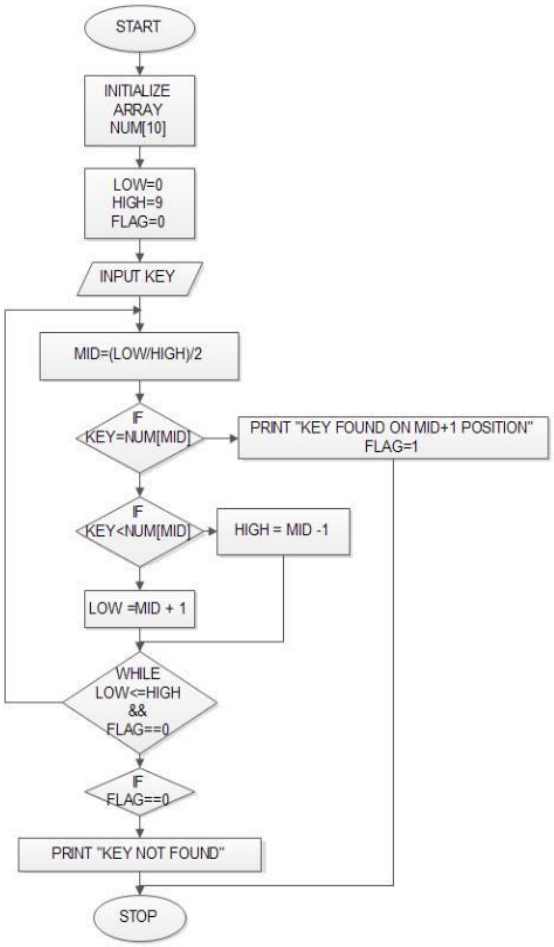

Fig. 1 **Binary Search Flowchart**

5. Advantages and Disadvantages of Binary Search

Each algorithm has advantages and disadvantages. In this section, that will be describe the advantages and disadvantages of Binary Search when compared to other searching algorithms based on related research (Pandey, 2014; Kumar, 2013):

Advantages:

- Searching big data using Binary Search is faster than using Linear Search.
- This search technique can be used in data in the form of a tree (Binary Search Tree).

Disadvantages:

- Binary Search is not suitable for use in a linked list data structure because it cannot access data at the midpoint of the linked list.
- Not suitable for data that can be modified by the user such as adding data or deleting data.

6. Quadratic Search

Quadratic Search is an algorithm that also uses a critical point as in Binary Search. The difference that can be seen from the Quadratic Search algorithm compared to Binary Search

is that in Binary Search there is only three critical points in the middle of the array. Whereas in Quadratic Search, there are five critical points in the section of start, ¼ or quarter point, ½ or mid-point, ¾ or third quarter point, and the end point. Each value at the five critical points is obtained from the formula as follows (Painthankar, 2017):

```
Mid = (Hi+Low)/2;
Quarter = Low + (Hi-Low)/4;
ThirdQuarter = Low + (Hi-Low) * 3/4;
```

After the values at the five critical points are obtained, if the data being searched is not found in one of the five critical points, so there are four conditions that can be done to find the data being searched. The four conditions are as follows:

- First Condition:

```
if (Data < array(Quarter) && Data < array(Mid)) {
    Hi = Quarter-1;
}
```

In this first condition, if the value of the data being searched is smaller than the value of ¼ array and value of ½ array, then the largest position (Hi) of the data being searched will be reduced in the range to ¼ *array* minus one.

- Second Condition:

```
if (Data > array(Quarter) && Data < array(Mid)) {
    Low = Quarter+1;
    Hi = Mid-1
}
```

In the second condition, if the data value being searched is between the value of ¼ array and the value of ½ array, then the smallest position (Low) of the data being searched will be ¼ array plus 1, and the largest position (Hi) of the data being searched will be ½ array minus one.

- Third Condition:

```
if (Data > array(Mid) && Data < array(ThirdQuarter)) {
    Low = Mid+1;
    Hi = Quarter-1;
}
```

In the third condition, if the data value being searched is between the ½ array value and the ¾ array value, then the smallest position (Low) of the data being searched will be ½ array plus 1, and the largest position (Hi) of the data being searched will be ¾ array plus 1.

- Fourth Condition:

```
if (Data > array(Mid) && Data > array(ThirdQuarter)) {
      Low = ThirdQuarter+1;
}
```

In the last condition, if the data value being searched is greater than the ½ array value and the ¾ array value, then the smallest position (Low) of the data being searched will be ¾ array plus 1.

The all four conditions will continue to be repeated as long as the data being searched has not been found at one of the five critical points, or as long as the smallest position (Low) is not greater than the largest position (Hi). The code below is sample code of Quadratic Search

```
while(Low <= Hi) {
    Mid = (Hi+Low)/2;
    Quarter = Low + (Hi-Low)/4;
    ThirdQuarter = Low + (Hi-Low) * 3/4;

    if(Data == array(Mid) || Data == array(Quarter) || Data == array(ThirdQuarter)){
        printf("Data Found");
        break;
    }
    if (Data < array(Quarter) && Data < array(Mid)) {
        Hi = Quarter-1;
    }
    if (Data > array(Quarter) && Data < array(Mid)) {
        Low = Quarter+1;
        Hi = Mid-1;
    }
    if (Data > array(Mid) && Data < array(ThirdQuarter)) {
        Low = Mid+1;
        Hi = Quarter-1;
    }
    if (Data > array(Mid) && Data > array(ThirdQuarter)) {
        Low = ThirdQuarter+1;
    }
}
```

7. Quadratic Search and Binary Search Analysis

For Binary serach, each iteration, the Binary Search algorithm divides the array that contains or does not contain the data being searched into two parts. In Fig2 shown chart overview of binary search

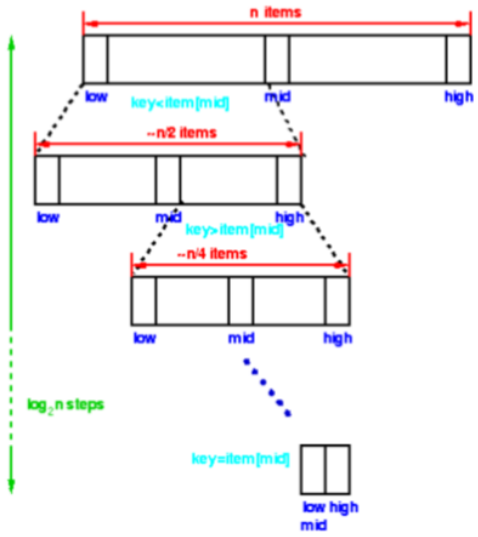

Fig. 2 **Chart Overview of Binary Search**

For Quadratic Search iteration, the algorithm divides the array that have contains or does not have contain the data that being searched into four parts. This division, divided into four parts to makes the time find the data half times faster than the time required on the Binary Search algorithm. In Fig4. Shown chart overview of quadratic search.

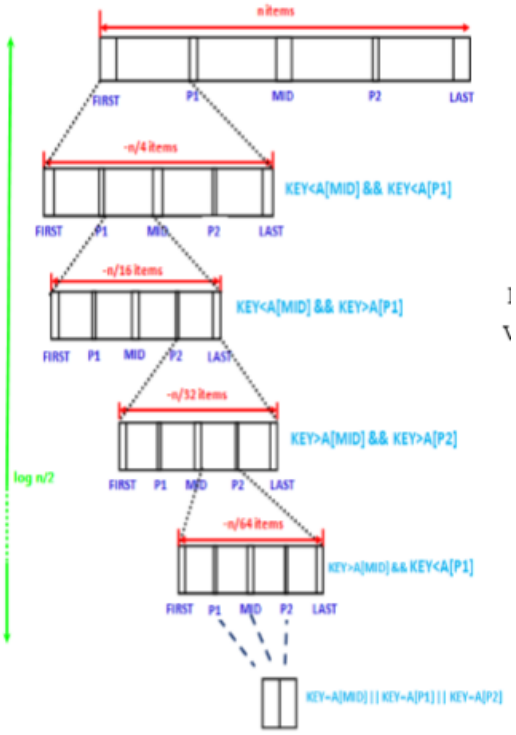

Fig. 3 **Chart Overview of Quadratic Search**

8. Methodology

In this research uses Vector as a method of storing small data and big data to see the difference in time efficiency produced by Quadratic Search and Binary Search. Vector is a dynamic array that can be change the size of the array automatically if there is an addition or deletion of an element in the array (Soulie, 2007; Kusnadi, 2017). Vector can be used in the C++ programming language that provides the Standard Template Library (STL).

Standard Template Library (STL) is a set of pre-programmed data structures and algorithms (Savitch, 2002). This Standard Template Library was created with the aim of facilitating programmers with the most commonly used collections of functions and data structures (Seed, 2001). In other words, the use of STL can improve time efficiency and also the efficiency of writing code in programming. In this study, the Standard Template Library from the C ++ programming language:

- **vector< >**, used to create dynamic arrays that can adjust their own size when there are elements that are being added or removed.
- **push_back( )**, used to add new elements to the dynamic array at the very back / last position.
- **sort( )**, used to sort elements in an array in ascending or descending order.
- **begin( )**, used to get the first element in a dynamic array.
- **end( )**, used to get the last element in a dynamic array.

9. Comparison of Binary Search and Quadratic Search Performance

To compare the performance provided by the Binary Search and Quadratic Search algorithms, we used several test cases with different amounts of data. In each experiment, we are using difference amount of data that will be test for six times to see the average time efficiency that generated from binary search and quadratic search, the result from each amount of data can be seen in Table1, Table2, Table3, Table4, Table5 and Table6.

Table1. Test Case1 with 10 amount of data

Amount Of Data	Data Being Searched	Number Of Iterations	
		Binary Search	Quadratic Search
10	7	4	1

Trial	Execution Time (second)	
	Binary Search	Quadratic Search
1	0.275	0.254
2	0.250	0.269
3	0.266	0.218
4	0.239	0.208
5	0.226	0.223
Average	0.2512	0.2336

Table2. Test Case2 with 100 amount of data

Amount Of Data	Data Being Searched	Number Of Iterations	
		Binary Search	Quadratic Search
100	7	7	3

Trial	Execution Time (second)	
	Binary Search	Quadratic Search
1	0.318	0.223
2	0.280	0.254
3	0.352	0.239
4	0.248	0.271
5	0.238	0.251
Average	0.2872	0.2476

Table3. Test Case3 with 1000 amount of data

Amount Of Data	Data Being Searched	Number Of Iterations	
		Binary Search	Quadratic Search
1000	7	10	4

Trial	Execution Time (second)	
	Binary Search	Quadratic Search
1	0.280	0.213
2	0.245	0.233
3	0.224	0.217
4	0.258	0.245
5	0.251	0.256
Average	0.2516	0.2328

Table4. Test Case4 with 10000 amount of data

Amount Of Data	Data Being Searched	Number Of Iterations	
		Binary Search	Quadratic Search
10000	7	14	7

Trial	Execution Time (second)	
	Binary Search	Quadratic Search
1	0.342	0.294
2	0.307	0.228
3	0.310	0.300
4	0.323	0.250
5	0.316	0.240
Average	0.3196	0.2624

Table5. Test Case1 with 100000 amount of data

Amount Of Data	Data Being Searched	Number Of Iterations	
		Binary Search	Quadratic Search
100000	7	17	9

Trial	Execution Time (second)	
	Binary Search	Quadratic Search
1	0.329	0.358
2	0.323	0.368
3	0.308	0.351
4	0.319	0.388
5	0.301	0.342
Average	0.316	0.3614

Table 6. Test Case6 with 999999 amount of data

Amount Of Data	Data Being Searched	Number Of Iterations	
		Binary Search	Quadratic Search
999999	7	15	8

Trial	Execution Time (second)	
	Binary Search	Quadratic Search
1	0.287	0.402
2	0.336	0.311
3	0.352	0.440
4	0.390	0.302
5	0.359	0.414
Average	0.3448	0.3738

10. Result and Discussion

Based on the results of research conducted using 6 different amounts of data, where each algorithm from 6 cases was repeated five times to get the average execution time, and the results obtained that Quadratic Search get faster time than Binary Search in the amount of data under 100,000 data. But after using more than 100,000 data, Quadratic Search takes longer than Binary Search to find the data. In addition to time efficiency, the number of iterations that needed by the Quadratic Search algorithm is smaller than Binary Search.

11. Conclusions and Suggestions

Based on the results of the study, it can be concluded that Quadratic Search is still more efficient to use than Binary Search. Even so, there are still many factors that can affect execution time besides the amount of data. Computer memory speed, RAM and memory capacity can also affect the execution time of both the Binary Search algorithm and the Quadratic Search algorithm.

For future research, is suggest that the research should also pay attention to the position factor of the data being searched. Data position at the beginning, middle, and end can produce different results. In addition, future research can also pay attention to the number of iterations performed to process a certain amount of data.

Acknowledgement

The authors are thankful to the Universitas Multimedia Nusantara, Indonesia which has become a place for researchers to develop presented research.

References

Mehta, A; Saxena, A; Patel, J.; Thanna, A; Review on comparison of binary search and linear search, International Journal of Engineering Sciences & Management Research, vol. 2, no. 10, pp. 85-89, 2015.

w3schools, Searching Techniques, [Online]. Available: https://www.w3schools.in/data-structures-tutorial/searching-techniques/. [Accessed 15 September 2019].

Balogun, B. G; Sadiku, J. S; Simulating Binary Search Technique Using Different Sorting Algorithms, International Journal of Applied Science and Technology, vol. 3, no. 6, pp. 67-75, 2013.

cssimplified, [Online]. Available: http://cssimplified.com/c-cpp-programming-data-structure/design-an-algorithm-draw-a-corresponding-flow-chart-and-write-a-c-program-for-binary-search-to-search-a-given-number-among-the-list-of-numbers-10m-dec2007. [Accessed 15 September 2019].

Kamlesh Kumar Pandey et al, A Comparison and Selection on Basic Type of Searching Algorithm in Data Structure, International Journal of Computer Science and Mobile Computing, Vol.3 Issue.7, July- 2014, pg. 751-758

Verma, D; Painthankar, D. K; Optimizing the Performance of Quadratic Search using Memorization, International Journal of Advanced Research in Computer Science , vol. 8, no. 3, pp. 481-484, 2017.

Kumar, P.; Quadratic Search: A New and Fast Searching Algorithm (An extension of classical Binary search strategy)," International Journal of Computer Applications, vol. 65, no. 14, pp. 43-46, 2013.

Soulie, J.; C++ Language Tutorial, cplusplus.com, 2007.

Savitch, W.; Absolute C++, Boston: Addison-Wesley, 2002.

Seed, G. M.; An Introduction to Object-Oriented Programming in C++ with Applications in Computer Graphics, Edinburgh: Springer, 2001.

Kusnadi, Adhi.; Perbandingan Algoritma Horspool dan Algoritma Zhu-Takaoka dalam Pencarian String Berbasis Desktop, Ultima Computing vol. 9, no. 1, pp. 12-16, 2017.

Winarno, Michael.; Design and Development of Computer Specification Recommendation System Based on User Budget With Genetic Algorithm, International Journal of New Media Technology, vol. 5, no. 1, pp 25-29, 2018

K. Hendy currently study at the Universitas Multimedia Nusantara, Tangerang Indonesia. Kevin does research in Algorithms and Data structure.

W. Istiono currently works at the Universitas Multimedia Nusantara, Tangerang Indonesia as lecturer and researcher. Wirawan does research in Algorithms, Computer Science and Information and applied computing.

DVEJETAINĖS IR KVADRATINĖS PAIEŠKOS EFEKTYVUMO ANALIZĖ DIDELIUOSE IR MAŽUOSE DUOMENYSE

Kevin Hendy, Wirawan Istiono

Santrauka

Dvejetainė paieška yra klasikinis algoritmas naudojamas paieškai surūšiuotuose duomenyse. Šio algoritmo veikimas paremtas norimų raktinių žodžių palyginimu su duomenų serijos pradžios, vidurio ir pabaigos reikšmėmis. Raktinių žodžių paieška atliekama sumažinant pradžios ir pabaigos taškų diapazoną, kad galiausiai būtų surasti norimi raktiniai žodžiai. Dvejetainės paieškos algoritmo laiko kompleksiškumas yra O(log2n) eilės, o reikalinga atminties talpa yra O(1) eilės, kai naudojamas iteratyvus algoritmas bei O(log2n), kai naudojamas rekursinis algoritmas. Straipsnyje pristatytas dvejetainės ir kvadratinės paieškos algoritmų palyginimas turint tikslą pasiekti optimalų veikimą mažuose ir dideliuose duomenų rinkiniuose. Tyrimas atliktas naudojant šešis skirtingos apimties duomenų rinkinius. Kiekvienas algoritmas su kiekvienu duomenų rinkinių buvo kartojamas penkis kartus, norint gauti vidutinį vykdymo laiką. Duomenų rinkiniuose mažesniuose nei 100 000 įrašų kvadratinė paieška pasirodė geriau, rezultatai gaunami greičiau nei taikant dvejetainę paiešką. Tačiau duomenų rinkiniuose didesniuose nei 100 000 įrašų laimi dvejetainė paieška.

Pagrindiniai žodžiai: dvejetainė paieška, veiklos analizė, dvejetainės paieškos reitingai