To what extent is the binary search algorithm suitable to find specified values from datasets of varying size?

Computer Science

Word Count: 3628

Candidate Number: xxxx

Supervisor: xxxx

EXAMPLE ESSAY – Received 34 points = A grade

Abstract

This investigation explores the how suitable the binary search algorithm is to find an integer from datasets of varying size, in order to determine its usability within larger data sets. The binary search algorithm is an algorithm which finds a certain value within a set of data that has already been sorted. To explore this, the concepts of big data and its problems are introduced, explaining why using a traditional method to manage data is not appropriate. Succeeding this, an experiment was carried out involving creating a program to test the efficiency of the algorithm where a number of pseudo-random integers were created and stored in array. The time taken to sort the array and find a randomly selected value from the array was measured. The integers were then stored in a text file in order to determine an estimate for the file size of these integers. A graph was plotted to measure the relationship between the size of the text file and time taken to sort the dataset and find the random integer. From this, an equation of the relationship was found and the data was extrapolated, to predict the time it would take to find a random integer within larger datasets. The relationship between the time taken and file size proved to be linear, so as file size increased, time taken also increased. For larger datasets, other methods of data management, such as the use of Apache Hadoop were explored. It was found that Apache Hadoop provided far greater functionality than utilising the simple algorithm in the management of large-scale data sets. The binary search algorithm was found to only be useful when managing small-scale sets of data that are structured in nature.

EXAMPLE ESSAY – Received 34 points = A grade

Contents

1 Introduction

The end of the 20th century marked the beginning of the digital era. The usage of computer systems skyrocketed, and to cope with this, data storage had to expand exponentially. The need for data management solutions grew rapidly, where in 2012, the total amount of digital information in the world was estimated hold more than 2.7 zettabytes of data[1]. Recently, the phenomenon of 'big data'[2] has taken large corporations by storm; involving problems associated with the management of large data sets. Managing this data can provide essential information for companies, acting as a predictor for the future. However, it is becoming increasingly difficult to manage large data sets, where both specialist software and hardware is required. For this reason, the research question '*To what extent the is binary search algorithm suitable to find specified values from datasets of varying size*?' will be investigated, to evaluate and thus appreciate why traditional methods of data management are insufficient.

1.1 Big Data

So what is big data? The phenomenon describes *"the exponential growth and availability of data, both structured and unstructured*[3]*"*. Structured data is data that is contained in a field and saved in a record or file[4]. A common example of this is data that is stored in a relational database or a log file. Conversely, unstructured data can be understood as not being organised in a particular way. An important example of this would be content on a social media site, such as Facebook. This content would include images, videos, text and advertisements to name a few, but the content will usually be too difficult to organise to be stored in a database or log file.

Requirements to big data management solutions can be broken down into four sections, also known as the four 'Vs' of big data. These are: volume, velocity, variety and veracity[5]. The volume of data used in big data analysis is far larger than that of typical data management solutions. Velocity describes the idea that companies that use big data solutions require real-time

---

[1] Pimentel, A. (2012). *Big Data: The Hidden Opportunity*. [online] Forbes.com. Available at: http://www.forbes.com/sites/ciocentral/2012/05/01/big-data-the-hidden-opportunity/#177b501168fa50ac3b1a68fa [Accessed 1 May 2012].

[2] Sas.com, (2015). *What Is Big Data?* [online] Available at: http://www.sas.com/en_us/insights/big-data/what-is-big-data.html [Accessed 19 Sep. 2015].

[3] Ibid.

[4] Webopedia.com, (2015). *What is Structured Data? A Webopedia Definition*. [online] Available at: http://www.webopedia.com/TERM/S/structured_data.html [Accessed 20 Sep. 2015].

[5] IBM Big Data Hub, (2015). *Infographic: The Four V's of Big Data | The Big Data Hub*. [online] Available at: http://www.ibmbigdatahub.com/infographic/four-vs-big-data [Accessed 19 Sep. 2015].

generation, so it can be used when needed. The variety problem details that data also takes different forms, so managing unstructured data becomes increasingly difficult and needs to be catered to. Finally, veracity involves the quality of data and uncertainties regarding these patterns, which play a large role in the field of big data.

The velocity problem was chosen as the main focus of this investigation in order to further understand how using traditional methods of data management can have consequences to the speed of result generation.

1.2 The Importance of Big Data

Data has become increasingly important and has adopted a major role in the global economy. Since its recent spike in 2012, big data analytics has provided a new method for businesses to analyse consumers' behaviours and their tendencies to buy certain products[6]. The use of big data allows these businesses to change their business models almost instantaneously to maximise their business profitability. Cisco, a multinational technology company, create approximately 30,000 different business models each year[7] in order to predict trends as accurately as possible. It can be said without argument that in the past, companies of this size would only be able to adopt a few static business models[8]. It is equally important for companies such as Netflix collect and manage data, where consumer habits are monitored and thus their tastes can be appropriately catered for[9].

The increased confidence that businesses gain from the results of data analysis allow them to make decisions based on said results. Approaches to analytics vary, and these can be either proactive or reactive[10]. The SAS institute considers these to be:

1) Business Intelligence: This is a reactive approach to analytics, whereby business reports and ad hoc reports are produced based on the results of analysis. This includes big data business

---

[6] Van der Hoek, A. (2015). *The Importance of Big Data is Growing Exponentially: Do You Have "Who" it Takes?* [online] Consumergoods.edgl.com. Available at: http://consumergoods.edgl.com/column/The-Importance-of-Big-Data-is-Growing-Exponentially--Do-You-Have--Who--it-Takes---97977 [Accessed 20 Sep. 2015].

[7] Ibid.

[8] Ibid.

[9] Smartdatacollective.com, (2016). *How Netflix Uses Big Data to Drive Business Success | SmartData Collective*. [online] Available at: http://www.smartdatacollective.com/bernardmarr/312146/big-data-how-netflix-uses-it-drive-business-success [Accessed 17 Jan. 2016].

[10] Sas.com, (2015). *Big data analytics: What it is and why it matters*. [online] Available at: http://www.sas.com/en_us/insights/analytics/big-data-analytics.html [Accessed 20 Sep. 2015].

intelligence, which is where decisions are made from reports of large datasets.

2) Big Analytics: As a proactive approach to data analytics, the aim of big analytics is to make decisions about the future. This involves methods such as optimisation, predictive modelling, text-mining, forecasting and statistical analysis. Through this, weaknesses and trends can be identified; however, it is unlikely that this can be achieved with big data, as it will take too long to process the data.
3) Big Data Analytics: similar to big analytics, big data analytics is a proactive approach to data analysis, allowing businesses to extract the relevant data from their storage and use it to make better decisions.

We can see that many businesses consider big data to play an important role within their business strategy by using analytics as a predictive tool to make decisions regarding the future, and as a reactive tool to make decisions based on existing information.

1.3 Challenges within Big Data

An important issue that has arisen in the data explosion is that standard data processing techniques are insufficient to cope with the complexity or size of datasets[11]. For some algorithms, such as the linear search algorithm[12] (where data is stored in a data structure and is looped through until a match is found), it may take too long to process the data, and so finding results would simply take too long. It was thus decided that the research question will be explored in order to challenge the size and volume aspects of big data.

The binary search algorithm is a simple, yet fundamental algorithm in the field of programming. It is used to find a specific value within an array that has already been sorted, where it starts at the middle of the array, compares if the number in question is higher or lower than the number that needs to be found, and then moves left or right along the array (depending on if it was higher or lower) to reach the number in question[13]. The binary search algorithm is a traditional type of data processing and will be used to investigate if its application is worth use in big data, or if other more suitable techniques are required.

---

[11] GCN, (2016). *What are big data techniques and why do you need them? -- GCN*. [online] Available at: https://gcn.com/microsites/2012/snapshot-managing-big-data/01-big-data-techniques.aspx [Accessed 21 Feb. 2016].

[12] www.tutorialspoint.com, (2016). *Data Structures & Algorithms Linear Search*. [online] Available at: http://www.tutorialspoint.com/data_structures_algorithms/linear_search_algorithm.htm [Accessed 21 Feb. 2016].

[13] Algorithms.openmymind.net, (2015). *Binary Search*. [online] Available at: http://algorithms.openmymind.net/search/binarysearch.html [Accessed 19 Sep. 2015].

1.4 Algorithmic Efficiency

It is important to understand all algorithms within the field of computing have an efficiency, describing how good an algorithm is under constraints of time (time complexity) or under constraints of memory within a computer (space complexity)[14]. Space complexity is relevant within this essay to understand that large companies must be aware of their system resources when managing large datasets, which must be as efficient as possible so to use as little memory as possible when completing a task. However, the concept of time complexity is more important within this investigation, as it is focused on determining the speed of the algorithm under datasets of varying sizes. Time complexity is concerned with the relationship between the complexity (often the size) of a problem, and the time taken to complete the problem.

Time complexity can thus be graphed, and an equation to state the relationship between the two variables can be found. For example, a particular algorithm could have the efficiency stated by the equation below:

$$f(n) = 2n^2 + 6n + 6$$

Where: n = size of the problem at hand

For larger and larger values of n, the values of $9n$ and 6 become insignificant[15], where the equation can be approximated to:

$$f(n) = 2n^2$$

From this equation, we can understand that $f(n)$ is proportional to n^2[16]. Thus, it can be said that the efficiency has an order of n^2. This can also be expressed where the time function $T(n) = O(n^2)$. This concept is known as Big-O notation, which has a fundamental role within computing. It describes the worst-case scenario for each particular algorithm[17] (meaning the situation that requires the most processing within a computer system[18]).

[14] Cs.bluecc.edu, (2016). *Algorithm Efficiency*. [online] Available at: http://cs.bluecc.edu/java/CS260/Notes/Efficiency.htm [Accessed 21 Feb. 2016].
[15] Ibid.
[16] Ibid.
[17] Rob-bell.net, (2009). *A beginner's guide to Big O notation - Rob Bell*. [online] Available at: https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/ [Accessed 21 Feb. 2016].
[18] Cs.bluecc.edu, (2016). *Algorithm Efficiency*. [online] Available at: http://cs.bluecc.edu/java/CS260/Notes/Efficiency.htm [Accessed 21 Feb. 2016].

2 Investigation

In order to investigate the research question, an experiment will be conducted. This involves creating a program in Java (see appendix) to test the binary search algorithm's efficiency over several datasets with varying sizes. The proposed algorithm will create pseudorandom numbers (of value 0 to 255 for simplicity) and store them into a file, move them into an array, sort them and then use the binary search algorithm to search for a randomly selected within the array. The number of data, file size and time taken to find the value will then be recorded. Searching algorithms are a necessity in large businesses. For example, a database of facial photographs may be searched through to find a certain person[19]. The program will therefore be a simplified model of such that can be found in larger businesses. In reality, it is likely that the algorithm used are far more complex in order to maximise efficiency.

For this investigation, I hypothesise that as the file size of the list increases, the time taken to process the data and find the integer will also increase. This is due to the computer having to process more data, which will thus take a longer.
The binary search method of the algorithm created (fully viewable in the appendix) can be viewed below[20].

Figure 1

```
public static int indexOf(int[] a, int key) {
        int lo = 0;
        int hi = a.length - 1;
        while (lo <= hi) {
            int mid = lo + (hi - lo) / 2;
            if      (key < a[mid]) hi = mid - 1;
            else if (key > a[mid]) lo = mid + 1;
            else return mid;
        }
        return -1;
    }
```

[19] ScienceDaily, (2016). *A simple solution for big data: New algorithm simplifies the categorization of data*. [online] Available at: http://www.sciencedaily.com/releases/2014/06/140626141650.htm [Accessed 18 Jan. 2016].
[20] Algs4.cs.princeton.edu, (2015). *BinarySearch.java*. [online] Available at: http://algs4.cs.princeton.edu/11model/BinarySearch.java.html [Accessed 20 Sep. 2015].

The method indexOf takes in the array being searched and takes in the value to be found as parameters Within the program, the method takes in an already sorted array and the specified integer to be found within the array. It takes two integers, lo, which compares to the first index in the array, and hi, which compares to the last index in the array. It then loops from lo to hi, finding the middle value between them. Since the array is sorted, if the key (value to be found) is greater than mid, it will move along the array towards the higher values until a match is found, and if it is less than mid, it will move along the array to the lower values until a match is found. If no match is found, -1 is returned.

The main process of the algorithm can be viewed below in figure 2. The timer is started, the data is sorted, a value from the array is randomly selected and that specific value is found from the array. The timer is then stopped after the process has been completed, and the time taken is printed (viewable in appendix).

Figure 2

```
//Calculate time to sort all data
long start = System.nanoTime();
Arrays.sort(data);
//searches for an integer at a pseudorandom location within the array.
int val = data[new RandomInteger().generate(data.length)];
int index = BinarySearch.indexOf(data, val);
long end = System.nanoTime();
```

2.1 Results

Figure 3 below tables the results of the experiments.

Figure 3

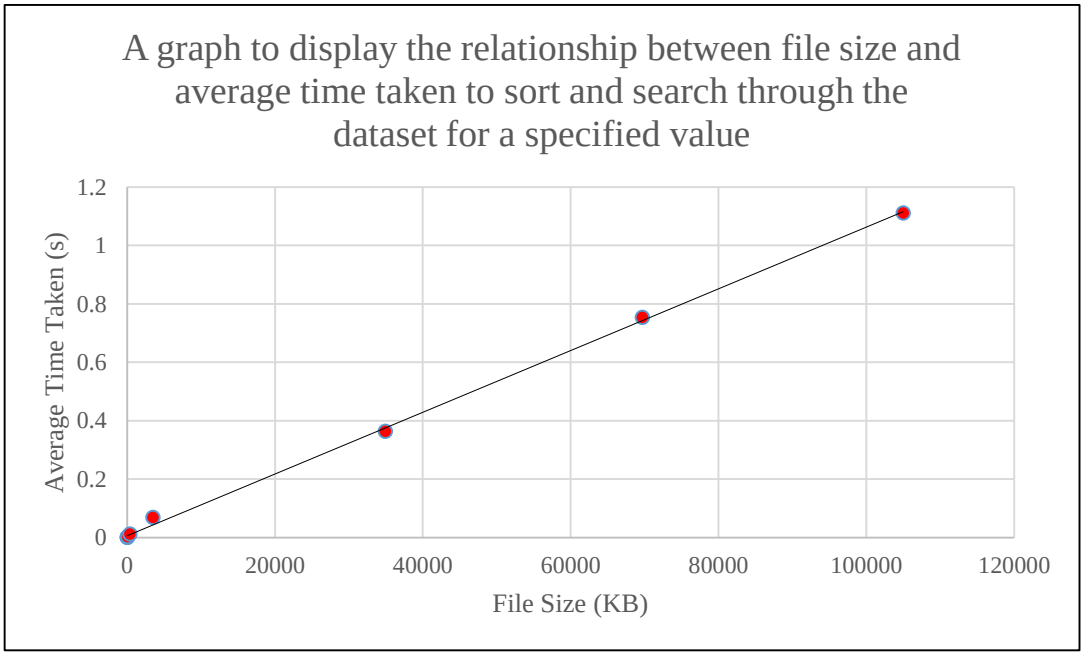
		Time taken to find integer (s)			
Number of integers	File Size (KB)	Repeat 1	Repeat 2	Repeat 3	Average time taken (s)
10	0.03	0.000856165	0.000608981	0.000537990	0.000668
100	0.36	0.000885674	0.000621383	0.000628653	0.000712
1000	3.48	0.001036636	0.000954526	0.001357377	0.001116
10000	51.8	0.003790735	0.003087670	0.003278831	0.003386
100000	365	0.010406131	0.012348541	0.012041912	0.011599
1000000	3490	0.073161094	0.073383475	0.060799723	0.069115
10000000	34900	0.373367267	0.362390216	0.356383800	0.364047
20000000	69700	0.729905877	0.745461404	0.786033897	0.7538
30000000	105000	1.144285694	1.075317759	1.113183195	1.110929

It is apparent that as the file size increases, it is also the case that the average time taken to find the value increases. For example, at a file size of 0.03KB, the average time taken is 0.000668 seconds. At a file size of 34900KB, the average time taken is 0.364047 seconds. It is therefore clear that there is a form of positive relationship between the file size and time taken to complete the task. In order to find a clear relationship between file size and average time taken, a graph has been plotted.

2.2 Graph

Figure 4 below graphs the results by measuring the relationship between the average time taken to sort and search for a randomly chosen value within the dataset at that file size.

Figure 4



2.3 Analysis

The graph shows a positively correlated linear relationship between average time taken to sort the array and the file size of the array. This means that the time taken, and thus the performance of the algorithm grows linearly with direct proportionality to the size of the dataset[21].

A linear regression has been calculated from the data (using a TI-84 Plus model calculator), as an equation in the form $y = mx + c$:

$$y = (1.0558326 \times 10^{-5})x + 0.006787749$$

Where: y = Time taken to complete operation (seconds)

x = Size of file containing integers (kilobytes)

[21] Rob-bell.net, (2009). *A beginner's guide to Big O notation - Rob Bell*. [online] Available at: https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/ [Accessed 21 Feb. 2016].

The equation describes that as the file size increases, time taken to process the data and thus find a certain value increases. Although it is the case that the binary search algorithm is known to have $O(\log_2 n)$ efficiency[22], the time taken to sort the array has also been factored in. This is due to the fact that other searching algorithms other than the binary search algorithm may not require a sorted dataset to perform a search operation. From the equation, it can be understood that the process of both sorting and searching has an efficiency of order $O(n)$.

The equation can be used to extrapolate data and predict the efficiency of the searching and sorting process with different file sizes. For example, the time taken for one terabyte of data (1000000000 kilobytes) to be processed can be calculated:

$$y = (1.0558326 \times 10^{-5})(1000000000) + 0.006787749$$
$$= 10565.11375 \text{ seconds}$$

It is estimated from the above calculation that the time taken to find a value within one terabyte of data using the binary search algorithm is approximately three hours. This can be applied to a greater context. For larger companies, it is fairly obvious that datasets could succeed even billions or trillions of pieces of data, with file sizes potentially reaching hundreds terabytes for larger companies. For example, Facebook collects over 500 terabytes of data per day[23]. Finding a specific value within a dataset of this size could take up to hundreds of hours using the binary search algorithm, and for this reason, it may be unsuitable for data of that size.

2.4 Implications

The investigation thus highlights a clear relationship between file size and the time it takes for the operation to complete. It is evident that the binary search algorithm is a completely feasible method to find specific data items within small datasets, because the time taken would be relatively small. However, using the binary search algorithm to search for values within large data sets would take a very large amount of time. Data analytics often involves making predictions about certain changes to come, so it is important that results of predictive analysis come quickly, so companies can react and plan accordingly.

---

[22] Research.cs.queensu.ca, (2016). *Search Algorithms & Algorithm Efficiency Lesson*. [online] Available at: http://research.cs.queensu.ca/home/cisc101spring/Spring2006/webnotes/search.html [Accessed 18 Jan. 2016].

[23] Kern, E. (2012). *Facebook is collecting your data — 500 terabytes a day*. [online] Gigaom.com. Available at: https://gigaom.com/2012/08/22/facebook-is-collecting-your-data-500-terabytes-a-day/ [Accessed 18 Jan. 2016].

Another example of where speed is critical could be when the police are searching through criminal records, matching a photograph to a certain record in order to retrieve the details of a possible suspect. It is likely that there are an extremely large number of criminal records in any country's database, and it would be necessary for the algorithm to be especially fast in this case, so to prevent any further danger that this suspect could cause. The binary search algorithm may take too long to provide a result in this scenario, which could allow said suspect to escape.

2.5 Limitations of the Investigation

After the investigation had been completed, it became apparent that there were several limitations that needed to be explicitly discussed. These are in regards to the investigation methodology and the binary search algorithm itself.

The first major limitation to the investigation was that any number of integers over roughly 30000000 could not be parsed by the program and returned an overflow error, leading to a maximum file size of 105000 kilobytes. The data therefore may not be as reliable as there could be a change in pattern further along the graph that is not evident from the data. It follows suit that it is less reliable to extrapolate the data for larger file-sizes, which could lead to inaccurate predictions of time taken for larger datasets.

Secondly, there was a decision that had to be made as to whether to include the time to sort the array of data or not. The experiment followed the argument that the use of algorithms such as the linear search algorithm[24] do not require sorting to be used, but it could also be argued that most searching algorithms (usually the best) require a sorted dataset, and thus there may not have been the need to account for the time taken to sort the array.

Another limitation that came to light was that the hardware of the computer used to obtain the data (see appendix for details) is far below the specification of that needed for large-scale data analysis. It is well known that computers that are used for this purpose often have specialised hardware, such as NVidia's Tesla range of GPU accelerators[25], which are mainly used in servers for scientific computing and data analysis. Using specialised hardware, the results of the same investigation will differ due to the computer being able to process comparisons and calculations

[24] www.tutorialspoint.com, (2016). *Data Structures & Algorithms Linear Search*. [online] Available at: http://www.tutorialspoint.com/data_structures_algorithms/linear_search_algorithm.htm [Accessed 21 Feb. 2016].

[25] NVidia, (2016). *High Performance Computing for Servers | Tesla GPUs|NVIDIA UK*. [online] Available at: http://www.nvidia.co.uk/object/tesla-server-gpus-uk.html [Accessed 18 Jan. 2016].

more quickly, and thus would mean that the time taken to find the random integer would be lower.

The final limitation of the investigation is that it does not address the problem of unstructured data as the algorithm and data used are effectively a simplification of the process that would occur when managing structured data sets. Unstructured data is a major problem within large-scale businesses[26], of which the investigation cannot be applied to.

3 Managing Larger Data Sets

It quickly became apparent that though the binary search algorithm can be used to find values within smaller datasets, it would take increasingly more time to find a specified value. In addition to this, the use of the binary search algorithm is limited to searching through structured sets of data, and thus may not be applicable for more data that is more random. For this reason, other solutions to the management of large data sets were explored. One of the major solutions to the management of large data sets is Apache Hadoop, a framework used to manage structured, and more specifically, help manage unstructured data.

In big-data based solutions such as Hadoop, it is often the case that clusters are used to manage data. Clusters are defined as a group of computers that work together, often connected in a network. In regards to data analytics, Clusters are used to process data together and help manage resources between each other. Hadoop manages large data sets by effectively distributing this data across the cluster and yet acting as a single unit when processing tasks.

Apache Hadoop is broken up into three components, YARN, MapReduce and the Distributed File System. Hadoop YARN provides resource management capabilities on a cluster (multiple) of servers[27]. This means that clusters can work together in tandem in order to process a task, rather than individually. Hadoop MapReduce provides the software framework for distributed processing, helping manage large data sets within computer clusters[28]. It is also in charge of

[26] Grimes, S. (2005). *Structure, Models and Meaning - InformationWeek*. [online] Available at: http://www.informationweek.com/software/information-management/structure-models-and-meaning/d/d-id/1030187? [Accessed 18 Jan. 2016].
[27] Dawson, R. (2015). *Untangling Apache Hadoop YARN, Part 1: Cluster and YARN Basics - Cloudera Engineering Blog*. [online] Cloudera Engineering Blog. Available at: http://blog.cloudera.com/blog/2015/09/untangling-apache-hadoop-yarn-part-1/ [Accessed 18 Jan. 2016].
[28] Webopedia.com, (2016). *What is Hadoop MapReduce? A Webopedia Definition*. [online] Available at: http://www.webopedia.com/TERM/H/hadoop_mapreduce.html [Accessed 18 Jan. 2016].

managing tasks: involving monitoring, scheduling tasks and correcting errors. This makes Hadoop a fault-tolerant system, meaning that errors can be handled without user input. Hadoop Distributed File System (HDFS) is a scalable file system that allows files to be distributed and stored across a cluster, allowing potentially huge files to be stored[29].

Hadoop has several uses, such as analysing and indexing data within ecommerce, finding the most appropriate advertisements and search results. It is also heavily used within data analytics to find precise relationships within data. It also has the capability of machine learning for pattern detection, such as those used within search engines.

Apache Hadoop provides features that a simple search algorithm such as binary search cannot, and is therefore far superior for the management of very large amounts of data. On the other hand, many of the features of Hadoop may not be correctly utilised when only managing smaller amounts of data, and its 'cluttered' nature may provide problems for people who try to do this.

Conclusion

Concluding the question '*To what extent the is binary search algorithm suitable to find specified values from datasets of varying size*?' depends on several factors. Sizes of data sets vary from small to large, but it is important to consider that the definition of a 'large' dataset itself can vary, so it is important to understand that 'large' is relative to the scale of operations it is needed for. A small company that, for example, wants to store account details for reward cards may consider a few thousand accounts to be a large number, of which the binary search algorithm is a perfectly appropriate method to find specific values. On the other hand, larger businesses that want to store data that is exponentially larger in size may find that the binary search algorithm is unsuitable: it will simply take far too much to sort all of the data and find a specific value.

Despite the limitations presented within the investigation, it was found that the binary search algorithm is suitable for small datasets that are structured in nature. The program created had an efficiency of order $O(n)$ (between file size and time taken). This linear relationship made it clear that the binary search algorithm is not efficient enough to be used within larger datasets. Furthermore, it cannot be applied to unstructured datasets at all. Instead, it is evident that data

[29] Garment, V. (2014). *Hadoop 101: The Most Important Terms, Explained*. [online] Plottingsuccess.com. Available at: http://www.plottingsuccess.com/hadoop-101-important-terms-explained-0314/ [Accessed 18 Jan. 2016].

management solutions with the correct framework to handle large data, such as Apache Hadoop, prove to be more effective as they have more appropriate functionalities to handle large data sets, such as being able to manage resources, tasks and files within a cluster. A program which utilises the simple binary search algorithm, on the other hand, is limited to a single computing unit, and thus its use is limited to very small-scale operations. Using Apache Hadoop for small-scale operations would be inappropriate, as it provides many features that would be unnecessary. It can be concluded that a program that utilises the binary search algorithm is thus most suitable to find specified values of small datasets.

References

Algorithms.openmymind.net, (2015). *Binary Search*. [online] Available at: http://algorithms.openmymind.net/search/binarysearch.html [Accessed 19 Sep. 2015].

Algs4.cs.princeton.edu, (2015). *BinarySearch.java*. [online] Available at: http://algs4.cs.princeton.edu/11model/BinarySearch.java.html [Accessed 20 Sep. 2015].

Backupify, (2015). *Bits & Bytes: A History of Data Storage*. [online] Available at: https://www.backupify.com/history-of-data-storage/ [Accessed 19 Sep. 2015].

Cs.bluecc.edu, (2016). *Algorithm Efficiency*. [online] Available at: http://cs.bluecc.edu/java/CS260/Notes/Efficiency.htm [Accessed 21 Feb. 2016].

Dawson, R. (2015). *Untangling Apache Hadoop YARN, Part 1: Cluster and YARN Basics - Cloudera Engineering Blog*. [online] Cloudera Engineering Blog. Available at: http://blog.cloudera.com/blog/2015/09/untangling-apache-hadoop-yarn-part-1/ [Accessed 18 Jan. 2016].

Garment, V. (2014). *Hadoop 101: The Most Important Terms, Explained*. [online] Plottingsuccess.com. Available at: http://www.plottingsuccess.com/hadoop-101-important-terms-explained-0314/ [Accessed 18 Jan. 2016].

GCN, (2016). *What are big data techniques and why do you need them? -- GCN*. [online] Available at: https://gcn.com/microsites/2012/snapshot-managing-big-data/01-big-data-techniques.aspx [Accessed 21 Feb. 2016].

Grimes, S., (2005). *Structure, Models and Meaning - InformationWeek*. [online] Available at: http://www.informationweek.com/software/information-management/structure-models-and-meaning/d/d-id/1030187? [Accessed 18 Jan. 2016].

IBM Big Data Hub, (2015). *Infographic: The Four V's of Big Data | The Big Data Hub*. [online] Available at: http://www.ibmbigdatahub.com/infographic/four-vs-big-data [Accessed 19 Sep. 2015].

Kern, E. (2012). *Facebook is collecting your data — 500 terabytes a day*. [online] Gigaom.com. Available at: https://gigaom.com/2012/08/22/facebook-is-collecting-your-data-500-terabytes-a-day/ [Accessed 18 Jan. 2016].

NVidia, (2016). *High Performance Computing for Servers | Tesla GPUs|NVIDIA UK*. [online] Available at: http://www.nvidia.co.uk/object/tesla-server-gpus-uk.html [Accessed 18 Jan. 2016].

Pimentel, A. (2012). Big Data: The Hidden Opportunity. [online] Forbes.com. Available at: http://www.forbes.com/sites/ciocentral/2012/05/01/big-data-the-hidden-opportunity/#177b501168fa50ac3b1a68fa [Accessed 19 Sep. 2015].

Research.cs.queensu.ca, (2016). *Search Algorithms & Algorithm Efficiency Lesson*. [online] Available at: http://research.cs.queensu.ca/home/cisc101spring/Spring2006/webnotes/search.html [Accessed 18 Jan. 2016].

Rob-bell.net, (2009). *A beginner's guide to Big O notation - Rob Bell*. [online] Available at: https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/ [Accessed 21 Feb. 2016].

Sas.com, (2015). *Big data analytics: What it is and why it matters*. [online] Available at: http://www.sas.com/en_us/insights/analytics/big-data-analytics.html [Accessed 20 Sep. 2015].

Sas.com, (2015). *What Is Big Data?* [online] Available at: http://www.sas.com/en_us/insights/big-data/what-is-big-data.html [Accessed 19 Sep. 2015].

ScienceDaily, (2016). *A simple solution for big data: New algorithm simplifies the categorization of data*. [online] Available at: http://www.sciencedaily.com/releases/2014/06/140626141650.htm [Accessed 18 Jan. 2016].

Smartdatacollective.com, (2016). *How Netflix Uses Big Data to Drive Business Success | SmartData Collective*. [online] Available at: http://www.smartdatacollective.com/bernardmarr/312146/big-data-how-netflix-uses-it-drive-business-success [Accessed 17 Jan. 2016].

van der Hoek, A. (2015). *The Importance of Big Data is Growing Exponentially: Do You Have "Who" it Takes?* [online] Consumergoods.edgl.com. Available at: http://consumergoods.edgl.com/column/The-Importance-of-Big-Data-is-Growing-Exponentially--Do-You-Have--Who--it-Takes---97977 [Accessed 20 Sep. 2015].

Webopedia.com, (2015). *What is Structured Data? A Webopedia Definition*. [online] Available at: http://www.webopedia.com/TERM/S/structured_data.html [Accessed 20 Sep. 2015].

Webopedia.com, (2016). *What is Hadoop MapReduce? A Webopedia Definition*. [online] Available at: http://www.webopedia.com/TERM/H/hadoop_mapreduce.html [Accessed 18 Jan. 2016].

www.credera.com, (2013). *Apache Hadoop Explained in 5 Minutes or Less - www.credera.com*. [online] Available at: https://www.credera.com/blog/technology-insights/open-source-technology-insights/apache-hadoop-explained-5-minutes-less/ [Accessed 18 Jan. 2016].

www.tutorialspoint.com, (2016). *Data Structures & Algorithms Linear Search*. [online] Available at: http://www.tutorialspoint.com/data_structures_algorithms/linear_search_algorithm.htm [Accessed 21 Feb. 2016].

Appendix

Relevant Computer Specifications

These are the computer specifications

Processor: Intel Core i7-4710MQ CPU @ 2.40 GHz

Memory: 8GB DDR3

Graphics Processor: NVidia GeForce GTX 765M Mobile Chipset (2GB GDDR5)

Operating System: Windows 10 Home Premium

Console Log

```
File replaced with 10 random integers.
Time taken to find 115 = 8.561650000000001E-4 seconds at index 4.
File replaced with 10 random integers.
Time taken to find 197 = 6.089810000000001E-4 seconds at index 5.
File replaced with 10 random integers.
Time taken to find 175 = 5.3799E-4 seconds at index 7.

File replaced with 100 random integers.
Time taken to find 136 = 6.286530000000001E-4 seconds at index 61.
File replaced with 100 random integers.
Time taken to find 217 = 6.213830000000001E-4 seconds at index 85.
File replaced with 100 random integers.
Time taken to find 243 = 8.85674E-4 seconds at index 94.

File replaced with 1000 random integers.
Time taken to find 21 = 0.001357377 seconds at index 72.
File replaced with 1000 random integers.
Time taken to find 185 = 9.54526E-4 seconds at index 725.
File replaced with 1000 random integers.
Time taken to find 236 = 0.001036636 seconds at index 929.

File replaced with 10000 random integers.
Time taken to find 139 = 0.0037907350000000004 seconds at index 5428.
File replaced with 10000 random integers.
Time taken to find 253 = 0.0032788310000000003 seconds at index 9921.
File replaced with 10000 random integers.
```

```
Time taken to find 254 = 0.0030876700000000003 seconds at index 9980.

File replaced with 100000 random integers.
Time taken to find 21 = 0.012041912 seconds at index 8201.
File replaced with 100000 random integers.
Time taken to find 237 = 0.012348541000000001 seconds at index 93358.
File replaced with 100000 random integers.
Time taken to find 30 = 0.010406131 seconds at index 12108.

File replaced with 1000000 random integers.
Time taken to find 65 = 0.06079972300000001 seconds at index 257811.
File replaced with 1000000 random integers.
Time taken to find 203 = 0.073383475 seconds at index 796874.
File replaced with 1000000 random integers.
Time taken to find 152 = 0.07316109400000001 seconds at index 597655.

File replaced with 10000000 random integers.
Time taken to find 36 = 0.35638380000000003 seconds at index 1445311.
File replaced with 10000000 random integers.
Time taken to find 12 = 0.36239021600000004 seconds at index 507811.
File replaced with 10000000 random integers.
Time taken to find 95 = 0.373367267 seconds at index 3749999.

File replaced with 20000000 random integers.
Time taken to find 182 = 0.786033897 seconds at index 14296874.
File replaced with 20000000 random integers.
Time taken to find 163 = 0.745461404 seconds at index 12812499.
File replaced with 20000000 random integers.
Time taken to find 241 = 0.7299058770000001 seconds at index 18906249.

File replaced with 30000000 random integers.
Time taken to find 105 = 1.1442856940000001 seconds at index 12421874.
File replaced with 30000000 random integers.
Time taken to find 135 = 1.075317759 seconds at index 15937499.
File replaced with 30000000 random integers.
Time taken to find 154 = 1.113183195 seconds at index 18164061.
```

Project Code

Driver Class (runs the program)

```
import java.io.FileNotFoundException;
public class Manipulator {
    public static void main(String args[]){
        AddToFile a = new AddToFile();
        a.setn(20000000);
        a.generate();
        int[] data = null;
        //Read from file and enter into an array.
        try {
            FileReader fr = new FileReader("file.txt");
            BufferedReader br = new BufferedReader(fr);
            String line;
            String[] temp = null;
            while((line=br.readLine()) != null)
            {
                temp = line.split(",");
            }
            br.close(); fr.close();

            data = new int[temp.length];
            //move and convert values to integers
            for(int i = 0; i < temp.length; i++) data[i] = Integer.parseInt(temp[i]);

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        //Calculate time to sort all data
        long start = System.nanoTime();
        Arrays.sort(data);
        //searches for an integer at a pseudorandom location within the array.
        int val = data[new RandomInteger().generate(data.length)];
        int index = BinarySearch.indexOf(data, val);
        long end = System.nanoTime();
        double total = (end-start)*(Math.pow(10, -9));
        System.out.println("Time taken to find "+val+" = "+total+" seconds at index "+index+".");
        }
}
```

Writing Integers to File Class

```
import java.io.PrintWriter;


public class AddToFile {
    private int n;
    private RandomInteger ranInt;
    public AddToFile(){
        n = 0;
        ranInt = new RandomInteger();
    }

    public void generate(){
        try {
            PrintWriter writer = new PrintWriter("file.txt");
            writer.print("");
            for(int i = 0; i < n; i++) writer.print(ranInt.generate(255)+",");
            System.out.println("File replaced with "+n+" random integers.");
            writer.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    public void setn(int n){
        this.n = n;
    }

}
```

Random Integer Generator Class

```
import java.util.Random;


public class RandomInteger {
    private Random rand = new Random();
    public RandomInteger(){

    }
    public int generate(int n){
        return rand.nextInt(n);
    }
}
```

EXAMPLE ESSAY – Received 34 points = A grade

Binary Search Algorithm Class

```
public class BinarySearch {

    private BinarySearch() { }

    public static int indexOf(int[] a, int key) {
        int lo = 0;
        int hi = a.length - 1;
        while (lo <= hi) {
            // Key is in a[lo..hi] or not present.
            int mid = lo + (hi - lo) / 2;
            if      (key < a[mid]) hi = mid - 1;
            else if (key > a[mid]) lo = mid + 1;
            else return mid;
        }
        return -1;
    }

}
```