Asian Journal of Advances in Research

4(1): 968-976, 2021

DEPLOYMENT OF AN EFFICIENT ALGORITHM FOR SEARCHING MOTOR VEHICLE DATABASE

DIPO THEOPHILUS AKOMOLAFE[1*] AND NGOZI MAUREEN NWANZE[2]

[1]Department of Mathematical Science, Olusegun Agau University of Science and Technology, Okitipupa, Nigeria.

[2]Department of Computer Education, Federal College of Education (Technical), Asaba, Nigeria.

AUTHORS' CONTRIBUTIONS

This work was carried out in collaboration between both authors. Both authors read and approved the final manuscript.

Received: 03 July 2021
Accepted: 07 September 2021
Published: 09 September 2021

Original Research Article

ABSTRACT

Transportation is a requirement for every nation regardless of its level of development. Moving goods and people from one place to another is crucial to maintaining strong economic and political ties between the various components of any given nation and among nations. How that movement takes place can be determined by several variables but the most reliable and widely embraced means of transportation is road. Road transportation is the most widely used means of transportation due to its cheapness and capability to penetrate remotest parts of any town and village. Therefore, various measures and devices had been and are being developed to enhance its continuity, sustainability and safety. Most of these measures rely on database. Database is a structured set of data held in a computer, especially one that is accessible in various ways. In order to enhance a smooth running of database developed for the purpose of road transportation generally and searching of vehicles in particular, there is the need to deploy an efficient searching algorithm.

In this research, a database developed for vehicles was used as a focus point. Different search algorithms were deployed to test the database, their complexities were derived and it was discovered that these were not fast enough, effective and efficient when implemented on road transport database. Consequently, the need to develop the most efficient algorithm that could guarantee fast and quick response to queries and present results with minimum delay was necessitated.

Keywords: Transportation; database; algorithm; search; complexity of algorithm; linear search; binary search.

1. INTRODUCTION

Transportation can be briefly defined as the movement of people and or goods from one place to another through a means. The means may be by walking on land, vehicle on road, train on the rail and plane in the air or ship on the sea. The concern of this research is the movement of vehicles on roads in general which will be henceforth referred to as road transportation [1]. Road transportation therefore is the conveyance of people and or goods on road from one place to another by a particular mode. Road transportation is characterized by its multimodal system, which may be by walking, cycling and driving. There are many benefits associated with the multimodal characteristics of road transportation and these benefits contributed to its popularity. Also, this multimodal feature and benefits of road transportation created diverse challenges that make it difficult to

**Corresponding author: Email: dtakomolafe@yahoo.com;*

present information associated with vehicles promptly and timely.

One of the requirements for effective management of road transport management is the development of road transportation database [2]. This can be seen in [3] where a relational distributed database of road safety was developed and implemented in an environment characterized by RAPPORT database. Also, [4] affirmed the necessity of database system in transportation system with emphasis on road transportation. Miller et al. [5] and [6] showed the relevance of database system to road transportation. In the design and development of road transportation database, some terms must be clearly understood and adopted and one of such terms is the study of the database models. A database model is **a** type of data model that determines the logical structure of **a** database. It fundamentally determines in which manner data can be stored, organized and manipulated. The most popular example of a database model is the relational model, which uses a table-based format. The main reason for identifying appropriate database model is to identify what to model, how to model and to present a scheme of what is modelled so that the characteristics of the resulting database could be identified. Database model is concerned with the appropriate way to arrange the data so that the component data can be efficiently transacted upon [7,8].

In the course of implementation of the database developed for road transport especially for searching of vehicles, it was discovered that precious time is either being wasted or lost in the course of searching because it usually takes a very long time before a particular vehicle could be searched and found [9]. It was consequently discovered that the type of searching algorithm deployed in the database determines how fast and efficient the database can respond to queries.

Therefore, it is important to evolve a searching algorithm that can quickly, effectively and efficiently search through the database and present the desired result with minimum delay. This research is being carried out to propose appropriate algorithm for searching of vehicles in a road transportation database.

1.1 Objectives

The primary objective of this research is to develop an efficient algorithm for road transportation database. Other objectives are to:

a. Reduce cost of transactions on road transportation database
b. Ensure prompt retrieval of information from the database
c. Reduce access time to information

2. SEARCHING ALGORITHMS

A searching algorithm [10-12] is that type of algorithm that allows the efficient retrieval of a particular item from a set of many items. Searching is the algorithm process of finding a specific item in a collection of item. A search typically answers the user whether the item being searched for is present or not. Computer systems are often used to store large amounts of data from which individual records can be retrieved according to some search criterion so, it is required to search and fetch the data in that manner so that it will take lesser time and will be efficient [11]. For this purpose some approaches are needed that not only saves time but also fetches the required data efficiently. One of the major characteristics of road transportation database is the volume and uniqueness of its data. Resultantly, there is the need to develop a very fast, reliable and cost effective algorithm that will meet the unique characteristics of the database. A lot of studies had been carried out in the field of algorithm and searching and these had led to specific search algorithms developed for various data structures.

Ahmad Shoaib Zia [13] presents the review of certain important and well discussed traditional as well as proposed search algorithms with respect to their time complexity, space Complexity , merits and demerits with the help of their realize applications. Ahmad Shoaib Zia [13] also highlights their working principles. As it is well known that every sorted or unsorted list of elements requires searching technique so many search algorithms have been invented. Among traditional search algorithms, a comparison table has been made in order to check and establish their benefits and drawbacks. Among some of the well known proposed search algorithms like fast string searching algorithm, multiple solution vector approach and bi linear search etc, a comparison has also been made. Ritu et al. [14] proposed algorithm for path-compressed trie for keyword searching in a database through a search engine. Faster searching optimizes the search engine and speeds up the complete process of creating final results. Thus, greater SEO (search engine optimization), faster will be Page Rank Algorithm.

The most popular of the search algorithms are the linear, binary and tree searches.

Linear search is the simplest of the algorithms. This search simply starts by searching from the first of the list or array of keys and compare each successive item against the key until it is able to find a match for the searched item or reach the end. In linear search, each item in the list is looked upon in turn, quitting once an item that matches the search term is found or once the end of the list is reached. The “return value” is the index at which the search term was found, or some indicator that the search term was not found in the list.

3.1.1 Algorithm for linear search

```
for (each item in list) {
        compare search term to current item
        if match,
                save index of matching item
                break
}
return index of matching item, or -1 if item not found
```

The advantages of this algorithm lie in its ease to program and its ability to search unsorted lists. Therefore, it is highly useful when searching small sized arrays or records. However, [15] pointed out that the algorithm is found to be very inefficient when used on a large data set. Therefore, it may not be appropriate for use in road transportation database.

2.1 Binary Search

Another search algorithm is binary search. Wu et al. [16] proposed binary search algorithm to meet the needs of quick shopping in supermarkets, to prevent from goods anti-collision, which makes it faster to scan and identify the goods. On this basis, the goods which have not been checked out will be scanned again to obtain security. Besides, it plays an important role in the supermarket shopping system to achieve the functions of self-purchase and membership scheme. For the RFID supermarket shopping system, the test of the checkout efficiency, product recognition rate, and security were done. The result showed that the binary search algorithm had good accuracy in the RFID supermarket shopping system. The result also proved the rationality of the RFID supermarket shopping system, which can meet the daily needs of supermarkets. The method was applying the emerging technology into the traditional supermarket shopping system, which improved the profitability and management level of supermarkets.

In binary search, the search begins with a sorted array of n keys. To search for key r, r is compared to the middle key thereby reducing the number of possible location of r in the arrays until it is found. During each stage of the search, the search for r which is an ITEM is reduced to a segment of elements of DATA:

DATA[START + 1],
DATA[START+2].......
.DATA[END].

It should be noted that variables START and END denote, respectively, the beginning and end locations of the segment under consideration. The algorithm compares r with the middle element DATA MID] of the segment where MID is obtained by: MID = INT ((START +END)/2).

The algorithm for binary search is formally stated thus:

BINARY (DATA, LB, UB, ITEM, LOC)
Where:
DATA is a sorted array with lower bound LB and upper bound UB,
ITEM is a given item of information.
The variables START, END and MID denote respectively the beginning, end and middle locations of a segment of elements of DATA.

This algorithm finds the location LOC of ITEM in DATA or sets LOC = NULL.

```
[Initialize segment variables.]
Set START:=LB, END :=UB and MID = INT((START+END)/2).
Repeat Steps 3 and 4 while START ≤ END and DATA [MID] ≠ ITEM
if ITEM <DATA[MID], then:
Set END : = MID – 1
                    Else:
                        Set START : = MID + 1.
[End of if structure.]
    Set MID : = INT ((START + END)/2)
    [End of Step 2 loop.]
If DATA[MID] = ITEM, then:
       Set LOC:= MID.
    Else:
       Set LOC: = NULL
[End of if structure.]
Exit.
```

In this algorithm, whenever ITEM does not appear in DATA, the algorithm eventually arrives at the stage that START = END = MID. Then the next step yields END<START and control transfers to Step 5 of the algorithm.

3. COMPLEXITY OF THE BINARY SEARCH ALGORITHM

The complexity of this algorithm is measured by the number f(n) of comparisons to locate ITEM in DATA where DATA contains n elements. It should be noted that each comparisons reduces the sample size by half therefore, we require at most f(n) comparisons to locate ITEM

Where

$2^{f(n)}>n$ or $f(n) = [\log_2 n]+1$

This means that the running time for the worst case is approximately equal to $\log_2 n$.

The limitation of this algorithm is that it spends a very long time before it concludes its search. Associated with this delay is cost and time. In other words, it is having a complexity (this is measured by the number f(n) of comparisons to locate ITEM in DATA

Where

DATA contains n elements) of $f(n) = [\text{Log}_2 n] + 1$.

Hui Yonghui [17] gave a greater importance to the search algorithm because they have assumed that the data will be complete and focused on Two search algorithms to learn the structure of a Bayesian network. The heuristic search algorithm is simple and explores a limited number of network structures. On the other hand, the exhaustive search algorithm is complex and explores many possible network structures. Searching is a process that cannot be issued for a transaction and communication process, many search algorithms that can be used to facilitate the search, linear, binary, and interpolation algorithms are some searching algorithms that can be utilized, the comparison of the three algorithms is performed by testing to search data with different length with pseudo process approach, and the result achieved that the interpolation algorithm is slightly faster than the other two algorithms.

Various attempts had been made to improve on this complexity and the most recent one is [12] which was an improvement on [18]. This study proposed double character search before dissolving to binary search. The algorithm proposed to reduce the search list in two stages by a comparison of characters. In the first stage, the list would be reduced based on comparison of the first character of the search key with the first character of the strings in the search list. This will divide the list into three parts; strings starting with character less or greater than or equal to the starting character of the search key. At this stage, strings with first character less or greater than the starting character of the search key will be eliminated. In the second stage, only strings with first character equal to the first character of the search key shall be considered. The search list would be reduced further by repeating the above comparison, on the second characters of the strings in the search list with the second character of the search key. All these would result in a ‘virtual’ sub array defined by new ‘low’ and ‘high’ index bounds without removing elements from the original array. The final results of these two stages are passed to binary search to search for the actual element. The algorithm of the Double Elimination Binary Search (DEBS) can be defined thus:

```
public static string doublecharEliminate(strind search array[], string searchkey)
IntaLenght = searchArray.lenght – 1  //alenght = length of array
charfirstchar = searchkey.charAt(0)    //firstchar = 1st char of element
intfirstindex = 0                      //firstindex = any counter
while (firstchar = ! searchArray[firstindex].charAt(0)&&firstindex<= a length)
firstindex ++;
if (first index == aLenght)
return -1
intfirstLow = firstindex
intfirstcharcount = firstLow
while (firstchar = = searchArray[firstcharcount].charAt(0) &&firstcharcount<= aLenght
firstcharcount ++;
intfirstHigh = firstcharcount
//repeat procedure for second character
charsecondchar = searchkey.charAt(1)
intsecindex = firstlow
```

```
17.   while (secondchar =! searchArray[secindex].charAt(1) &&secindex<= firstHigh)
18.   secIndex ++;
19    if (secindex = =firstHigh)
20    return -1
21    intsecondLow = secindex
22.   secondcharcount = secondLow
23    while (secondChar = = serachArray[secondCharCount].charAt(1) &&secondCharCount<= firstHigh)
24    secondCharCount ++;
25    intsecondHigh = secondCharCount
26    return secondlow, secondHigh.
```

4. AVERAGE CASE ANALYSIS

In this work, the average case complexity of traditional binary search was used to find the complexity of the algorithm. In the algorithm, 19 basic operations are executed in order to obtain the sub array and by considering the 19 basic operations, and the searchable elements ‘s’:

S = m/676 (1.1)

And the complexity of binary search which had been found approximately to be :

T = 9log2n-9 (1.2)

By considering the 19 basic operations, we can rewrite equation (2) as

T = 9log2n-9+19
or T = 9log2n + 10 (1.3)

Finally, (1) was substituted for the number of elements “n” in array (3) to obtain:

T = 9log2(m/676) + 10 (1.4);

which results

T = 9log2n-75 which is the complexity of double elimination binary search.

However, this algorithm is very efficient in array search but may not be too good in road transportation database search because of the peculiarity of vehicle numbers. It is in view of this that a new algorithm is being proposed.

4.1 First and Last Character Sieve Binary Search Algorithm

The major features of this algorithm are:

1 It attempts to reduce the list of records by sieving out members that have first character and last character different from those of the keys
2 Reduce the database records by comparing the primary key with the first character and last character of the search key
3 The reduced list is now passed to a binary search to locate the record.

4.2 Algorithm

```
Public Static String First Last Sieve (string search key)
Int. rec.lenght = recount-1
Char  firstchar = searchkey.char At (0)
Char  last Char =search key.char At (search key length – 1)
Int first index = 0
While (first char! = first char (primary key)
αα first index <= reclenght
first index ++
return -1
Int first flow = first index
Int first char count = first flow
While (firstchar = = firstchar (primary key) &&
Firstcharcount<= reclenght
If lastchar = lastchar (primary key)
Set filter to firstchar = firstchar (primary key)
&&lastchar = lastchar (primary key)
```

CODIFICATION USING JAVA

```
//package komojava;
/* This algorithm searches a database of Nigeria Vehicle plate
   Numbers by reducing the search list to elements that have first
and the last element of the search key.
   Author:  Akomolafe, D.T.
as research work on development of algorithm for tracking and searching vehicle
system
*/
importjavax.swing.*;
importjava.awt.*;
importjava.awt.event.*;
import java.io.*;
public class sievegui extends JFrame
{     privateJTextFieldarrayField,keyField, inputField, outputField;
privateJLabelarrayLabel,keyLabel, inputLabel, outputLabel;
privateJButtonsrchButton, extButton, rsetButton ;
privateJPaneltopPanel, midPanel,butPanel;
public static JTextAreaaTextArea;
publicsievegui() throws IOException
     {      super("Sieve Algorithm");
           Container c=getContentPane();
c.setLayout(new FlowLayout());
topPanel = new JPanel();
topPanel.setLayout(new GridLayout(3,2));
aTextArea=new JTextArea(10,10);
arrayLabel=new JLabel("Enter Array Length");
arrayField=new JTextField(10);
keyLabel=new JLabel("Enter Search key");
keyField=new JTextField(10);
topPanel.add(arrayLabel);
topPanel.add(arrayField);
topPanel.add(keyLabel);
topPanel.add(keyField);
c.add(topPanel,BorderLayout.NORTH);
midPanel = new JPanel();
midPanel.setLayout(new GridLayout(5,1));
inputLabel=new JLabel("Enter input file path");
inputField=new JTextField(20);
outputLabel=new JLabel("Enter output file path");
outputField=new JTextField(20);
midPanel.add(inputLabel);
midPanel.add(inputField);
midPanel.add(outputLabel);
midPanel.add(outputField);
c.add(midPanel,BorderLayout.CENTER);
butPanel = new JPanel();
butPanel.setLayout(new GridLayout(1,3));
srchButton =new JButton("Search");
extButton =new JButton("Quit");
rsetButton =new JButton("Reset");
butPanel.add(srchButton);
butPanel.add(extButton);
butPanel.add(rsetButton);
c.add(butPanel,BorderLayout.SOUTH);
c.add(new JScrollPane(aTextArea),BorderLayout.EAST);
setSize(400,350);
setVisible(true);
srchButton.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent event)
{ try{ String arrayFieldContent=arrayField.getText();
                String keyFieldContent=keyField.getText();
                String inPathContent=inputField.getText();
                String outPathContent=outputField.getText();
if(arrayFieldContent.equals("")||keyFieldContent.equals("")
                  || inPathContent.equals("") ||outPathContent.equals(""))
                {     JOptionPane.showMessageDialog(null,"Please make entry in all
            fields","SIEVE",JOptionPane.INFORMATION_MESSAGE);
return;
                }
keyFieldContent=keyFieldContent.toString().toLowerCase();
int length=Integer.parseInt(arrayFieldContent);
aTextArea.append(String.valueOf(length));
JOptionPane.showMessageDialog(null,"lengthis"+length);
sieveBS.fls(length,keyFieldContent,inPathContent,outPathContent);
               }
catch (Exception e)
{  System.err.println("Caught IOException:"+e.getMessage());
               }
          }
        });
rsetButton.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent event)
{  arrayField.setText("");
keyField.setText("");
inputField.setText("");
outputField.setText("");
          }
        });
extButton.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent event)
{  System.exit(0);
          }
        });
      }
public static void main(String[] args) throws IOException
      {    sieveguiappl = new sievegui();
appl.addWindowListener(
newWindowAdapter(){
public void windowClosing(WindowEvent e)
               {   System.exit(0);
               }
          }
        );
   }
}
```

5. AVERAGE CASE ANALYSIS

The average case complexity of this algorithm tries to improve on the complexity of DEBS that had been already discussed, therefore it uses the complexity of DEBS; T = 9 $\log_2 n$-75 to find the complexity of the algorithm. In the algorithm, eleven (11) basic operations are to be executed in order to obtain the sub record and by considering the 11 basic operations that must be performed before the final result are turned to binary search.

The searchable elements s = m/26 (2.1)

$T = 9 \log_2 n - 75$ (2.2)

By considering the 11 basic operations, we can rewrite equation (2.2) as

$T = 9 \log_2 n - 75 + 11$ (2.3)
$T = 9 \log_2 n - 64$

Substituting (2.1) into 2.3, we have

$T = 9 \log_2 (m/26) - 64$ (2.4)

$= 9[\log_2^{m} - \log_2^{26}] = 64$
$= 9 \log_2^{m} - 9 * 4.5 - 64$
$= 9 \log_2^{m} - 40.5 - 64$
$= 9 \log_2^{m} - 104.5$
$= 9 \log_2^{n} - 104.5$ (2.5)

Equation (2.5) is the complexity of the algorithm which when compared to the complexity of DEBS in equation (2.2) shows substantial reduction in the number of comparisons required to find a key. Therefore, First and Last Character Sieve binary search algorithm is more efficient than the previous ones and most suitable to reduce time and cost of searching in road transportation database.

This shows that irrespective of the volume of the database, this algorithm will work faster and turn in result more than any other algorithms

6. CONCLUSION

As earlier noted, the future of road transportation will definitely be driven by technology because its operation and activities cannot be excluded from the wind of information technology blowing across all fields and activities. As road transportation system is being accorded its rightful position in patronage, there is the need for proper control and coordinated means by which law enforcement agents can apprehend road traffic offenders, have current statistics of vehicles plying the roads and the passengers therein. It is against these backdrops coupled with the increasing number of accidents on the highways that it is being recommended that the time spent searching for vehicle or driver details should be considerably reduced. It is therefore compelling to deploy a fast and efficient search algorithm hence the adoption and deployment of the developed algorithm.

In this research, an algorithm called first and last character sieve was proposed and developed. The algorithm used JAVA programming language for its codification as presented for the purpose of searching vehicle plate number database. The double character search was initially carried out before dissolving to binary search. The algorithm proposed to reduce the search list in two stages by a comparison of characters.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Akomolafe DT, et al. Enhancing road monitoring and safety through the use of geospatial technology" International Journal of Physical Sciences. 2009;4(5):343-348.
2. Adigun MO. The specification for a national police command and control system. COAN Conference Series. 1994;5:16-25.
3. Akinyokun OC. A methodology for the automatic design of database systems. Ph.D. Thesis, University of East Anglia, Norwich, England; 1984.
4. Akinyokun OC. The design and implementation of road safety relational database. International Journal of Information Technology for Development Published by the Oxford University Press, United Kingdom. 1987;2(2):147-156.
5. Miller MJ, Vucetic B, Barry L. Satellite communications mobile and fixed services. Norwell, M A Kluiver Academic Publishers; 1993.
6. Heath S. Effective PC networking: An imprint of butter worth. Heinemann Limited; 1993.
7. James Armold. Surface transportation and global positioning system improvement (1992): L5 and DGPS; 1992. webmaster@aero,org
8. Krammer G. An outline of an automatic road transportation system with radar guidance and precision navigation; 2001. m.dg.k@t-online.de
9. Janerdan JR, Li Q. GPCA: An efficient dimension reduction scheme for image comprehension and retrieval. In Prroc. 10th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining. 2004;354-363.
10. Tahira Mahboob, Fatima Akhtar, Moquaddus Asif, Nitasha Siddique, Bushra Sikandar, Survey and Analysis of Searching Algorithms, International Journal of Computer Science Issues (IJCSI); 2015.
11. Subbarayudu B, Lalitha Gayatri L, Sai Nidhi P. Ramesh R. Gangadhar Reddy, Kishor Kumar Reddy C. Comparative analysis on sorting and searching algorithms. International Journal of

Civil Engineering and Technology (IJCIET); 2017.

12. Robbi Rahim et al. J. Phys.: Conf. Ser. 2017;930 012007.
13. Ahmad Shoaib Zia. A Survey on Different Searching Algorithms International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056. 2020;07(01)Jan 2020
14. Ritu Sachdeva (Sharma), Sachin Gupta A Novel Algorithm for Enhancing Search Engine Optimization International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958. April 2019;8(4C).
15. Najma Sultana, Chandra, Sourabh, Paira, Smita, Alam, Sk. A brief study and analysis of different searching algorithms; 2017.
16. Wu L, Liu S, Zhao B, et al. The research of the application of the binary search algorithm of RFID system in the supermarket shopping information identification. J Wireless Com Network. 2019;27.
    DOI:https://doi.org/10.1186/s13638-019-1343-2
17. Hui Liu, Yonghui Cao. The research on search algorithms in the machine learning IJCSI International Journal of Computer Science. 2013;10(1):No 1, January 2013 ISSN (Print): 1694-0784 | ISSN (Online): 1694-0814
    Available:www.IJCSI.org
18. Roopa K, Reshma J. A comparative study of sorting and searching algorithms. International Research Journal of Engineering and Technology (IRJET); 2018.