Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 7, July 2014, pg.751 – 758

RESEARCH ARTICLE

A Comparison and Selection on Basic Type of Searching Algorithm in Data Structure

Kamlesh Kumar Pandey[1], Narendra Pradhan[2]

[1]Computer Science, IGNTU Amarkantak, India

[2] Computer Science, SS College of Education, Pendra Road, India

[1] kamleshamk@gmail.com; [2] pradhan.narendra627@gmail.com

Abstract— A lot of problems in different practical fields of Computer Science, Database Management System, Networks, Data Mining and Artificial intelligence. Searching is common fundamental operation and solve to searching problem in a different formats of these field. This research paper are presents the basic type of searching algorithms of data structure like linear search, binary search, and hash search. We have try to cover some technical aspects to this searching algorithm. This research is provides a detailed study of searching algorithms working process, select on Searching Algorithm according To Problem and compares them on the basis of different parameters like total number of comparison, type of data Structure, time complexity and space complexity.

Keywords — Sequential Search, Binary Search, Hashing, Hash function, Comparisons, Application, Time Complexity

I. INTRODUCTION

Sorting and Searching are two fundamental operations in a computer science. Sorting means arranging on data in given order such that increment or decrement. Searching means find out location or find out an element of a given item in a collection of item. Many data structures are used to store information but Arrays, linked lists and tree are basic data structures used to sorting and searching operation. Searching element is any type of a numerical data, alphabet, String, character data. A number of searching algorithms have been developed like that sequential search, binary search, tree search and hashing etc**.** Every searching algorithm depends on specific problem, property of data and algorithm complexity. This research paper gives brief introduction about searching algorithm, we define to which type of searching algorithm used to which type of Problem and we compare to different type of searching algorithm in a different important parameter like time complexity, space complexity, associated key, no of comparisons etc. A group of element is originated to a file or table. Each element is a called to record. Search technique is applying to a file or table and find to a specific record with location. Each record is associated to a key and this key is separated to different record. If key are store on start of a record so this type of key is called to internal key. In other case key are store on separate table including pointer to the record so this type of key is called external key. Every file or tables have a two set of key. First set are define a uniquely data this key is a called primary key and this set have an internal and external key. Second set are define none uniquely data this key is a called secondary key. We search to an element with location we need to first set after that need to second set.

The searching technique can be divided into two categories: 1. **Internal Searches: -** if file or table are kept in main memory this type of searching is called internal search. 2. **External Searches**: - if file or table is kept in auxiliary memory (hard disk, floppy, tape etc) this type of searching is called External Search.

II. WORKING PROCEDURE OF SEARCHING ALGORITHMS

1. Sequential Search:- Simplest form of searching technique is a sequence search or linear search. This search is applicable to a small size of array or linked list data structure. Find out on searching element in sequential manner on unordered list or ordered list. In this method searching process are started at begins to end, scans the elements one by one of the list from left to right until the searching record/element is found. If we taken on ordered list or table then time it given to fast searching and good efficient as a comparison on unordered list.

Algorithm: Here L is a location variable, I is a variable which value is a element position and A is Array/List[0-N]. SE is search element.

LINEAR _SEARCH (A,I,SE)

1. Initialize L=0
2. For I=0 to length[*A*] //scan on element start to end
   3. If (SE = =A [*I*]) //check on searching element
      4. L=L+1 // increment on location variable
      5. Return A[I] //print on finding element
   6. END If
   7. END For LOOP
   8. If L=0 then return "not find out element"
   9. Exit

Example:

Unordered Array A[1-6]	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	Output
	33	22	55	11	33	44	
Step:-1 SE= 33	33	22	55	11	33	44	Return A[0],L=1
Step:-2 SE=33	33	22	55	11	33	44	
Step:-3 SE=33	33	22	55	11	33	44	
Step:-4 SE=33	33	22	55	11	33	44	
Step:-5 SE=33	33	22	55	11	33	44	Return A[1],L=2
Step:-6 SE=33	33	22	55	11	66	44	

Example of Linear Search

2. Binary Search: - Another simplest form of searching technique and best efficient method is a binary search. It can be used if the table is stored as an array and mid size of array. If we use to linked list and insert/delete on data in linked list during searching time then problem are generated. This algorithm is also based on Divide-and-Conquer algorithm.

Divide: In sorted list is divided into two parts.

Conquer: We first compare to search/input element with the mid element of the list. If do not match then we check search element and mid element if search element is less then to mid element then go to first part/left part otherwise go to second/right part.

Combine: we apply to divide and conquer part whenever our search element is not found. When our search element is found then list are automatic combined and return.

Algorithm: Here L is a location variable, LOW is start index and HIGHT is last index element position of Sorted Array/List A[0-N]. SE is search element

BINARY _SEARCH (A,LOW,HIGHT,MID,SE)

```
1. Initialize LOW=0, HIGHT=N,L=0
2. While(LOW<=HIGHT)                    //scan on element start to end
        3. MID=(LOW+HIGHT)/2            //divide on list
          4. IF (SE==A[MID])            // check on searching element
              5. L=L+1                  // increment on location variable
              6. Return A[MID]          //print on finding element
           7. Else If(SE<A[MID])
                8. HIGHT=MID-1          // go to Left/first part
           9. Else
                10. LOW=MID+1           // go to right/second part
           11. END If Statement
11. END While LOO
12. If L=0 then return "not find out element"
13. Exit
```

Example:

Unordered Array A[1-6]	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	
	33	22	55	11	33	44	

Sort to Array A and result is Array A is	11	22	33	33	44	55	Operation

Step:-1 *LOW=0, HIGHT=5,* MID= (0+5)/2=2, *A [MID] =33, SE=33*

Check SE==A[MID]	11	22	33	33	44	55	Return A[MID],L=1

Step:-2 *LOW=MID+1, 2+1=3, HIGHT=5,* MID= (3+5)/2=4, *A [MID] =44, SE=33*

Check SE==A[MID]	11	22	33	33	44	55	

Step:-3 *LOW=3, HIGHT=MID-1, 4-1=3,* MID= (3+3)/2=3, *A [MID] =33, SE=33*

Check SE==A[MID]	11	22	33	33	44	55	Return A[MID],L=2

Example of Binary Search

3. Hashing: - Best searching Technique and most efficient method is a hash search. It searches technique whose search times can be independent of the number of entries in a table. Hashing has a worst-case behavior that is linear for finding a target value, but with some case, hashing can be dramatically fast in the average-case. With this approach, the position of a particular entry in the table is determined by the key for every record. This association is realized through the use of a hashing function. If H is a hash function and *K* is key, H(K) is called hash of key. So a hash function H(K) transforms a key into an address/hash table index position. if key is a non integer value then we convert to a integer value. Some hashing function is

1. Division Method (tablesize = prime):- choose a prime number M which are equal to a table size and K is a key. The hash function H is define as H(K)= K MOD M or H(K)= (K MOD M) +1. First function denotes the remainder when K is divided by M. second function is used when we want the hash table to range from 1 to M rather than from 0 to M-1.

2. Midsquare Method (tablesize = 2^n):- The multiplication method may be used for a HashTableSize that is a power of 2. In this method a key is multiplied by itself and the address is obtained by selecting a number from the middle of the square. The hash function H is define as H(K)= L is obtained $K^{2.}$

3. Folding Method:- in this method key is portioned into a equal number of parts. this part are added together and ignoring the final carry, to from an address. The hash function H is define as $H(K)=K_1+K_2+\ldots\ldots\ldots+K_n$

4. Variable string exclusive-or method (tablesize = 256):- To obtain a hash value in the range 0-255, all bytes in the set of character are exclusive-or together. However, in the process of doing each exclusive-or, a random component is introduced. The exclusive-or method has its basis in cryptography

Some time hash function given to map several key into the same address this situation is called Collision. And remove this collision using Buckets and Chaining method.

Algorithm: Many algorithms are use to hashing according to problem. Basically algorithm are divided in four categories Cyclic Redundancy Checks Algorithm, Checksum Algorithm, Cryptographic hash Algorithm(SECURE HASH ALGORITHM) and searching on data(Open addressing, Static and dynamic hashing).

Example: Hash Function

We given an example are find out an index number using an above hash function. A company have 3 employee is assigned three digit employee_id (External Key) and our Hash Table length is 10 from 0 to 9.

Employee id= 111, 113, 119

1. **Division Method:- Hear address are started at 0 to 9 then we Taken on prime number 7 which are close to Table size and We can choose to H(K)= (K MOD M)+1**

H(111)=(111 MOD 7)+1 =6+1 =7	H(113)=(113 MOD 7)+1 =1+1 =2	H(119)=(119 MOD 7)+1 =0+1 =1

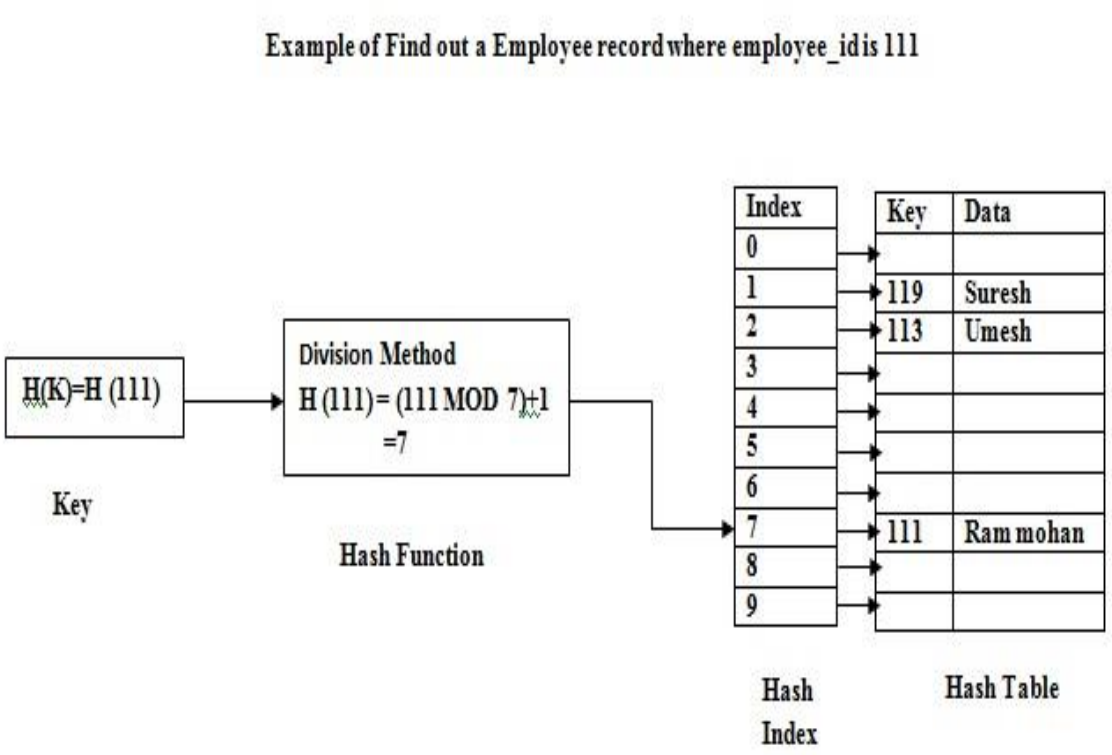
2. **Midsquare Method:-**

K	111	113	119
K^2	12321	12769	14161
H(K)	3	7	1

3. **Folding Method:-**

H(111)=1+1+1 =3	H(113)=1+1+3 =5	H(119)=1+1+9 =11 Remove to left digit of 11 =1

Example: Hashing Process



III. PROBLEM DEFINITION AND SEARCHING ALGORITHM

All searching algorithm are problem specific. In this section we described some problem and analysis which searching algorithm is more suitable for that problem

TABLE 1:-SEARCHING ALGORITHM ACCORDING TO PROBLRM

Problem Definition	Searching Algorithm
1. Find out a particular value in a small size of data origination 2. Resource Efficient Problem 3. Searching on unsorted data list 4. Find out data at a time, without jumping	**Sequential Search**
1. Used to access ordered data quickly when memory space is tight 2. Fast searching in large data origination and mid data origination 3. Find out path on mixed analog circuits 4. Maintains a contiguous subsequence of the starting sequence 5. where the target value is surely located	**Binary Search**
1. Constructing Indices 2. Storage for Object-Oriented Databases 3. Achieve Authentication and Data Integrity in a Cryptography 4. Fast searching in large data origination 5. How to Search documents on the internet for documents similar to a given one	**Hashing**

IV. COMPARITIVE STUDY OF ALL ALGORITHMS

TABLE 2:- COMPARISONS OF COMPARISON BASED SEARCHING TECHNIQUES ON VARIOUS PARAMETERS

Parameter	**Sequential Search**	**Binary Search**	**Hash Search**
Total no of comparison	(N+1)/2 for successful, N for unsuccessful	Each comparison reduces the number of possible candidates by a factor of 2. approximately comparison is $2*\log_2 N$	In double hashing tech. log(Tablesize+1)-0.5 for successful, (Tablesize+1)/2 for unsuccessful
Time Complexity			
1.Best Case	O(1)	O(1)	O(1)
2.Average Case	O(N)	O(log N)	O(1)
3.Worst Case	O(N)	O(log N)	O(N)
Space Complexity	O(N)	O(N)	O(N)
Associated key	Internal key	Internal key	External Key
Searching type	Internal Searches	Internal Searches	External Searches
Sorting	Do not needed, depending on user	Yes , use to any type of Sorting method	Do not needed, depending on user
Data Structure	Array, Linked List	Array	Array, Linked List and pointer
Simplicity	Easy	Average	Hard
Efficient	Low	Medium	High
Algorithm	Straightforward algorithm	Divide-and-Conquer	Cryptographic(SHA,MD5) and non Cryptographic(Dynamic search) algorithm
Implementation on programming	Easy	Average	Average implementation to data Structure/hard implementation to cryptography
Strategy	Scan on list start to end and match search element one by one	Divide on list match on mid element and search element. If the search element is less than the middle element, do again searching on the left half; otherwise, search the right half.	Location of storage is computed using the key(search element key) and a hash function, and direct go to desired element
Table and file are	Unordered/Ordered	Ordered	Unordered/Ordered

Size of table/file	Suitable for small size	Suitable for mid size/large size	Suitable for large size of
Java function on implementation	No specific function	binarySearch(array,search_key)	hashcode()

V. ADVANTAGE/DISADVANTAGE AND APPLICATION OF SEARCHING ALGORITHM

. In this section we cover some basic Advantage/Disadvantage and Application of searching algorithm which helpful to select on search algorithm according to our needed.

TABLE 3 ADVANTAGE/DISADVANTAGE OF SEARCHING TECHNIQUES

Searching Techniques	Advantage	Disadvantage
Sequential Search	• For smaller lists linear search may be faster because the speed of the simple increment. • Linear search very simplicity, resource efficient and memory efficient. • It can operate sorted and unsorted array and link list.	• Linear search technique is low efficient and slower than other searching algorithm. • Not suitable for large data set.
Binary Search	• The general moral is that for large lists binary search is very much faster than linear search. • This search technique given a searching approach on binary search tree and other tree.	• Binary search is not appropriate for linked list structures (no random access for the middle term) • Apply searching before we needed searching Algorithm. • Not suitable for inserted/deleted a data in a searching time as a compare to other searching algorithm.
Hash Search	• Searching is faster and more efficient in large list as compares to other searching algorithm. • More reliable and flexible of data retrieval then other Data Structure. • Various type of Hashing Algorithm is use to different filed of computer. Hashing use in network security for Authentication and Data Integrity	• Hash function and key are must be needed. • Large Memory size is required. • Some time hash function given to same index in hash index table in different key. Then time we needed a collision remove technique. • It is not efficient in a small Hash Table.

TABLE 4:- APPLICATION OF SEARCHING ALGORITHM

Sequential Search	Binary Search	Hash Search
• Index sequential search implementations • Recognition system • Analysis to Hacking system • Phonebook	• Binary search Tree/Tree search implementations • Microprocessor and many analog mixed signal circuits • 3D games	Application of Hashing • Construction on symbol table for compiler • Accessing tree or graph nodes

• Cable and Telephone Networks for Data Transmission • Address Mapping in computer network • Find out an object in a raster-scan system display. • Checking for spell-checking	• Number guessing game • Searching a telephone book • Packing rectangles • Prefix search • Client and server communication	by name • Maintain a transposition table in games • Dictionary lookups Application of Hash function • Construction on message authentication code(MAC) • Use to digital signature • Cryptography • Time stamping

VI. CONCLUSIONS

This paper discusses three basic type of searching algorithms and their example. Searching is the process to finding to location of The given data elements in the data structure. Hashing are faster for large lists as a compare Binary search and Binary search are faster for mid size lists or small list as a compare Linear search. We have compared the searching algorithm on the basis of various factors like complexity, data structure, searching type, application etc. we introduced which type of problem solve through which type of searching algorithm it help to selection on best searching algorithm.

REFERENCES

[1] Data structures using c and c++ by Yedidyah langsam,Aaron M. Tenenbaum second Indian printing (Prentice Hall of India private limited), New Delhi-110001
[2] An Introduction to Data Structures with Application by Jean-paul Tremblay Tata McGraw Hill
[3] Data Structures by Seymour Lipschutz and G A Vijayalakshmi Pai (Tata McGraw Hill companies), Indian adapted edition-2006,7 west patel nagar,New Delhi-110063
[4] Debadrita Roy, Arnab Kundu," A Comparative Analysis of Three Different Types of Searching Algorithms in Data Structure" International Journal of Advanced Research in Computer and Communication Engineering) Vol. 3, Issue 5, 2014
[5] Sorting and Searching Algorithms: A Cookbook by Thomas Niemann in Portland, Oregon
[6] NOTES ON HASHING by Jayakanth Srinivasan
[7] Knuth, D.E., 1988. The Art of programming-Sorting and Searching. 2nd Edn.,Addison Wesley, ISBN: 020103803X.
[8] Cormen, T.H. et al. Introduction to Algorithms. 2nd Edn., 2001. ISBN: 0262032937
[9] Asagba P. O, Osaghae E. O. and Ogheneovo, E. E.Department of Computer Science, University of Port Harcourt, Choba, Port Harcourt, Rivers State. Is Binary search technique faster than Linear search technique?
[10] Web help in wikipedia.org and stackoverflow.com
[11] Cryptography and Network Security Principles and Practice, Fifth Edition EDITION by William Stallings,Pearson
[12] Database System Concepts, Fourth Edition by Silberschatz−Korth−Sudarshan
[13] Data Structures and Algorithms Alfred V. Aho, John E. HopCroft and Jelfrey D. Ullman, Addison –Wesley, 1983
[14] Data Structures and Algorithm Analysis in C++ by Mark Allen Weiss.
[15] S.K. Shrivastava, Deepali Shrivastava,"Searching, Hashing and Storage Management", "Data Structures Through C In Depth", BPB Publications, Eighth Edition,ISBN:81-7656- 741-8

