How to Select a Searching Algorithm-A Comparative Study

Sobha Xavier P[#1], Jisna V A[*2], Bineesh M[#3]
[#1,2,3]Assistant Professor, Department of Computer Science, Jyothi Engineering College
Cheruthuruthy, Trissur, Kerala, India

Abstract *— Searching is a process of checking whether the required item is available or present in a certain data set. Searching can done based on the key element. A number of searching algorithms have been developed like that sequential search, binary search, tree search and hashing etc.*

Keywords *— Linear search , Binary search, pseudo code, Hashing, Collission, Probing.*

I. INTRODUCTION

Every searching algorithm depends on specific problem, property of data and algorithm complexity [2].Following are the three important searching algorithms:

- LINEAR SEARCH,
- BINARY SEARCH AND
- HASHING

This paper compares these three basic searching algorithms and gives a brief description about its applications, advantages and disadvantages etc.

II. LINEAR SEARCH

Linear search technique requires scanning and comparing one by one element of a list with a search key. So the time required by linear search is proportional to the total number of elements in a list. If an element to be searched is located nearer to first element it requires less time as opposed to an element which is located at the end of the list or an element does not exist at all in the list requires maximum time because it requires maximum comparisons.[4]

A function to search a given element X in A[].

INPUT: Given an array A[] of n elements

Table 1: Linear search - pseudo code

1.	**SET I=1**
2.	**CHECK WETHER A[I]=X IF YES GO TO 4**
3.	**IF I != N THEN SET I=I+1 AND GO TO 2 ELSE GO TO 5**
4.	**PRINT "ELEMENT FOUND AT:",I**
5.	**STOP**

III. BINARY SEARCH

Binary search algorithm requires all the elements in the given list in ordered manner. It first searches the list with its centre element by comparing it with a given value. If match is found algorithm terminates by returning its position and if no match is found it reduces the search space to the half of the original size. If search element is less than the middle element then only first half of the list is searched and in case if it is greater than the middle element then only second half of the list is searched in next iteration. In either of the case the list to be searched is reduced to half at each iteration of the search process. Following lists the binary search algorithm.[4]

A function to search a given element X in A[].

INPUT: Given a sorted array A[] of n elements

Table 2: Binary search - pseudo code

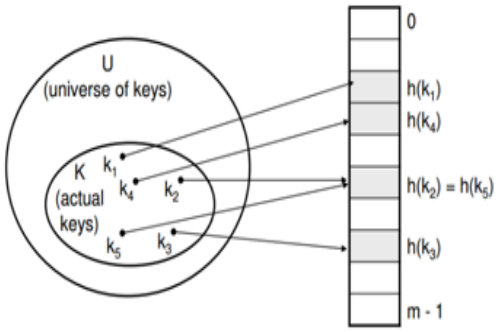
1.	**COMPARE X WITH THE MIDDLE ELEMENT.**
2.	**IF X MATCHES WITH MIDDLE ELEMENT, WE RETURN THE MID INDEX.**
3.	**ELSE IF X IS GREATER THAN THE MID ELEMENT, THEN X CAN ONLY LIE IN RIGHT HALF SUBARRAY AFTER THE MID ELEMENT. SO WE RECUR FOR RIGHT HALF.**
4.	**ELSE (X IS SMALLER) RECUR FOR THE LEFT HALF.**

IV. HASHING

Hashing is the technique used for performing almost constant time search in case of insertion, deletion and find operation. The essence of hashing is to facilitate the next level searching method when compared with the linear or binary search. It is the process of mapping large amount of data item to a smaller table with the help of a hashing function. The advantage of this searching method is its efficiency to hand vast amount of data items in a given collection (i.e. collection size). [3]

A perfect Hash function is a function which when applied to all the members of the set of items to be stored in a hash table, produces a unique set of integers within suitable range. A good hash function minimize collisions by spreading the elements uniformly throughout the table.[5]

Fig.1: Illustration of Hashing



.

IV- A. HASHING METHODS:

The main advantage of hashing is the time required to search any element is constant and is proportional to the time required by hashing function.[2]

Following methods are using for hash key generation:

1) *Truncation Method*

2) *Mid Square Method*

3) *Folding Method*

4) *Modular Method*

5) *ASCII Method.*

1) Truncation Method:

In this method ignore a part of the key and use the remaining part directly as the index

2) Mid Square Method:

In Mid-Square method, the key element is multiplied by itself to yield the square of the given key. It generates random sequences of hash keys, which are generally key dependent.

3) Folding Method:

Partition the key into several parts and combine the parts in a convenient way (often addition or multiplication) to obtain the index.

4) The Division Method:

If the size of the hash table is j then key k= i modulo division j

Key: 45; Size of Hash table 10

45%10

⇩

5

5) A SCII meth od:

The procedure to find out hash key using ASCII method is as follows:

H(k)=∑(ASCII values of each characters in the key)

In this method we can find out hash key for both integer and string data type input.

Key: ‘set’

115+101+116

⇩

332

Key: 4589

45892=210**58**921

⇩

58

Key: 525371726

⇩

726

Key: 458921

45+89+21

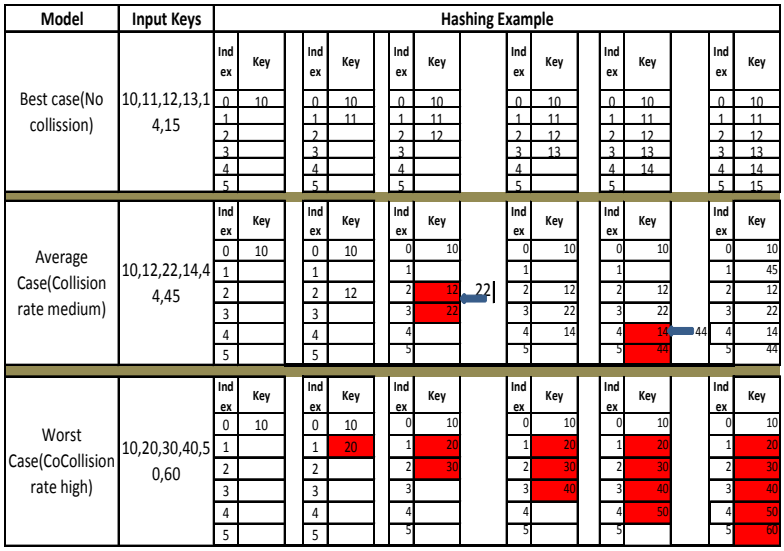
⇩

155

IV .B. COLLISIONS IN HASHING:

Suppose if the user want to add a new key value to ith index in the hash table and that location is already filled with another key then that situation is called as collision. Table 3 shows hashing example based on division method. Here we compared 3 cases: best case, average case and worst case based on different input combinations. Red colour in the cells represents collision situation. Here collision is resolved by occupying the key value to the very next vacant position(probing). But the disadvantage of this method is that keys tend to ‘CLUSTER’ i.e., appear next to one another. Such clustering increases the average search time for a key. Two collision resolution techniques are **open addressing and chaining**.

Table 3: Hashing Example – Division method (Size of Hash table 5)



As shown in the above table division method is simple to calculate but the collision rate is depends on the input given.So a good programmer has to select appropriate method based on the situation. Table 4 give s a comparison of different hashing methods based on three parameters: Hash key Calculation over head, Collision rate, Data type of the given input key.

Table 4: comparison of hashing methods

V. COMPARITISON OF SEARCHING ALGORITHMS:

In this section we cover Time complexity and Space complexity analysis of three searching methods explained in the previous sections. In terms of time complexity we can say that hashing is faster than other two methods.

Method no.	Method Name	Hash key Calculation over head	Collision rate	Input data type
1	Truncation Method	Low	Low	Integer only
2	Mid Square Method	Higher than method 1 and 4	Low	Integer only
3	Folding Method	Higher than method 1 and 4	Low	Integer only
4	Division Method	Low	Higher than other methods	Integer only
5	ASCII method	Higher than method 1 and 4	Low	Both strings and integers

Table 5: Comparison of searching algorithms.

VI. ADVANTAGE,DISADVANTAGE AND APPLICATION OF SEARCHING ALGORITHM [2]

Searching Method	Time Complexity			Space Complexity
	Best Case	Average Case	Worst Case	
Linear Search	O(1)	**O(N)**	O(N)	O(N)
Binary Search	O(1)	**O(Log n)**	O(Log n)	O(N)
Hashing	O(1)	**O(1)**	O(N)	O(N)

Here we cover some basic Advantages, Disadvantages and Applications of searching algorithms which helpful to select on search algorithm according to our need.

Table 6: advantage/disadvantage/applications of searching techniques

Searching Techniques	Advantage	Disadvantage	Applications
Linear Search	Simple Not need ordered /sorted input array	Linear search technique is low efficient and slower than other searching algorithm. Not suitable for large data set.	To search in linked list structures To search in data list
Binary Search	Faster than sequential searching, Less number of comparisons.	Binary search is not appropriate for linked list structures (no random access for the middle term) Not suitable for inserted/deleted a data in a searching time as a compare to other searching algorithm.	Binary search Tree/Tree search implementations
Hash Search	Faster than other two methods.	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.	can be used for on-line spelling checkers Game playing programs use hash tables to store seen positions, thereby saving computation time if the position is encountered again

VII. CONCLUSION

This paper compares three basic searching algorithms and gives a brief description about its applications, advantages and disadvantages etc. . Every searching algorithm depends on specific problem, property of data and algorithm complexity [2] .The task is to find a particular data in the list in the shortest possible time. So the users can go through the application, advantage and disadvantage of each method and can select the best method as per their requirement.

REFERENCES:

[1] *Classic Data Structures Samanta (Author), Debasis (Author).*

[2] A Comparison and Selection on Basic Type of Searching Algorithm in Data Structure Kamlesh Kumar Pandey1, Narendra Pradhan2IJCSMC, Vol. 3, Issue. 7, July 2014

[3] Designing And Implementing Data Structure With Search Algorithm To Search Any Element From A Given List Having Constant Time Complexitydr. Vimal P. Parmar ,Dr. CK Kumbharana, E-ISSN No : 2454-9916 Volume : 3 | Issue : 1 | Jan 2017ternIERJational Education & Research Journal [IERJ]

[4] Data Structures Using C ,Udit Agarwal

[5] http://www.geeksforgeeks.org/

[6] An Introduction to Data Structures with Application by Jean-paul Tremblay Tata McGraw Hill

[7] 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