Journal of Information & Communication Technology
Vol. 1, No. 2, (Fall 2007) 49-58

Multiple Values Search Algorithm

Muhammad Sharif *
Aman Ullah Khan *
COMSATS Institute of Information Technology Islamabad, Pakistan

ABSTRACT

This paper introduces a fast searching technique for multiple values to be searched which is even faster than the binary search. In this technique, a sorted list of values can be searched from another sorted list very efficiently as compared to other existing searching techniques specifically binary search algorithm. The overall time for sorting the values to be searched and searching is less than the time taken by the binary search. In this technique, the size of targeting list is reduced for every next value to be searched. As a result the searching time of remaining values to be searched is reduced, while in binary search the size of target list remains same for every value to be searched. In other words, we can say that the targeting list is traversed completely each time for every value to be searched from beginning to end. So binary search takes more time than the Multiple Values Search Algorithm (MVSA)..

INSPEC Classification : C7250R, C4240C

Keywords : Multiple Values, Binary Search, Complexity, Data Warehouse

1) INTRODUCTION

There are many searching techniques available while Binary Search is considered as an efficient and faster one. It starts searching for every value starting from top, so it performs many comparisons but consumes a lot of time. When number of values is to be searched, this searching time can be reduced by avoiding searching every time for each value from the top. Multiple Values Search Algorithm (MVSA) is based on this concept. One thing is common in both searching techniques that target list should be sorted but the difference is that the values being searched should also be sorted for the Multiple Values Search Technique.

This paper is organized as follows; Section II presents a short description of the proposed solution. Section III presents the algorithm used for the proposed solution and also tabular examples. In Section IV the complexity of the algorithm is presented with asymptotic, time and statistical analysis. Section V ends with concluding remarks.

* The material presented by the authors does not necessarily portray the viewpoint of the editors and the management of the Institute of Business and Technology (BIZTEK) or COMSATS Institute of Information Technology Islamabad, Pakistan.

* Muhammad Sharif : muhammadsharifmalik@yahoo.com
* Aman Ullah Khan : auk_pk@yahoo.com

2) DESCRIPTION OF THE PROPOSED ALGORITHM

In this algorithm, searching will start from top of the target list for first value. If value exists in the list, the location of the found value will be stored in a location pointer (LP). For next value the searching will start from that pointer until the value is found or target list is finished? If it is also found then the LP is updated by the location of the found value. This process continues for all values and searching time gradually decreases for the upcoming values because the size of the target list is decreasing each time for every next coming value.

3) ALGORITHM

Main steps of Multi Search are

```
LP: = 0

found: = false
low: = LP
high: = size-1
while high >= low
 begin
    mid: = (low + high)/2
     If key < mid then
        list: = first half of list
     else if key > mid then
         list: = second half of list
     else if key = mid then
          LP:= mid
          return found and get next value to  search
  end
LP: = low
return not found and get next value to search
```

Table 1
Binary Search

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Data	23	27	29	32	34	41	46	47	49	52	55	68	71	74	77	
For52																
1st Iterations	F							M								L
2nd Iterations									F			M				L
3rd Iterations									F	M	L					
For77																
4th Iterations	F							M								L
5th Iterations									F			M				L
6th Iterations													F	M		L
7th Iterations															FM	L
For55																
8th Iterations	F							M								L
9th Iterations									F			M				L
11th Iterations									F	M	L					
12th Iterations											FML					

Table 2
Multi Search

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Data	23	27	29	32	34	41	46	47	49	52	55	68	71	74	77	
For52																
1st Iterations	F							M								L
2nd Iterations									F			M				L
3rd Iterations									F	M	L					
For55																
4th Iterations										F			M			L
5th Iterations										F	M	L				
For77																
6th Iterations											F		M			L
7th Iterations														F	M	L

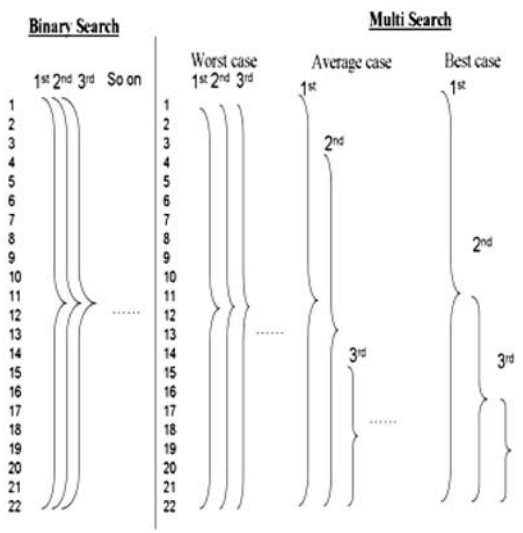
In tables 1 & 2, the working scenario of both binary and multiple search algorithms are shown with example. First row of the table, starting from the top represent the array indices while second row show the data stored at array locations. Three pointers First (F), Last (L) and Middle (M) are used. Let us suppose that three elements 52, 77 & 55 are to be searched. Using binary search the elements are searched in 12 iterations as shown in table 1.

To find these same elements by using multiple values searching technique, first sort them as 52, 55 and 77. By this technique the elements are searched in just 7 iterations as compared to binary search technique which takes 12 iterations on same data as shown above.

4) COMPLEXITY OF ALGORITHM

It has been observed that if searched values exist at the start of the target list then it will be the worst case and searching time nearly equals to that of simple binary search, result still better than binary search. If the searched value exists in the middle of target list every time then it will be the best case as shown in Fig 1.

Fig 1
Binary Vs Multi Search Comparison



analyzed this technique by using three types of analysis:-

- Asymptotic analysis
- Time analysis
- Statistical analysis

A. Asymptotic Analysis

Average case

Let n be the total no of items in the target list and r be the no of items to be searched from the target list.

Complexity of binary search algorithm is $\log_2 n$; where data contains n-values. If we have to locate r data items from n, each time we have to compare approximately $\log_2 n$ items. So for r data items to be searched **$r\log_2$ (n)** comparisons are required for binary search.

When r-data items are to be searched by using multiple values search algorithm, for 1st value the searching time will be the same as that of binary search.

For 1st value to be searched

No. of items = n

No. of comparisons =$\log_2 n$

2nd time search will start from the location of 1st searched item and will have a list of reduce data items i.e. n will be reduced by a fraction of itself.

Now reduced n can be calculated using the location of 1st searched element as

Reduced no. of data items

= (n (n- location of 1st value searched)/n)

$c_1 n$ = n (n- location of 1st value searched)/n,

Where c1 is reducing factor when 1st value is searched

=> c_1 = (n- location of 1st value)/n

For 2nd value to be searched

No. of items = $c_1 n$
No of comparisons = $\log_2 c_1 n$
It should be noted that $0<c_1<=1$

Because c_1 cannot be equal to zero, for $c_1=0$
n-location of 1st value +1= 0
=> location of 1st value= n+1, but list contains n items
3rd time n will be reduced again and can be calculated as $c_2 n$ where

c_2 = (n-location of 2nd value)/n

For 3rd value to be searched

No. of items = $c_2 n$
No of comparisons = $\log_2 c_2 n$
And $0<c<=1$

Continuing this process, the r^{th} data item is to be searched
For r^{th} value to be searched

No. of items = $c_{r-1}\ n$
No of comparisons = $\log_2 c_{r-1}\ n$
And $0<c<=1$
Thus total numbers of comparisons are
$f(n) = \log_2 n + \log_2 c_1\ n + \ldots.. + \text{Log}_2\ c_{r-1} n$

Analysis

$= \log_2 n\ (c_1 n)\ (c_2 n) \ldots.\ (c_{r-1}\ n)$

$= \log_2\ n^r\ c_1 c_2 \ldots\ldots c_{r-1}$

$= r\log_2 n + \log c_1 c_2 \ldots c_{r-1}$

where $0< c_1, c_2, c_3 \ldots cr_{-1} <=1$

As all c's are positive numbers less than or equal to 1
$=> x = c_1 * c_2 * \ldots.. * c_{r-1}$, $0<x<=1$

$=> \log_2 x <=0$ because the log of numbers between 0 and 1 is always negative and zero when number is equal to 1.
$=> f(n) = r\log_2 n$ for multiple values search
=>Number of comparisons using multiple values search is always less than or equal to binary search. Comparisons are equal only when all c's are 1 and it is possible when either no data value is located or only last is located.

Worst case

Worst case can occur only when no data item is located. In this case of n will not reduce and then all c will be 1 so f (n) becomes

$f(n) = n\log_2 n + \log c_1 c_2 \ldots c_{r-1}$

$f(n) = n\log_2 n + \log (1*1*1*\ldots..*1)$

$f(n) = n\log_2 n + \log_2 1$

$f(n) = r\log_2 n + 0$

$f(n) = r\log_2 n$

Best case

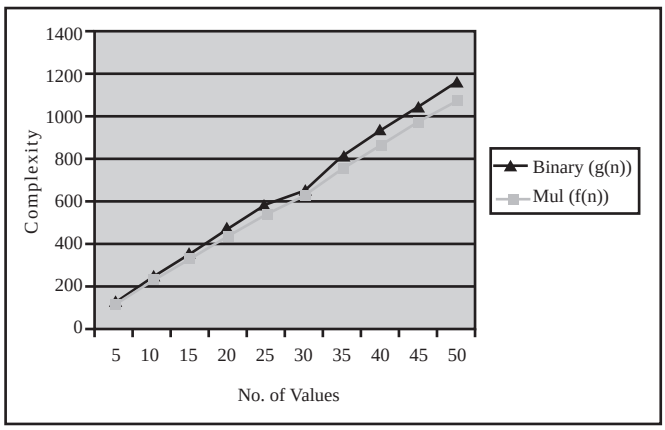
Best case occurs when at each comparison a value is found, in this case

f (n) = r = 1+1+1+…. +1 (r-times)

Table 3
Complexity comparison of binary vs. multiple values searches. Total List values = 10000000

No.of Values	Comparisons		Complexity	
	Binary	**Multi**	**Binary (g(n))**	**Multi (f(n))**
5	109	105	116.26748	108.234016
10	226	231	232.534967	218.762603
15	334	311	348.80245	321.812137
20	445	407	465.06993	431.077751
25	557	518	581.337417	539.993376
30	647	605	647.351403	625.493012
35	778	725	813.872383	752.849707
40	893	834	930.139867	862.250953
45	1001	939	1046.40735	969.804985
50	1114	1044	1162.67483	1079.78546

Fig 2
Complexity comparison of binary vs. multiple values searches.



Big O notation

In big O notation we find that f (n) = O (rlog2n + log2x)

Table 3 shows that the complexity of multiple values search is better than binary search

B. Time Analysis

A list of 10000000 values is generated by the random number and the different number of values is searched from the same list by using binary search and multiple values search algorithm. Following are the results of these two algorithms in Table 4 and table 5.

Table 4 Result-1
Total List values = 10000000

	Binary		Sorting	Multi		Total
No of Values	**Comp***	**Time**	**Time**	**Comp**	**Time**	**Sor+Mul***
100	2228	419	23	1404	317	340
200	4449	639	56	2672	485	541
400	8922	1154	113	5383	896	1009
800	17837	2200	245	10543	1584	1829
1600	35676	3978	514	21024	3166	3680

* Comp = Comparisons
* Sot+ Mul= Sum of Sorting Time + Searching Time of Multiple Values Search
*Time in microseconds

Fig 3
Result-1

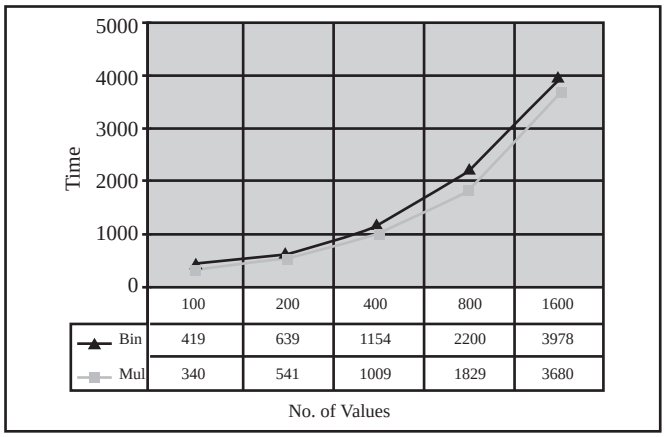


Table 5
Result-2

	Binary		Sorting	Multi		Total
No of Values	**Comp**	**Time**	**Time**	**Comp**	**Time**	**Sor+Mul**
500	11174	1517	145	6709	1287	1432
1000	22286	2373	310	13219	1915	2225
2000	44611	4624	661	26576	3561	4222
4000	89261	8484	1414	52959	6655	8069
8000	178744	18738	2958	104812	15299	18257

Fig 4
Result-2

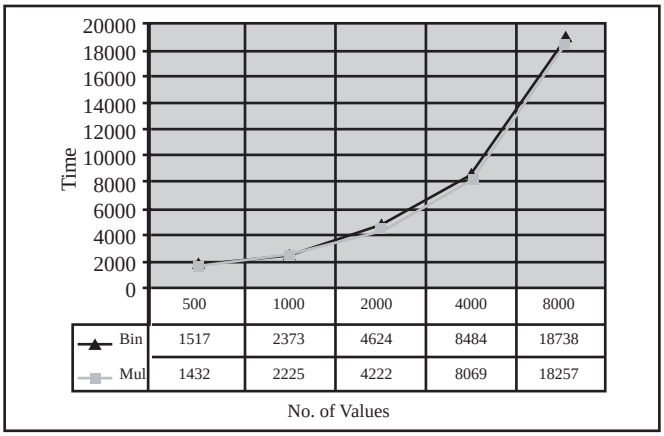


Table 6
Result-3

	Binary		Sorting	Multi		Total
No of Values	**Comp**	**Time**	**Time**	**Comp**	**Time**	**Sor+Mul**
10000	223499	23450	3732	131670	17093	20825
20000	447427	44471	8114	264492	33582	41696
40000	894659	87869	21081	529467	59674	80755
80000	1788717	168311	42637	1056468	110063	152700
160000	3577892	337166	91014	2114074	229472	320486

Fig 5
Result-3

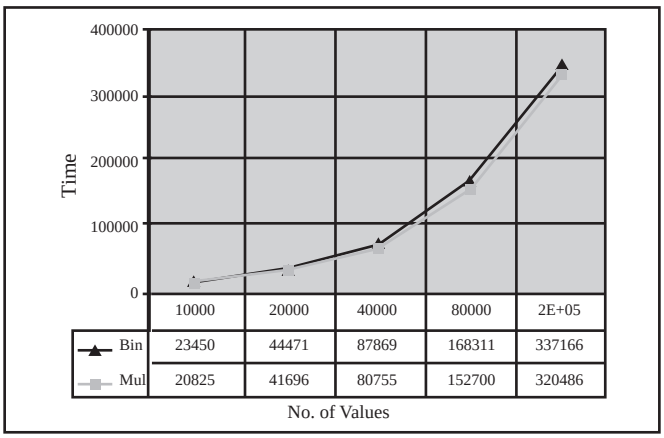
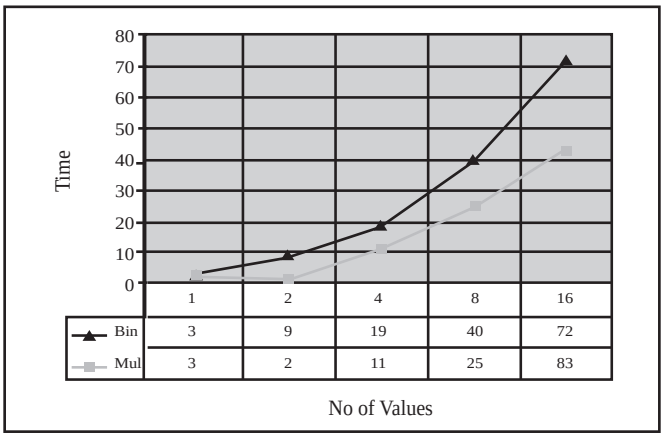


Table 7
Result-4

	Binary		Sorting	Multi		Total
No of Values	**Comp**	**Time**	**Time**	**Comp**	**Time**	**Sor+Mul**
1	0	3	2	0	1	3
2	22	9	1	0	1	2
4	69	19	4	44	7	11
8	160	40	4	114	21	25
16	338	72	5	192	38	43

Fig 6
Result-4



C. Statistical Analysis

The statistical analysis can be done with the help of standard deviation method by using the formula

$\sigma = s\ ((3x^2 / n) + (3x/n)^2)$

The standard deviation of the above results from table 4 to 7 is as follows:-

Table 8
Standard Deviation Results

Results	**Binary Search**	**Multiple Values Search**
Result -1	1457.88	1356.92
Result -2	7017.70	6878.53
Result -3	127282.39	121158.05
Result -4	28.04	17.30

The data with less standard deviation is better. In each of the above results the multiple values search is more reliable than binary search because it attains less time as compare to binary search.

5) CONCLUSION

It is clear from the above results that the multiple values search algorithm is better as compared to the binary search algorithm and it is useful for multiple values to be searched from huge amount of data like data warehouse.

REFERENCES

www.cs.ecu.edu/~hochberg/fall2002/class14/class14.pdf
www.cs.rutgers.edu/~ryder/111/111sp98/1998splectures/complexity. PDF
www.csis.gvsu.edu/~grissom/163/Analysis (ch5).ppt
Binary search algorithm, http://en.wikipedia.org/wiki/Binary_search
Big O notation, http://en.wikipedia.org/wiki/Big_O_notation
http://www.nist.gov/dads/HTML/binarySearch.htm