Quadratic Search: A New and Fast Searching Algorithm (An extension of classical Binary search strategy)

Parveen Kumar

Department of Computer Science & Engineering
National Institute of Technology, Hamirpur
Himachal Pradesh, India

ABSTRACT-

It consider the problem of computing efficient strategies for Searching. As a generalization of the classical binary search for ordered array, suppose one wishes to find a specific element in an ordered array by making comparisons, where each comparison indicates the endpoint closer to the desired element. Given the likelihood of each element being the one searched, the objective is to compute a search strategy that minimizes the expected number of comparisons. Practical applications of this problem include file system synchronization and software testing. Here this paper presents a new algorithm which is 2 times faster than classical Binary Search. This represents a significant improvement over previous Binary Search Algorithm having worst case complexity O(log n).This New algorithm Quadratic Search having worst case complexity O(log n/2). It is shown in this paper that by splitting and checking the ordered array in a way that the searching speed of a specific element is twice than the classical Binary search.

Keywords: Binary Search, Complexity analysis

1. INTRODUCTION

Searching in ordered structures is a fundamental problem in theoretical computer science. In one of its most basic variants, the objective is to find a special element of a totally ordered set by making queries which iteratively narrow the possible locations of the desired element. A search strategy is a procedure that decides the next query to be posed based on the outcome of previous queries.[1]

Binary search method is a well-known standard technique to search a linearly ordered array for a given key. Concrete algorithms of binary search are found in many books such as [2], [3], etc. The binary search method used most commonly tries always to check the middle key of the remaining continuous part of the ordered array. Since binary search itself does not necessarily imply adopting equal splitting criterion, we call the usual method as above the ordinary binary search [4].

In this paper, A new algorithm is introduced having worst case complexity O(**log**n/2) by using a slight modification in already existing binary search algorithm. In Section 2, A little introduction is discussed about Binary search and work which is already done on this topic. Section 3, Describe the Proposed algorithm and its implementation steps. In Section 4, Comparison between performance of Binary search and Quadratic search. In Section 5,Test results and performance graph, and finally the conclusion.

2. BINARY SEARCH ALGORITHM

However, if the items are placed in an array and sort them in either ascending or descending order on the key first, then a better performance can be obtained in searching with an algorithm called **binary search**.

In binary search, first compare the key with the item in the middle position of the array. If there's a match, Then return immediately. If the key is less than the middle key, then the item sought must lie in the lower half of the array; if it's greater then the item sought must lie in the upper half of the array. So repeat the procedure on the lower (or upper) half of the array.

Binary search requires a more complex program than the original search and thus for small ***n*** it may run slower than the simple linear search. However, for large ***n***,

$$\lim_{n\to\infty}\left(\frac{logn}{n}\right)$$

Thus at large ***n***, **log *n*** is much smaller than ***n***, consequently an ***O*(log *n*)** algorithm is much faster than an ***O(n)*** one.

3. QUADRATIC SEARCH ALGORITHM

In the proposed algorithm, if items are placed in an array and sort them in either ascending or descending order on the key first, then a much better performance can obtained in searching then binary search by using **Quadratic Algorithm.**

In Quadratic Algorithm, first calculate the middle element,$1/4^{th}$ element and $3/4^{th}$ element . then compare the key with the item in the middle ,$1/4^{th}$ and $3/4^{th}$ positions of the array. If there's a match, Then return immediately. If the key is not matched then it check with the certain conditions which will discussed in algorithm :

```
MID = (FIRST+LAST)/2;
P1 = FIRST+(LAST-FIRST)/4;
P2 = FIRST+(LAST-FIRST)3/4;
```

Now, A concrete algorithm for Quadratic Search which is slightly different from the conventional binary search.

3.1 ALGORITHM :

```
Quadratic_Search( int A[ ],int KEY,int FIRST,int LAST )
  {
      int P1=0,P2=0,MID=0;
      FIRST = 1;
      LAST = n;

      While( FIRST<=LAST )
       {
          MID = (FIRST+LAST)/2;
          P1 = FIRST+(LAST-FIRST)/4;
          P2 = FIRST+(LAST-FIRST)3/4;
```

```
            If(KEY==A[MID]    ||    KEY==A[P1]    ||
KEY==A[P2])
                  return element found;
               Elseif(KEY<A[MID] && KEY<A[P1])
                 LAST=P1-1;
               Elseif(KEY<A[MID] && KEY>A[P1])
                 FIRST=P1+1;
                 LAST=MID-1;
               Elseif(KEY>A[MID] && KEY>A[P2])
                 FIRST=P2+1;
               Elseif(KEY>A[MID] && KEY<A[P2])
                 FIRST=MID+1;
                 LAST=P2-1;
      }
         Return element not found
  }
```

The given Algorithm will check 4 conditions after calculating MID,P1,P2 when the key is not found at the middle,P1 and P2 :

Condition 1 : IF KEY is less than Middle and also Less than P1.Then

(P1-1) will become the LAST and the procedure will repeated.

Condition 2 : IF KEY is less than Middle and greater than P1.Then

(P1+1) will become the FIRST and (MID-1) will become the LAST and the procedure will repeated.

Condition 3 : IF KEY is greater than Middle and also greater than

P2.Then (P2+1) will become the FIRST and the procedure will repeated.

Condition 4 : IF KEY is greater than Middle and Less than P2.Then

(MID+1) will become the FIRST and (P2-1) will become the LAST and the procedure will repeated.

4. COMPARISON BETWEEN THE PERFORMANCE OF BINARY SEARCH AND QUADRATIC SEARCH:

This Section will compare the performance between the Classical Binary Search and the Quadratic Search. The Splitting criteria is Represented Diagrammatically for both algorithms and also a performance Graph is plotted for both of Algorithms. The graph is plotted between Time "f(n)" and number of elements "n". As the number of elements are increased the time will decrease and it will be clear to distinguish the difference of performance between the linear search and Binary search and the Quadratic Search technique. And Lastly, The comparison graph of Binary Search and Quadratic search is Plotted on a single Graph to compare the performance of both search technique. The comparison graph will shows the difference, and the difference is that, Quadratic search is 2 times faster than the Classical Binary Search Technique.

4.1 ANALYSIS OF BINARY SEARCH:

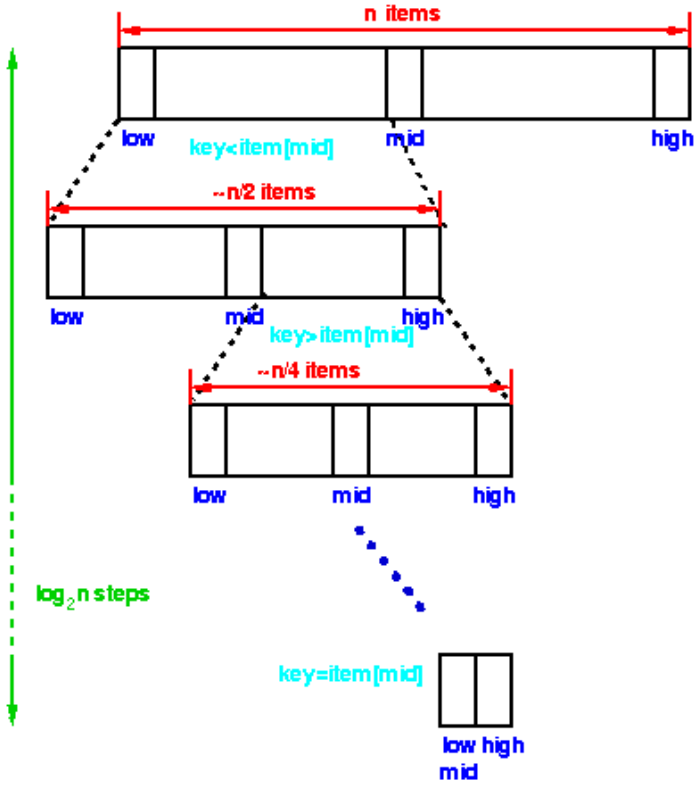

Figure 1. Analysis of Binary search

Each step of the algorithm divides the block of items being searched in half. A set of ***n*** items can divided in half at most $\log_2 n$ times. Thus the running time of a binary search is proportional to **log *n*** and can be said as ***O*(log *n*)** algorithm.

4.2 PERFORMANCE GRAPH OF BINARY SEARCH :

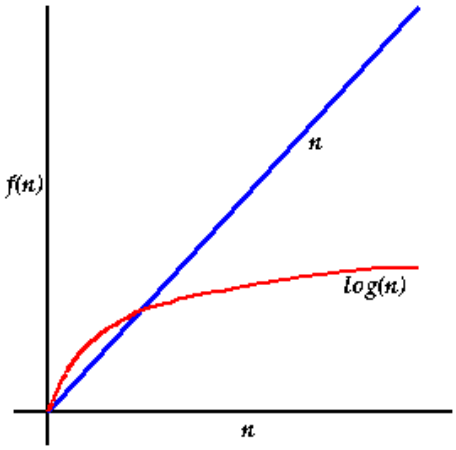

Figure 2. Plot of *n* and log *n* vs *n* .

4.3 ANALYSIS OF QUADRATIC SEARCH:

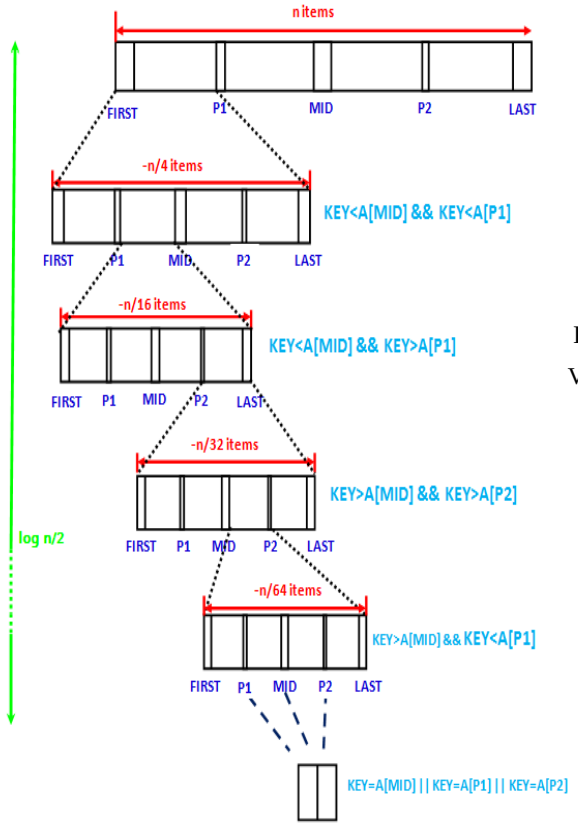

Figure 3. Analysis of Quadratic Search

Each step of the algorithm divides the block of items being searched in 4 Parts.A set of ***n*** items can be divided in 4 parts at most **log *n/2*** times. Thus the running time of a Quadratic search is proportional to **log *n/2*** and can be said as ***O*(log *n/2)*** algorithm.

4.4 PERFORMANCE GRAPH OF QUADRATIC SEARCH:

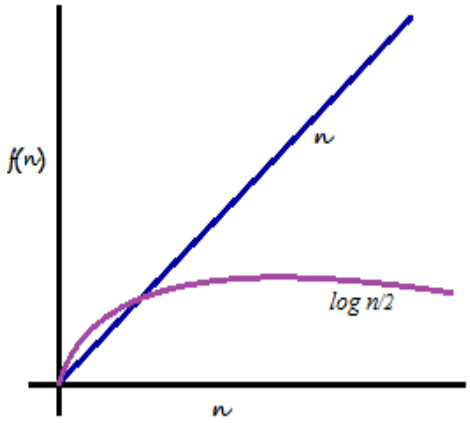

Figure 4. Plot of *n* and log *n/2 vs n* .

5. EXPERIMENTAL TEST AND RESULT:

An ordered array having 5000 elements is used and the values stored in the array are uniformly distributed with a difference of 7 Then both algorithms are applied Binary search as well as Quadratic Search for searching of same elements who take worst time of algorithm to become found and the result of both algorithms are shown below, also the no. of steps algorithms take are mentioned in comparison with the respected values of comparison.

ARRAY :

Figure 5 Array used in Algorithm

Index →	1	2	3		2499	2500	2501		4998	4999	5000
Value →	7	14	21	….	17493	17500	17507	….	34986	34993	35000

Table 1 . Find Key =14

Search Key = 14		
Steps	***Binary***	***Quadratic***
Initial	low= 1 mid=2500 high=5000	low= 1 p1= 1250 mid= 2500 p2=3750 high=5000
1	low= 1 mid=1250 high=2499	low= 1 p1= 313 mid= 625 p2=937 high=1249
2	low= 1 mid=625 high=1249	low= 1 p1= 78 mid= 156 p2=234 high=312
3	low= 1 mid=312 high=624	low= 1 p1= 20 mid= 39 p2=58 high=77
4	low= 1 mid=156 high=311	low= 1 p1= 5 mid= 10 p2=14 high=19
5	low= 1 mid=78 high=155	low= 1 p1= 1 mid= 2 p2=3 high=4
6	low= 1 mid=39 high=77	
7	low= 1 mid=19 high=38	
8	low= 1 mid=9 high=18	
9	low= 1 mid=4 high=8	
10	low= 1 mid=2 high=3	
Total:	**10**	**5**

Table 2 . Find Key = 34993

Search Key = 34993		
Steps	***Binary***	***Quadratic***
Initial	low= 1 mid=2500 high=5000	low= 1 p1= 1250 mid= 2500 p2=3750 high=5000
1	low= 2501 mid=3750 high=5000	low= 3751 p1= 4063 mid= 4375 p2=4687 high=5000
2	low= 3751 mid=4375 high=5000	low= 4688 p1= 4766 mid= 4844 p2=4922 high=5000
3	low= 4376 mid=4688 high=5000	low= 4923 p1= 4942 mid= 4961 p2=4980 high=5000
4	low= 4689 mid=4844 high=5000	low= 4981 p1= 4985 mid= 4990 p2=4995 high=5000
5	low= 4845 mid=4922 high=5000	low= 4996 p1= 4997 mid= 4998 p2=4999 high=5000
6	low= 4923 mid=4961 high=5000	
7	low= 4962 mid=4981 high=5000	
8	low= 4982 mid=4991 high=5000	
9	low= 4992 mid=4996 high=5000	
10	low= 4997 mid=4998 high=5000	
11	low= 4999 mid=4999	

	high=5000	
Total:	11	5

Table 3 . Find Key =34994

Search Key = 34995		
Steps	***Binary***	***Quadratic***
Initial	low= 1 mid=2500 high=5000	low= 1 p1= 1250 mid= 2500 p2=3750 high=5000
1	low=2501 mid=375 high=5000	low= 3751 p1=4063 mid=4375 p2=4687 high=5000
2	low=3751 mid=4375 high=5000	low= 4688 p1=4766 mid=4844 p2=4922 high=5000
3	low= 4376 mid=4688 high=5000	low= 4923 p1=4942mid= 4961 p2=4980 high=5000
4	low= 4689 mid=4844 high=5000	low= 4981 p1=4985mid= 4990 p2=4995 high=5000
5	low= 4845 mid=4922 high=5000	low=4996 p1=4997 mid= 4998 p2=4999 high=5000
6	low= 4923 mid=4961 high=5000	low=5000 p1=5000 mid=5000 p2=5000 high=5000
7	low= 4962 mid=4981 high=5000	
8	low= 4982 mid=4991 high=5000	
9	low= 4992 mid=4996 high=5000	
10	low= 4997 mid=4998 high=5000	
11	low= 4999 mid=4999 high=5000	
12	low= 5000 mid=5000 high=5000	
13	low= 5001 mid=5000 high=5000	
	Element not found	Element not Found
Total	13	6

5.1 COMPARISON GRAPH OF BS AND QS:

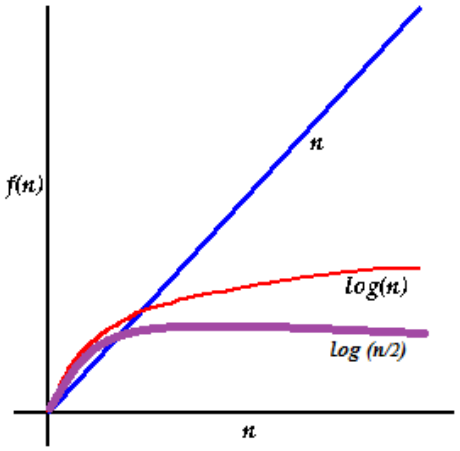

Figure 6. Plot of *n* and log *n/2 vs logn vs n* .

6. CONCLUSION :

In this paper, A new Algorithm is presented and implemented to search an element in an ordered list of items with worst case complexity as ***O(log(n/2))***.The multiple splitting Technique is used to implement the algorithm. An experimental result and performance graph comparing the performance of the Binary search and Quadratic search Algorithm is also presented. This constitutes a significant improvement over the Classical Binary Search with ***O(logn)*** approximation algorithm available in the literature. In Future Scope the algorithm can be optimized as per the requirement. Recursion can also be used to optimize the code. Finding the optimal splitting method that will give the best efficiency in terms of Time is open problem.

7. REFERENCES:

[1] An Approximation Algorithm for Binary Searching in Trees duardo Laber Marco Molinaro : Algorithmica (2011) 59: 601–620 DOI 10.1007/s00453-009-9325-0

[2] D. E. Knuth, The Art of Computer Programming, Vol. 3: Sorting and Searching. Reading, MA: Addison-Wesley,1973 Structures.

[3] E. Horowitz and S. Sahni, fundamental of Data Structure Rockville, MD: Computer Science Press, 1982.

[4] K. J. Overholt, "Optimal binary search methods," BIT, vol. 13, no.1, pp. 84-91, 1973.

[5] D. Coppersmith, "Fast evaluation of logarithms in finite fields of characteristic two," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 587-594, 1984.

[6] Ben-Asher, Y., Farchi, E., Newman, I.: Optimal search in trees. SIAM J. Comput. **28**(6), 2090–2102 (199Carmo, R., Donadelli, J., Kohayakawa, Y., Laber, E.: Searching in random partially ordered sets.Theor. Comput. Sci. **321**(1), 41–57 (2004)

[7] Knight, W.: Search in an ordered array having variable probe cost. SIAM J. Comput. **17**(6), 1203–1214 (1988)

[8] Navarro, G., Baeza-Yates, R., Barbosa, E., Ziviani, N., Cunto, W.: Binary searching with nonuniform costs and its application to text retrieval. Algorithmica **27**(2), 145–169 (2000)

[9] The *m*-Version of Binary Search Trees: An Average Case Analysis, Hindawi Publishing CorporationISRN Combinatorics,Volume 2013, Article ID 450627, 8 pages,http://dx.doi.org/10.1155/2013/450627

[10] Machine Vision and Applications,DOI 10.1007/s00138-013-0483-3A 3-degree of freedom binary search pose estimation technique,Robert Ross · Andrew Martchenko · John Devlin,Received: 18 September 2011 / Revised: 29 October 2012 / Accepted: 11 january 2013,© Springer-Verlag Berlin Heidelberg 2013

[11] IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-5, NO. 4, JULY 1979,Multidimensional Binary Search Trees in Database Applications,JON L. BENTLEY, MEMBER, IEEE

[12] Non-blocking Binary Search Trees, *PODC'10,* July 25–28, 2010, Zurich, Switzerland.Copyright 2010 ACM 978-1-60558-888-9/10/07 ...$10.00.

[13] A Non-Blocking Internal Binary Search Tree,SPAA'12, June 25–27, 2012, Pittsburgh, Pennsylvania, USA.Copyright 2012 ACM 978-1-4503-1213-4/12/06 ...$10.00.

[14] Noisy binary search and its applications,Computer Science Division, University of California, Berkeley. Supported in part by NSF grant CCF-0515259 karp@icsi.berkeley.edu, rdk@cs.cornell.edu