Rational Search Algorithm

Raj Asha

Undergraduate student, Department of Information Technology, Thakur College of Engineering and Technology

Kandivali East, Mumbai, Maharashtra, India

Abstract *- The Rational Search algorithm is a searching algorithm used to search for a particular element in an array. The array should be sorted in ascending order for using the algorithm. The Rational Search algorithm works in a similar manner as a human would search a word in a dictionary by directly jumping in the part where there is a high chance of finding the required item. It outperforms linear search as well as binary search when the data items in the array are approximately uniformly distributed, that is, the difference between any two adjacent elements are approximately the same for any other two adjacent elements in the array. Otherwise its performance is equivalent to a linear search algorithm.*

Keywords—*Search Algorithm, Rational Search Algorithm, Searching.*

I. INTRODUCTION

The Rational Search algorithm searches for a particular element in the array. The array must be in ascending order for using the rational search algorithm. The Rational search algorithm performs excellently when the elements in the array are approximately uniformly distributed, that is, the difference between any two adjacent elements are approximately the same for any other two adjacent elements in the array.

Rational Search algorithm searches for an element in the array just like a human would search for a word in the dictionary by directly jumping in the part where there is a high chance of finding the required word.

Rational Search uses the first and last element of the sorted array and the element to search in an equation which returns an index which is very close to the element that is to be searched if present. The above process is repeated where the first and last element changes according to the subarray at every iteration until the element is found or the entire array is checked. If the element is found, then it returns its index otherwise return null.

Rest of the paper is organized as follows: section 2 describes the Related Work in which I have discussed about linear search and binary search algorithms. Section 3 describes the new Rational Search algorithm. Section 4 describes Proof of Correctness of the rational search algorithm. Section 5 describes Comparison between linear, binary and rational algorithm. Conclusion, acknowledgment and references is provided in section 6.

II. RELATED WORK

A. *Linear Search*

Linear search is also called sequential search. It is one of the simplest searching algorithm that finds an element from a list of elements. It starts from the first element in the list and moves in sequence one element at a time to search for the target value. If target element is found then search stops and returns the target element index otherwise it reaches the end and returns null.

Linear search worst case running time is n, where n is the number of elements in the list.

Algorithm

1. Set i to 0
2. If $A_i = E_{target}$, then target element found; return i.
3. Increment i by 1.
4. If $i < n$, go to set 2. Otherwise, the target element not found; return null.

B. *Binary Search*

Binary search is a popular searching algorithm which requires the array to be sorted in order to search for the target element. Binary search finds out the middle element and then compares it with our target value; if the middle element matches with the target value then simply return index of the middle element otherwise check whether our target value is smaller or larger than the middle element and accordingly consider only the part of array where the target value may be present and eliminate the other half. Continue this process until the target value is found or the remining part is empty.

Binary search runs in at worst logarithmic time, making $O(\log n)$ comparisons, where n is the number of elements in the array.

Algorithm

1. Set low to 0 and high to n-1.
2. If low > high, the search stops and target value is not found so return null.
3. Set mid(middle element) to (low+high)/2.
4. If $A_{mid} < E_{target}$, then low = mid + 1 and go to step 2.
5. If $A_{mid} > E_{target}$, then high = mid – 1 and go to step 2.

6. If $A_{mid} = E_{target}$, then target element found; return mid.

III. Rational Search

Rational search is a search algorithm that finds the position of a target value within a sorted array. Rational search makes use of an equation involving value of element to be searched, value of first and last elements of remaining part of array. The equation returns an index whose value is compared with our target element; If they are unequal then the part of the array where the target element cannot lie is eliminated and search continues on the remaining part until it is successful or remaining part becomes empty.

Rational search has a worst case when elements are not at all uniformly distributed at that time it takes the same time as linear search. But its best case is when the elements are uniformly distributed then it takes constant time to search an element. Whereas for the average case it performs better than linear search but may perform better/worse than binary search.

Algorithm

```
1. Set L to 0 and R to n-1
2. Set Lval to $A_L$ and Rval to $A_R$
3. If L > R, the search terminates as unsuccessful.
4. Set perc to T * 100 / (Lval + Rval).
5. Set index to (perc * (R – L + 1) / 100) + L.
6. If $A_{index}$ < T, set L to index + 1 and go to step 2.
7. If $A_{index}$ > T, set R to index – 1 and go to step 2.
8. Now $A_{index}$ = T, the search is done; return index.
```

For example, if we have an array of size 10 which is uniformly distributed, that is, the difference between any two adjacent elements are approximately the same for any other two adjacent elements in the array and is stored in ascending order as:

A[10] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}

If we want to search for 90 then in such a case:

1. L = 0 and R = 9
2. Lval = 10 and Rval = 100.
3. Since L < R we continue.
4. Perc = 90 * 100 / (10 + 100) = 81.
5. Index = (81 * (9 – 0 + 1) / 100) + 0 = 8.
6. Now A_{index} = T, the search is done; return index 8.

Linear search:
It would take **9 comparisons** for searching using linear search as linear search takes O(n) time for searching in average and worst cases.

Binary search:
It would take **3 comparisons** for searching using binary search as binary search takes O(log n) time for searching in average and worst cases.

Rational search:
It would only take **1 comparison** for searching the element using rational search as it takes constant time (O(1) time) for rational search when the data is uniformly distributed.

IV. Proof of Correctness for Rational Search

To prove correctness of the algorithm, we need to prove two things:

1. algorithm terminates
2. algorithm produces correct output

A. Rational Search Terminates After Finite Number of Steps

Variable L is set to 0 and R is set to n – 1. After each iteration either the value of L is increased at least by 1 or the value of R is decreased at least by 1. The termination condition is given as L > R. Therefor after a finite number of steps L will become greater than R and so the loop will terminate.

B. Rational Search Produces Correct Output

In the last line, we can see that if A_{index} = T, where T is the target element then we return the index of the element otherwise the process continues until L <= R after which we return null and the program terminates indicating the element is not present in the array.

V. Comparison of Searching Algorithms

We will compare three search algorithms which are linear search, binary search and rational search.

TABLE 1
TIME COMPLEXITY OF DIFFERENT ALGORITHMS

	Best	Average	Worst
Linear	O(1)	O(n)	O(n)
Binary	O(1)	O(log n)	O(log n)
Rational	O(1)	Less than O(n)	O(n)

Best Case:

The best case for rational search occurs when all the elements are uniformly distributed. In such a case, it requires constant time O(1) to search for an element irrespective of the size of the array.

Average Case:
The average case for rational search occurs when some elements are uniformly distributed whereas others are not uniformly distributed. In such a case, small clusters are formed causing delay in finding the element quickly as compared to best case. So, in such a case, it requires less time than linear search, that is, less than O(n) time.

Worst Case:
The worst case for rational search occurs when all elements are not uniformly distributed at all. In such a case, it requires the same time as linear search for searching an element, that is, it takes O(n) time.

VI. CONCLUSION

Rational search is thus a good searching algorithm which can be used in many applications for a quick search result than linear search and also than binary search in many cases. Rational search, when used in application where data is uniformly distributed, searches an element in constant time outperforming both linear and binary search. Therefore, rational search gives best results as compared to linear or binary search in applications where data does not contain many outliers.

ACKNOWLEDGMENT

The author wishes to thank Rupa Asha, author’s mother, who helped him by providing suggestions and improvements in the algorithm.

REFERENCES

[1] Linear search algorithm, https://en.wikipedia.org/wiki/Linear_search
[2] Binary search algorithm, https://en.wikipedia.org/wiki/Binary_search_algorithm
[3] Knuth, Donald (1998). *Sorting and Searching*. The Art of Computer Programming. 3 (2nd ed.).
[4] Uniform distribution (continuous), https://en.wikipedia.org/wiki/Uniform_distribution_(continuous)
[5] Time Complexity, https://en.wikipedia.org/wiki/Time_complexity
[6] Alfred V., Aho J., Horroroft, Jeffrey D.U. (2002) Data Structures and Algorithms