DESIGN AND ANALYSIS OF PARALLEL LINEAR SEARCH ALGORITHM

Rahul Sharma¹, Rohit Kumar²

Abstract—Searching is an important operation that is performed on array. Searching is used in software development, web development and database to perform certain operation. Linear search algorithms is used to perform searching operations on array. In case of linear search, operation will be delayed when the size of array is large, and the search value is at the end of the array. This paper presents Parallel Linear search algorithm, which will help in performing the search operations faster. In each iteration, the algorithm compares four values of array with the search value and returns the index when the value of array matches with the search value. Search is performed from two directions, which makes it faster even if the value is at the last.

Keywords—array, data structure, linear, parallel, search, searching algorithm

1. INTRODUCTION

Evolution of computer and advancement of technology had made operations easier and faster. Several algorithms are useful in solving problems. Searching algorithms are used to search the desired value in a huge data. An efficient algorithm is needed to perform the operations, so that the time and memory consumption is less. The modern digital computer was intended as a machine or device that will facilitate and speed up the computation of complicated problems. Searching operations are also used by the search engines. The data is searched from the large database. To search the data from a large database is a time consuming process, but if the searching algorithm is fast then the search operation is performed in less time. To perform a search operation on array, there are many search algorithms that are used according to the requirement. Linear search [1], Binary search [2], Jump search [3] are used for searching values from array. Each searching algorithms has different mechanisms. Linear search is easy to understand and implement. Linear search compares each value of the array in a linear fashion. Worst case complexity of linear search is O(n). There are searching algorithms that have some conditions. Binary search works only on sorted array. Linear search can work on unsorted array. Binary search divides the array and search the value. Linear search is not used for large sized array.

2. PROPOSED ALGORITHM

To make the search process faster, here is the pseudocode of Parallel linear search given below:

```c
int search(int a[], int val, int size)
for (i ← 0; i <= size/2; i ← i+2)
if (a[i] = = val)
return i
else if (a[i+1] = = val)
return i+1
else if (a[size-i-1] = = val)
return size-i-1
else if (a[size-i-2] = = val)
return size-i-2
return -1
```

There is one method, which performs the search operation. The method take array, search value and size of array as parameter. There is a loop, which iterates from 0 to size/2. Every time the loop increment by 2, Search value is compared with i index of array. If both matched, then it return the index position of the value otherwise i+1 index of array is compared with the search value. If both the values are matched then it return i+1. Now, the size-i-1 index is compared with the search value. If both matched, then it return size-i-1, and at last it compares the index of size-i-2 with the search value. Similarly, in the next iteration the above steps are repeated.

¹ Department of Computer Science and Engineering, IIMT Engineering College, Meerut, UP, India
² Department of Computer Science and Engineering, IIMT Engineering College, Meerut, UP, India
2.1 Working—
Fig. 2 shows the working of the parallel linear search algorithm. In first iteration, i = 0, which will lead to the comparison of first and second value of the array with the search value. Then it compares the size-i-1 and size-i-2 value with the search value. In case, if any value of array is same as the search value, then it return the index position of that value. When the search value is not found, i increment by 2 and in second iteration, i = 2. Again i and i+1 is compared with the search value, size-i-1 and size-i-2 values are compared with the search value. The loop will iterate till the condition of the loop is fulfilled. In each iteration, four values are compared with the search value.

2.2 Analysis—
For an array with n value, the best case is when the search value is equal to the first value of the array. In this case only one comparison is required. The best case of parallel linear search algorithm will be:
O(1)
The proposed algorithm has a for loop, which iterates from 0 to size/2 and increment by a constant value. In worst case, the search value is not present in the array, and the loop has to execute completely. The time complexity of a for loop is considered as O(n) when the loop variable is incremented by a constant value. The worst case of parallel linear search algorithm will be:
O(n)

2.3 Example—
Fig. 3 is the example of parallel linear search algorithm. The size of array is 8, and the search value is 2. In first iteration, four values are compared with the search value. In the second iteration, next four values are compared. In the given example, search value is found in the second iteration at 5th index.
3. EXPERIMENT AND RESULT

Time taken by both the algorithms are measured with same data on a machine of 64-bit operating system with Intel(R) Core(TM) i5 1.60GHz and 8 GB RAM.

Table - 1 Time taken by linear search and parallel linear search in worst case

<table>
<thead>
<tr>
<th>N</th>
<th>Linear search(seconds)</th>
<th>Parallel linear search(seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>0.000289</td>
<td>0.000288</td>
</tr>
<tr>
<td>20000</td>
<td>0.000568</td>
<td>0.000553</td>
</tr>
<tr>
<td>30000</td>
<td>0.000782</td>
<td>0.000677</td>
</tr>
<tr>
<td>40000</td>
<td>0.001064</td>
<td>0.000836</td>
</tr>
<tr>
<td>50000</td>
<td>0.001219</td>
<td>0.000947</td>
</tr>
</tbody>
</table>
CONCLUSION
It is not important to sort the data when the searching operation is performed using linear search. In parallel linear search algorithm, it is not important to sort the data before searching. The time complexity of both the algorithms are same, but parallel linear search compares four value in one iteration. This help in searching the value faster as compared to linear search. In linear search the loop iterates from the first value to the last value of the array, but in multiple comparison linear search algorithm the loop iterates from 0 to size/2. Both the algorithms are tested with equal sized array. We conclude that in worst case the parallel linear search algorithm works faster than the linear search.

REFERENCES