Utilizing Centralized Diamond Architecture for Searching Algorithms

Masumeh Damrudi, Kamal Jadidy Aval
Computer Science Department, Islamic Azad University, Firoozkooh Branch, 3981838381, Iran
m.damrudi@yahoo.com, k.jadidy@yahoo.com

Abstract. Searching data elements and information in a fraction of time is still an important task for many areas of works. The importance of finding the best answer leads to use different algorithms which consume time. To improve the speed of searching information within mass information, employing parallel processing is inevitable. A search algorithm with constant time(CS) is proposed on centralized diamond architecture. We have optimized this algorithm to find the location and the number of occurrence of data, if there exists any.

1. Introduction
Finding the appropriate answer in the search algorithms is a critical work in some areas of study. In computer science and mathematics, a search algorithm receives an input and after evaluating the possible solutions, returns the answer. Various searching algorithms are presented by different researchers which are employed in applications. Some of these applications include expert systems, database systems, game systems, and robot control systems [1].

Some of the search algorithms are considered in [2]. A parallel search algorithm on sorted matrices with $O(\log\log n)$ execution time using $O(n/\log\log n)$ processors on a CREW PRAM is presented in [3]. Their selection is considered to be an $n \times m$ sorted matrix. Two parallel search algorithms with different complexities in different ways are explained on [4]. A parallel search algorithm on tree is presented with $O(\log^2(n+1))$ time complexity in [5].

We issue an improved algorithm on the new searching algorithm on Centralized Diamond architecture presented in [6]. Diamond architecture which is heterogeneous was proposed in [7]. This architecture has $(7/4)n$ processor elements. NOD and ENOD sorting algorithms are using this architecture [7, 8]. The Centralized Diamond issued in [6], is a variation of Diamond exploiting one more node in the center. This improved architecture leads to a better result in the design of algorithm which the search algorithms are presented on centralized diamond architecture.

The first search algorithm is the improved of CS algorithm [6] which give is getting the key value location as the result of search too. The second search algorithm count the number of searched value as the result of algorithm.

2. Improved of CS Search on Centralized Diamond Architecture
The main CS search algorithm and the Centralized Diamond architecture are completely explained in [6]. In this paper, assume $n$ is the number of data elements and $N$ is the number of processor elements. The Centralized Diamond architecture for $n=16$ and $N=29$ is shown in Figure 1.
In the first place, all data elements are in the third level as inputs. The aim is to search the key value of \( k \) among data elements as well as its location. It means that if the data element exists, which processor element has it. It is assumed that the data elements are distinct. Step 1: At first, \( k \) is entered to all nodes in the third layer of the architecture at the same time. All nodes compare their data elements with the received value \( k \). If these two data were equal in the node, the node sends the PE number to its parent which means the data exists as shown in Figure 2. Otherwise the node sends 0 to the parent.

**Figure 2. Operations in the third layer**

In this figure, node 18 has the \( k \) and sends PE number which is 18 to its parent. Step 2: At the second level, each processor performs OR operation with the received values which are 0 or a PE number. The result of OR operation will be sent to its parent in the first level. Figure 3 shows this operation.
Figure 3. Operations in the second layer

Step 3: At the first level, each processor element makes OR operation with the received values. The result of OR operation will be sent to its parent which is the central node in the zero level. The received values are either zero or number of a PE that has the $k$ value. Figure 4 shows this operation.

Figure 4. Operations in the first layer

Step 4: Level number zero receives the results from first level. These data are the number of a PE or zeros. The OR operation on the received values present a number. If this number is 0, it means the key $k$, doesn’t exist and if the number becomes greater than 0, it means that data exists and the number is the PE number which has the key.
3. Counting of Searching Value on CS Search

In some cases, it is important that how many of the searching value exists. This algorithm is called counting. The algorithm is explained via an example. In this example, it is considered that PEs 18 and 21 have the \( k \) value. At the first place, all data elements are in the third level as inputs. Step 1: At first, \( k \) is entered to all nodes in the third layer of the architecture at the same time. All nodes compare their data elements with the received value \( k \). If these two data were equal in the node, the node sends 1 to its parent which means the data exists. Otherwise the node sends 0 to the parent. The results are shown in Figure 5. It is considered that the PEs with number 18 and 21 have the key value.

Step 2: At the second level, each processor calculates the sum of received values which are 0 or 1. The result of addition will be sent to its parent in the first level. Figure 6 shows this operation.

Step 3: At the first level, each processor element adds the received values. The result of addition will be sent to its parent which is the central node in the zero level. The received values are either zero or a number greater than zero. Figure 7 shows this operation.
Step 4: Level number zero receives the results from first level. The addition of the received values presents a number. If this number is 0, it means the key \( k \), doesn’t exist and if the number becomes greater than 0, it means that data exists and the number shows how many of data \( k \) exist.

4. Analysis and Cost
Total execution time for both searching algorithms is constant. In the improved of CS search algorithm, Step one has just one comparison, which is \( O(1) \). In step two, two received data elements perform OR operation which take place in \( O(1) \). The third step is just like the second step which means it needs \( O(1) \) execution time. The last step requires two OR operations. This step requires \( O(2) \).

The total execution time is constant \( t_n=O(5)=O(1) \), therefore the total cost is the number of processor elements \( c_n=O(N) \).

The execution time for the second algorithm (counting algorithm) is the same as the improved CS algorithm. The difference is the type of operations. The operations in the second algorithm is addition while in the first algorithm is OR operation.

Considering the first algorithm, in following example assume that we are looking for \( k=5 \) among data elements. It’s seen that having \( N=29 \) and \( n=16 \) is like four threes are connected to each other. The number of trees differs by the number of data elements and processor elements as it’s considered in Figure 8.
All these four parts work in parallel during the algorithms. The steps from one to three are shown in Figure 9 for improved CS algorithm.

**Figure 9.** The procedure of algorithm from step 1 to step 3 of improved CS algorithm

In step one, the architecture from the left side, 13 is not equal to the key 5, so sends 0 to level two and also 26 is not equal too, then it sends 0 to level 2 as shown in the figure. The value 5 is equal to the key 5, then the value 18 which is the number of PE is sent to level 2. The squares show the number of registers of each PEs in each level.

In level two, in the left side, two received zeros do OR operation, and the result which is 0 is sent to the first level. In the right side, the OR operation of zero and 18 is 18. The result is sent to the level one.

In level 1, one of two received values is 18 then 18 as the results of OR is sent to level 0 because this is the result of third section in Figure 8. The other sections 1, 2 and 4 do not have the key while the third section has it. It is because the data are assumed to be distinct. You see the procedure of step 4 in Figure 10.

**Figure 10.** The procedure of step 4

In the last step, the centralized node does OR operation among the four received data and the final result is 18.

The second algorithm (counting algorithm) is the same as the first algorithm. The difference is the operation. Figure 11 shows the steps of this algorithm for the following example.

**Figure 11.** The procedure of algorithm from step 1 to step 3 of counting algorithm

In step one, the architecture from the left side, 5 is equal to the key 5, so sends 1 to level two and also 26 is not equal to key 5, then it sends 0 to level 2 as shown in the figure. The value 25 is not equal to the key 5, then the value 1 is sent to level 2.

In the level two, in the left side, two received values do addition operation, the result which is 1 is sent to the first level. In the right side, the addition operation of zero and 1 is 1, and the result is sent to the level one.

In level 1, the two received values are 1 then, the sum of these two values which is 2, as the results of addition is sent to level zero. This is the result of third section in Figure 8 for counting algorithm. The results of other sections are calculated and sent to level zero like the third section. The procedure of step 4 is shown in Figure 12.
Figure 12. The procedure of step 4

In the last step, the centralized node adds the results from four sections and the final result in this example is 2.

5. Conclusion
We proposed the improved of our previous search algorithm (CS) with constant execution time which gives the place of the search value as the result. The proposed algorithm has a suitable order using the centralized diamond architecture where the total cost is . It is assumed that the data is distinct in this algorithm.

The second proposed algorithm using mentioned architecture is counting which shows how many of the searched data exists. This algorithm needs constant execution time too.

References
[1] M. Damrudi, K. Jadidy Aval, 2011, Searching data using centralized diamond architecture, Journal of Communication and Computer, 8(9), 807-811.
[2] D. Eppstein, Z. Galil, 1988, Parallel algorithmic techniques for combinational computation, Annual review of computer science, 3(1), 233-283.
[3] R. Sarnath, X. He, 1992, Efficient parallel algorithms for selection and searching on sorted matrices, In Proceedings of Sixth International Parallel Processing Symposium, Washington, DC, USA, 1992, 108-111.
[4] K.T. Yar, N.S.S Aung. 2009, The optimal running time from computational models by comparing two parallel searching procedures, In IEEE International Workshop on Open-source Software for Scientific Computation (OSSC), china, 2009, 73-78.
[5] K. A. Berman, J. L. Paul, 2005, Algorithms: sequential, parallel, and distributed, Course Technology Ptr, USA.
[6] M. Damrudi, Masumeh, K. Jadidy Aval, 2011, Searching data using centralized diamond architecture, Journal of Communication and Computer, 8(9), 807-811.
[7] M. Damrudi, K. Jadidy Aval, 2009, Diamond architecture with NOD sorting, IEEE Youth Conference In Information, Computing and Telecommunication, 2009, 431-434.
[8] M. Damrudi, K. Jadidy Aval, 2010, A new parallel sorting on Diamond architecture," In Proc. 4th Conf. Europ. Comput. Conf., Univ. Politechnica of Bucharest, Bucharest, Romania, 2010, 284-287.