A Succinct Novel Searching Algorithm

Celine[1], Shinoj Robert[2], Maria Dominic[3]
Research Scholar
Department of Computer Science
Sacred Heart College, Tirupattur, India

***Abstract*—A searching algorithm was found to be effective in producing acutely needed results in the operation of data structures. Searching is being performed as a common operation unlike other operations in various formats of algorithms. The binary and linear search book a room in most of the searching techniques. Going with each technique has its inbuilt limitations and explorations. The versatile approach of different techniques which is in practice helps in bringing out the hybrid search techniques around it. For any tree representation, the sorted order is expected to achieve the best performance. This paper exhibits the new technique named the biform tree approach for producing the sorted order of elements and to perform efficient searching.**

***Keywords*—Time complexities; space complexities; searching algorithm; biform tree; pre-order traversal**

I. Introduction

Data structure performs as fundamental in the area of computing. The efficient search and sort are possible only if the data is organized into the headed process as structured delegacies.

The data organization plot is exemplified as a well-known data structure representation from Fig. 1 and clearly states the individual representation on data-based classification. It also projects the clear representation takes away to the immediate access and handling of data. Understanding data structure and algorithm is very difficult unless searching and sorting are not made and also brought into effect. Each desirable algorithm is chosen based on the data structure type [1]. All searching algorithms lead to efficient retrieval of a specific element from the listed aggregation of elements [2]. Until the desired result is found the search process continuous in all versatile techniques.

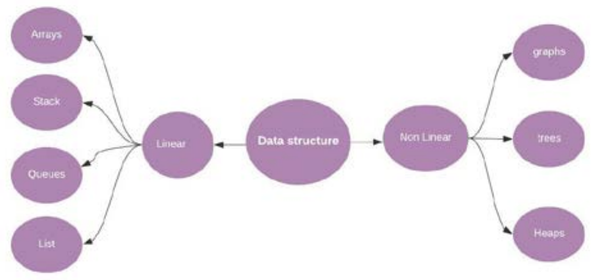


Fig. 1. Data Organization Representation.

II. Searching Techniques

A. Eccentrics of Searching

Searching is a process of accruing and discovering factors from the given list. The searching and sorting algorithms assist in arranging the elements in some order. In deliverance to the efficiency of algorithms including merge and sort technique, sorting is needed [3]. In general, searching is applied to alphabets, strings, and characters other than numbers.

The search algorithms projected in Fig. 2 are on the mind map view featuring significant divisions and classifications of search algorithms that are widely applied on Artificial Intelligence techniques-based algorithms. From the root of the search algorithm, the classification is divided into major divisions as clueless (uninformed) and communicated (informed). The first division clueless of searching explicit the exploration in each step. The goal is to split into each state of activity and explored if not. Less domain knowledge is expected in this type of search and it increases in time complexity. The operations are executed in a brute force method and the result of the current step advances to the next level of implementation. The brute force carries the selective information of traversing from a tree to a festinated step. The communicated search comes with information patterns in each step to find the solution which results quickly in the process. This pattern includes the domain knowledge that results in the heuristic way of approach within fair timing. The complex rich problems are focused and lead to a better solution in this way of approach [4]. The classifications above represent the different searching techniques by all possible means of representing the utmost classification needed.

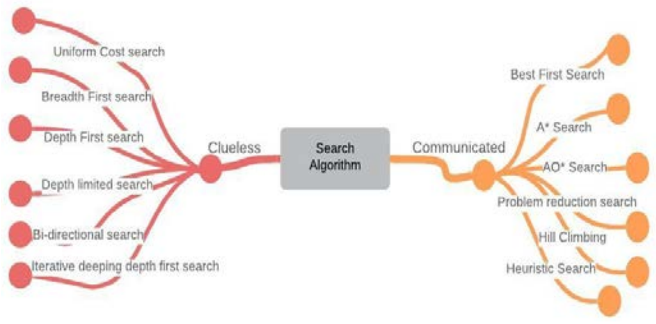


Fig. 2. Searching Types Classifications.

B. Handed-Down (Traditional) Searching Techniques

The search algorithm is one of the class algorithms among the existing classifications on constant, logarithmic, linear, quadratic, and exponential algorithms. The searching is to fill one more piece of the above classifications that are valuable. A searching algorithm is a kind of obvious statement where the word is predominantly mentioned mechanism in the web portal. A search is a way to find a group of items from an implicit or an explicit way of collection. Any searching technique is made easy along with the properties provided. The properties search to reach completeness including the time and space complexities. Every sorting technique comes with a prerequisite in which the effective searching was done looking over the data provided. The comparison is on existing traditional searching techniques assures the complexities they built-in. The searching is made reasonable and efficient through the possibilities as figured and mentioned in search type classifications.

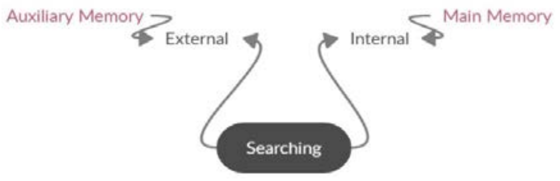


Fig. 3. Searching Proficiencies.

Fig. 3 projects the searching operations on two main classes as external searching and internal searching. The external searching is colligated with auxiliary memory occupies in the files hived away on disk storage. Internal searching is concerned with minimum data that resides on the data processor's main memory [5].

Search classification
Sorting
Searching

Fig. 4. Searching Classifications.

The searching technique has the base classification of Traditional sorting and searching as projected in Fig. 4. The comparative analysis of the different searching algorithms is possible from the base classification field.

III. Comparative Analysis

A. Comparison on Searching Performance

The performance and efficiency of each algorithm differ based on the data provided for each separate or repeated task. The methodology applied to assess the performance is time & space complexity which have a better modification of words over time and memory space in CPU [6]. Any search algorithms are calculated based on certain attributes of their complexities. Efficient searching fulfills the completeness of the searching algorithm. Table I projects the performance of the binary searching on sorted techniques that is brought through a comparative study in three strategies Performance, effectiveness & output.

TABLE I. A Comparative Study on Different Searching Algorithms over Searching

Algorithm	**Technique**	**Performance**		
		Best-face	***Worst-face***	***Fair-face***
Binary search	Divide &conquer	O(1)	O(log2 n)	O(log n)
Sequential Search	Linear search	O(1)	O(n)	O(n)
Hash Search	Hashing	O(1)	O(n)	O(1)
Tree search	Divide & conquer	O(1)	O(n)	O(n)
Interpolation search	Binary search	O(1)	O(n)	O(n)
Jump Search	Linear search	O(1)	O(n)	O(1)
Hybrid Search	Interpolation & Binary	O(1)	O(√n)	O(n)
Exponential search	Sorting	O(1)	O(log n)	O(log n)
Fibonacci search	Comparison-based	O(1)	O(log n)	O(log n)
DFS	Graph data structure	O(1)	O(|V|+|E|)	O(n+m)
BFS	Graph data structure	O(m)	O(b^m)	O(|V|+|E|)
Heuristic search	Greedy search	G(n)	O(bm)	O(bm)
Bi-directional	Graph search	O(bd)	O(bd/2)	O(bd/2)
Sequential Search	Linear search	O(1)	O(n)	O(n)

{N = 23, 10, 75, 05, 15, 82, 19, 07, 31,100}

Fig. 5. Set of N Random Elements.

The purpose of the comparison table is to equate the different searching techniques based on the individual effective performance. The running complexities include best, worst, and average (fair) cases as a comparative study. Beginning with the binary search each algorithm is classified with the technique built-in. The sequential algorithm derived from linear search finds for a particular element starting apiece, considered to be efficient with classified order. Hashing searching techniques have got their advantage over larger data volume of data sets. The results are implemented without any collision possible actions through open and close addressing methods such as family collisions [7]. Tree search addresses the issue of involving combinations with the basic idea of divide and conquer in better measure and conquer technique and the process continues till it pruned [8]. The interpolation search falls another page from the binary search techniques for the particular data set provided the best effort with the sorted factors. Jump search has a limitation over certain block representations with the intervals and has control as block search. Hybrid search is constructed over the sorted and unsorted distribution of arrays. It blends from the staple of binary and interpolation search for efficient search acquiring the advantage point over both algorithms [9]. The exponential search is selfsame to binary search in which the proportions are equal at the depth of the node, this projects the minor level enhanced by an increasing factor of 2 [10]. This exponents the children at each level in order.

From the classification on different searching algorithms, BST is one of the best-practiced techniques of linear data representation types. BST holds a special name in terms of representing in sorted order which pushes the process much effective & easier. It results in optimal performance for the end-user. The construction of a binary tree protrudes from the root node [11]. The tree has the best picture once it is portrayed in the hierarchy of parent-child with a single character as a parent node (root node).

IV. Proposed Methodology

The success story of each searching algorithm will result in finding the desired element. This paper proposes the construction of effective searching. This methodology was applied to reduce the time complexities of searching principles. This paper effort in the construction of an algorithm that is capable of causing searching efficiently. To achieve such efficiency, the given N random numbers will be converted into binary form and a binary tree for the binary form will be constructed. Performing the technique of pre-order traversal on a binary tree provides a sorted set of output.

A. *Phase I: Binary Conversion*

The given N numbers will find a sorted position so that any given number is made available in the tree.

The set of N numbers listed is shown as the example for constructing a binary tree. The step begins by involving binary conversion for the given 10 digits into binary form. For the given values projected in Fig. 5, and converted into binary representation in an unsorted order.

B. *Phase II: Grouping*

The binary conversion is practiced for the set of given N numbers. The presentment of numbers into binary format is projected in Table II. These binary format numbers are grouped based on the number of bits as depicted below representation Fig. 6.

C. *Phase III: Biform Tree Approach*

Construction of biform tree for all groups G1, G2, G3, and G4 are projected in the respective Fig. 7(a) to 7(d). The tree construction is initiated from the root node. If the binary number is 1, then it is skewed to the right-hand side of the root node otherwise to the left-hand side of the root node. This operation is continued for all the n bits.

TABLE II. BINARY REPRESENTATION ON N RANDOM ELEMENTS

N	23	10	75	05	15	82	19	07	31	100
	10111	1010	1001011	101	1111	1010010	10011	111	11111	1100100

G1 = {101, 111}
G2 = {1010. 1111}
G3 = {10111, 10011, 11111}
G4 = {1001011, 1010010, 1100100}

Fig. 6. Elements in Group-Wise.

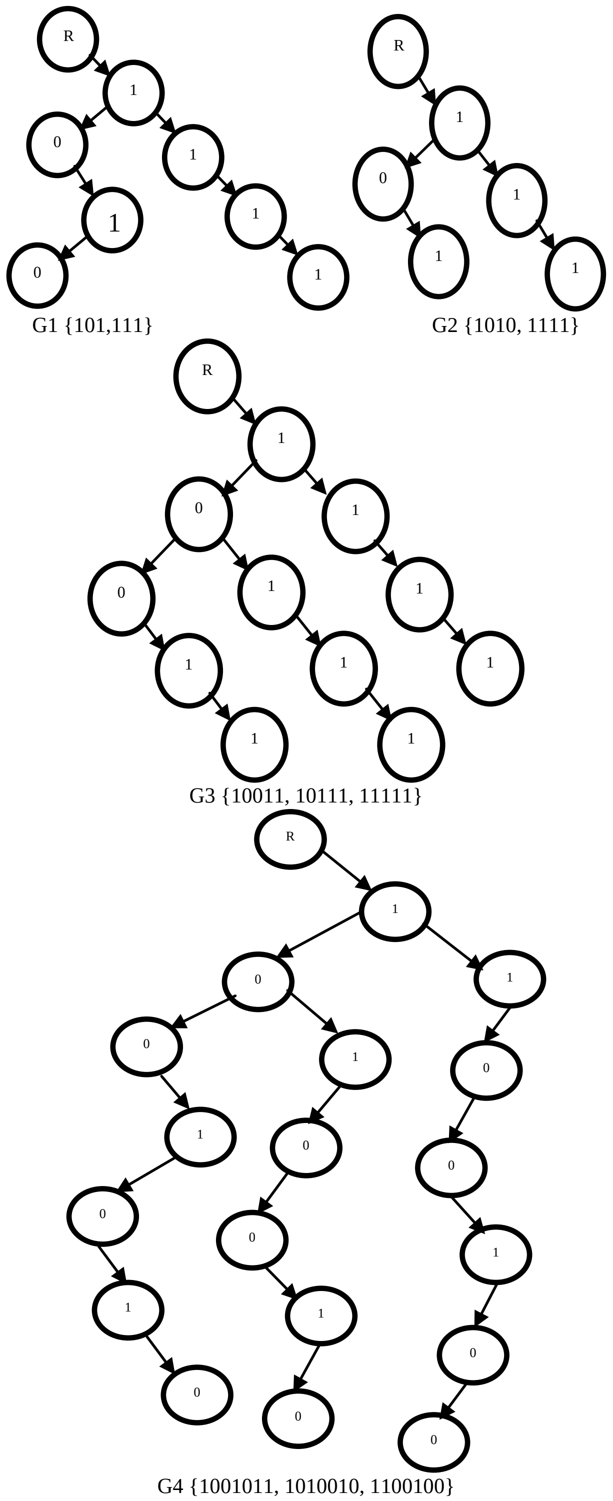


Fig. 7. Biform Tree Representation for G1, G2, G3, and G4.

The effective searching for storing any sorted element is made easy in this way of tree representation. This tree construction helps in making the search faster and effective in paving an easy representation. The grouping of digits will help in looking for the needed elements by avoiding the time in search of the rest.

D. *Phase IV: Tree Traversal*

As a tree is a self-referential data structure, traversal can be defined by recursion. The traversal algorithm can be broadly classified into depth-first and breadth-first search as shown in Fig. 8. The depth-first algorithm is branched towards pre-order, in-order, and post-order traversal.

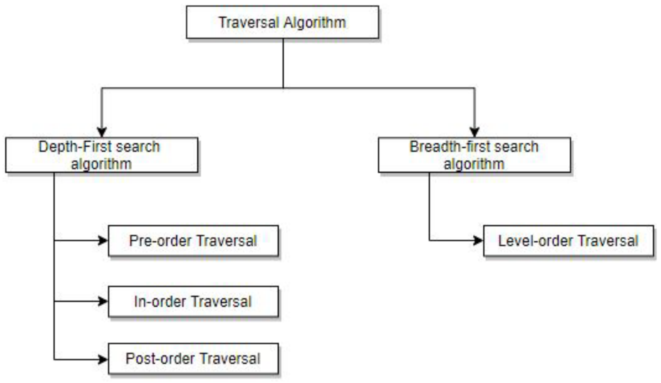


Fig. 8. Traversal Algorithm Classifications.

Classifying the traversal techniques from the depth-first search algorithm, this paper utilizes pre-order traversal to receive the numbers in sorted order.

E. *Phase V: Searching Outgrowth*

To arrange the elements in sorted order and search, pre-order traversal is performed on the tree in a depth-first manner as projected in Fig. 7. To search the particular element 1010, the list made available in the searching process is to be carried in G2 as follows.

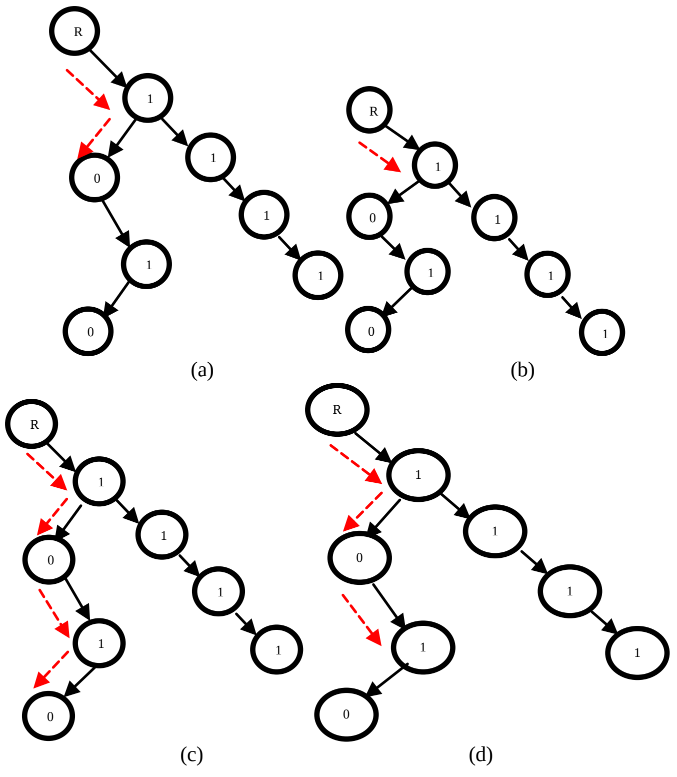


Fig. 9. Searching Process of Each Node.

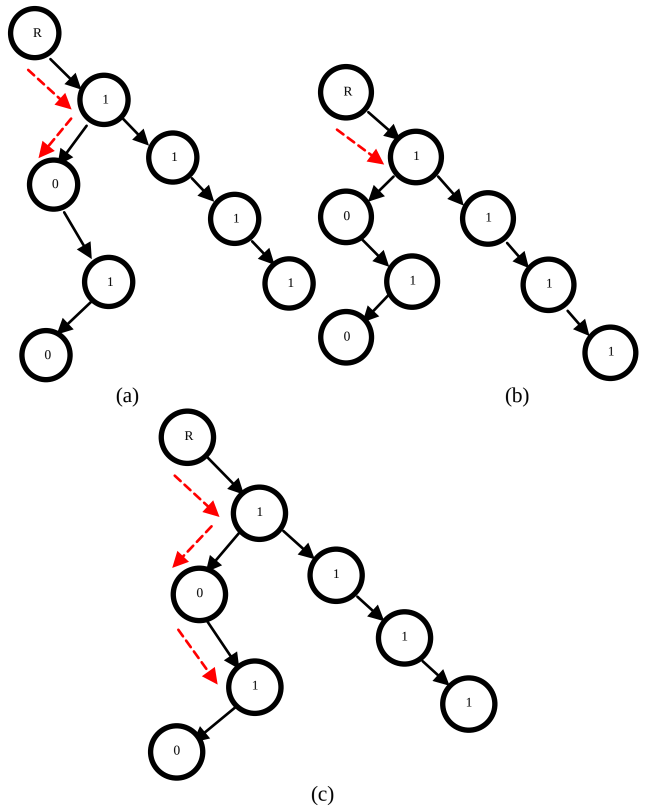


Fig. 10. Searching Process of Each Node (a, b, and c) for Element Not Found.

The traversal begins from the root node R and moves towards the left child of its own. Since the unavailability in the left child for R, the tree travels through the right child as projected in Fig. 9(a). The travel continues through the left child of the current node as it is shown in Fig. 9(b) and the process is repeated in the following Fig. of 9(c) and 9(d). The traveling process is completed once it reaches the leaf node. Now arrange the visited node from the root node as 1010. Hence the element is found in G2.

The traversal begins from the root node R and moves towards the left child and proceeds for the right child as imaged in Fig. 10(a) by visiting the right child node labeled 1 and to the left child labeled 0. The travel continues for the right child labeled 1. Instead of the availability for the right child with the label 1, the left child labeled as 0 presence is found in the tree. The conclusion is derived from the unavailability of element 10111 in the G2. In the search made the availability of left child with label 0 is found but not the other child named right with the label of 1. Grouping in order digit representation which the binary conversion is made emphasizes on a searching part, reckon the given element to be searched and let's spot the random element as 82. The element state is to be found and the number of comparisons made here is spotted as second since it has the immediate search of prefix value. In case of the element 84, the search is ineffective resulting in not found status since the given random element is not found in any of the ordered group in particular to G4, instead if it is then the searching group falls under the G4 classification. The searching is made according to the value given as input. Here the given value is 84, in mechanical the search begins at G4 since the value ranged to 84 lies within that region. The immediate result of found or not found status on any random search is made possible by group classifications.

V. Programme Encryption of the Algorithm

The proposed algorithm is assorted into separate parts classifying sort and search functions. The algorithm produced is made effective in the C++ language. The algorithm for sorting and searching is represented easier as the given element pitch in.

VI. Experimental Result

Searching for a particular element is projected clearly under phase II, classifying bit and group-wise respectively. The bitwise operation of search is the best fit in the case of integer data type [12]. Since the main advantage over the proposed algorithm is found to be effective searching and the arrangement of given numbers is on bitwise. Searching is made easier based on group classification.

TABLE III. Algorithm Representation for Sorting

Sorting Representation	
Step 1	*Consider N random numbers*
Step 2	*Convert all N numbers into binary form*
Step 3	*Group the converted binary numbers based on the number of bits. Let it be m.* *G1(m)=3, G2(m)=4, G3(m)=5, G4(m)=7*
Step 4	see below
Step 5	*Perform the pre-order traversal in the binary tree to receive the numbers in sorted order*
Step 6	*Repeat step 4 and step 5 for all Groups*
Step 7	*Stop the Process*

Step 4:

```
Step 4: Construct a binary tree for each Group G
        for (i = 1 ; i < = m ; i ++) // where m ≥ 3
    {
     insert (Node node, key values) // Insert a new node in the tree
      {
      node ← input node object
      value ← Actual value of the node // values are either 0 or 1
      if (node = = null) then
          return Node (value);
      else if (value [i] = = 0) then
          node.left ← insert (node.left , value)
      Else
          node.right ← insert (node.right, value)
      return node;
      }
    }
```

TABLE IV. Algorithm for Searching a Particular Element in the List

searching representation	
Step 1	*Start a process*
Step 2	*Get the input from the binary trees G_n, where all the numbers in sorted order and n are the number of Groups*
Step 3	see below
Step 4	*Stop the process*

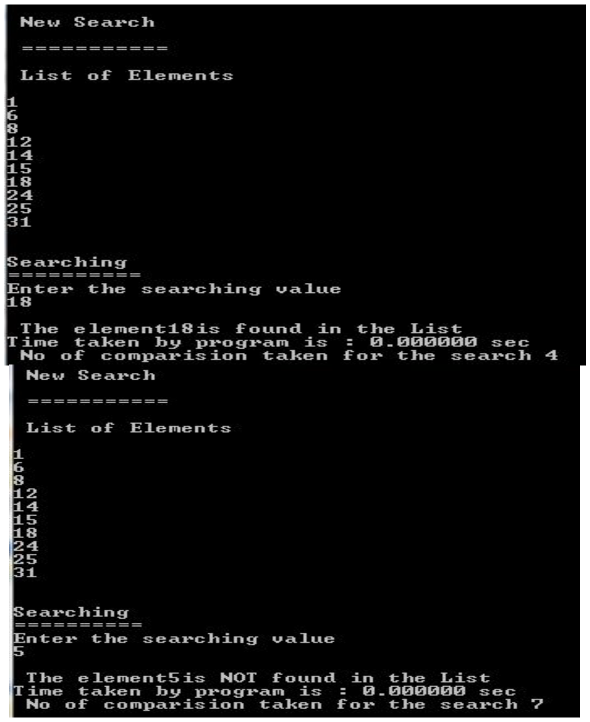
Step 3:

```
Enter the element to be searched.
        Let it be L.
    Check the element L belongs to which Group.
    $\forall_i$ $G_i$ where 1<= i <=n
    if ( L $\in$ $G_n$ )
        {
      Start from the root node R
          for (i = 1 ; i <=$G_i$(m) ; i ++ )
          {
              if ( L[i] == 0 )
            Perform the searching in left sub-tree
                search (root → left)
            Else
                Perform the searching in right sub-tree
                search (root → right)
      }
    Else
          print element L is not available in N
    }
```

The algorithm projected in Tables III and IV on sort and search functionalities is applied to produce the experimented result on comparison over linear, binary, and the proposed search. In the next phases of comparisons, the results are proven over a tried-out study between linear, binary, and the proposed search. The prediction time, an element found and the comparison made are taken as measuring facts for the comparisons. Below Table V shows the comparison involves the searching technique.

TABLE V. Comparison between Linear, Binary, and Proposed Search

Elements (in random)	Algorithms		
	Binary	Linear	Proposed
	Prediction Time	Prediction Time	Prediction Time
1	0.00100	0.00200	0.00000
6	0.00100	0.00200	0.00000
8	0.00100	0.00100	0.000000
12	0.00100	0.00100	0.000000
14	0.00200	0.00100	0.000000
15	0.00100	0.00200	0.000000
18	0.00100	0.00100	0.000000
24	0.00100	0.00000	0.000000
25	0.00100	0.00100	0.000000
31	0.00100	0.00200	0.000000



```
 New Search
 ===========
 List of Elements
1
6
8
12
14
15
18
24
25
31

Searching
==========
Enter the searching value
18
 The element18is found in the List
Time taken by program is : 0.000000 sec
 No of comparision taken for the search 4
 New Search
 ===========
 List of Elements
1
6
8
12
14
15
18
24
25
31

Searching
==========
Enter the searching value
5
 The element5is NOT found in the List
Time taken by program is : 0.000000 sec
 No of comparision taken for the search 7
```

Fig. 11. The Experimented Result on the Element Found and Not Found in the Proposed Searching Technique.

Table V focus on emphasizing searching by making a comparison on linear, binary, and the proposed search numbers in an individual search mode. The experimented result is proven with the given number of 10 random numbers Consider the element 18 to be found in binary search, the prediction time taken by the binary search includes 0.00100 seconds where linear carries the same.

The biform approach tree technique is the searching time projected in Fig. 11 consists of 0.000 seconds and considers to be the quickest in delivering the search result compared with the two existing searches. In the case of an element not found represented in Fig. 10, the total time taken for traversal in search of the given element 5 is found to 0.000000 seconds. This includes the tree representation as stated in the above discussion on searching occurrences in the proposed methodology topic.

VII. Graphical Representation

The graphical representation for the effective searching over the above discussed different programming on searching techniques were applied on a set of linear, binary, and the proposed search.

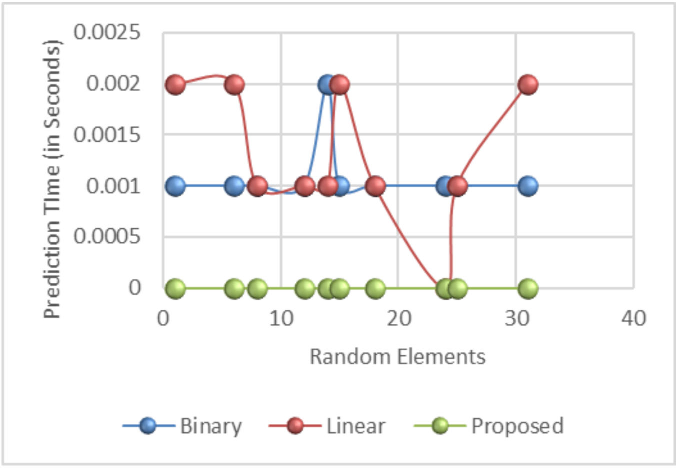


Fig. 12. Graphical Representation of Binary, Linear, and Biform Search.

A. *Graphical Representation of Searching an Element among for 10 Elements*

The graph representation is presented over linear, binary and the proposed search is exemplified as graphical representation in Fig. 12. The X-axis represents the random elements given in numbers and the Y-axis indicates the arrival of searches in seconds. The individual search includes linear, binary, and proposed are represented in separate line charts. The dots connected with blue represents the binary and others simultaneously. The graph withstands the elements up to 10 combinations in a random approach in the above graphical representation. The data projected as X-axis is made an approximate joint on Y-axis in which the total time is taken for each data and mapped into a line chart representation. Similarly, the remaining searches for the given set of random elements take place in the graphical representation.

VIII. Graphical Representation of Searching an Element among for 50 Elements

The graph projected in Fig. 13 explains the prediction time taken for each set of linear, binary, and proposed search techniques. This line chart is represented on 50 random sets of elements. The comparison time increases when the bit sizes get increased in each representation.

The maximum threshold for the binary, linear search is considered as 0.004 seconds while the minimum search happens and succeeds on the proposed search as 0.001 seconds when the given element bit sizes increase. While the elements are distributed in random the classification takes place based on group phases as discussed in the structured methodology. From the given program for efficient searching, the line chart is discussed to prove the efficient technique for searching is on a proposed searching method.

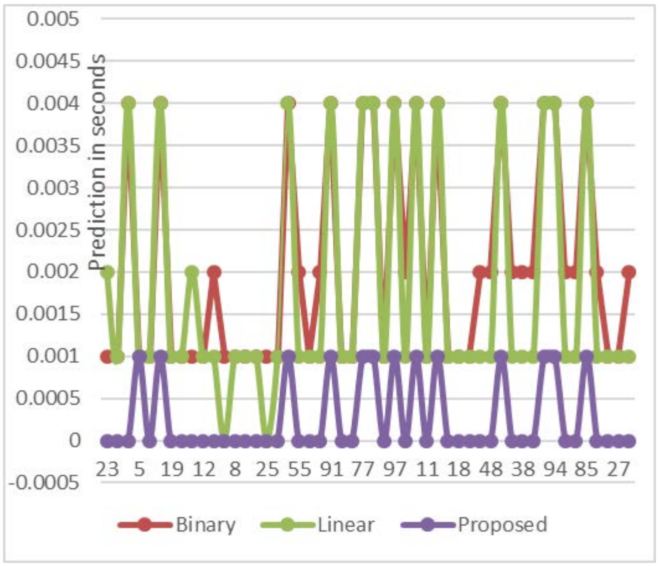


Fig. 13. Graphical Representation of Binary, Linear, and Biform Search Over 50 Elements.

IX. Graphical Representation of Searching an Element among for 100 Elements

The Graphical chart representation of Fig. 14 projects the status for the possible search of carrying a maximum of 100 random values in a given element. Fig. 14 illustrates how the searching takes place as efficient when more elements are into it.

As stated in the combination of more than 50 inputs the same criteria fit in for the case of 100 inputs. Here in Fig. 13 indicates the proposed search goes to the maximum prediction time of 0.001 seconds when the bit size increases in a large volume of size while the linear and binary goes to the extent of 0.004 seconds. From the given every efficient search in the graphical format of presentments the tree structure stood as a prerequisite for easier delegacies. Alike sorted data for binary search, tree structure for the biform tree approach act as a prerequisite inefficient search. All the above three graphical representation initialize and emphasizes on searching techniques that deliver the fast result on any values given in.

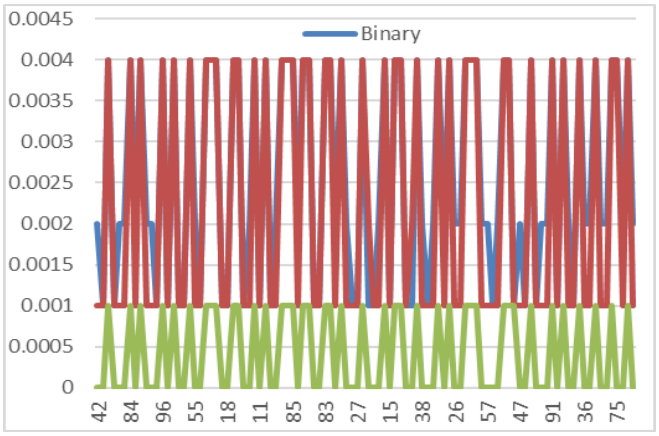


Fig. 14. Graphical Representation of Binary, Linear and Proposed Search over 100 Elements.

X. Practical Applications and Enhancements

The tree structure proposed and experimented in the graphical representation defined for searching any random element will be an effective and efficient way. The tree representation way of organizing the data was brought into effect by searching for any random value and it is highlighted in all the experimented results that search took place. Both the result of found and not found was done efficiently and it was proven in the sample searching on the experimented result page. The time complexity for the biform approach search for best face and fair face is O(log n) and the worst face is O(n).

The biform approach search compared with the existing (linear and binary) will utilize the tree structure that has a binary tree nature in organizing the data that act as an integral to lots of existing applications considering the binary tree node representation. The pointer includes left-right and parent deliberated in space consumption in the large representation of values.

The proposed search was applied in the renowned areas of acquiring knowledge from a knowledge base using tree representation, in the prediction grounded model for directly learn word representation. In the field of synonyms detection and word prediction where the words are evaluated for easy word representation. The tree structure is represented as a prerequisite for any search that is being done in the proposed method. For effective searching, a data structure in the form of a sorted order tree structure is a prerequisite like the existing searches including binary, linear, etc. The biform tree approach search was found to be effective in the finding of the learning content for the learner from the repository. Once the content is identified in the repository the corresponding knowledge graph has been created for the system to understand and learn about the learner to provide the needed learning content. The novel biform tree structure is a prerequisite for efficient searching and also for sorting the elements while doing the pre-order traversal over that tree.

XI. Conclusion

The different searching techniques make way to crystalize and clarify the problem in an effective way. It is an evolutionary method among the infinities in the improvement of time complexities. The biform tree approach inquiry on searching techniques introduces the binary representation in digits that makes the search easier & faster with well-defined generalized fit in by making a comparison table study over performance on the search and the new data structure has a built-in form of tree representation using the searching technique. The beauty of the biform tree approach is the same binary tree is utilized to produce the sorted order numbers and for the element search. The precursor for element searching is provided by the sorted binary tree in which the elements are in sorted order. Altogether searching was made effective over designing a data structure in the form of tree representation which makes the searches an efficient one with the biform approach search.

References

[1] N. Sultana, S. Paira, S. Chandra and S.S Alam, "A brief study and analysis of different searching algorithms", International conference on electrical, computer and communication technologies, Vol 4, pp. 1-4, February 2017.

[2] A. Shoaib "A survey in different searching algorithms", International research journal of Engineering and Technology, vol. 7,p. 275, January 2020.

[3] Kamlesh Kumar Pandey and N. Pradhan, "A comparison and selection on the basic type of searching algorithm in data structure", International journal of computer science and mobile computing, Vol 3, pp. 751-758 July 2014.

[4] Choing R, Sultano J. H and W. J. Jap, " A comparative study on informed and uniformed search for intelligent travel planning in borneo island", International symposium of Information Technology, IEEE. Vol. 3, pp. 1-5, August 2008.

[5] Ana Bell, Eric Grimson, and John Guttag, "6. 0001 Introduction to computer science and programming using python, Massachusetts Institute of Technology: MIT, OpenCourseware, Fall 2016.

[6] K Roopa and J Reshma, "A comparative study on sorting and searching algorithms, International Journal of engineering and technology, Vol 5, p. 1416, January 2018.

[7] Dapeng Liu and Shaochum Xu, "An Empirical study on the performance of Hast table", Dapeng Liu et all, 13th International conference on Computer and Information science (ICIS), Vol 3, pp 60-68, January 2015.

[8] Henning Fernau and Daniel Raible, "Searching trees: an essay", International conference on Theory and applications of Models of computation, pp 59-70, May 2009.

[9] A. S. Mohammed, S. E. Amrahov and F. Celebi, "Efficient hybrid search algorithm on ordered dataset", computer engineering department, Turkey, August 2017.

[10] Xinguo Deng, Yangguang Yao, Jia Chen and Yufeng Lin "Combining breadth-first with depth-first search algorithms for VLSIwire routing", International conference on advance computer theory and engineering, pp. V6-486, 2010.

[11] Inayat Rehman, S. Khan and M. S.H. Khayal, "A survey on maintaining binary search tree in optimal shape" International conference on Information Management, and Engineering,pp. 365-369 , June 2009.

[12] K. Yordzhev, "The bitwise operations related to a fast sorting algorithm", International Journal of Advanced Computer science and applications, Vol. 4, pp. 103-107, November 2013.

Authors' Profile

S. Celine is a part-time research scholar in the Department of Computer Science, Sacred Heart College, Tirupattur district, and working as an Assistant professor in the Department of Computer Science, Government of Arts College for Men, Krishnagiri, Tamil Nadu. Her area of research is in the field of e-Learning using Deep Learning.

Shinoj Robert is a part-time research scholar in the Department of Computer Science, Sacred Heart College, Tirupattur District, and working as an Assistant professor from 2014 in the Department of Computer Application, Don Bosco College, Yelagiri Hills, and His area of research in the field of Machine learning and E-learning.

Dr. M. Maria Dominic obtained his B.Sc., M.Sc., and M.Phil. and Ph.D. in Computer Science. He has been working in Sacred Heart College, from 1996 onwards in various capacities He has also worked in Multimedia University, Malaysia on a Contractual Basis. He has co-authored a book on OOP using C++ published by Pearson education. He has published more than 20 research articles in International Journals. He has 4 Ph.D. Research scholars working under him in the field of Artificial Intelligence especially in Machine learning and deep learning.