IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-9 E-ISSN: 2347-2693

A Review on Specific Data Structures Using Data Preprocessing and Refinement of Existing Algorithms in Order to Improve Time Complexities

S. Hrushikesava Raju[1*] and M.Nagabhusana Rao[2]

[1]*Professor, Department of CSE, SIETK, NarayanaVanam Road, Puttur,A.P. -India*
[2]*Professor & HOD, Department of IT, SRK Institute of Technology, Vijayawada, A.P. -India*

Available online at: www.ijcseonline.org

Received: 22/Aug/2016 Revised: 02/Sept/2016 Accepted: 20/Sept/2016 Published: 30/Sep/2016

Abstract- The data preprocessing is helpful in removing noise, inconsistency in the given data and produce quality data. The output of the data preprocessing is then given to refinement of existing algorithm that can later applied over the data structures called external sorting, optimal binary search trees, and pattern matching algorithms. In external sorting(first case), user data entered can be qualified using Data preprocessing, then separate algorithms used to different data items such as numeric and alphabets. In Optimal binary search trees (second case), user entered data can be made quality data using data preprocessing (second case), then refined algorithm used over the data elements that produce OBSTs separately for numeric items, and String items. In pattern matching (third case), user entered data can be made quality data, then refined algorithm used over the text which immediately finds out index for the pattern along with history of indices for the substring which further helpful in manual identification of the given pattern in the large given text. The results and graphs were also demonstrated based on certain examples. This also differentiates between time complexities obtained of the existing and proposed algorithm used over the data structures such as external sorting, OBST, and pattern matching.

Keywords—*Data preprocessing, data structures, external sorting, Optimal Binary Search Trees, Pattern Matching, Time Complexities.*

I. Introduction

Certain data structures are taken into consideration such as external sorting, pattern matching, and optimal binary search trees. These data structures also used in important real time applications such as bank applications in which account information going to be sorted based on balance, or date of opening the account, or based on IFSC Code etc. where external sorting is used. The other applications such as finding a particular data in the huge amount of related data like gas holder's information or PAN card information where pattern matching is used. The last but not least applications such as construction of decision induction trees or binary search trees where optimal binary search trees are used so that visual effect gives more meaning simply than lot of English like statements.

A. External Sorting: It is the data structure required whenever the data stored in external storage devices such as tape, disk, drum etc. The existing algorithm consumes more number of disk accesses, more runs, more input and output costs. The time complexity $2*N*(\log_B (N/M) + 1)$ where N is number of items in the data set, B is order of sorting such as 2-way or 3-way or k-way etc. and M is initial run size been reduced still further after data preprocessing is applied on the initial data set and modified algorithm is applied on that data set. The output of this is producing sorted data elements in less than $2*N*(\log_B (N/M) + 1)$. The related works on this are:

Firstly, certain types of lemmas are used to achieve efficient external sorting but it works on only one disk model although it takes less Input / Output operations than normal merge sort.

Secondly, the external sorting applied on the data although that are of disk accesses or inputs / outputs costs are huge compared to disk accesses on data without redundancy. The time complexity of k-way merging and poly-phase merging on the data that possess redundancy are also taking more time. But poly-phase is somehow reduce time compared to k-way merge sorting.

The following table lists existing methods used for

Method name	Irregularity	output
Data cleaning	Incomplete, noise, inconsistent, missing	Quality data Before integration
Data Integration and transformation	Object identity problem	Quality data with care taken
Data reduction	Data set is high dimensional	Reduced size
Data Discretization and summarization	Data continuous	Simplified data sets

Table 1 : Data preprocessing method's irregularities & their output

To perform external sorting efficiently, first data preprocessing should be applied which process the initial data set and produce quality data. Then, refined existing algorithm is used to sort that data. The output is sorted data elements in less time compared to earlier external sorting techniques.

Data preprocessing separates the data based on the type such as numeric, alphabetic, and string type.

Based on type, sorting is applied but this preserves original count of content.

B. Efficient Pattern Matching: In this data structure, a pattern is searched in the huge given text, outputs the index of the pattern. There are existing methods used which are all searches the pattern only once whether it exists in multiple places.

The related works on this are Brute Force which compare pattern with every character of the text until match is found, Boyer more which computes last occurrence function that decides from which index in the text, the pattern going to be searched, KMP computes failure function that also determines from which position in the text, the pattern going to be searched. All these search the pattern only once in text. The other method Robin Karp used to search the pattern in the text and produce indices for the pattern that was found in the text and also search multiple patterns in the text. The disadvantages are time became worst when pattern is of large size, works by converting text and pattern into decimals, and many patterns have same has function.

The following table lists existing methods used for pattern matching:

PM Algorithm	Computation	Time Complexity
Brute Force	Comparing with every Text index	O(n*m)
Boyer Moore	Last Occurrence table	O(|E|+n*m)
KMP	Failure function	O(n+m)
Rabin Karp Method	Converting pattern and text into decimals	More time complexity depends on the context
BWT Method	Rotating the block of text	
Regular Expressions	Effort to learn the style for each new language	

TABLE 2: Drawbacks of Pattern Matching Algorithms

The time complexity is reduced by introducing data preprocessing on the initial data set and refined pattern matching with the help of one time look indexing is used. This maintains the indices for the substrings of given pattern. This history helps to search a pattern in single time or in very less time compared to earlier methods.

C. Optimal Binary Search Trees: In this data structure, many unnecessary attributes are computed. This is going to be reduced by using post dynamic programming. Initially, the clean data set is going to be obtained using data preprocessing method. The time complexity taken by this method is reduced significantly.

The related works on this are as follows:

Technique	Advantage	Disadvantage
Randomization	Simple, easy and mandatory task	Takes more time in giving right tree with minimum cost
Using sets	Optimal sub-structures	Expensive in avoiding overlapping sub-problems
Technique	**Advantage**	**Disadvantage**
Greedy	Guarantee the optimal in each case	Using Recursion cause
Traditional Dynamic Programming	Giving a tree optimally	Leads many Unnecessary computations
Proposed Dynamic Programming(Post DP)	Gives a tree optimally with little time complexity	NIL

Table3: Related Methods advantages and disadvantages for OBSTs

To perform efficient construction of OBST, First data preprocessing is applied on the initial data set that produce clean data set. Later, Refined Dynamic Programming is developed which is also called post Dynamic Programming. It has same characteristics as Dynamic Approach but it starts computing from best attribute that gives minimum cost, the next nodes are computed from there onwards.

II. Proposed Methodology

The existing algorithms used over external sorting, pattern matching, and optimal binary search trees are refined and data preprocessing algorithm is developed before construction of data structure. The Pseudo codes for each case are as follows:

A. Development of Data preprocessing and refinement of existing external sorting:

First data preprocessing is applied on the initial data set and later refined pattern matching is applied where former gives clean data without noise and latter gives history of indices first and then finds the pattern in the given text in O(1) time.

These procedures are described neatly through the following flow charts.

The flowchart to apply data preprocessing is as below:

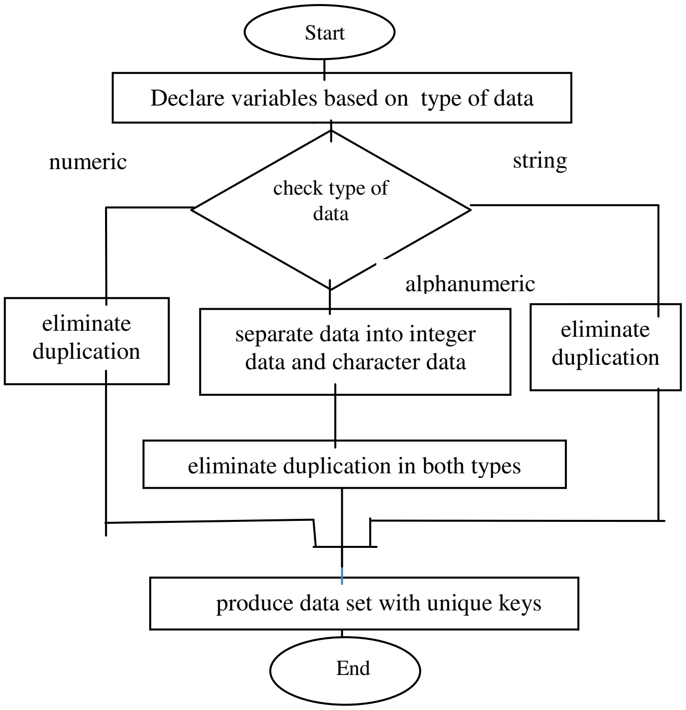

Start
Declare variables based on type of data
numeric
check type of data
string
alphanumeric
eliminate duplication
separate data into integer data and character data
eliminate duplication
eliminate duplication in both types
produce data set with unique keys
End

Figure 1: Development of Data Preprocessing methodology

The flowchart to perform external sorting is as below:

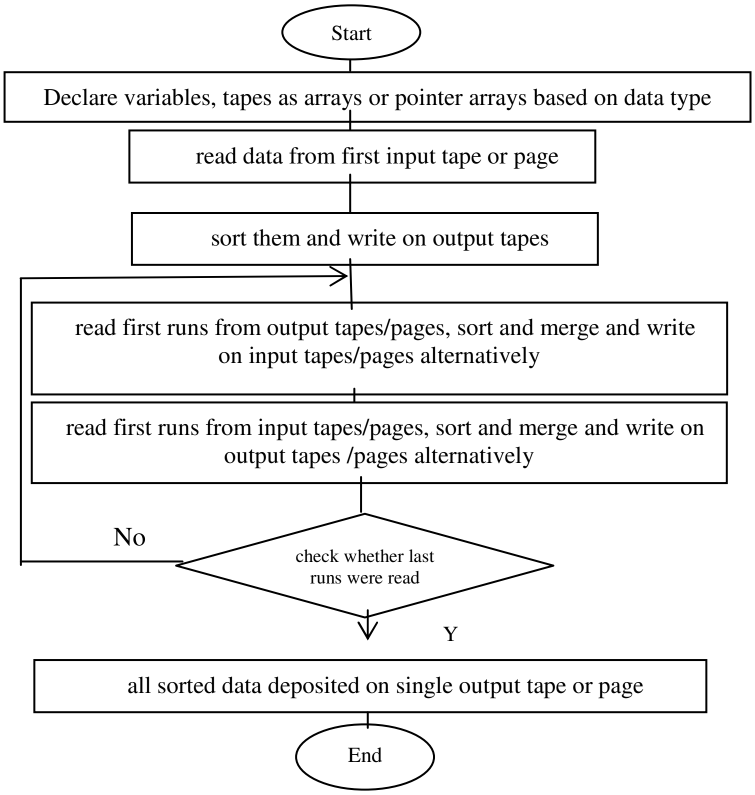

Start
Declare variables, tapes as arrays or pointer arrays based on data type
read data from first input tape or page
sort them and write on output tapes
read first runs from output tapes/pages, sort and merge and write on input tapes/pages alternatively
read first runs from input tapes/pages, sort and merge and write on output tapes /pages alternatively
No
check whether last runs were read
Y
all sorted data deposited on single output tape or page
End

Figure 2 : External Sorting Steps

B. Development of data preprocessing method and refining existing algorithm as Dynamic Pattern Matching algorithm with the help of onetime look indexing:

In this, first data preprocessing applied on the initial data set which produce clean data set. Later, refined pattern matching along with onetime look indexing is used which reduce time complexity significantly greatly.

The following flow chart gives the working procedure:

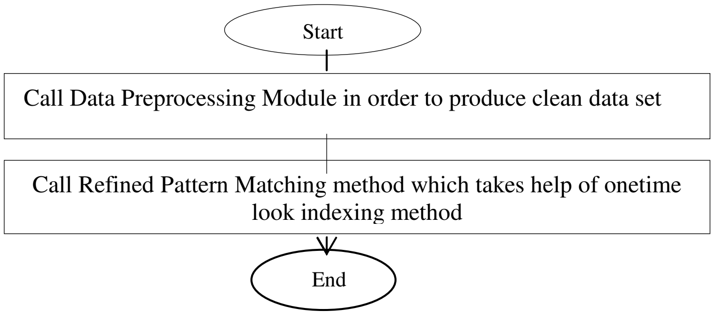

Start
Call Data Preprocessing Module in order to produce clean data set
Call Refined Pattern Matching method which takes help of onetime look indexing method
End

Figure 3: Procedure of Efficient Pattern Matching

First step, the data preprocessing works as follows:

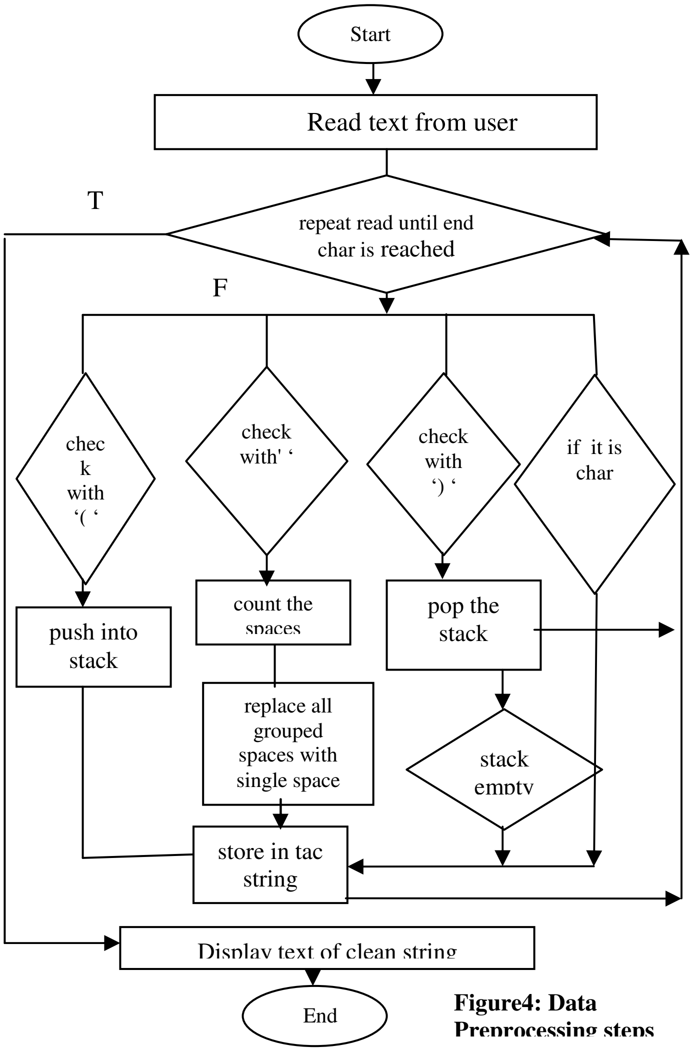


Figure4: Data Preprocessing steps

The dynamic pattern matching procedure is depicted in the following flow graph which takes the help of one time look indexing method is as follows:

The work of D-PM is depicted in and is as follows:

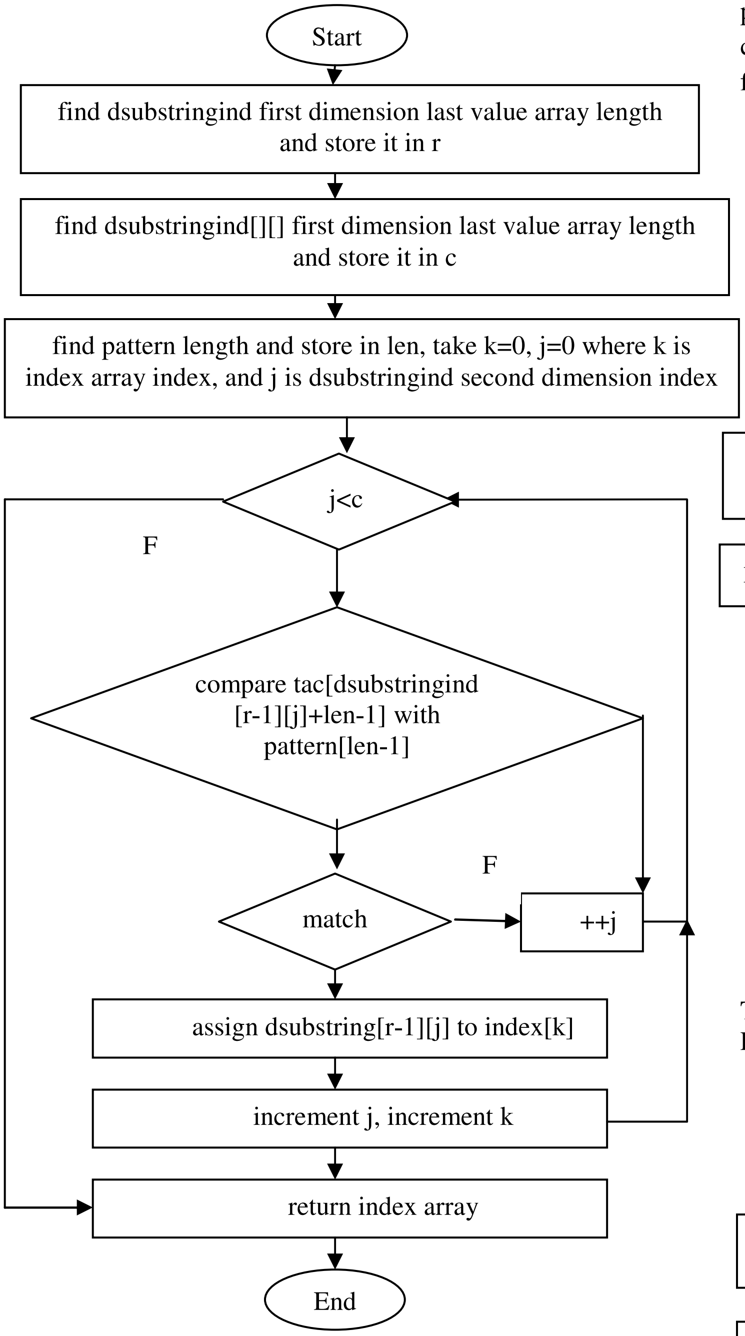

Start
find dsubstringind first dimension last value array length and store it in r
find dsubstringind[][] first dimension last value array length and store it in c
find pattern length and store in len, take k=0, j=0 where k is index array index, and j is dsubstringind second dimension index
j<c
F
compare tac[dsubstringind [r-1][j]+len-1] with pattern[len-1]
F
match
++j
assign dsubstring[r-1][j] to index[k]
increment j, increment k
return index array
End

Figure 5: Dynamic Pattern Matching with the help of one time look indexing

C. Development of Data preprocessing and refinement of optimal binary search trees:

The initial data set can be refined first using data preprocessing which eliminates redundancy and separates the elements of different type. Later, Post dynamic programming is applied which avoid unnecessary computations and reduce computation time greatly. The following illustrates data preprocessing:

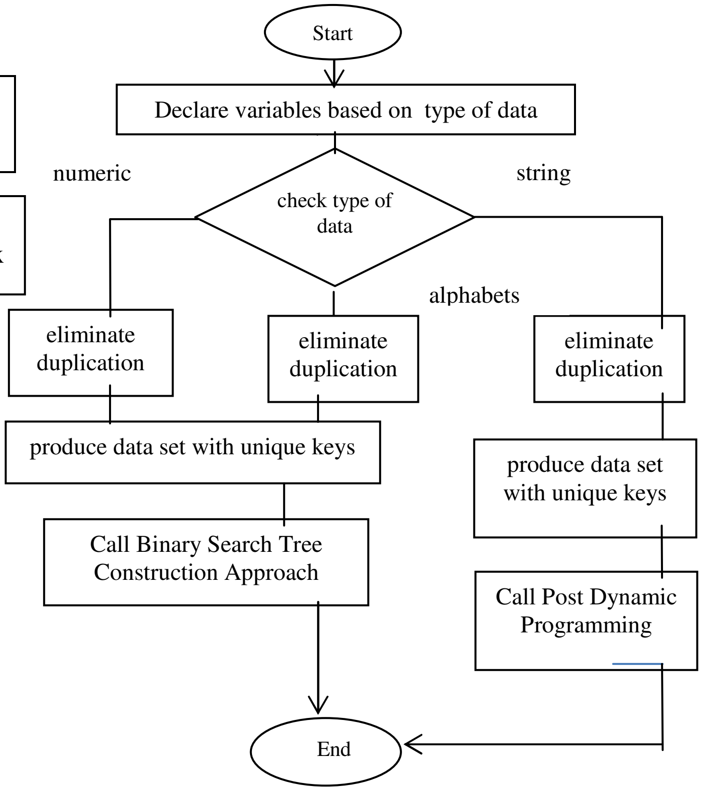
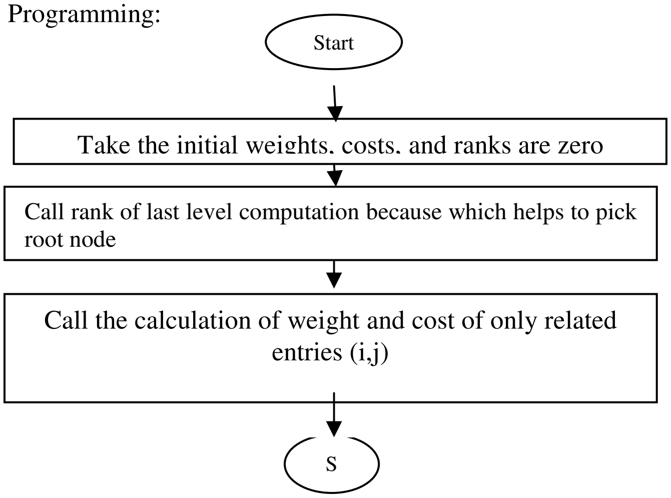

Start
Declare variables based on type of data
numeric
string
check type of data
alphabets
eliminate duplication
eliminate duplication
eliminate duplication
produce data set with unique keys
produce data set with unique keys
Call Binary Search Tree Construction Approach
Call Post Dynamic Programming
End

Figure 6: Application of Data Preprocessing approach

The following demonstrates the post dynamic Programming:


Start
Take the initial weights, costs, and ranks are zero
Call rank of last level computation because which helps to pick root node
Call the calculation of weight and cost of only related entries (i,j)
S

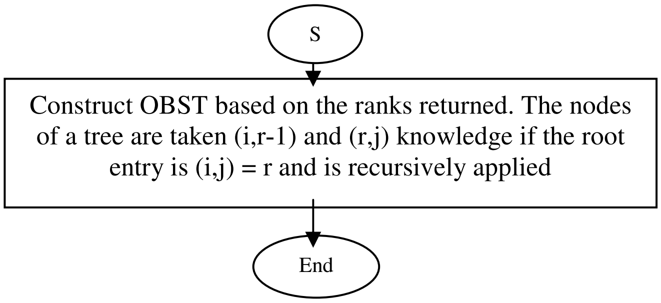

S
Construct OBST based on the ranks returned. The nodes of a tree are taken (i,r-1) and (r,j) knowledge if the root entry is (i,j) = r and is recursively applied
End

Figure 7: Post Dynamic Programming

III. Results and Comparison

The results for each data structure is illustrated based on existing and refinement of existing methods based on examples. The examples are taken from Results column in the papers for external sorting, for pattern matching, and for post dynamic computation. The time complexities observed for existing and proposed methods are demonstrated below:

Data Structure	Existing Method Drawbacks, time complexities incurred	Refinement of existing method using data preprocessing advantages and their time computations
External Sorting	More no. of input and output costs, no. of runs, no. of passes. **For redundancy involved in example Considered:** 2*N* (log $_B$ (N/M) + 1)= 120 disk accesses, 7 runs, 4 passes	Redundancy eliminated, Minimum no. of passes, runs, disk accesses are consumed. The estimated costs are 36 disk accesses, 2 runs, 3 passes
Pattern Matching	Unable to process for second time pattern search in single search by certain methods, one method search multiple patterns but involve overhead **For considered Example:** Brute Force:21 Boyer Moore:10 KMP:20 Robin Karp:21	Able to search given pattern in the text in al occurrences and also produce history of substring indices for future need. **For considered Example:** Dynamic PM using one time look indexing: O(1)

Optimal Binary Search Trees	More space, more computation time
	For Considered Example:
	Space= 45*sizeof(item)
	Time= 4 * sizeof(item)
		Less space, less computation time.
		For Considered Example:
		Space: O(15*3)=45
		Time: O(4 * 3 + 5 for initial variables + some intermediate variables assume k) < 45

Table 3.1: Comparison among existing and proposed methods

IV. Conclusion

The background data structures used in most of applications such as banking where external sorting is used, finding a particular candidate gas details in huge gas members information or finding a pan data in huge information where pattern matching is used, construction of real time scenarios for a lot of detailed descriptions where optimal binary search trees are used. These are also used in most of other suitable applications. Implementation of these might helpful to apply in many new applications as indirect data structures. For external sorting, data preprocessing eliminates redundancy and preserves the data set. Then, a refined algorithm is applied separately for numeric, alphabets, and strings. For redundant data, existing algorithm won't have capability to process the data set in certain odd times and proposed algorithm applied merge sorting separately on the each category of different data items. For Pattern matching, existing algorithms won't find a pattern for second time in single attempt by certain algorithms and involve overhead by Robin Karp method although it search multiple patterns at a time. The time complexity consumed by proposed Dynamic Approach is significantly reduced and history of the substrings of the given pattern also recorded. For OBSTs, the existing methods take more time and involve space for unnecessary computations. The proposed initially eliminate the redundancy in the data set and construct the efficient OBST for the given strings and binary trees for numeric and alphabets but designing OBST for the given strings is done in less time and space also saved using Post Dynamic Computing.

V. References:

[1] Mark Allen Weiss, "Data Structures and Algorithm Analysis in C++", Fourth Edition, Chapter7, Page No (297 – 347).

[2] Mark Allen Weiss, "Data Structures and Algorithm Analysis in Java" , Third Edition, Chapter7, Page No (297 – 347).

[3] Alfred V. Aho, John E. HopCroft and Jelfrey D. Ullman, "Data Structures and Algorithms", Sorting, Addison – Wesley, 1983.

[4] Micheline Kamber and Jiawei Han, "Data Mining Principles and Techniques", Data Preprocessing, Morgan Kaufmann, 2006, Page No (13 -30)

[5] Margaret H Dunham, "Data Mining Introductory and Advanced Topics", Pearson Education, 3e, 2008

[6] Sam Anahory and Dennis Murray, "Data Ware housing in the Real World", Pearson Education, 2003

[7] D. E. Knuth (1985), "The Art of Computer Programming", Sorting and Searching, Vol. 3, Addison –Wesley, Reading, MA, 1985

[8] Alok Aggarwal, Jeffrey Scott Vitter, "Algorithms and Data Structures", Input and Output Complexity of Sorting and related problems, AV88.pdf.

[9] Leu, Fang-Cheng; Tsai, Yin-Te; Tang, Chuan Yi, "An efficient External Sorting Algorithm", pp (159-163), Information Processing Letters 75 2000.

[10] Ian H. Witten, Eibe Frank, Morgan Kaufmann, "Data Mining: Practical Machine Learning Tools and Techniques", Second Edition (Morgan Kaufmann Series in Data Management Systems), 2005.

[11] Zhi – Hua Zhou, Dept. of CSE, Nanjing University , "Introduction to Data Mining", part3: Data Preprocessing, Spring 2012, Pt03.pdf.

[12] Chiara Rebso, "Introduction to Data Mining: Data Preprocessing", KDD- LAB, ISTI – CNR, Pisa, Italy.

[13] Michael T.Good Rich, Roberto Tamassia,"Data Structures and Algorithms in java",6th Edition.

[14] Akepogu Ananda Rao, Radhika Raju Palagiri, "Data Structures and Algorithms using C++ ".

[15] Donald Adjeroh, Timothy Bell, Amar Mukharjee, "The Burrows Wheeler Transform".

[16] Machael McMillan," Data Structures and Algorithms using Visual Basic.NET".

[17] Svetlana, Eden,"Introduction to String Matching and modification in R using Regular expressions", march,2007.

[18] Jeffrey.E.F.Fredl, "Mastering Regular Expression" , 3rd Edition, 3rd Edition, O'reilly publications.

[19] "Regular expressions and Matching in Modern Perl", 2011-12 edition.

[20] S. S. Sheik,Sumit K. Aggarwal, Anindya Poddar, N. Balakrishnan, K. Sekar,.,"A FAST Pattern Matching Algorithm", J. Chem Inf. Comput. Sci. 2004, 44, 1251-1256.

[21] Micheline Kamber, Jiawei Han, " Data Mining Concepts and Techniques", Second Edition.

[22] Dorian Pyle, "Data preparation for Data Mining", Morgan Kaufmann Publishers, Inc.

[23] Pang-Ning Tan, Vipin Kumar, Michael Steenbach, "Introduction to Data Mining", Addition-Wesley Companion book site, Page No (19 – 88).

[24] E.Horotiwz, S.Sahni, Dinesh Mehta, "Fundamentals of data structures in C++" , Second Edition.

AUTHORS PROFILES:

Mr. S. HrushiKesava Raju, working as a Associate Professor in the Dept. of CSE, SIETK, Narayanavanam Road, Puttur. He is pursuing Ph.D from Rayalaseema University in the area " Development of Data preprocessing on certain advanced Data Structures by refining their existing algorithms for getting improved time complexities". His other areas of interest are Data Mining, Data Structures, and Networks.

E-mail: hkesavaraju@gmail.com

Dr. M.Nagabhushana Rao, working as Professor & Head of the department in the Dept. of IT, SRK Institute of Technology, Vijayawada, A.P. He had completed Ph.D from S.V. University in the area of Data mining. He is presently guiding many scholars in various disciplines.