String Matching Algorithms and their Applicability in various Applications

Nimisha Singla, Deepak Garg

Abstract- In this paper the applicability of the various strings matching algorithms are being described. Which algorithm is best in which application and why. This describes the optimal algorithm for various activities that include string matching as an important aspect of functionality. In all applications test string and pattern class needs to be matched always.

Keywords: Databases, Dynamic programming, Search engine, String matching algorithms

I. INTRODUCTION

The problem of string matching is that there are two strings one is text T [1…..n] i.e. is main string given and the other is pattern P [1…….m] i.e. is the given string to be matched with the given main string given m<=n. String matching is in variably used in real word applications like Database schema, Network systems. There are main 2 techniques of string matching one is exact matching (Needleman Wunsch, Smith Waterman, KMP, Dynamic Programming, BMH) and other is approximate matching (fuzzy string searching, Rabin Karp, Brute Force**)**. Various string matching algorithms are used to solve the given above problems like wide window pattern matching, approximate string matching, polymorphic string matching, string matching with minimum mismatches, prefix matching, suffix matching, similarity measure, longest common subsequence (dynamic programming algorithm), Boyer Moore Horspool (BMH), Brute Force, Knuth Morris Pratt (KMP), Quick search, Rabin Karp.

Some of the applications are Text editors in computing machines, Database queries, Bioinformatics and Cheminformatics, two dimensional mesh, network intrusion detections system, wide window pattern matching (large string matching), music content retrievals, language syntax checker, ms word spell checker, matching DNA sequences, digital libraries, search engines and many more applications.

Combinatorial pattern matching addresses issues of searching and matching strings and more complicated patterns such as trees, regular expressions, graphs, arrays and point sets. The goal is to derive non trivial combinatorial properties for such structures and then exploit these properties in order to achieve improves performance.

This area is expected to grow even more due to increasing demand of speed and efficiency that comes from molecular biology, information retrieval, pattern recognition, compiling, data compression, program analysis and security. Also long strings create natural variations and random noise.

II. RELATED WORKS/DEFINITIONS

A brief introduction to all the basic algorithms of string matching:-

2.i *Approximate String Matching Algorithm (fuzzy string searching):* It is the technique of finding strings that match a pattern approximately (rather than exactly). The problem of approximate string matching is typically divided into two sub problems: finding approximate substring matches inside a given string and finding dictionary strings that match the pattern approximately.

2.ii *Rabin Karp Algorithm:* It is a string searching algorithm that uses hashing to find any one of a set of pattern strings in a text. For text of length *n* and *p* patterns of combined length *m*, its average and best case running time is O (*n+m*) in space O (*p*), but its worst-case time is O (*nm*).A practical application of Rabin Karp is detecting plagiarism. Rabin Karp can rapidly search through a paper for instances of sentences from the given source material, ignoring details such as case and punctuation.

2.iii *Needleman Wunsch Algorithm:* It performs a global alignment on two sequences. It is commonly used in bioinformatics to align protein or nucleotide sequences. The Needleman Wunsch algorithm is an example of dynamic programming, and was the first application of dynamic programming to biological sequence comparison.

2.iv *Knuth Morris Pratt Algorithm:* KMP string searching algorithm searches for occurrences of a "word" W within a main "text string" T by employing the observation that when a mismatch occurs, the word itself embodies sufficient information to determine where the next match could begin, thus bypassing re-examination of previously matched characters.

2.v *Dynamic Programming Algorithm (longest common subsequence (LCS) problem):* It is to find the longest subsequence common to all sequences in a set of sequences. The subsequence is different from a substring, It is a classic computer science problem, the basis of different (a file comparison program that outputs the differences between two files), and has applications in bioinformatics.

Manuscript received January 10, 2012.

Nimisha Singla, Department of Computer Science, Thapar University, Ludhiana, India, nimishasingla@gmail.com

Dr.Deepak Garg, Department of Computer Science, Thapar University, Patiala, India, dgarg@thapar.edu

2.vi *Algorithm using Levenshtein Distance:* Levenshtein distance is a metric for measuring the amount of difference between two sequences (i.e. an edit distance). The term edit distance is often used to refer specifically to Levenshtein distance. The Levenshtein distance between two strings is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character.

2.vii *Smith Waterman Algorithm:* It is a well-known algorithm for performing local sequence alignment; that is, for determining similar regions between two nucleotide or protein sequences. Instead of looking at the total sequence, the Smith Waterman algorithm compares segments of all possible lengths and optimizes similarity measure.

2.viii *Brute Force Algorithm:* It is also known as proof by exhaustion, also known as proof by cases, perfect induction, or the brute force method, is a method of mathematical proof in which the statement to be proved is split into a finite number of cases and each case is checked to see if the proposition in question holds. A proof by exhaustion contains two stages: proof that the cases are exhaustive; i.e., that each instance of the statement to be proved matches the conditions of (at least) one of the cases and a proof of each of the cases.

2.ix *Boyer Moore Algorithm:* It is a particularly efficient string searching algorithm, and it has been the standard benchmark for the practical string search literature. The algorithm preprocesses the target string (key) that is being searched for, but not the string being searched in (unlike some algorithms that preprocess the string to be searched and can then amortize the expense of the preprocessing by searching repeatedly). The execution time of the Boyer Moore algorithm, while still linear in the size of the string being searched, can have a significantly lower constant factor than many other search algorithms: it doesn't need to check every character of the string to be searched, but rather skips over some of them. Generally the algorithm gets faster as the key being searched for becomes longer. Its efficiency derives from the fact that with each unsuccessful attempt to find a match between the search string and the text it is searching, it uses the information gained from that attempt to rule out as many positions of the text as possible where the string cannot match. BMH approach uses only the Bad Character Heuristic of Boyer Moore for skipping comparisons rather than both the Bad Character and Good Suffix Heuristics.

2.x *Jaccard Similarity:* It is a measure of the similarity between two binary vectors. It can be simply used to measure the similarity of the strings.

Standard Boyer-Moore bad character shift	Before Shift: p -> six plus two * t -> two plus three equals five After Shift: p -> six plus two * t -> two plus three equals five The characters are matched starting at * and compared in right to the left manner. The whole pattern shift one by one along the text to be searched from left to right. The first only comparison fails on the 'r'. No r exists in the pattern; it can be shifted by 12 characters as shown. The next comparison begins after shifting and at the second *.
Standard Boyer-Moore repeated substring shift	Before Shift: p ->two plus two * t -> count to two hundred four After Shift: p -> two plus two * t ->count to two hundred four The characters are matched starting at * and continues examining characters from right to left. It fails on second 'o' from the text. Since two is a repeated substring in the pattern, the pattern is shifted by 9 characters so that the two of pattern is aligned with the matching part of the text. The next comparison begins after shifting and at the second *.

III. VARIOUS ALGORITHM ANALYSIS TABLE

Algorithm name	Comparison Order	Preprocess time complexity	Search time complexity	Main characteristic
Boyer Moore	Right to Left	O(m+n)	O(mn)	Use both good suffix shift and bad character shift

BM Horspool	Not relevant	O(m+n)	O(mn)	Use only bad character shift with rightmost character
Brute Force	Not relevant	No preprocessing	O(mn)	Use one by one character shift. Not an optimal one
KMP	Left to Right	O(m)	O(m+n)	Independent of alphabet size, use the notion of border of the string, increases performance, decrease delay and decrease time of comparing
Quick Search	Not relevant	O(m+n)	O(mn)	Use only bad character shift, very fast
Rabin Karp	Left to Right	O(m)	O(mn)	Use hashing function, very effective for multiple pattern matching in 1D matching
Approximate string matching	Not relevant	-	O(mn)	First matching approximate then searching dictionary
Needleman Wunsch	Left to Right	-	O(mn)	Global alignment of proteins and nucleotides
Smith Waterman	Left to Right	-	O(mn)	Align local sequence in segments of proteins and nucleotides

IV. APPLICATIONS WITH THEIR OPTIMAL ALGORITHMS

3.i *Text Editor, Digital Library and Search Engine:* Every person uses a text editor and every user of a digital library or search engine, needs to find patterns in a text. The Boyer Moore algorithm is directly implemented the search command of practically all text editors. The longest common subsequence dynamic programming algorithm is implemented in system commands that test differences between files.

3.ii *Multimedia and Computational Biology:* It has shown that a much more generalized theoretical basis of pattern matching could be of tremendous benefit. Pattern matching has to adapt itself to increasingly broader definitions of matching. In computational biology one may be interested in finding a close mutation, in communications one may want to adjust for transmission noise, in texts it may be desirable to allow common typing errors. In multimedia one may want to adjust for loss compressions, occlusions, scaling, affine transformations or dimension loss. The largest overlap heuristic for finding the shortest common superstring has been used in DNA sequencing. The algorithms that are used in Needleman Wunsch and Smith Waterman.

3.iii *Medical Tests:* The BMH algorithm achieves the best overall results when used with medical tests. This algorithm usually performs at least twice as fast as the other algorithms tested. The time performance of exact string pattern matching can be greatly improved if an efficient algorithm is used. Considering the growing amount of text handled in the electronic patient record, it is worth implementing this efficient algorithm.

3.iv *String Prefix Matching Problem:* this refers to the matching the prefixes of the pattern and the text. It also checks the longest prefix of some given sequence/text. This occurs at the starting of the patterns. It also includes preprocessing of the pattern. KMP algorithm and deterministic sequential comparison model are used to solve this problem. This can be done by assigning the lower and the upper bounds of the prefix to be matched.[1]

3.v *Polymorphic String Matching:* In this some basic string matching algorithm are combined to make one or more efficient algorithms for further computation as in fusion of the algorithms some functions become easy to define and some of the time consuming parameters need not be used.

3.vi

3.vii In this data representation technique used is tree. One example is KMP and Boyer Moore fusion. There combination completes the task in amortized constant time and cost is also equal to equality check cost. The quadratic time is dropped. The features of both the algorithms are combined and used for producing a better functional algorithm.[2]

3.viii *Network Intrusion Detection System:* This problem requires exact pattern matching technique. This is open source IDS snort. It reduces computational time and higher order of context matching is performed. To solve this Boyer Moore algorithm is used as it needs exact matching of the given pattern. To achieve this tree data structure is used in modified algorithms that convert the bad character shift to good prefix shift and resulting in far better performance.[3]

3.ix *Denoising of the Pattern:* When the text is long it creates noise in the pattern. To remove these denoise technique i.e. DUDE algorithm is used for this. It comprises of simple string matching technique with radix sort for getting a noise free sequence. It has 2 passes: 1st collects the empirical probability distribution and observe the noise pattern within the double sided window and in 2nd pass those noise patterns are replaced by denoise pattern/sequence.[4]

3.x *Approximate Similarity Measure:* This algorithm is used in real world applications. This is for finite length strings. It searches the optimal alignment for pattern searching. It is far-far better than the any other algorithms. In the previous similarity measure algorithm the variations in the pattern decreases but now in the new proposed algorithm the numbers of the comparisons are reduced. This is done by picking only those strings from the database whose lengths are either equal or greater than the pattern to be matched. Time variation depend on the length of query string, L and number of strings in the database of the length U where (L-L/2) <=U<= (L+L/2).[5]

3.xi *Retrieving Music Pattern from Musical Database:* When musical note from musical database are to be retrieved then we need string matching. There are four similar techniques for this: edit distance, dice similarity, jaccard similarity and cosine similarity. The musical notes are retrieved by QBE (query by example) approach. So the best scheme for this problem is Levenshtein distance with jaccard similarity. This is an approximate music search technique. As the jaccard similarity performs excellent in passing a query when a pitch change scenario is selected.[6]

3.xii *String Matching through Two Dimensional Meshes:* This is the optimal time algorithm that is used for computational biology, cellular automata. It reduces time spent on the comparisons in string matching by using mesh or 2D matrix. Data partition technique is used here. The process is as find the occurrences of pattern, pattern preprocessing, text searching, decomposing the pattern given and finally creating sparse table. The maximum time will be less than half of the size of the string to be matched. This is a optimal algorithm for the mesh network structure.[7]

3.xiii *Bioinformatics and Cheminformatics:* The string matching is also used in Bioinformatics and Cheminformatics. Local data warehouse in which genes, DNA, proteins and chemists information is stored is present. Chemicals are represented by using lines representing chemical bond between atoms and shaped on 2D structural formulae known as SMILES representation in the NMRShiftDB (database for organic structures). Linear formulae are better read by computer than graphical structures. Nowadays the string matching problem received an enormous deal of attention due to its various applications in computational biology. It include some steps like searching the antimicrobial structures followed development of local data warehouse followed by development of user interface for replying queries and finally checking the presence of the antimicrobial structures in the local data warehouse. The interface tool used is JME editor.[8]

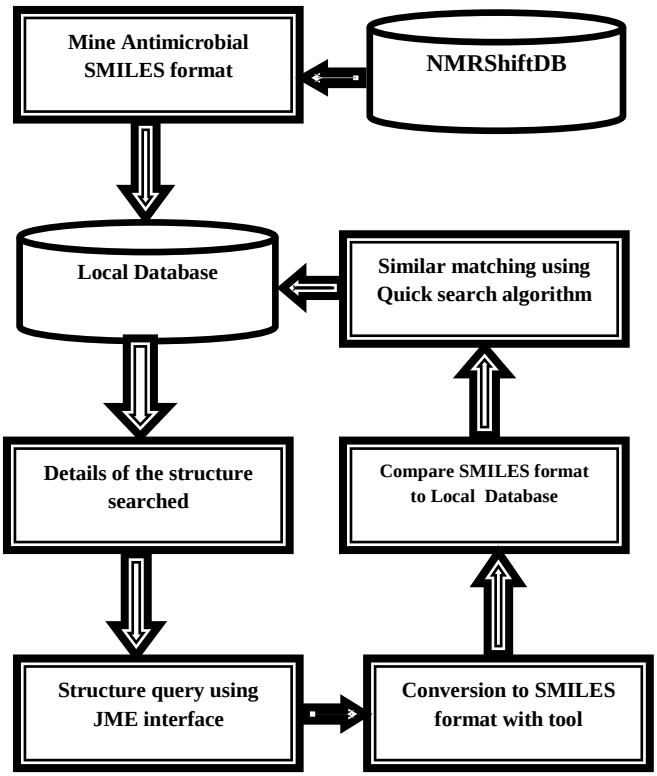

Fig. Bioinformatics and Chemin-formatics

V. CONCLUSION

In today's online scenario finding the appropriate content in minimum time is most important. String algorithms play a vital role for this. Different groups are working on software and hardware levels to make pattern searching faster. By applying various algorithms in various applications the approximate best algorithm for different applications is determined. The recommended algorithms give the reduced complexity and also reduced computation time. The algorithms assigned to various applications may not be best optimal algorithm but better than the general algorithms. Rather than applying each algorithm to every application one application is explained with particular optimal algorithm. And it has been noted that most applications uses Boyer Moore, BMH or KMP algorithms for their effective and efficient functionality and other applications uses

the basics of these algorithms for their functionalities as the KMP algorithm has less time complexity and Boyer Moore and BMH algorithms has preprocessing time complexity less. Other algorithms depends upon the type of input and is efficient for certain or particular application.

REFERENCES

1. Dany Breslauer, Livio Colussi and Laura Toniolo, 'Tight Comparison Bounds for the String Prefix Matching Problem', Stiching Mathematisch Centrum, Amsterdam, 1-9, 1992.
2. Richard S. Bird, 'Polymorphic String Matching', 110-115, Haskell'05 2005, Tallin, Estonia.
3. C. Jason Coit, Stuart Staniford and Joseph McAlerney, 'Towards Faster String Matching for Intrusion Detection or Exceeding the Speed of Snort', 1-7.
4. S. Chen, S. Diggavi, S. Dusad and S. Muthukrishnan, 'Efficient String Matching Algorithms for Combinatorial Universal Denoising',1-10.
5. Narendra Kumar, Vimal Bibhu, Mohammad Islam and Shashank Bhardwaj, 'Approximate string matching Algorithm', International Journal on Computer Science and Engineering, Vol. 02, No. 03, 2010, 641-644.
6. Yu-lung Lo Chien-Chi Huang, 'Fault Tolerant Music Retrieval by similar String Matching', National Science Council of ROC Grant NSC98-2221-E-324-027,1-10.
7. S. Viswanadha Raju and A. Vinayababu. 'Optimal Parallel algorithm for String Matching on Mesh Network Structure', International Journal of Applied Mathematical Sciences ISSN 0973-0176 Vol.3 No.2 2006, 167-175.
8. Ahmad Fadel Klaib, Zurinahni Zainol, Nurul Hashimah Ahamed, Rosma Ahmad, and Wahidah Hussin, 'Application of Exact String Matching Algorithms towards SMILES Representation of Chemical Structure', World Academy of Science, Engineering and Technology 34 2007,36-40.
9. Venkata Padmavati Metta, Kamala Kritivasan, Deepak Garg, "On String Languages Generated by SN P Systems with Anti-Spikes", International Journal of Foundations in Computer Science, World Scientific May 2011.
10. www.cs.biu.ac.il.
11. Venkatesan T. Chakaravarthy, Rajasekar Krishnamurthy, 'The Problem of Context Sensitive String Matching', 1-12.

AUTHORS PROFILE

Nimisha Singla has received her Bachelor of Technology in Computer Science & Engineering from Punjab Technical University, Jalandhar, India. She is pursuing her Master in Engineering in Computer Science & Engineering from Thapar University, Patiala, India. She is Secretary of Student Chapter of Technical Society viz. Association for Computing Machinery (ACM). Her main research interests include: String matching problems.

Dr.Deepak Garg is Chair ACM SIGACT North India and Secretary of IEEE Computer Society, Delhi Section. He is senior member of IEEE and senior member of ACM. He is Life Member of CSI, IETE, ISCA and ISTE. He has undertaken research projects funded of Indian Govt. He is on various advisory boards and technical committees of International workshops and conferences. He is a certified on various technologies. He is guiding many MTech and PhD students. His active area of research is advance algorithm design and theory of computer science. He has more than 80 Publications in International Journal and Conferences of Repute. He is Faculty at Computer Science and Engineering Department at Thapar University, Patiala.