ISSN:2319-9873

Research & Reviews: Journal of Engineering and Technology

KareemNaaz-Vasavi (KV) Pattern Matching Algorithm

Shaik Kareem Basha* and G. Vasavi

Department of CSE, HITAM, India

Research Article

Received date: 07/11/2015

Accepted date: 07/12/2015

Published date: 12/12/2015

***For Correspondence**

Shaik Kareem Basha, Assistant professor, Department of CSE, HITAM, India

E-mail: kareembashas.cse@hitam.org

Keywords: Algorithm, Program, Software, Development.

ABSTRACT

In Computer Science, pattern matching is the process of checking a given sequence of tokens for the presence of pattern. The patterns generally have the form of either sequences or tree structures. Uses of pattern matching includes outputting the locations of a pattern within a token sequence, to output some component of the matched pattern and to substitute the matching pattern with some other token sequence. Sequence patterns such as text string are often described using regular expressions and matched using techniques such as backtracking. Pattern matching can be used to filter data of a certain structure. It is used to find a pattern which is relatively small in a text, which is very large. Patterns and texts can be One Dimensional or Two Dimensional. In the case of One Dimensional, example can be text editor. The proposed algorithm KareemNaaz-Vasavi (KV) pattern matching algorithm is applicable to One Dimensional patterns and texts. We followed the various stages of Software Development Life Cycle to demonstrate the proposed KV pattern algorithm. Pattern matching and proposed KareemNaaz-Vasavi (KV) pattern matching algorithm are introduced in detail. The proposed algorithm is analyzed and designed. The proposed algorithm is implemented by using C programming Language. We test the implementation of proposed algorithm using different test cases.

INTRODUCTION

In Computer Science, pattern matching is the process of checking a given sequence of tokens for the presence of some pattern. In contrast to pattern recognition the match usually has to be exact. The patterns generally have the form of either sequences or tree structures. Uses of pattern matching includes outputting the location of a pattern with in a token sequence, to output some component of the matched pattern and to substitute the matching pattern with some other token sequence. Sequence patterns such as text string are often described using regular expressions and matched using techniques such as backtracking. The first computer programs to use pattern matching were text editors. The simplest pattern in pattern matching is an explicit value or a variable. More complex patterns can be built from the primitive patterns. Pattern matching can be used to filter data of a certain structure [1]. It applies to the structures of expressions. By far the most common form of pattern matching involves strings of characters. In many programming languages, a particular syntax of strings are used to represent regular expressions, which are patterns describing string characters. Pattern matching is to find a pattern, which is relatively small in a text, which is supposed to be very large. Patterns and texts can be One Dimensional or Two Dimensional. In the case of One Dimensional, example can be text editor. In the text editor, we have 26 characters and some special symbols. In the case of 2 dimensional, the typical application is a pattern matching in computer vision. Either One Dimensional or Two Dimensional, the text is very large and a fast algorithm to find the occurrence of pattern in it is needed. In this paper, the proposed algorithm KareemNaaz-Vasavi Pattern Matching Algorithm takes pattern and text as input. The indices of characters of text, which are same as first character of pattern are placed into start array, s indicates the number of characters of text, which are equal to first character of pattern. The indices of characters of text, which are same as last character of pattern are placed into end array, e indicates the number of characters of text, which are equal to last character of pattern. The difference of elements of start array

and end array is calculated and if the difference is equal to strlength(pattern)-1, then the remaining characters of pattern from indices 2 to strlength (pattern)-2 are compared in the range of indices from start[i] to end[j] within text.

ANALYSIS AND DESIGN OF PROPOSED ALGORITHMS

Figure 1 shows the Text, which consists of 26 characters "INDIAISMYCOUNTRYALLINDIANS" starting from index 0 to index 25. So, the length of text (n) is 26.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
I	N	D	I	A	I	S	M	Y	C	O	U	N	T	R	Y	A	L	L	I	N	D	I	A	N	S

Figure 1. Indices and Characters of Text.

Figure 2 shows the pattern, which is to be searched in text shown in Figure 1. It consists of 6 characters "TRYALL" starting from index 0 to index 5. So, the length of pattern (m) is 6.

0	1	2	3	4	5
T	R	Y	A	L	L

Figure 2. Indices and characters of Pattern.

In the proposed KareemNaaz-Vasavi (KV) pattern matching algorithm, we are using start array with index s. It is used to store the indices of text characters, which are same as first character of pattern. In Figure 2 T is the first character of pattern and in Figure 1, the indices of first character with in text are 13 only. So it is placed into start array, s indicates the number of indices of text characters, which are same as first character of pattern. In this example s is 1, since only one index 13 of text is placed into start array **(Figure 3).**

13

Figure 3. start array with index s.

In the proposed KV algorithm, we are also using end array with index e, which is used to store the indices of text characters, which are same as last character of pattern. In Figure 2 A is the last character of pattern and in Figure 1, the indices of last character with in text are 17 and 18. These indices of text are placed into end array, e indicates the number of indices of text characters, which are same as last character of pattern. In this example e is 2, since two indices {17,18} of text are placed into end array **(Figure 4).**

17	18

Figure 4. end array with index e.

After placing indices of text characters into start array and end array, Each element E_i (i=1 to s) of start array is substracted from each element E_j (j=1 to e) of end array. If we found difference equal to strlength(pattern)-1 ie. (E_j – E_i)= m-1. Then the remaining text characters of range (E_{i+1} to E_{j-1}) are compared with pattern characters of range (1 to m-2). **Figure 5** shows the comparision of text characters of range (14 to 17) with pattern characters of range (1 to 4). Since the Equality status of all comparisions is YES, so it means pattern is found in text.

Text Index	Text Char	Pattern Index	Pattern Char	Equality
14	R	1	R	YES
15	Y	2	Y	YES
16	A	3	A	YES
17	L	4	L	YES

Figure 5. Matching of pattern characters with text characters.

Algorithm 1 finds whether all the characters of pattern of indices (1 to m-1) are same as the characters of text of indices (s+1 to e-1). It takes two inputs s and e. s indicates the index of text character, which is same as first character of pattern and e

indicates the index of text character, which is same as last character of pattern. If all the characters of text of indices (s+1 to e-1) are matched with all the characters of pattern of indices (1 to m-1), then Algorithm 1 returns true, which means pattern exist with in text, other wise it returns false, which indicates pattern does not exist with in text [2].

In **Algorithm 1** while loop repeats from (s+1 to e) or (1 to m-1). So the overall time taken by Algorithm found(s,e) is O(m). m is the length of pattern, which is to be searched in text.

In **Algorithm 2** each element E_i (i=1 to s) of start array is substracted from each element E_j (j=1 to e) of end array. If the difference is equal to strlength(pattern)-1 ie. ($E_j - E_i$)= m-1, then the remaining text characters of range (E_{i+1} to E_{j-1}) are compared with pattern characters of range (1 to m-2). If all the characters of text of range (E_{i+1} to E_{j-1}) are same as all the characters of pattern of range (1 to m-2), then pattern exist with in text, otherwise pattern does not exist within text.

In Algorithm 2 outer loop repeats s+1 times, inner loop repeats s.(e+1) times, Algorithm found() is called s.e times. Each time found() takes O(m) time to match characters of text and pattern. The overall time taken by Algorithm 2 is O(m.s.e) ie. O(m).

Algorithm 3 takes pattern and text as input and places indices of text characters, which are same as first character of pattern into start array and also places indices of text characters, which are same as last character of pattern into end array. In Algorithm 3 s indicates the number of characters of text, which are same as starting character of pattern and e indicates the number of characters of text, which are same as ending character of pattern.

In Algorithm 3 for loop repeats n times from 0, match() call takes O(m) time. So the overall time taken by Algorithm 3 is O(n) + O(m).

IMPLEMENTATION OF PROPOSED ALGORITHM USING C

The Proposed Algorithm is implemented by using C Programming Language, which is a robust, structure oriented programming language, which provides different concepts like functions, pointers, structures, arrays and so on to solve complex problems.

TESTING OF PROPOSED ALGORITHM

Different Test cases are considered to test the result of Program 1. Consider the following Test Case in which text consists of 26 characters “indiaismycountryallindians”, starting from index 0 to index 25. Pattern consist of 6 characters “tryall”, starting from index 0 to index 5. For the given pattern, the result “pattern tryall found” is displayed [3] **(Figure 6).**

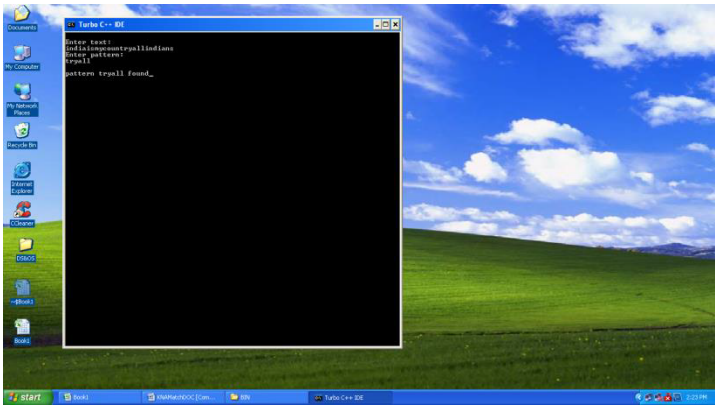

Figure 6. output screen shot of program 1.

CONCLUSION

We conclude that in the proposed KareemNaaz-Vasavi (KV) pattern matching algorithm, we are taking pattern and text as input and places indices of text characters, which are same as first character of pattern into start array and also places indices of text characters, which are same as last character of pattern into end array. each element E_i (i=1 to s) of start array is substracted from each element E_j (j=1 to e) of end array. If the difference is equal to strlength(pattern)-1 ie. ($E_j - E_i$)= m-1, then the remaining text characters of range (E_{i+1} to E_{j-1}) are compared with pattern characters of range (1 to m-2). If all the characters of text of range (E_{i+1} to E_{j-1}) are same as all the characters of pattern of range (1 to m-2), then pattern exist with in text, otherwise pattern does not exist within text.

REFERENCES

1. Aho Alfred V and Jeffrey D U. Data Structures and Algorithms. Addison Wesley, Reading, Massachusetts 1983.
2. Cormen TH, et al. Introduction to Algorithms. McGraw-Hill, New York 1990.
3. Knuth Donald E. The Art of Computer Programming, Volume 3, Sorting and Searching. Addison-Wesley, Reading, Massachusetts 1998.