Fascurate Pattern Searching Algorithm

Manoj Kumar Nama[1], Namita Mittal[2], Emmanuel Pilli[2]
[1,2]Department of Computer Engineering
[1]Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur
[2]Malviya National Institute of Technology, Jaipur
Email-[1]mk.nama@gmail.com, [2]nmittal@mnit.ac.in, [3]espilli.cse@mnit.ac.in

Received 27 February 2015, received in revised from 24 March 2015, accepted 25 March 2015

Abstract: **As there are various pattern matching algorithms are present now but if we try to make them advanced by some new algorithms than it can be made faster easily. In such way the idea is to make a new file and store it in the file system which will be containing the metadata (lengths of words) about the file. Whena system call is made, new associated file will be opened in hidden mode. As this new file is containing metadata about the file mainly containing the lengths of all the words by their position; by which the length of any word in file could be determined. When we create a file then an algorithm works which calculates lengths of all the words present in file and store the lengths in a positional way like first phrase is length of first word and 99th phrase is length of 99th word. This algorithm will work on a file a single time until it is not modified or updated. Once this file has been made after that on a search request it calculates the length of the searched word and finds that length in metadata file. The Positions at which length is matched, only those words will be checked for matching of patterns. So we do not need to compare all the words in file with searched word by which it will reduce its time complexity and searching process will be faster with some storage overhead. The most required thing in computer system is reduced time complexity meanwhile some storage overhead is allowed. Like in binary search firstly we sort the data in ascending order then the binary search is performed so over all reduced complexity is occurred.**

1. INTRODUCTION

There are two cases in this searching algorithm.

1. When a single word is searched. Example –"Laptop"

2. When a string containing a sentence or more than one word is searched. Example – "Integrated webcam" or "Fastener matching algorithm for various patterns".

<u>In the 1st case when only a single word is searched.</u>

When a single word is searched the length of the word is calculated and matched in the metadata file then all the words are selected which are having same length as searched word then those words are compared with searched word and if a match occurs it is displayed. The lengths in metadata files will be kept separated by using any specified symbol due to ambiguity issues.

<u>In the 2nd case when a string containing a sentence or more than one word is searched.</u>

When a string more than one word is searched the length of all the words is calculated and a new sting is formed which will be containing all the lengths of words, will be matched in the metadata file and if the same pattern of lengths is present in metadata file then only the comparison of words is done and if a match occurs, it is displayed.

Firstly, this is not a pattern matching algorithm besides it is a selection process of words which will be compared. In this work is totally focused on selection of words and there is no any alteration in pattern matching so as traditional methods Rabin-Carp and Boyer-Moore will be used[1-5].

What's new?

In this work new thing is selection of words which will be further fed to pattern matching algorithms. In this, words will be selected which are having same length of any searched word. In this manner all words will not be compared by which time complexity will decrease

2. PROPOSED WORK

In this work there are two algorithms.

A. Algorithm to make a new file which will contain length of all the words. (Metadata File)

B. Algorithm to select the matched length words and match them with search string.

In this work the first algorithm is just reading a file character by character and separates words on basis of occurrences of "white space", "dot (.)", "comma (,)etc. lengths of separated words are written to a new file with any separation mark like any special symbol(Here '+' is being used).

In the second algorithm selection of words is being done. In this algorithm firstly the length of the searched word is calculated and metadata file is read till "special symbol '+'". In such manner the location of matched length words is obtained. In other words; the words have been selected to be compared. So now these selected words will be compared with the searched word using any traditional approach of pattern matching.

Under this work the process is being added of selecting of words only and these algorithms do not affect any pattern matching algorithm so it can be called a pattern searching algorithm.

3. PROPOSED ALGORITHMS

As there are two algorithms

A. Mtdt algorithm:

This algorithm is responsible for creating a new binary file containing binary values separated by any special symbol (Here '+').

This binary file is human unreadable.

According to this algorithm when a white space or '.' Or ',' is encountered, length is calculated till there using counters and that integer value is stored in metadata file. Length of all words present in file will be stored, separated by any special symbol.

Example:

Pattern file: "This is advancement in pattern search".

Metadata file: "4+2+11+2+7+6" (In character view.)

So length of all words will be stored in such a manner.

This algorithm will be executed for once till the file is not updated or modified.

B. *Search algorithm:*

This algorithm is responsible for finding words which are having same length as searched string. And after finding these words, finding exactly words present in the file and to show their no. and locations.

Example:

Searched word: "algorithm"

Metadatafile : "4+9+2+4+9+2+6+5+1+5+8+9+4+3+8".

Firstly length of "algorithm" is 9 will be calculated.

This length will be calculated and will be matched in metadata file, so the matched positions are: 2,5,12.

The words on these positions will be identified and now the pattern matching will be processed on these three words only. Hence less time will be required for pattern matching process.

3.1 Mtdt Algorithm (I Algorithm)

Algorithm:Mtdt

Input: Text file in which search has to be done.

Output: create a new Binary file containing length of all words present in file.

```
1.  Begin
2.  Initialize i: =0;
3.  Open input file in read mode with file pointer *fp;
4.  Make output file in write mode with file pointer *fpn;
5.  While (! feof (fp)) do
6.  ch: =getch(fp);
7.  If ch==32 or ch=='.' or ch==',' then
8.  Putw (i, fpn);
9.  Reinitialize i: =0;
10. goto step 5;
11. end if;
12. Increase i by 1;
13. end while;
14. end;
```

3.2 Search Algorithm (II Algorithm)

Algorithm: Search.

Input: file created by mtdt algorithm and text file in which search has to be done.

Output: position and number of matched words.

```
1.  Begin;
2.  Initialize all integers with 0;
3.  print"enter to be searched";
4.  gets(ch);
5.  len =strlen(ch);
6.  Open file created by mtdt algorithm in read mode with file
    pointer *fp;
7.  while(!feof(fp)) do
8.  chnint=getw(fp);
9.  Increment i by 1;
10. end while;
11. arr=(int*)malloc(i*sizeof(int));
12. fclose(fp);
13. Re-initialize i=0;
14. Re-open file which is closed in step 11;
15. while(!feof(fp)) do
16. chnint=getw(fp);
17. if(len==chnint) then
18. Assign arr[k]:=i;
19. Increment j and k by 1;
20. end if;
21. Increment i by 1;
22. end while;
23. fclose(fp);
24. Re-initialize i:=0;
25. Open text file in which search has to be done with file
    pointer *fpn.
26. while(p!=arr[i]) do
27. chnew=fgetc(fpn);
28. if chnew==32 or chnew=='.' or chnew==',' then
29. Increment p by 1;
30. end if;
31. end while;
32. fgets(msg,len+1,fpn);
33. Increment i by 1;
34. if(strcmp(msg,ch)==0)
35. Increment m by 1;
36. end if;
37. if(p!=arr[j-1])
38. goto step 25;
39. end if;
40. print"match for string searched";
41. print"valueof m";
42. end;
```

4. PATTERN SEARCHING V/S PATTERN MATCHING

Both of the things are very different but on combining both of them make a new process with decreased time complexity which makes easy to fulfil the objective of searching.

"Pattern searching" takes place before "pattern matching".

TABLE
Comparison

S. No.	Pattern searching v/s Pattern Matching		
	Basis	Pattern searching	Pattern Matching
1	Time of execution	Before Pattern Matching	Independent
2	Space overhead	Yes	No
3	Time Complexity	Reduced	More

5. ADVANTAGEs, DISADVANTAGEs AND LIMITATIONS

The main advantage of using this is to use its reduced time complexity which will make overall process of pattern matching faster.
According to this work pattern matching will not be performed on all words of a file besides only some words will be selected to be matched.
This work has only one dis-advantage that some space overhead will be there or in other words more storage will be required.
This work also has a limitation that only exact words present, can be searched. Means if we have to find out the result in reduced time complexity then we have to search precisely.
Because the words of same lengths are searched first means the result is given only for exact words so it is to be remembered "search precisely".
As according to this study reduced time complexity in overall process of pattern matching has been obtained.
Example:
Pattern file: "This algorithm is quite fast".
Metadata file: "4+9+2+5+4".
Searched word: "algo".
The length of this searched word is 4 will be calculated and this matching to this length will be dine in metadata file.
In this case the matched positions are 1 and 5 and corresponding words to these positions are words "this" and "fast".
When pattern matching will be processed on these two words, the final outcome says no match occurs.
Here the word "algorithm" is not even selected. So we have to search as precise as possible for fast response.

6. ALGORITHM WORKING WITH AN EXAMPLE

A text file is made of abstract part of this paper which is containing 593 words.
Now if we search for the word like "algorithms"; length of this word is 10, so algorithm finds out location of all the words of length 10.so in this file there are 19 words of length 10, after that pattern matching will take place.so now pattern matching will be done on only 19 words, so processing with 19 words will be much easier and less time will be required in computation. After that, by any pattern matching algorithm those 19 words will be matched with searched word and appropriate results will be provided.
Pattern file: Abstract part of this paper.
Metadata File: After execution of mtdt algorithm.
Searched word: "Algorithms"
Firstly the length of "algorithms" will be calculated as 10.
Now the search will be done in metadata file on the length basis.
Matched length words of length: 19
Now the pattern matching will be done on only 19 words instead of 593 words.

7. CONCLUSION

This work reduced time complexity which will make overall process of pattern matching faster.According to this work pattern matching will not be performed on all words of a file besides only some words will be selected to be matched. In this work two algorithm are proposed, both of the things are very different but on combining both of them make a new process with decreased time complexity which makes easy to fulfil the objective of searching.

REFERENCES

[1] T. Lecroq, New experimental results on exact string-matching, Rapport LIFAR 2000.03, Université de Rouen, 2000.

[2] Frantisek Franek, Christopher G. Jennings, W.F. Smyth, A simple fast hybrid pattern- matching algorithm, Journal of Discrete Algorithms, Volume 5, Issue 4, December 2007, Pages 682-695, ISSN 1570-8667.

[3] J. Holub, W.F. Smyth, Algorithms on Indeterminate strings, in: 14th Australasian Workshop on Combinatorial Algorithms, 2003, pp. 36–45.

[4] Zhaoyang Qu, Xiaobo Huang, The Improving pattern matching algorithm of Intrusion detection, Procedia Engineering, Volume 15, 2011, Pages 2841-2846, ISSN 1877-7058.

[5] L. Colussi, Z. Galil, R. Giancarlo, On the exact complexity of string matching, in: 31st Symp. Found. Comp. Sci. vol. I, St. Louis, Missouri, 22–24 October 1990, IEEE, 1990, pp. 135–144.

◆ ◆ ◆