PATTERN MATCHING ALGORITHM OF FINITE AUTOMATA

INTRODUCTION:

Pattern searching is an important problem in computer science. When we do search for a string in notepad/word file or browser or database, pattern searching algorithms are used to show the search results.

PROBLEM STATEMENT:

Certain known nucleotideor amino acid sequences have properties known to biologists. E.g.,ATG is a string which must be present at the beginning of every protein (gene) a DNA sequence. A primer is a conserved DNA sequence used in the Polymerase Chain Reaction(PCR) to identify the location of the DNA sequence that will be amplified(amplification starts at the location immediately following the 3' end primer, known as the forward primer). Finding if a DNA sequence contains a specific(candidate) primer is therefore paramount to the ability to run correct PCR. Finding such modified strings is an important process for mapping DNA of a new organism, based on the known DNA of a related organism.

PROBLEM SIZE:

The input to all string matching algorithms is a pair of strings S and P. Their lengths n and m respectively, supply the problem size .Note that $m < n$. In some cases, especially in bioinformatics applications $m << n$: DNA sequences may be millions of nucleotides long, while the pattern P looking for (e.g., for primer identification) may be on the order of tens of nucleotides. In such cases, we may consider the size of input to be n, rather than $n + m$. In our studies we fix the alphabet . In bioinformatics, it makes sense since the two known alphabets nucleotide and amino acid ones have 4 and 20 characters in them respectively. Occasionally, = K may be considered part of the size of input

FINITE AUTOMATA:

The string-matching automaton is very efficient: it examines each character in the text exactly once and reports all the valid shifts in O(n) time.

A finite automaton M is a 5-tuple (Q,q0,A,S,d), where

•Q is a finite set of states.

•q0 Q is the start state.

•A Q is a distinguish set of accepting states.

• S is a finite input alphabet

• d is a function from Q × S into Q, called the transition function of M.

AIM:

To calculate the exact matching state and find the correct pattern in a set of DNA sequence.

INPUT:

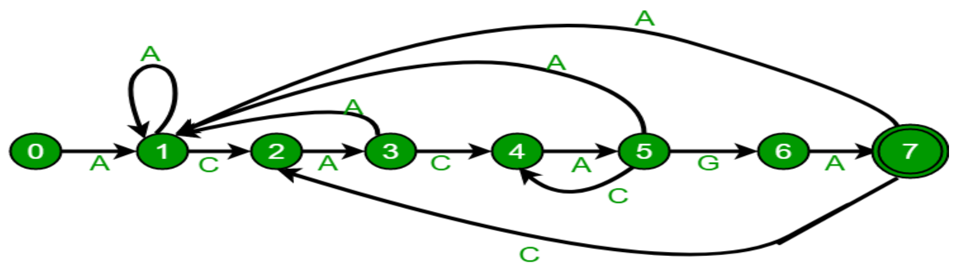
txt[] = "AABAACACAGAAAGAACCACACAGAAT"

pat[] = "ACACAGA"

PROCEDURE:

In this we will discuss Finite Automata (FA) based pattern searching algorithm. In FA based algorithm, we preprocess the pattern and build a 2D list that represents a Finite Automata. Construction of the FA is the main tricky part of this algorithm. Once the FA is built, the searching is simple. In search, we simply need to start from the first state of the automata and the first character of the text. At every step, we consider next character of text, look for the next state in the built FA and move to a new state. If we reach the final state, then the pattern is found in the text. The time complexity of the search process is O(n).

Before we discuss FA construction, let us take a look at the following FA for pattern ACACAGA.



	character			
state	A	C	G	T
0	1	0	0	0
1	1	2	0	0
2	3	0	0	0
3	1	4	0	0
4	5	0	0	0
5	1	4	6	0
6	7	0	0	0
7	1	2	0	0

•Each match sends the automation into a new state.

•If all the characters in the pattern has been matched, the automaton enters the accepting state.

• Otherwise, the automaton will return to a suitable state according to the current state and the input character such that this returned state reflects the maximum advantage we can take from the previous matching.

•The matching takes O(n) time since each character is examined once.

The above diagrams represent graphical and tabular representations of pattern ACACAGA.

Classes ,Variables and Methods were used:

Class :Pattern

Variables:

1. No of characters
2. State(iterable variable)
3. M(length of pattern)
4. N(length of txt)
5. X(iterable variable)
6. Ns(contains longest prefix)

Methods:

getNextState(pat,M,state,x) :To calculate the next state

compute TF(pat,M):Builds the TF table which represents the Finite Automata for a given pattern

search(pat,txt):Prints all occurrences of pattern in text

OUTPUT:

```
Pattern found at index: 4

Pattern found at index : 18
```

CONSTRUCTOR:

To initialize the object with the pattern,text,number of characters.

PSEUDOCODE:

```
1. Start the code.
```

2. Initialize the variable NO_OF_CHARS=256

3. Calculate the next state.

4.If the character is same as next character in pattern ,then simply incremented state.

5. Initialize i->0

6. ns stores the result which is next state.

7. ns finally contains the longest prefix which is also suffix in “pat[0..state-1]c”

8. Start from the longest possible value and stop when you find a prefix which is also suffix.

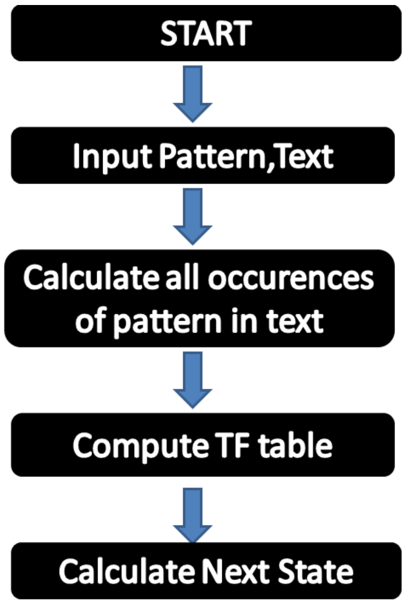
9. Next definition method “computeTF” this function builds the TF table which represents finite automata for a given pattern.

10.Next method “search” the pattern and comparison with text and print all occurences of pattern in text.

11. Process text over FA.

12. Driver program to text these functions(main program).

WORKFLOW:



Submitted by

L.Tamizhini

M.Nivethika