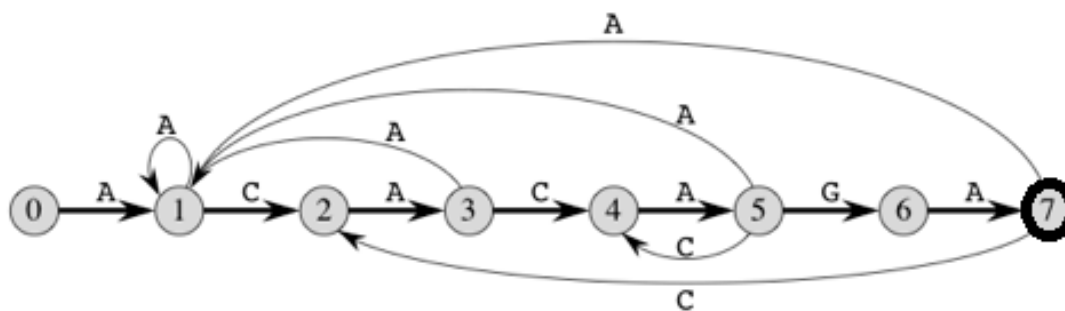


Pattern Searching | Set 6 (Efficient Construction of Finite Automata)

In the [previous post](#), we discussed Finite Automata based pattern searching algorithm. The FA (Finite Automata) construction method discussed in previous post takes $O((m^3) \cdot \text{NO_OF_CHARS})$ time. FA can be constructed in $O(m \cdot \text{NO_OF_CHARS})$ time. In this post, we will discuss the $O(m \cdot \text{NO_OF_CHARS})$ algorithm for FA construction. The idea is similar to lps (longest prefix suffix) array construction discussed in the [KMP algorithm](#). We use previously filled rows to fill a new row.



Google™ Custom Search



GeeksforGeeks



52,731 people like [GeeksforGeeks](#).



Facebook

[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

state	character			
	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

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

Algorithm:

- 1) Fill the first row. All entries in first row are always 0 except the entry for pat[0] character. For pat[0] character, we always need to go to state 1.
- 2) Initialize lps as 0. lps for the first index is always 0.
- 3) Do following for rows at index i = 1 to M. (M is the length of the pattern)
 -a) Copy the entries from the row at index equal to lps.
 -b) Update the entry for pat[i] character to i+1.
 -c) Update lps "lps = TF[lps][pat[i]]" where TF is the 2D array which is being constructed.

Implementation

Following is C implementation for the above algorithm.

```
#include<stdio.h>
#include<string.h>
#define NO_OF_CHARS 256

/* This function builds the TF table which represents Finite Automata
   given pattern */
void computeTransFun(char *pat, int M, int TF[][NO_OF_CHARS])
{
    int i, lps = 0, x;

    // Fill entries in first row
    for (x = 0; x < NO_OF_CHARS; x++)
        TF[0][x] = 0;
    TF[0][pat[0]] = 1;

    // Fill entries in other rows
```

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)



Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and](#)

[Data Packing](#)

Intersection point of two Linked Lists

Lowest Common Ancestor in a BST.

Check if a binary tree is BST or not

Sorted Linked List to Balanced BST

```
for (i = 1; i<= M; i++)
{
    // Copy values from row at index lps
    for (x = 0; x < NO_OF_CHARS; x++)
        TF[i][x] = TF[lps][x];

    // Update the entry corresponding to this character
    TF[i][pat[i]] = i + 1;

    // Update lps for next row to be filled
    if (i < M)
        lps = TF[lps][pat[i]];
}
```

```
/* Prints all occurrences of pat in txt */
void search(char *pat, char *txt)
{
    int M = strlen(pat);
    int N = strlen(txt);

    int TF[M+1][NO_OF_CHARS];

    computeTransFun(pat, M, TF);

    // process text over FA.
    int i, j=0;
    for (i = 0; i < N; i++)
    {
        j = TF[j][txt[i]];
        if (j == M)
        {
            printf ("\n pattern found at index %d", i-M+1);
        }
    }
}

/* Driver program to test above function */
int main()
{
    char *txt = "GEEKS FOR GEEKS";
    char *pat = "GEEKS";
    search(pat, txt);
    getchar();
    return 0;
}
```

Output:

```
pattern found at index 0
pattern found at index 10
```

Time Complexity for FA construction is $O(M \cdot \text{NO_OF_CHARS})$. The code for search is same as the [previous post](#) and time complexity for it is $O(n)$.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

695



Subscribe



Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

[Backtracking | Set 7 \(Sudoku\)](#) · 16 minutes ago

RVM Can someone please elaborate this Qs from above...

[Flipkart Interview | Set 6](#) · 36 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

[Software Engineering Lab, Samsung Interview | Set 2](#) · 36 minutes ago

@meya Working solution for question 2 of 4f2f round....

[Amazon Interview | Set 53 \(For SDE-1\)](#) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

Related Topics:

- [Printing Longest Common Subsequence](#)
- [Suffix Array | Set 2 \(nLogn Algorithm\)](#)
- [Rearrange a string so that all same characters become d distance away](#)
- [Recursively remove all adjacent duplicates](#)
- [Find the first non-repeating character from a stream of characters](#)
- [Dynamic Programming | Set 33 \(Find if a string is interleaved of two other strings\)](#)
- [Remove "b" and "ac" from a given string](#)

- [Dynamic Programming | Set 29 \(Longest Common Substring\)](#)



3



Tweet

0



0

Writing code in comment? Please use [ideone.com](#) and share the link here.

10 Comments**GeeksforGeeks**

Sort by Newest ▾



Join the discussion...

**Guest** • 6 months ago

This code runs in $O(n)$ time with constant space ...please correct me if it is inc

```
int position[26] = {0};
```

```
int delta(int state,char input,char *fa,int i){
```

```
if( fa[state] == input ){
```

```
position[input - 97] = state;
```

```
return state+1;
```

```
}
```

```
return position[input-97];
```

```
}
```

```
int pattern_match(char *str,char *p,int n,int m){
```

```
int q = 0;
```

```
for(int i = 0 ; i < n; i++){
```

```
int old_q = q;
```

```
q = delta(q,str[i],p,i);
```

```
if(old_q != 0 && old_q - q > 1)
```

```
i--;
```

```
else if(q == m){
```

```
cout<<"pattern found at : "<<i-m+1<<endl; q="0;" }="" }="" int="" main(){='
```

AdChoices

[▶ C++ Algorithms](#)[▶ Algorithm Java](#)[▶ Java to C++](#)

AdChoices

[▶ Pattern Matching](#)[▶ Java Pattern](#)[▶ Automata](#)

AdChoices

[▶ It Efficient](#)[▶ Pattern State](#)[▶ Pattern Search](#)

```
char="" pattern[]="geeks" ;="" pattern_match(str,="" pattern,="" sizeof(str)="" ;
sizeof(char)="" -1);="" return="" 0;="" }="">
```

^ | v • Reply • Share ›



alien • 8 months ago

@GeeksforGeeks: Could you please explain why this algorithm is able to fill TF at a given time?

1 ^ | v • Reply • Share ›



Dhiren • 10 months ago

Consider this example

Pattern – A C A C

At state-0, we have only “A”, so lps = 0

Transition from state-0 to state-1, probable cases may be

Case-1 a new ‘A’ comes, then we go back to our longest prefix suffix till now what if a ‘A’ comes, in this case it is 1

Case-2 a new ‘G’ comes, then we go back to our longest prefix suffix till now it will be 0

Case-3 a new ‘C’ comes, then also value will be 0

That’s why we are first copying the lps row values into the current ith row.

Then we update the state transition for patt[i] in this case for ‘C’ state will be 2.

Then we calculate the current lps value, that is “AC” but still lps =0 as there is

Calculation of lps can be clear from state transition-2 to 3.

Current lps is 0, now ‘A’ comes so that new lps is 1 for “ACA” which can be fo

^ | v • Reply • Share ›



Arvind • 10 months ago

How does this algorithm work ? where is the proof for this ?

1 ^ | v • Reply • Share ›



Ram • 10 months ago



Wher is the proof ???????????? please post the proof

1 ^ | v • Reply • Share ›



Akshay khare • 11 months ago

when i will become equal to M

then $TF[i][pat[i]] = i+1$. will give segmentation fault

since pat has length M and its index can be upto M-1 only

how will $TF[M][pat[M]]$ will work..pat[M] -> this location not exists..pls explain h
me if i am wrong...

2 ^ | v • Reply • Share ›



Shiwakant Bharti → Akshay khare • 8 months ago

Akshay khare Nice findings! I got the same error in Java.

Here is the test case which should break:

```
char[] txt2 = "AABAACAADAABAAABAA".toCharArray();
char[] pat2 = "AABA".toCharArray();

//This code fix worked for me. Not sure if this robust enough.
for (i = 1; i <= M; i++) {
    // Copy values from row at index lps
    // Is this powerful enough to handle case of i = M(
    // transition)?
    for (ch = 0; ch < NoOfChars; ch++) {
        TFDP[i][ch] = TFDP[lps][ch];
    }
    // This is special case where the last halt state i
    // for regular processing.
    // Here pat[M] is out of bound and further calculat
    //
```

[see more](#)

^ | v · Reply · Share ›



abhishek08aug · a year ago

Intelligent :D

^ | v · Reply · Share ›



vinu · 2 years ago

Yes... @GeeksForGeeks can you please me more clear with reasoning of ste

```
/* Paste your code here (You may delete these lines if not writing c
```

^ | v · Reply · Share ›



spandan · 2 years ago

If someone can please explain how has the transition function been computed

^ | v · Reply · Share ›



Subscribe



Add Disqus to your site

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team