GeeksforGeeks

A computer science portal for geeks

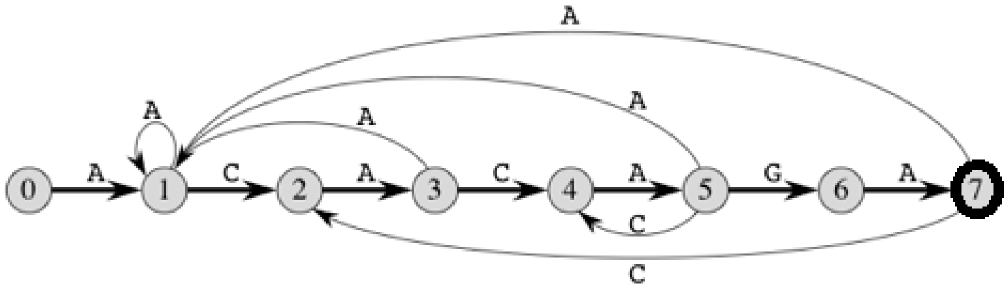
GeeksQuiz

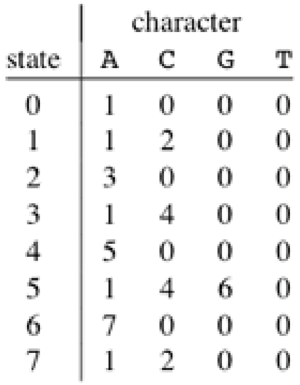
- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

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)*NO_OF_CHARS) time. FA can be constructed in O(m*NO_OF_CHARS) time. In this post, we will discuss the O(m*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.





	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

The abvoe 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 for a
   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
    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*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.

Related Topics:

- Recursively print all sentences that can be formed from list of word lists
- Check if a given sequence of moves for a robot is circular or not
- Find the longest substring with k unique characters in a given string
- Function to find Number of customers who could not get a computer
- Find maximum depth of nested parenthesis in a string
- Find all distinct palindromic sub-strings of a given string
- Find if a given string can be represented from a substring by iterating the substring "n" times
- Suffix Tree Application 6 – Longest Palindromic Substring

Tags: Pattern Searching

Tweet 0 +1 1

Writing code in comment? Please use **ideone.com** and share the link here.

17 Comments GeeksforGeeks 1 Login

Recommend 1 Share Sort by Newest

Join the discussion…

logic • a month ago

for those looking for proof go to coursera for algorithms kmp...that guy has explained without this lps shit...extremely well...pure dfa...here is a link for my implementation...quite awesome....http://ideone.com/sqqYZu

^ | v • Reply • Share ›

Rahul Kumar • 10 months ago

in this line,

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

whats the logic for choose the value of lps,please explain.

1 ^ | v • Reply • Share ›

venugopal ➦ Rahul Kumar • 6 months ago

Also Can't we use just

lps = TF[i][pat[i]] = i+1;

lps = TF[i][pat[i]] = i+1;

as a replacement of the two lines

TF[i][pat[i]] = i + 1;
if (i < M)
lps = TF[lps][pat[i]];

if i==M then this means this was the last iteration so whatever becomes of the value of lps in this iteration it will not affect the algo. So I think we can remove the if(condition) and making it simpler. we may use

TF[i][pat[i]] = i + 1;
lps = TF[lps][pat[i]];

or just lps = TF[i][pat[i]] = i+1;

Correct me if I am wrong

^ | ⌄ • Reply • Share ›

Guest ➦ venugopal • 6 months ago

Use and updation of lps can be understood as below:

state 0: lps value not applicable here pat[0]=A
state 1: lps=0 used and NF[0][...] pat=AC used for , similarly
state 2: lps=0 used pat=ACA

state 3: lps=1 used pat=ACAC

state 4: lps=2 used pat=ACACA

state 5: lps=3 used pat=ACACAG

state 6: lps=0 used pat=ACACAGA

state 7: here i=7(i !< M)
This tells us two points i and lps are not interchangeable. so we can't do lps = TF[i][pat[i]] = i+1;
Again lps value has no relation with previous i, it can change back to 0 if new character is found in pattern.

why if(i<m) used?="" we="" saw="" that="" for="" i="state=7" if="" we="" will="" use="" lps="TF[lps][pat[i]]" ,="" then="" we="" will="" get="" a="" segmentation="" fault="" as="" pat[i]="pat[7]" is="" required="" but="" pattern="" has="" values="" from="" pat[0...6].="" we="" don't="" have="" pat[7].="" that's="" why="" we="" are="" using="" a="" check="" condition="" for="" the="" last="" iteration="" of="" i="M.">

^ | ⌄ • Reply • Share ›

Saurabh • a year ago

I have doubts in how the transition function is computed. Furthermore, in statement
// Update the entry corresponding to this character
TF[i][pat[i]] = i + 1;
for i = M it will access pat[M] which is segmentation fault. So how is this algorithm working?
Anyone please help.

⌃ | ⌄ • Reply • Share ›

gambler ➦ Saurabh • 6 months ago

thats y it is taken as M+1 size.

⌃ | ⌄ • Reply • Share ›

Guest • a year ago

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

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

⌃ | ⌄ • Reply • Share ›

alien • 2 years ago

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

1 ⌃ | ⌄ • Reply • Share ›

Dhiren • 2 years 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 which is "A" which is state-0 and see 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 and see what if 'G' comes in this case 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 no longest prefix suffix.

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 found out in row [lps]['A']

^ | ⌄ • Reply • Share ›

Arvind • 2 years ago

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

1 ^ | ⌄ • Reply • Share ›

Ram • 2 years ago

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

1 ^ | ⌄ • Reply • Share ›

Akshay khare • 2 years 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 how last row is calcualated..or correct me if i am wrong...

2 ^ | ⌄ • Reply • Share ›

Shiwakant Bharti ➦ Akshay khare • 2 years 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(halt state
            // transition)?
            for (ch = 0; ch < NoOfChars; ch++) {
                TFDP[i][ch] = TFDP[lps][ch];
            }
            // This is special case where the last halt state is also considered
            // for regular processing.
            // Here pat[M] is out of bound and further calculation isn't needed.
```

see more

• Reply • Share ›

Suthar Shiwakant Bharti • a year ago

for (x = 0; x < NO_OF_CHARS; x++)
TF[i][x] = TF[lps][x];

Update last entry similar to previous ones, using lps. Just that now there is no more remaining character in pattern so no need to update it again as we are doing for other rows.

You can add this for loop at the end of method.

• Reply • Share ›

abhishek08aug • 2 years ago

Intelligent :D

• Reply • Share ›

vinu • 3 years ago

Yes... @GeeksForGeeks can you please me more clear with reasoning of steps in transition function?

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

• Reply • Share ›

spandan • 3 years ago

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

• Reply • Share ›

Subscribe Add Disqus to your site Privacy

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- 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

Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

Java Programming Language · 2 hours ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 2 hours ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 2 hours ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago

-

@geeksforgeeks, Some rights reserved Contact Us!
Powered by WordPress & MooTools, customized by geeksforgeeks team