Searching for Patterns | Set 4 (A Naive Pattern Searching Question)

Question: We have discussed Naive String matching algorithm here. Consider a situation where all characters of pattern are different. Can we modify the original Naive String Matching algorithm so that it works better for these types of patterns. If we can, then what are the changes to original algorithm?

Solution: In the original Naive String matching algorithm , we always slide the pattern by 1. When all characters of pattern are different, we can slide the pattern by more than 1. Let us see how can we do this. When a mismatch occurs after j matches, we know that the first character of pattern will not match the j matched characters because all characters of pattern are different. So we can always slide the pattern by j without missing any valid shifts. Following is the modified code that is optimized for the special patterns.

```
#include<stdio.h>
#include<string.h>

/* A modified Naive Pettern Searching algorithn that is optimized
   for the cases when all characters of pattern are different */
void search(char *pat, char *txt)
{
    int M = strlen(pat);
    int N = strlen(txt);
    int i = 0;

    while(i <= N - M)
    {
        int j;

        /* For current index i, check for pattern match */
        for (j = 0; j < M; j++)
        {
            if (txt[i+j] != pat[j])
                break;
        }
        if (j == M)  // if pat[0...M-1] = txt[i, i+1, ...i+M-1]
        {
           printf("Pattern found at index %d \n", i);
           i = i + M;
        }
        else if (j == 0)
        {
           i = i + 1;
        }
        else
        {
           i = i + j; // slide the pattern by j
        }
    }
}

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

Output:
Pattern found at index 4
Pattern found at index 12

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

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

Rohit Asthana • 8 months ago

one more clean way http://ideone.com/ce77B4 in O(n) O(1)

• Reply • Share ›

sijayaraman • a year ago

#include<iostream>
#include<string>
using namespace std;
int main()
{
char str[]="ABCEABCDABCEABCD";
char pat[]="ABCD";
for(int i=0;i<strlen(str);i++) {="" int="" j,k;="" for(j="0,k=i;j<strlen(pat);j++)" {=""
if(str[k++]!="pat[j])" {="" break;="" }="" }="" if(j="strlen(pat))" {=""
cout<<"startindex="<<i<<" and="" endindex="<<(i+j-1)<<endl;
}
}
}">

• Reply • Share ›

zyzz • 2 years ago

```
 /#include<stdio.h>
#include<string.h>


void pattern(char *s,char *p){
    int n=strlen(p);
int i,flag=0;
while(*p!=*s)
{s++;}

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

    if(*p==*s){

        p++;
        s++;

        flag++;
```

see more

• Reply • Share ›

seeker • 3 years ago

it can easily be done using following simple loop

```
 void search(char *pat, char *txt)
{
    int M = strlen(pat);
    int N = strlen(txt);
    int i = 0;  //points to location in text
    int j =0; //points to location for pattern
    for( int i = 0 ; i <= N-M ;i++)
    {
        if( txt[i] == pat[j] ) {
                j++;
        }else {
                j=0;
        }
        if( j == M ) {
                printf("Pattern found at index %d \n", i-M);
                 j=0;
        }
   }

}
```

```
}
```

5 • Reply • Share ›

rohit batra seeker • 3 months ago

it fails for this

char *txt = "AABCD";

char *pat = "ABCD";

• Reply • Share ›

user seeker • a year ago

Pattern found at i-m+1 not only i-m

• Reply • Share ›

user seeker • a year ago

I think, In loop condition should be i<n, instead="" of="" i<="n-m." above="" code="" will="" not="" match="" the="" last="" occurrence="" i.e="" if="" pattern="" matches="" at="" the="" end="" of="" given="" text="">

• Reply • Share ›

saurabh • 3 years ago

Above algorithm not work for

char *txt = "AgneepathAgneepathkanchaAgneepathKancha";
char *pat = "Kancha";

• Reply • Share ›

GeeksforGeeks saurabh • 3 years ago

@saurabh: Please take a closer look at the problem statement. The pattern given by you is not a valid pattern as 'a' occurs two times. You can apply the algorithm discussed here.

1 • Reply • Share ›

Naveen Makwana • 4 years ago

I think , i got a better solution to this ...accepting all types of patterns and yes with just O(n) time complexity.....
so here is the code.....
[sourcecode]
int main()
{
char *s="ARAARAJAAARAJAAJARRAARAJRARAJAA";
char *p="ARAJAA";
int i=0 j=0 ls lp;

```
int i=0,j=0,ls,lp;
lp=strlen(p); // length of pattern
ls=strlen(s); // length of text

while(ls--){
if((s[i]==p[j])){
j++;
}
else{
j=0;
```

see more

• Reply • Share ›

Agniswar ➚ Naveen Makwana • 4 years ago

@Naveen:Hi,your code gives wrong output in case of inputs like char *s="aabaacaaba" and char *p="aaba".Accd to your code the output is "Pattern found at 0" and "Pattern found at 6"..but it essentially missed out position 1..I guess it's because you have incremented i one place always in the same loop as j.So,i guess you will need two loops in order to print all the positions !

• Reply • Share ›

Naveen Makwana ➚ Naveen Makwana • 4 years ago

u can remove use of strlen here....like

[sourcecode]

```
int main()
{
char *s="ARAARAJAAARAJAAJARRAARAJRARAJAA";
char *p="ARAJAA";
int i=0,j=0;
while(s[i]){
if((s[i]==p[j])){
j++;
}
else{
j=0;
if((s[i]==p[j])){
j++;
}
}
if(p[j+1]=='&#092&#048'){
printf("Pattern found at %d\n",i-j+1);
j=0;
}
i++;
```

```
}
return 0;
}
```

• Reply • Share ›

student • 4 years ago

still the solution will not work for cases like this

```
char *txt = "AABCD";
char *pat = "ABCD";
```

• Reply • Share ›

GeeksforGeeks ➦ student • 4 years ago

@student: Thanks for pointing this out. We have changed the code to handle this case.

• Reply • Share ›

student ➦ GeeksforGeeks • 4 years ago

thanks for correcting the error.

• Reply • Share ›

Raja • 4 years ago

Will it work for "RARAJA" if the pattern is "RAJA"

• Reply • Share ›

Sandeep ➦ Raja • 4 years ago

@Raja: Please take a closer look at the question. The pattern "RAJA" is not a valid pattern for the given question as all characters of the pattern must be different.

• Reply • Share ›

Venki • 4 years ago

The above function misses few corner cases. For example see the following input,

```
char *txt = "AABAACAADAABAAAABA&quot"
char *pat = "AABA&quot"
```

There are three matching patterns. But the code prints only two.

Here is the correct version of program (or increment i by (M + 1) in original program after match is found (if clause)),

```
#include <stdio.h>
#include <string.h>
```

```
#include <string.h>

#define TEXT "AABAACAADAABAAAABA"
#define PATT "AABA"

// Improvded pattern matching
```

see more

• Reply • Share ›

Sandeep ➚ Venki • 4 years ago

@venki: Please take a closer look at the question. The pattern "AABA" is not a valid pattern for the given question as all characters of the pattern must be different.

• Reply • Share ›

shanker • 4 years ago

@geeksfoegeesk..can you post Boyce Moorrie string matching algo with explanation :) keep it up

• Reply • Share ›

Simran • 4 years ago

This code is not right.. I feel you have modified the algorithm from Cormen a little.. In the lowermost else case of your function, you cannot increment 'i' by 'j+1'.. It has to be incremented by 1 only..
Test Case where your code fails..

char *txt = "ABAABABACBCABCABABA";
char *pat = "ABABAC";

• Reply • Share ›

Simran ➚ Simran • 4 years ago

Sorry, didn't see your statement about pattern string having different characters..

• Reply • Share ›

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