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.

Popular Posts

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

Related Tpoics:

- Print all possible words from phone digits
- 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

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

18 Comments GeeksforGeeks

Sort by Newest

sijayaraman • 14 days 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&amp;lt;strlen(pat);j++)" {="'
}="" if(j="strlen(pat))" {="" cout<<"startindex="&amp;lt;&amp;lt;i&amp;lt;&amp;lt;" and="" endindex
}
}
}">
```

zyzz • 11 months 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++;
```

705

Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 2 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 42 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 45 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

see more

seeker • 2 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;
        }
   }

}
```

4

saurabh • 2 years ago

Above algorithm not work for

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

GeeksforGeeks ➚ saurabh • 2 years ago

@saurabh: Please take a closer look at the problem statement. The pa
as 'a' occurs two times. You can apply the algorithm discussed here.

Naveen Makwana • 3 years ago

I think , i got a better solution to this ...accepting all types of patterns and yes w
so here is the code.....
[sourcecode]
int main()
{
char *s="ARAARAJAAARAJAAJARRAARAJRARAJAA";
char *p="ARAJAA";
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;
if((s[i]==p[j])){

see more

Agniswar ➚ Naveen Makwana • 3 years ago

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

Naveen Makwana ➚ Naveen Makwana • 3 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;
}
```

student • 3 years ago

still the solution will not work for cases like this

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

GeeksforGeeks ➚ student • 3 years ago

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

student ➚ GeeksforGeeks • 3 years ago

thanks for correcting the error.

Raja • 3 years ago

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

Sandeep ➚ Raja • 3 years ago

@Raja: Please take a closer look at the question. The pattern "RAJA" i
question as all characters of the pattern must be different.

Venki • 3 years ago

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

```
 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 prc

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

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

// Improvded pattern matching
```

see more

Sandeep Venki • 3 years ago

@venki: Please take a closer look at the question. The pattern "AABA" question as all characters of the pattern must be different.

shanker • 3 years ago

@geeksfoegeesk..can you post Boyce Moorrie string matching algo with expla

Simran • 3 years ago

This code is not right.. I feel you have modified the algorithm from Cormen a lit 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";

Simran Simran • 3 years ago

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