

# 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](#)

## Searching for Patterns | Set 2 (KMP Algorithm)

Given a text  $txt[0..n-1]$  and a pattern  $pat[0..m-1]$ , write a function  $search(char\ pat[], char\ txt[])$  that prints all occurrences of  $pat[]$  in  $txt[]$ . You may assume that  $n > m$ .

Examples:

1) Input:

```
txt[] = "THIS IS A TEST TEXT"
pat[] = "TEST"
```

Output:

Pattern found at index 10

## 2) Input:

```
txt[] = "AABAACAADAABAAABAA"
pat[] = "AABA"
```

## Output:

```
Pattern found at index 0
Pattern found at index 9
Pattern found at index 13
```

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.

We have discussed Naive pattern searching algorithm in the [previous post](#). The worst case complexity of Naive algorithm is  $O(m(n-m+1))$ . Time complexity of KMP algorithm is  $O(n)$  in worst case.

## KMP (Knuth Morris Pratt) Pattern Searching

The [Naive pattern searching algorithm](#) doesn't work well in cases where we see many matching characters followed by a mismatching character. Following are some examples.

```
txt[] = "AAAAAAAAAAAAAAAAAAB"
pat[] = "AAAAB"

txt[] = "ABABABCABABABCABABABC"
pat[] = "ABABAC" (not a worst case, but a bad case for Naive)
```

The KMP matching algorithm uses degenerating property (pattern having same sub-patterns appearing more than once in the pattern) of the pattern and improves the worst case complexity to  $O(n)$ . The basic idea behind KMP's algorithm is: whenever we detect a mismatch (after some matches), we already know some of the characters in the text (since they matched the pattern characters prior to the mismatch). We take advantage of this information to avoid matching the characters that we know will anyway match. KMP algorithm does some preprocessing over the pattern `pat[]` and constructs an auxiliary array `lps[]` of size `m` (same as size of pattern). Here **name lps indicates longest proper prefix which is also suffix..** For each sub-pattern `pat[0...i]` where  $i = 0$  to  $m-1$ , `lps[i]` stores length of the maximum matching proper prefix which is also a suffix of the sub-pattern `pat[0..i]`.

```
lps[i] = the longest proper prefix of pat[0..i]
        which is also a suffix of pat[0..i].
```

## Examples:

For the pattern "AABAACAABAA", `lps[]` is [0, 1, 0, 1, 2, 0, 1, 2, 3, 4, 5]

For the pattern "ABCDE", `lps[]` is [0, 0, 0, 0, 0]

For the pattern "AAAAA", `lps[]` is [0, 1, 2, 3, 4]

For the pattern "AAABAAA", `lps[]` is [0, 1, 2, 0, 1, 2, 3]

For the pattern "AAACAAAAAC", `lps[]` is [0, 1, 2, 0, 1, 2, 3, 3, 3, 4]

## Searching Algorithm:

Unlike the Naive algo where we slide the pattern by one, we use a value from `lps[]` to decide the next sliding position. Let us see how we do that. When we compare `pat[j]` with `txt[i]` and see a mismatch, we know that characters `pat[0..j-1]` match with `txt[i-j+1...i-1]`, and we also know that `lps[j-1]` characters of `pat[0...j-1]` are both proper prefix and suffix which means we do not need to match these `lps[j-1]` characters with `txt[i-j...i-1]` because we know that these characters will anyway match. See

KMPSearch() in the below code for details.

### Preprocessing Algorithm:

In the preprocessing part, we calculate values in lps[]. To do that, we keep track of the length of the longest prefix suffix value (we use len variable for this purpose) for the previous index. We initialize lps[0] and len as 0. If pat[len] and pat[i] match, we increment len by 1 and assign the incremented value to lps[i]. If pat[i] and pat[len] do not match and len is not 0, we update len to lps[len-1]. See computeLPSArray () in the below code for details.

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

```
void computeLPSArray(char *pat, int M, int *lps);
```

```
void KMPSearch(char *pat, char *txt)
{
    int M = strlen(pat);
    int N = strlen(txt);

    // create lps[] that will hold the longest prefix suffix values for patte
    int *lps = (int *)malloc(sizeof(int)*M);
    int j = 0; // index for pat[]

    // Preprocess the pattern (calculate lps[] array)
    computeLPSArray(pat, M, lps);

    int i = 0; // index for txt[]
    while (i < N)
    {
        if (pat[j] == txt[i])
        {
            j++;
            i++;
        }

        if (j == M)
        {
            printf("Found pattern at index %d \n", i-j);
            j = lps[j-1];
        }

        // mismatch after j matches
        else if (i < N && pat[j] != txt[i])
        {
            // Do not match lps[0..lps[j-1]] characters,
            // they will match anyway
            if (j != 0)
                j = lps[j-1];
            else
                i = i+1;
        }
    }
}
```

```
    free(lps); // to avoid memory leak
}


void computeLPSArray(char *pat, int M, int *lps)
{
    int len = 0; // length of the previous longest prefix suffix
    int i;

    lps[0] = 0; // lps[0] is always 0
    i = 1;

    // the loop calculates lps[i] for i = 1 to M-1
    while (i < M)
    {
        if (pat[i] == pat[len])
        {
            len++;
            lps[i] = len;
            i++;
        }
        else // (pat[i] != pat[len])
        {
            if (len != 0)
            {
                // This is tricky. Consider the example AAACAAA and i = 7.
                len = lps[len-1];

                // Also, note that we do not increment i here
            }
            else // if (len == 0)
            {
                lps[i] = 0;
                i++;
            }
        }
    }
}

// Driver program to test above function
int main()
{
    char *txt = "ABABDABACDABABCABAB";
    char *pat = "ABABCABAB";
    KMPSearch(pat, txt);
    return 0;
}
```



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

1

+1

9

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

118 Comments

GeeksforGeeks

1

Login ▾

♥ Recommend 1

🔗 Share

Sort by Newest ▾



Join the discussion...



**Siya** • a day ago

Check this video

<https://www.youtube.com/watch?...>

^ | v • Reply • Share ›



**AB\_De\_Villers** • a day ago

Here is the cormen implementation in C++:

<http://ideone.com/mwRJSk>

^ | v • Reply • Share ›



**Madhup Pandey** • 10 days ago

check this :

it is simple and takes less time and memory than KMP :

<https://ideone.com/0U1uNT>

^ | v • Reply • Share ›



**Agostinho Junior** ➔ Madhup Pandey • 7 days ago

It doesn't work:

Text = "acdacdab"

Pattern = "acdab"

The KMP algorithm exists for a reason. Your logic is very naive.

1 ^ | v • Reply • Share ›



**Madhup Pandey** → Agostinho Junior • 6 days ago

thank you Agostinho Junior.  
I have rectified my code.  
Let me know if you still can find any discrepancy.  
check code @ <http://ideone.com/KfPQIL>

^ | v • Reply • Share ›



**Agostinho Junior** → Madhup Pandey • 6 days ago

Text = "aaaaaaaaaaaaaaaaaaaaa"  
Pattern = "aa"

Missing a few indexes.

Please test your code in several computer generated random strings with a brute force algorithm before posting it. Using the first 2 - 6 letters should cover enough corner cases.

And even if you get correct answers for all of those don't forget to make sure that your algorithm is able to compute all indexes for strings with length up to 1000000 in less than 2 seconds. Without printing the indexes 2 seconds should be more than enough if the performance of your algorithm in anyway closer to linear algorithms like KMP.

^ | v • Reply • Share ›



**Abhigyan Mehra** → Agostinho Junior • a day ago

Well said, Sir.

It annoys me when people don't test their code thoroughly and claim otherwise.

^ | v • Reply • Share ›



**Akash Goel** • 16 days ago

Most crucial line (LPS function):

"// Also, note that we do not increment i here"

^ | v • Reply • Share ›



**Waterbottle** • 21 days ago

Really bad explanation

^ | v • Reply • Share ›

**Viraj** • 24 days agoCan someone explain why we assign `len = lps[len-1]` in the `computeLPSArray` function ?

^ | v • Reply • Share ›

**Amruta Borkar** • a month agowhy did you assign `j = lps[j-1]`; in `KMPSearch` function?

^ | v • Reply • Share ›

**piyush jain** • 2 months ago

Poor Explanation. couldnt understand a thing :(

1 ^ | v • Reply • Share ›

**prashant jha** • 2 months ago

here is my c code using suffix tree

<http://ideone.com/Ovv3Vo>

^ | v • Reply • Share ›

**Guest** • 2 months ago

my c code for pattern searching using suffix tree

<http://ideone.com/7ZZZw7>

^ | v • Reply • Share ›

**Guest** • 2 months agoFor the pattern "AABAACAABAA", `lps[]` is [0, 1, 0, 1, 2, 0, 1, 2, 3, 4, 5]

this line is wrong it should be

For the pattern "AABAACAABAA", `lps[]` is [0, 1, 0, 1, 2, 0, 1, 2, 3, 1, 5]

^ | v • Reply • Share ›

**Sahil Mutneja** • 2 months ago

Explanation with the Sample Spoj Problem and Code ::

<http://goo.gl/XbOm7F>

1 ^ | v • Reply • Share ›

**Aditya Goel** • 2 months ago

To avoid getting brain fucked by the "prefix of the suffix of the prefix of the...", refer this article-

<http://jakeboxer.com/blog/2009...>

2 ^ | v • Reply • Share ›

**Shubham Sharma** ➔ Aditya Goel • 2 months ago

You saved my time :)

^ | v • Reply • Share ›

**Guest** → Aditya Goel • 2 months ago

Thanks Aditya Goel for this link.....

^ | v • Reply • Share ›

**Raj** • 2 months ago

Can anybody please explain why did we use below condition

if (j == M)

{

printf("Found pattern at index %d \n", i-j);

j = lps[j-1];

}

when match occurred.

^ | v • Reply • Share ›

**Nono** • 3 months ago

hi, do you have an example of KMP with levenstein ?

for each entry, i want to check the distance and return if it is higher than a specified distance.

thank you

^ | v • Reply • Share ›

**Guest** • 4 months ago

why is lps[1] always equal to 1 ? if pattern is aa then a matches a . so length should be 1 .

Please clarify.

^ | v • Reply • Share ›

**Anurag Singh** → Guest • 4 months ago

lps[1] IS NOT always equal to 1. It depends on pattern.

In pattern AA, longest proper prefix is A which matches with suffix A, so lps[1] is 1.

In pattern "AB", there is no longest proper prefix which matches with suffix, so lps[1] = 0.

lps[0] is ALWAYS zero because with pattern length, there is no proper prefix at all, so lps[0] = 0.

3 ^ | v • Reply • Share ›

**The Internet** • 4 months ago

I think indexing arrays with int's is something we should finally get away from on this site. There are machines with 64bit address ranges on which int is only 32bit. If you want to address memory, of which array indexing is just one case, use size\_t. There have been plenty of bugs resulting from the lack of proper use of size\_t.

Also it's far more convenient to construct the backtrack array in such a fashion that the value



Also, it's far more convenient to construct the backtrack array in such a fashion that the value at position "x", as opposed to "x - 1", is the index we need to jump to when a mismatch occurs. That way, the backtrack array is far easier to index.

I think I have a functionally equivalent implementation here: <http://ideone.com/jDjoot>

It's a bit different, though. I return the index at which the pattern is found in the string. I find that is much closer to the type of signature most class libraries provide. Also, I only return the first match. The method can be called several times for all matches.

Last but not least, does everything on here have to be in plain C? A lot of the function signatures are a lot nicer and things such as memory cleanup a lot simpler in C++. Notice the use of the C++11 `unique_ptr`, in order to make use of a simple form of RAII.

---

[see more](#)

1 ^ | v • Reply • Share ›



**neelabhsingh** • 5 months ago

Please go to serial no 5. which is important.

Text: bacbababaabcbab

Pattern: abababca

In KMP, we preprocess the given pattern, Mean we will find the maximum length of Pattern repeating itself.

```
patter:| a | b | a | b | a | b | c | a |
index:| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
length:| 0 | 0 | 1 | 2 | 3 | 4 | 0 | 1 |
```

Explanation. index=0, 1, length is zero,

index=2, pattern is repeating itself so length is 1 for index=2

similarly for index=3, length of pattern is 2, for index=4 and 5, length are 3 and 4.

abababca

\*\*abab

But index=6, 'c' character comes so there is no so length is zero,

But index=7, 'a' character again come so length for index=7 is zero.

abababca

---

[see more](#)

4 ^ | v • Reply • Share ›



**Rasmi Ranjan Nayak** → neelabhsingh • 3 months ago

Excellent explanation, I think even better than the article. Could you pls make me understand how

“AAACAAAAAC”, lps[] is [0, 1, 2, 0, 1, 2, 3, 3, 3, 4], ???

I think it should be [0,1,2,0,1,2,3,4,5].

Am I right or wrong?

if I am wrong then pls guide me here am I wrong

^ | v • Reply • Share ›



**neelabh Singh** → Rasmi Ranjan Nayak • 2 months ago

AAACAAAAAC this the good example to understand,

AAACAAAAAC

A => A is repeating itself 0 time=>0

AA=> This String (AA) is repeating itself 1 time=>1

AAA=>This String(AAA), Max length of this SubString Which repeat itself from staring is 2.

AAA

\*AAA

AAAC=> Because we are searching max length of substring who suffix and prefix are same,

AAAC

---

see more

^ | v • Reply • Share ›



**Frank** → Rasmi Ranjan Nayak • 3 months ago

the first difference in your answer to the given one is 4 VS 3.

The prefixes of AAACAAAA are: AAACAAA, AAACAA, AAACA, AAAC, AAA, AA, A;

The suffixes of AAACAAAA are: AACAAAA, ACAAAA, CAAAA, AAAA, AAA, AA, A.

So the length of the Longest Common Substring is 3 (AAA) NOT 4.

^ | v • Reply • Share ›



**popina** • 6 months ago

I just want to thank the guy who wrote this article.

This is excellent!

2 ^ | v • Reply • Share ›



**deenak h** • 6 months ago



deepak b • 6 months ago

In the function KMPSearch, isn't the while loop returns ArrayIndexOutOfBoundsException exception in case of pattern not found in the text. If the last character `text[i = n-1]` matches with `pattern[j]`, then `i++` will lead to `text[n]`.

^ | v • Reply • Share ›



**Anurag Singh** → deepak b • 6 months ago

It will be a problem if it was a java code. Since it's a C code, it works well. Because after last character (`txt[n-1]`) match on text against pattern, if pattern not found (i.e. `j != M`), then next character of text and pattern are matched where `txt[n]` is NULL and so it doesn't create any problem. Recall that if you create a string of length N in C, compiler assigns a memory of N+1 bytes where (N+1)th location will have NULL in it as string terminator.

So even though C doesn't report any error here, it's a logical problem and we should put a boundary check on `txt` as (`i < N`) in the else condition.

It will be corrected soon.

^ | v • Reply • Share ›



**deepak b** → Anurag Singh • 6 months ago

Thanks Anurag for clarifying quickly and also for brushing up the C way of dealing things.

^ | v • Reply • Share ›



This comment was deleted.



**Krishna** → Guest • 6 months ago

<http://www.quora.com/What-is-t...> : Go through this

May be sometime Geeksforgeeks unable to provide best material but Don't abuse on this platform.

2 ^ | v • Reply • Share ›



**Shubhang** • 7 months ago

In the KMPSearch method the second else if statement should have `i < n` condition="" also="" along="" with="" `pat[j] != txt[i]` as="" you="" are="" incrementing="" the="" i="" after="" the="" while="" loop="" entry=""

2 ^ | v • Reply • Share ›



**Rahul** • 8 months ago

`computeLPSArray()` method can be solved this way more conveniently :

`lps[0] = 0;`

```
i = 1;
while(i < M)

{

if(pat[i] == pat[lps[i-1]])

lps[i] = lps[i-1]+1;

else

lps[i] = 0;

i++;

}
```

^ | v • Reply • Share ›



**theoretical\_physicist** → Rahul • 7 months ago

try for "AAACAAAA"..

1 ^ | v • Reply • Share ›



**Rahul** → theoretical\_physicist • 7 months ago

thanks for pointing out ...

^ | v • Reply • Share ›



**VISHAL60** • 8 months ago

i think lps array of last example is wrong..

1 ^ | v • Reply • Share ›



**Chaudhary** • 8 months ago

If somebody not understand till now , i am referring you a link

**Best link To understand**

21 ^ | v • Reply • Share ›



**Asimov** → Chaudhary • 8 months ago

You are a rockstar dude!

1 ^ | v • Reply • Share ›



**Kaidul Islam Sazal** • 9 months ago

Bad and insufficient explanation!

24 ^ | v • Reply • Share ›



**SHIVAM DIXIT** → Kaidul Islam Sazal • 8 months ago

CHECK THIS OUT- <http://jakeboxer.com/blog/2009...>

3 ^ | v • Reply • Share ›



**test** → SHIVAM DIXIT • 15 days ago

<https://www.youtube.com/watch?...>

^ | v • Reply • Share ›



**Kaidul Islam Sazal** → SHIVAM DIXIT • 8 months ago

Yes bro, I searched that day and learnt from there :)

^ | v • Reply • Share ›



**Chowdhury Osman** • 9 months ago

Thanks for such wonderful note .... Really helpful :)

^ | v • Reply • Share ›



**Guest** • 9 months ago

There is a small problem in code.

It will fail in cases where pattern matches partially at the end of text, e.g.

`txt = "fdfsfdfdf";`

`pat = "fsf";`

In KMPSearch function, in line

"else if(pat[j] != txt[i])", we need to add one more condition "i < N", i.e. it should be:

"else if(i < N && pat[j] != txt[i])"

1 ^ | v • Reply • Share ›



**krishna** • 10 months ago

good explanation

<https://www.youtube.com/watch?...>

and implementation of that algorithm

<http://ideone.com/BEYgY8>

^ | v • Reply • Share ›



**<HoldOnLife!#>** → krishna • 10 months ago

I was wondering KMP algorithm is  $O(N)$  but in your solution `constructdfa()` is  $O(mn)$  ..so overall your kmp method becomes  $O(mn)$  too ?

^ | v • Reply • Share ›



**raman singh** • 10 months ago

Guys Read it on Wikipedia for more clearer understanding ...

1 ^ | v • Reply • Share ›

Load more comments



- 
- 
- 
- 
- - [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

- [It\\_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