OUTLOOKING LIFE

SUFFIX ARRAYS – A SIMPLE TUTORIAL

July 1, 2012 | Ankur | Array, Programming, SPOJ, Strings, Suffix, Suffix Array, Topcoder

Suffix Arrays

It is the data structure to look out for when you are going for string Searching and want to reduce the time complexity greatly. So, Let us examine and start from basics.

What are Suffix Arrays ?

As simple as it can get , suffix arrays represent the rank of all possible suffixes of a given string .For example ,

String S = “ABDCASD”

Then the suffixes for the string S (with the numbers in front represent are ,

7 $ *

6 D

5 SD

4 ASD
3 CASD

2 DCASD

1 BDCASD

0 ABDCASD

*$, A null character can be called a suffix of every string as it does no addition to the string itself . We will later on see why we chose to add NULL character to our algorithm . For NULL character that we used, we assume have highest lexicographic priority .

Now,

What happens when we sort those suffixes lexicographically ?

We get the order

Rank	Index from which starting	String
1	7	$
2	0	ABDCASD
3	4	ASD
4	1	BDCASD

5	3	CASD
6	6	D
7	2	DCASD
8	5	SD

So, the Suffix Array in our case will be ,

Suffix Array Index	**Sorted Suffixes (index of suffixes when sorted in lexicographic order)**
1	7
2	0
3	4
4	1
5	3

6	6
7	2
8	5

Now let us take a different example and we introduce the concept of **Lowest Common Prefix** .

S=”abaabba”

Let us construct the Suffix Array for the above string first

Rank	Index	String	LCP
1	8	$	
2	7	a	
3	3	aabba	
4	1	abaabba	

5	4	abba
6	6	ba
7	2	baabba
8	5	bba

Now you see a column of LCP (lowest common prefix) , what is it ?

So , What is **Longest Common Prefix ?**

It is the length of prefix that is common between the two consecutive suffixes in an sorted suffix array. To simplify , see this way , we have created an suffix array and we have ordered suffixes . We need to find out the number of consecutive characters common in the two suffixes (from starting of suffix).

e.g.

LCP of a and aabba is 1 .

LCP of abaabba and abba is 2 .

We put LCP in first column as 0 , always .

So , we get the following table :

Rank	Index	String	LCP
1	8	$	0
2	7	a	0
3	3	aabba	1
4	1	abaabba	1
5	4	abba	2

6	6	ba	0
7	2	baabba	2
8	5	bba	1

Why would I want to use suffix array ?

Well , They are extremely useful string searching , where text is limited and substrings to be searched are not limited . This approach is in contrast to KMP algorithm where there was one substring and you were given continuous stream of characters to search from.

Following is one out of the many important usages of the Suffix Arrays :

Finding Total Number of distinct Substrings.

Well how do we do this ?

We see that , any string , let us say

ABC will give following distinct substring ,

A

B

C

AB

BC

ABC

Which is equivalent to choosing an end point and a start point in a string, and what is left is a simple matter of finding the number of ways you can choose a start and end point . Which is ${}^{n}C_{2}$ (where n is the string length) .

Problem happens when we have repeated characters . What will you do then ? that is when suffix arrays come into play .

Now if I am given a string , axyz , and let us say , I ask you how many strings can you make with this string such that they start from the first letter. Then the answer will be 4 (a , ax, axy, axyz) .

So , I have a string of length 7 , then I have 7 suffixes (excluding NULL , I am not counting it because it's length is 0 , does not contribute towards any distinct substring) . So , going by the above logic , I will 7+6+5+4+3+2+1 distinct strings , which is equivalent to ${}^{n}C_{2}$ which was also the result of above discussion .

Unfortunately for us , this result doesn't hold for the strings with repeated characters.

Take this one

S= "aabaab"

Suffix Possible strings such that they start from first letter of suffix

b b

ab a,ab

aab a,aa,aab

baab b,ba,baa,baab

abaab a,ab,aba,abaa,abaab

aabaab a,aa,aab,aaba,aabaa,aabaab

Now as you see strings a,b,aab,etc are repeated but we also know that if we sort the above suffixes , those strings which are common , there length is given in Lowest Common Prefix .

So , the total number of distinct Substrings from a given string is given by

$^{n+1}C_2$ – Sum of All LCP

And that is the answer, as simple as that.

Problems Based on the above concept :

http://www.spoj.pl/problems/SUBST1/

http://www.spoj.pl/problems/DISUBSTR/

We will see more about the Suffix array and LCP implementation in the next part.

Please feel free to leave suggestions. Question are encouraged. Do not forget to comment :)

Adios ,

Ankur Khurana

SHARE THIS:

★ Like

8 bloggers like this.

← Namaste

Immutable Var vs Mutable Val →

35 THOUGHTS ON "SUFFIX ARRAYS – A SIMPLE TUTORIAL"

MEDHA YADAV SAYS: Nice post !!.Clear explanation

July 26, 2016 at 2:59 pm • Reply »

GANEH SAYS: Awesome post. Very simple explanation.

July 5, 2016 at 4:04 am • Reply »

RAHUL BANSAL SAYS: Thanks man!..great explanation…

May 29, 2016 at 1:30 pm • Reply »

PEREFECTO SAYS: please explain how the searching is done with the help of the LCP in case of repeated sequences, its not clear thank you

March 21, 2016 at 4:37 am • Reply »

KONIGPRIYADARSI SAYS: Superb man.. way too good (y)

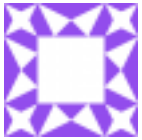
January 14, 2016 at 7:59 am • Reply »

SHOHAGHCSESUST SAYS: Nice tutorial..but it should be Longest Common Prefix (LCP) not Lowest Common Prefix😀

January 12, 2016 at 11:02 pm • Reply »

KOUSHIK SAYS: Nice Explanation…. Thanks

July 3, 2015 at 10:00 am • Reply »

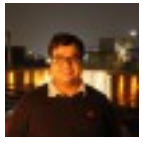


GAURAV CHANDEL SAYS: Brilliantly explained……Thanks

April 10, 2015 at 12:49 pm • Reply »

ANURAG SAYS: it must be (n+1)C2 otherwise we miss out all the cases with a single character [1,1] [2,2] …

January 17, 2015 at 11:45 am • Reply »



ANKUR SAYS: I agree. Updated.

July 6, 2016 at 2:14 am • Reply »

ASPIRING CODER SAYS: Seriously, the best explanation I've seen for this problem. You'll get a lot of good karma for this🙂

January 16, 2015 at 9:31 pm • Reply »

SAHILDUA2305 SAYS: Reblogged this on Sahil Dua and commented:
An Awesome Post on Suffix Arrays !

January 16, 2015 at 2:31 am • Reply »

THARUN SAYS: easy to understand. But where are posts about the suffix array and lcp implementation?

December 9, 2014 at 10:39 am • Reply »

SUBHABRATA DEBNATH SAYS: Nice explanation. However, there is a small mistake. While finding number of distinct substrings, the answer will be (N+1)C2 – sum of all LCA

October 30, 2014 at 9:57 pm • Reply »

THARUN SAYS: I think you are wrong. Lets say that length of string be N.Then you can pick 2 characters (starting and ending points) in NC2 possible ways.

December 9, 2014 at 12:19 pm • Reply »

ANKUR SAYS: I think Subhabrata is right. Because we allow the same string to be selected again (single character substring), the number of options become n+1. and by taking an example of string “ab”, it gives us 3 substring (excluding null, ofcourse) which is also mathematically equals to 3C2

July 6, 2016 at 2:13 am

ABHISHEK SAYS: just awesome….
simple and easy to understand
great work!!!

October 29, 2014 at 11:20 pm • Reply »

ASIF SAYS: Thanks for this nice tutorial

July 3, 2014 at 2:17 am • Reply »

SUFFIX ARRAY TUTORIAL | ANKUR KHURANA'S BLOG SAYS: […] appeared here first(written by yours truly) and I am cross posting it […]
June 30, 2014 at 12:37 am • Reply »

AKASH AGRAWALL SAYS: Awesome tutorial. I must say. Thanks for the same.

June 25, 2014 at 4:18 pm • Reply »

VIVEK KUMAR YADAV SAYS: Thanks for such a nice tutorials…..

June 23, 2014 at 5:28 pm • Reply »

ASHISH SAYS: Very nice tutorial,It clears all my doubts .I had wasted most of my time in codechefe and topcoder tutorial but when I read this I get it in single shot :):)…..

May 13, 2014 at 3:04 pm • Reply »

TARAK SAYS: Could you please explain the http://www.spoj.com/problems/ABA12B/ question ….
I would be thankful if you can do this……

August 16, 2013 at 8:26 pm • Reply »

NISCHAL KUMAR SAYS: nice tutorial, but inclusion of code would have been appreciated.

July 31, 2013 at 9:46 am • Reply »

SAI KRISHNA SAYS: Good one am eagerly waiting for the implementation part…. Please post it soon….!!

By the way number of possible substrings in a string of length n is

1+2+3+...+n = n*(n+1)/2=(n+1)C2

Its (n+1)C2 not nC2

July 15, 2013 at 3:24 pm • Reply »

ANKUR SAYS: Ah, you are right. Because we are giving the choice of selecting the same start and end point while selecting a string, it should be (N+1)C2 rather than NC2. good catch. updated.

July 6, 2016 at 2:09 am • Reply »

VAIBHAVATUL47 SAYS: Very nicely written and explained. Great tutorial for newbies...🙂

July 14, 2013 at 12:56 pm • Reply »

PRAKASH SAYS: continue the good work...

It is tutorials like this which help(encourage) noobs to become pros .🙂

July 7, 2013 at 7:18 pm • Reply »

AHMED SAYS: really Good one..Thanks

June 7, 2013 at 12:58 pm • Reply »

RISHAB SAYS: Awesome Article

May 23, 2013 at 12:54 pm • Reply »

ANURAG SAYS: Tera post top par aa gaya google mein😛

January 6, 2013 at 12:36 am • Reply »

ANKUR SAYS: Thanks guys🙂
I will be posting about the implementation and issues related to it soon🙂 .

July 3, 2012 at 9:45 pm • Reply »

SUP SAYS: It is too good.Even a non-programmer guy could code it now😀

July 3, 2012 at 9:37 pm • Reply »

DEVASHISH SAYS: awasum dude

July 2, 2012 at 11:55 pm • Reply »

AMIT GUPTA SAYS: A very nice tutorial, simple and easy to understand:)

July 2, 2012 at 11:31 pm • Reply »

LEAVE A REPLY

Enter your comment here...