

Suffix Array | Set 2 (nLogn Algorithm)

A **suffix array** is a **sorted array of all suffixes of a given string**. The definition is similar to **Suffix Tree** which is compressed trie of all suffixes of the given text.

Let the given string be "banana".

0 banana		5 a
1 anana	Sort the Suffixes	3 ana
2 nana	----->	1 anana
3 ana	alphabetically	0 banana
4 na		4 na
5 a		2 nana

The suffix array for "banana" is {5, 3, 1, 0, 4, 2}

We have discussed **Naive algorithm for construction of suffix array**. The Naive algorithm is to consider all suffixes, sort them using a $O(n\text{Log}n)$ sorting algorithm and while sorting, maintain original indexes. Time complexity of the Naive algorithm is $O(n^2\text{Log}n)$ where n is the number of characters in the input string.

In this post, a **$O(n\text{Log}n)$ algorithm** for suffix array construction is discussed. Let us first discuss a $O(n * \text{Log}n * \text{Log}n)$ algorithm for simplicity. The idea is to use the fact that strings that are to be sorted are suffixes of a single string.

We first sort all suffixes according to first character, then according to first 2 characters, then first 4 characters and so on while the number of characters to be considered is smaller than n . The important point is, if we have sorted suffixes according to first 2^i characters, then we can sort

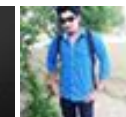
Google™ Custom Search



GeeksforGeeks



52,731 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

ITT Tech - Official Site

itt-tech.edu

Tech-Oriented Degree Programs.
Education for the Future.



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](#)

suffixes according to first 2^{i+1} characters in $O(n \log n)$ time using a $n \log n$ sorting algorithm like Merge Sort. This is possible as two suffixes can be compared in $O(1)$ time (we need to compare only two values, see the below example and code).

The sort function is called $O(\log n)$ times (Note that we increase number of characters to be considered in powers of 2). Therefore overall time complexity becomes $O(n \log n \log n)$. See <http://www.stanford.edu/class/cs97si/suffix-array.pdf> for more details.

Let us build suffix array the example string “banana” using above algorithm.

Sort according to first two characters Assign a rank to all suffixes using ASCII value of first character. A simple way to assign rank is to do “str[i] – ‘a’” for ith suffix of strp[]

Index	Suffix	Rank
0	banana	1
1	anana	0
2	nana	13
3	ana	0
4	na	13
5	a	0

For every character, we also store rank of next adjacent character, i.e., the rank of character at str[i + 1] (This is needed to sort the suffixes according to first 2 characters). If a character is last character, we store next rank as -1

Index	Suffix	Rank	Next Rank
0	banana	1	0
1	anana	0	13
2	nana	13	0
3	ana	0	13
4	na	13	0
5	a	0	-1

Sort all Suffixes according to rank and adjacent rank. Rank is considered as first digit or MSD, and adjacent rank is considered as second digit.

Index	Suffix	Rank	Next Rank
5	a	0	-1

1	anana	0	13
3	ana	0	13
0	banana	1	0
2	nana	13	0
4	na	13	0

Sort according to first four character

Assign new ranks to all suffixes. To assign new ranks, we consider the sorted suffixes one by one. Assign 0 as new rank to first suffix. For assigning ranks to remaining suffixes, we consider rank pair of suffix just before the current suffix. If previous rank pair of a suffix is same as previous rank of suffix just before it, then assign it same rank. Otherwise assign rank of previous suffix plus one.

Index	Suffix	Rank	
5	a	0	[Assign 0 to first]
1	anana	1	(0, 13) is different from previous
3	ana	1	(0, 13) is same as previous
0	banana	2	(1, 0) is different from previous
2	nana	3	(13, 0) is different from previous
4	na	3	(13, 0) is same as previous

For every suffix $\text{str}[i]$, also store rank of next suffix at $\text{str}[i + 2]$. If there is no next suffix at $i + 2$, we store next rank as -1

Index	Suffix	Rank	Next Rank
5	a	0	-1
1	anana	1	1
3	ana	1	0
0	banana	2	3
2	nana	3	3
4	na	3	-1

Sort all Suffixes according to rank and next rank.

Index	Suffix	Rank	Next Rank
5	a	0	-1
3	ana	1	0

1	anana	1	1
0	banana	2	3
4	na	3	-1
2	nana	3	3

We stop here as the next value is 8 which is greater than number of characters in “banana”.

```
// C++ program for building suffix array of a given text
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

// Structure to store information of a suffix
struct suffix
{
    int index; // To store original index
    int rank[2]; // To store ranks and next rank pair
};

// A comparison function used by sort() to compare two suffixes
// Compares two pairs, returns 1 if first pair is smaller
int cmp(struct suffix a, struct suffix b)
{
    return (a.rank[0] == b.rank[0])? (a.rank[1] < b.rank[1] ?1: 0):
        (a.rank[0] < b.rank[0] ?1: 0);
}

// This is the main function that takes a string 'txt' of size n as an
// argument, builds and return the suffix array for the given string
int *buildSuffixArray(char *txt, int n)
{
    // A structure to store suffixes and their indexes
    struct suffix suffixes[n];

    // Store suffixes and their indexes in an array of structures.
    // The structure is needed to sort the suffixes alphabatically
    // and maintain their old indexes while sorting
    for (int i = 0; i < n; i++)
    {
        suffixes[i].index = i;
        suffixes[i].rank[0] = txt[i] - 'a';
        suffixes[i].rank[1] = ((i+1) < n)? (txt[i + 1] - 'a'): -1;
    }

    // Sort the suffixes using the comparison function
```

Recent Comments

affizerv Your example has two 4s on row 3, that's why it...

[Backtracking | Set 7 \(Sudoku\)](#) · 11 minutes ago

RVM Can someone please elaborate this Qs from above...

[Flipkart Interview | Set 6](#) · 31 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

[Software Engineering Lab, Samsung Interview | Set 2](#) · 31 minutes ago

@meya Working solution for question 2 of 4f2f round....


[Amazon Interview | Set 53 \(For SDE-1\)](#) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

AdChoices 

[► Algorithm Java](#)

[► Java Array](#)

[► An Array](#)

```

// defined above.
sort(suffixes, suffixes+n, cmp);

// At this point, all suffixes are sorted according to first
// 2 characters. Let us sort suffixes according to first 4
// characters, then first 8 and so on
int ind[n]; // This array is needed to get the index in suffixes[
            // from original index. This mapping is needed to ge
            // next suffix.
for (int k = 4; k < n; k = k*2)
{
    // Assigning rank and index values to first suffix
    int rank = 0;
    int prev_rank = suffixes[0].rank[0];
    suffixes[0].rank[0] = rank;
    ind[suffixes[0].index] = 0;

    // Assigning rank to suffixes
    for (int i = 1; i < n; i++)
    {
        // If first rank and next ranks are same as that of previous
        // suffix in array, assign the same new rank to this suffix
        if (suffixes[i].rank[0] == prev_rank &&
            suffixes[i].rank[1] == suffixes[i-1].rank[1])
        {
            prev_rank = suffixes[i].rank[0];
            suffixes[i].rank[0] = rank;
        }
        else // Otherwise increment rank and assign
        {
            prev_rank = suffixes[i].rank[0];
            suffixes[i].rank[0] = ++rank;
        }
        ind[suffixes[i].index] = i;
    }

    // Assign next rank to every suffix
    for (int i = 0; i < n; i++)
    {
        int nextindex = suffixes[i].index + k/2;
        suffixes[i].rank[1] = (nextindex < n)?
                               suffixes[ind[nextindex]].rank[0] : -1;
    }

    // Sort the suffixes according to first k characters
    sort(suffixes, suffixes+n, cmp);
}


```

AdChoices 

[► MSD Multi Array](#)

[► Algorithm PDF](#)

[► Suffix Prefix](#)

AdChoices 

[► Array Int](#)

[► Array of Arrays](#)

[► Is Array](#)

```

// Store indexes of all sorted suffixes in the suffix array
int *suffixArr = new int[n];
for (int i = 0; i < n; i++)
    suffixArr[i] = suffixes[i].index;

// Return the suffix array
return suffixArr;
}

```

```

// A utility function to print an array of given size

```

```

void printArr(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

```

```

// Driver program to test above functions

```

```

int main()
{
    char txt[] = "banana";
    int n = strlen(txt);
    int *suffixArr = buildSuffixArray(txt, n);
    cout << "Following is suffix array for " << txt << endl;
    printArr(suffixArr, n);
    return 0;
}

```

Output:

```

Following is suffix array for banana

```

```

5 3 1 0 4 2

```

Note that the above algorithm uses standard sort function and therefore time complexity is $O(n \log n \log n)$. We can use **Radix Sort** here to reduce the time complexity to $O(n \log n)$.

Please note that suffix arrays can be constructed in $O(n)$ time also. We will soon be discussing $O(n)$ algorithms.

References:

<http://www.stanford.edu/class/cs97si/suffix-array.pdf>

<http://www.cbcb.umd.edu/confcour/Fall2012/lec14b.pdf>

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Better Than Hadoop.

HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.

 LexisNexis® [Learn More](#) 

Related Tpoics:

- [Print all possible words from phone digits](#)
- [Printing Longest Common Subsequence](#)
- [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](#)
- [Dynamic Programming | Set 29 \(Longest Common Substring\)](#)



11



Tweet

3



1

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

5 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion...



devesh kumar • 6 days ago

Above solution is having a bug....check for the test case dfkdfkdfjeowewwwok

the out put is

3 6 0 22 26 19 9 12 7 1 4 8 5 2 23 17 28 21 25 16 10 27 20 24 18 11 15 14 13

which is wrong. Correct output is

6 3 0 22 26 19 9 12 7 4 1 8 5 2 23 17 28 21 25 16 10 27 20 24 18 11 15 14 13

1 ^ | v • Reply • Share ›



devesh kumar → devesh kumar • 6 days ago

The buggy is code is this

```
int rank = 0;
suffixes[0].rank[0] = rank;
ind[suffixes[0].index] = 0;
int prev_rank = suffixes[0].rank[0];
```

right code should be because prev_rank is becoming 0

```
int rank = 0;
int prev_rank = suffixes[0].rank[0];
```

```
suffixes[0].rank[0] = rank;
ind[suffixes[0].index] = 0;
```

1 ^ | v • Reply • Share ›



Thanks for pointing this out. We have updated the code.

^ | v • Reply • Share ›



kaushik Lele • 13 days ago

I have developed code using "bucket" or "map" methodology.

e.g. for string "banab"

Starting character is 'b' so bucket of 'b' with value "anab" ('b' is node value)

Then next substring is "anab". So bucket of 'a' with value "nab" ('a' is node value)

Similarly another bucket 'n' for value "anab"

Now next string is "ab". We already have bucket for 'a' with value "anab".

So we will create subbuckets

Bucket 'a' -> sub-bucket 'n' with value "ab" (starting "a" and "n" are from bucket)

Bucket 'a' -> sub-bucket 'b' with value null (starting "a" from bucket)

If there was substring "anan..". Then we would have repeated steps

Bucket 'a' -> sub-bucket 'n' -> sub-sub-bucket 'a' and so on

Refer my code for word "banana". It prints value as expected.

5 a

3 ana

[see more](#)

^ | v • Reply • Share ›



Bhagwat kumar Singh • 21 days ago

awesome job i was waiting for this . its very interesting for me thanks alot.

^ | v • Reply • Share ›

