

# 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](#)

## Suffix Array | Set 1 (Introduction)

We strongly recommend to read following post on suffix trees as a pre-requisite for this post.

[Pattern Searching | Set 8 \(Suffix Tree Introduction\)](#)

***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](#). Any suffix tree based algorithm can be replaced with an algorithm that uses a suffix array enhanced with additional information and solves the same problem in the same time complexity (Source [Wiki](#)).

A suffix array can be constructed from Suffix tree by doing a DFS traversal of the suffix tree. In fact Suffix array and suffix tree both can be constructed from each other in linear time.

Advantages of suffix arrays over suffix trees include improved space requirements, simpler linear time construction algorithms (e.g., compared to Ukkonen's algorithm) and improved cache locality (Source:

[Wiki](#))

### Example:

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

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

### Naive method to build Suffix Array

A simple method to construct suffix array is to make an array of all suffixes and then sort the array. Following is implementation of simple method.

```
// Naive algorithm 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;
    char *suff;
};

// A comparison function used by sort() to compare two suffixes
int cmp(struct suffix a, struct suffix b)
{
    return strcmp(a.suff, b.suff) < 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 alphabetically
    // and maintain their old indexes while sorting
    for (int i = 0; i < n; i++)
    {
        suffixes[i].index = i;
        suffixes[i].suff = (txt+i);
    }

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

```

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

// 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

```

The time complexity of above method to build suffix array is  $O(n^2 \log n)$  if we consider a  $O(n \log n)$  algorithm used for sorting. The sorting step itself takes  $O(n^2 \log n)$  time as every comparison is a comparison of two strings and the comparison takes  $O(n)$  time.

There are many efficient algorithms to build suffix array. We will soon be covering them as separate posts.

### ***Search a pattern using the built Suffix Array***

To search a pattern in a text, we preprocess the text and build a suffix array of the text. Since we have a sorted array of all suffixes, [Binary Search](#) can be used to search. Following is the search function. Note that the function doesn't report all occurrences of pattern, it only report one of them.

```

// This code only contains search() and main. To make it a complete running
// above code or see http://ideone.com/1Io9eN

// A suffix array based search function to search a given pattern
// 'pat' in given text 'txt' using suffix array suffArr[]
void search(char *pat, char *txt, int *suffArr, int n)

```

```

{
    int m = strlen(pat); // get length of pattern, needed for strncmp()

    // Do simple binary search for the pat in txt using the
    // built suffix array
    int l = 0, r = n-1; // Initilize left and right indexes
    while (l <= r)
    {
        // See if 'pat' is prefix of middle suffix in suffix array
        int mid = l + (r - l)/2;
        int res = strncmp(pat, txt+suffArr[mid], m);

        // If match found at the middle, print it and return
        if (res == 0)
        {
            cout << "Pattern found at index " << suffArr[mid];
            return;
        }

        // Move to left half if pattern is alphabtically less than
        // the mid suffix
        if (res < 0) r = mid - 1;

        // Otherwise move to right half
        else l = mid + 1;
    }

    // We reach here if return statement in loop is not executed
    cout << "Pattern not found";
}

// Driver program to test above function
int main()
{
    char txt[] = "banana"; // text
    char pat[] = "nan";    // pattern to be searched in text

    // Build suffix array
    int n = strlen(txt);
    int *suffArr = buildSuffixArray(txt, n);

    // search pat in txt using the built suffix array
    search(pat, txt, suffArr, n);

    return 0;
}

```

Output:

Pattern found at index 2

The time complexity of the above search function is  $O(m \log n)$ . There are more efficient algorithms to search pattern once the suffix array is built. In fact there is a  $O(m)$  suffix array based algorithm to search

a pattern. We will soon be discussing efficient algorithm for search.

### *Applications of Suffix Array*

Suffix array is an extremely useful data structure, it can be used for a wide range of problems. Following are some famous problems where Suffix array can be used.

- 1) Pattern Searching
- 2) [Finding the longest repeated substring](#)
- 3) [Finding the longest common substring](#)
- 4) [Finding the longest palindrome in a string](#)

See [this](#) for more problems where Suffix arrays can be used.

This post is a simple introduction. There is a lot to cover in Suffix arrays. We have discussed [a  \$O\(n \log n\)\$  algorithm for Suffix Array construction here](#). We will soon be discussing more efficient suffix array algorithms.

### References:

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

[http://en.wikipedia.org/wiki/Suffix\\_array](http://en.wikipedia.org/wiki/Suffix_array)

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

### Related Topics:

- [Find Union and Intersection of two unsorted arrays](#)
- [Pythagorean Triplet in an array](#)
- [Maximum profit by buying and selling a share at most twice](#)
- [Design a data structure that supports insert, delete, search and getRandom in constant time](#)
- [Print missing elements that lie in range 0 – 99](#)
- [Iterative Merge Sort](#)
- [Group multiple occurrence of array elements ordered by first occurrence](#)
- [Given a sorted and rotated array, find if there is a pair with a given sum](#)

Tags: [Advance Data Structures](#), [Advanced Data Structures](#), [Pattern Searching](#)



Tweet

3

+1

5

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

27 Comments

GeeksforGeeks

1

Login ▾

♥ Recommend 1

🔗 Share

Sort by Newest ▾



Join the discussion...



Guest • 2 months ago

```
import java.util.*;
```

```

class SuffixNode
{
    int index;

    String suffix;

    public SuffixNode(int data ,String value){

        index = data;

        suffix = value;

    }

    public static Comparator<suffixnode> compareNode = new Comparator<suffixnode>(){

        public int compare(SuffixNode a, SuffixNode b){

```

---

[see more](#)[^](#) | [v](#) • [Reply](#) • [Share](#) ›**Jitendra Sangar** • 3 months ago

Good explanation is provided here :

<http://algorithmsandme.blogspot...>[^](#) | [v](#) • [Reply](#) • [Share](#) ›**Dipankar Jana** • 4 months ago

In c++ you don't need to write "struct suffix a" . Only "suffix a" will do fine.

[^](#) | [v](#) • [Reply](#) • [Share](#) ›**Cracker** • 5 months ago<http://algods-cracker.blogspot...>[^](#) | [v](#) • [Reply](#) • [Share](#) ›**Bikash Hazarika** • 8 months ago

How do we do that sort thing in java, i use TreeMap its automatically sort data, but is that a efficient way to do the sorting ? please help

[^](#) | [v](#) • [Reply](#) • [Share](#) ›**GuojiaAgain** ➔ [Bikash Hazarika](#) • 8 months ago

Build comparator and use Arrays.sort()

[^](#) | [v](#) • [Reply](#) • [Share](#) ›

**Tauqueer Ahmad** • 8 months ago

what is the result of "txt+suffArr[mid]" in the line

```
int res = strncmp(pat, txt+suffArr[mid], m);
```

^ | v • Reply • Share ›

**Abdelkarim Mamdouh** → Tauqueer Ahmad • 8 months ago

txt+suffArr[mid] is a pointer to the txt[suffArr[mid]]

```
int res = strncmp(pat, txt+suffArr[mid], m);
```

in this line we match pat with characters start from suffArr[mid] to suffArr[mid]+m in the text.

^ | v • Reply • Share ›

**Saurabh** • 9 months ago

Nice Post!!..

^ | v • Reply • Share ›

**codex** • a year ago

Any one please explain the working of sort(suffixes, suffixes+n, cmp); and cmp function

1 ^ | v • Reply • Share ›

**GOPI GOPINATH** → codex • a year ago

sort () is a inbuilt function present in "algorithm" header.

Here first parameter is the starting position and second parameter is end , where the elements between start and end positions must be sorted using the constraint given in the cmp() function.

for better understanding follow the link:

<http://www.cplusplus.com/refer...>

1 ^ | v • Reply • Share ›

**ram** • a year ago

This uses hell of memory. isnt it ?

1 ^ | v • Reply • Share ›

**Abdelkarim Mamdouh** → ram • 8 months ago

this is naive!

^ | v • Reply • Share ›

**kaushik Lele** • a year ago

I have followed "bucket" or "map" methodology to developed code.

e.g. for string "banab"

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

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

2 3 4 5

[see more](#)

^ | v • Reply • Share ›



**sambhavsharma** • a year ago

What if we search for "bana" or "ban"? I doesn't work for these patterns?

^ | v • Reply • Share ›



**anonymous** → sambhavsharma • a year ago

we are using strncmp( text, pattern, pattern\_length ) that why it matches strncmp( banana, ban, 3 ).

^ | v • Reply • Share ›



**Kartik** → sambhavsharma • a year ago

Please take a closer look. The program works fine for these cases also.

^ | v • Reply • Share ›



**sambhavsharma** → Kartik • a year ago

I think  $mid = l + (r-1)/2$  might not be the right condition,  
It should be  $(l+r)/2$

^ | v • Reply • Share ›



**Kartik** → sambhavsharma • a year ago

Please see <http://googleresearch.blogspot...>

^ | v • Reply • Share ›



**sambhavsharma** → Kartik • a year ago

This code doesn't work for the current condition.

for value "bana"



for value "ana".

and the document says one possible fix could be:

```
int mid = low + ((high - low) / 2);
```

^ | v • Reply • Share ›



**Bikash Hazarika** → sambhavsharma • 8 months ago

the same they have did,

```
int mid = l + (r - l)/2;
```

```
mid=low(l) +( high(r)-low(l))/2
```

got it !

^ | v • Reply • Share ›



**Saurabh Verma** • a year ago

how to write code in comment box??

^ | v • Reply • Share ›



**Kartik** → Saurabh Verma • a year ago

Looks like code in Disqus comment system doesn't work very well. Using a site [ideone.com](http://ideone.com) and sharing the link seems to be a good idea.

^ | v • Reply • Share ›



**CREATORS93** • a year ago

this is great

^ | v • Reply • Share ›



**geek** • a year ago

There is a problem in the implementation of pattern search function search. The string comparison function strcmp should not be used, because it searches for exact match. It is possible that pattern is a substring of suffix array element, so instead of strcmp I would suggest using of Find() function. For example :- If I search for pattern "nan", your implementation fails and returns pattern not found although it is present, Please correct it

1 ^ | v • Reply • Share ›



**GeeksforGeeks** → geek • a year ago

Geeks, thanks for pointing this out. We have updated the code to use strncmp()

<http://www.cplusplus.com/refer...>

^ | v • Reply • Share ›



**Thewar** • a year ago

Most awaited post :)

2 ^ | v • Reply • Share ›



- 
- 
- 
- - [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