

**IMPLEMENTATION SUFFIX ARRAY**

Written to fulfill the 2nd task of

Algorithm and Data Structure

Group 4

Names : Andrean Ahmad Fauzi

Alifiano Rifqi Athallah

Dimas Arya Kusuma Putra

Class : 2CC10

CEP CCIT

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

2019

**PREFACE**

First of all, author say prays and gratitude to Allah SWT. Which has led us to complete our ISAS paper.

Not forgetting author also say a lot gratitude for the help of those who have contribute by giving both the body and mind to make this paper.

Author realize this paper still far from the perfect word, we hope this paper can help and add insight to readers. Therefore, we hope contructive critisim and

suggestions to improve this paper.

Last, author say thanks and apologize if there many mistakes of words.

Depok, 12 June 2020

Author

**TABLE OF CONTENTS**

PREFACE .............................................................................................................. i

INTRODUCTION .................................................................................................. 5

I.1 Background ...................................................................................................... 5

I.2 Writing Objective ............................................................................................ 5

I.3 Problem Domain .............................................................................................. 5

I.4 Writing Methodology .............................................................. ……………....5

I.5 Writing Framework ................................................................. ……………….6

BASIC THEORY ................................................................................................... 7

II.1 Brief Explanation of Algorithm ...................................................................... 7

II.2 Brief Explanation of Data Structure ............................................................... 7

II.3 Brief Explanation of Array ............................................................................. 8

II.4 Brief Explanation of Suffix Tree ................................................................... 8

II.5 Suffix Array ................................................................................................... 8

PROBLEM ANALYSIS ..................................................................................... 12

III.1 Building Suffix Array ..................................................................................12

CONCLUSION AND SUGGESTION .................................................................22

IV.1 Conclusion ................................................................................................. 22

IV.2 Suggestion ......................................................................... ……………….22

BIBLIOGRAPHY................................................................................................. 23

**TABLE OF IMAGE**

Image 2.1 ……………………………………………………………………. 11

Image 3.1 ……………………………………………………………………. 12

Image 3.2 ……………………………………………………………………. 12

Image 3.3 ……………………………………………………………………. 13

Image 3.4 ……………………………………………………………………. 13

Image 3.5 ……………………………………………………………………. 14

Image 3.6 ……………………………………………………………………. 14

Image 3.7 ……………………………………………………………………. 15

Image 3.8 ……………………………………………………………………. 15

Image 3.9 ……………………………………………………………………. 16

Image 3.10 ……………………………………………………………………. 16

Image 3.11 ……………………………………………………………………. 17

Image 3.12 ……………………………………………………………………. 17

Image 3.13 ……………………………………………………………………. 18

Image 3.14 ……………………………………………………………………. 18

Image 3.15 ……………………………………………………………………. 19

Image 3.16 ……………………………………………………………………. 20

Image 3.17 ……………………………………………………………………. 20

**CHAPTER I**

# INTRODUCTION

## I.1 Background

To find string matching there are various ways. Among them is using Suffix Array. This paper will discuss data structures that can solve string problems, namely suffix arrays. This paper will also discuss the use of suffix arrays, algorithms, and a little about algorithms, data structures, arrays, and suffix trees.

## I.2 Writing Objective

The purpose of writing this paper is to give better understanding about data structure especially at Suffix Array. The explanation also cover definition of algorithm, data structure, and how implementation suffix array. The author hope the reader will get better knowledge about operating system by read this paper.

## I.3 Problem Domain

This paper will discuss about the Implementation of Suffix Array, which includes introducing Algorithm, Data Structure, and array. And in this paper also discuss about algorithm for making suffix array.

## I.4 Writing Methodology

In this paper, the author use two method. The first method that used in writing this paper is library research. Collecting data with browsing information from reference source contain on online sites that relate with the topic of this paper The second method that used in **writing** this paper is discussion method. After collecting data from reference source, we discuss and compose the data into structure contents for completing this paper.

**I.5 Writing Framework**

Here is a systematic writing of a paper entitle “Building Suffix Array”

CHAPTER I : INTRODUCTION

- Background

- Writing Objective

- Problem Domain

- Writing Methodology

- Writing Framework

CHAPTER II : BASIC THEORY

- Brief Explanation of Algorithm

- Brief Explanation of Data Structure

- Brief Explanation of Array

- Brief Explanation of Suffix Tree

- Suffix Array

CHAPTER III : PROBLEM ANALYSIS

- Building Suffix Array

CHAPTER IV : CONCLUSION AND SUGGESTION

- Conclusion

- Suggestion

# CHAPTER II

# BASIC THEORY

# II.1 Brief Explanation of Algorithm

# Basically algorithm is a logical and systematic arrangement that is used to solve or solve a particular problem. So, each logical arrangement ordered by a particular systematics used to solve the problem can be classified as an algorithm.

# In the world of programming, algorithms are used to build various kinds of software used on computer devices. Application of the algorithm system in making software is usually done through several types of programming languages such as the C programming language, the C # programming language, and the Visual Basic programming language, etc.

# II.2 Brief Explanation of Data Structure

# Data structure is a data management so that the data can be used more efficiently and effectively. In programming languages, data structures are often shown physically in the form of tables (usually in programming languages / database management based on visuals), but in some programming languages that are not visual based, data structures are more about managing data with certain rules.

# Data structures are divided into two types, simple and compound data structures. Simple data structures are divided into two, arrays and records. Whereas compounds are divided into linear (stack, queue, list and multi-list) and non-linear (binary-tree, graph).

# II.3 Brief Explanation of Array

An array data type is a data type that consists of a collection of other data types. In Indonesian, arrays are also known as arrays. With arrays, the process of storing data into variables becomes more efficient and easier, especially if we have large amounts of data.

Members or the contents of the array itself must be of one type of data type, for example consisting of a collection of integers only, only a collection of characters (char), and a collection of only fractional numbers (float). In C language, we cannot create 1 array with various data types (must be 1 type only).

**II.4 Brief Explanation of Suffix Tree**

In computer science (computer science), a suffix tree is a data structure that presents the suffix of the string so that it's easy certain implementations quickly for various operation on a string.

The suffix tree of the S string is a tree with edges labeled string which is the suffix of S and fit exactly one path from root to leaf. In building this type of tree for the S string, it will takes time and linear space depending on length S. If it has formed, several operations can be performed faster, for example searching for substrings in S, searching substring if how many small errors in writing allowed, and look for regular expression similarities.

Suffix tree also provides a one-time solution first linear for the longest common substring problem. This causes: save the suffix tree of a string usually requires more than more space on storing the string itself.

**II.5 Suffix Array**

Suffix array is data structures that are designed for optimize and streamline search in big text. The data structure is as simple as an array / array contains all the pointer to the text suffix sorted alphabetically or alphabetically. Each suffix is a string that starts at certain positions in the text and end at the end of the text.

Searching for text can be done with binary search using suffix array.

# II.5.1 Algorithm Which Construct Suffix Array

# To make suffix arrays, here are some algorithms that can be used to create suffix arrays.

# II.5.1.1 Algorithm O(N2 lg N)

The easiest way to make is to make all existing suffixes of the given string, then sort it by quicksort or like to get an ordered suffix. After that, an index search is performed that corresponds to that suffix to produce array suffix is desirable. The time complexity of this algorithm is fleeting is O (N lg N) of the time it takes to sorting, but the correct one is O (N2 lg N), because each suffix comparison operation has complexity O (N). An example is the string "aaaaaaaaaaaaaa".

**II.5.1.2 Algorithm O (N (lg N)2 )**

A better way to make suffix arrays on basically the same as above, but done such that comparative operations can done constantly. The trick is sorting first suffix prefixes with length 2i, then use the results of that order for speed up the sequencing of the suffix prefixes by length 2 i + 1. The ordering is complete after the length of the suffix prefix used is at least the same as the length of the string. Case the base is the length of the suffix 1 prefix, that is the character alone. The process of accelerating the comparison between 2 pieces prefix Suffix is done by comparing the results of sorting from the first half of the prefix of the suffix, then later with the latter half of the suffix prefix. Because of the results of this order in the form of an integer, then this comparison has constant time complexity.

For example, the process of creating an suffix array from the word "banana" is :

1. Step 1: { 2 1 3 1 3 1 }, because of the existing characters is {a, b, n}

2. Step 2: { 3 2 4 2 4 1 }, which is for {ba, an, na, an, na, a}

3. Step 3: { 4 3 6 2 5 1 }, which is for {bana, anan, nana, ana, na, a}

4. Step 4 : { 4 3 6 2 5 1 }, yaitu untuk {banana, anana, nana, ana, na, a}

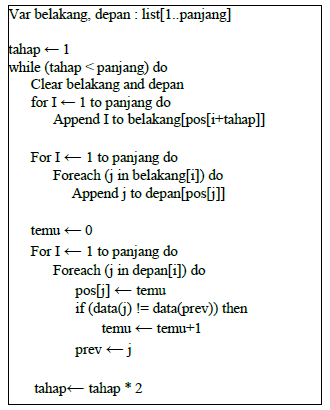
After the ordering process above is complete, array suffix can be made quickly, because the contents of each array index is the location of the suffix in the suffix array. Results the end is {6, 4, 2, 1, 5, 3}.

The method of comparison above has time complexity equal to O (N (lg N) 2), because there is another N stage sorting, where each sorting has time complexity O (N lg N). The pseudocode given above is still very rough,therefore the authors suggest paper as a reference for those who want to try to make own program.

**II.5.1.3 Algorithm O(N)**

Algorithm for making suffix arrays with complexity linear time already exists, one of which is making suffix tree first with a linear algorithm, like the Ukkonen Algorithm, then traversal is performed with depth-first search in that tree. It is just this method is actually not recommended, because suffix tree itself has the ability equivalent to suffix array that has been added by LCP Array.

**II.5.1.4 Algorithm O(N lg N)**

 The above algorithm can be improved by observation that the order of the results of the previous sorting will not uses a number greater than N, which is the length of the string the suffix array is currently being made. Because it's technique sorting that is not based on comparisons can be used, such as counting sort or radix sort. In a way this, the complexity of the ordering time of each stage is O (N), so the overall time complexity is O (N lg N). The following is pseudocode implementation using a linked-list data structure.

Images 2.1 Algorithm O(N lg N)

**BAB III**

**PROBLEM ANALYSIS**

**III.1 BUILDING SUFFIX SEARCHING PROGRAM WITH C++ AND JAVA**

Suppose we have a "sekolah" sample text and will build a suffix array from it.

**III.1.1 Assign "index point" to the sample text.**

The point index determines the position so that a search can be performed. In this case, the point index is paired on character by character. After that, we can do a search on the text with suffix array with the initial position anywhere according to our needs.

Image 3.1 assign index point

**III.1.2 Sort the index points based on each corresponding suffix.**

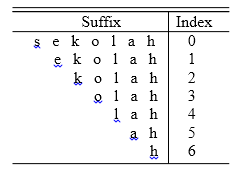
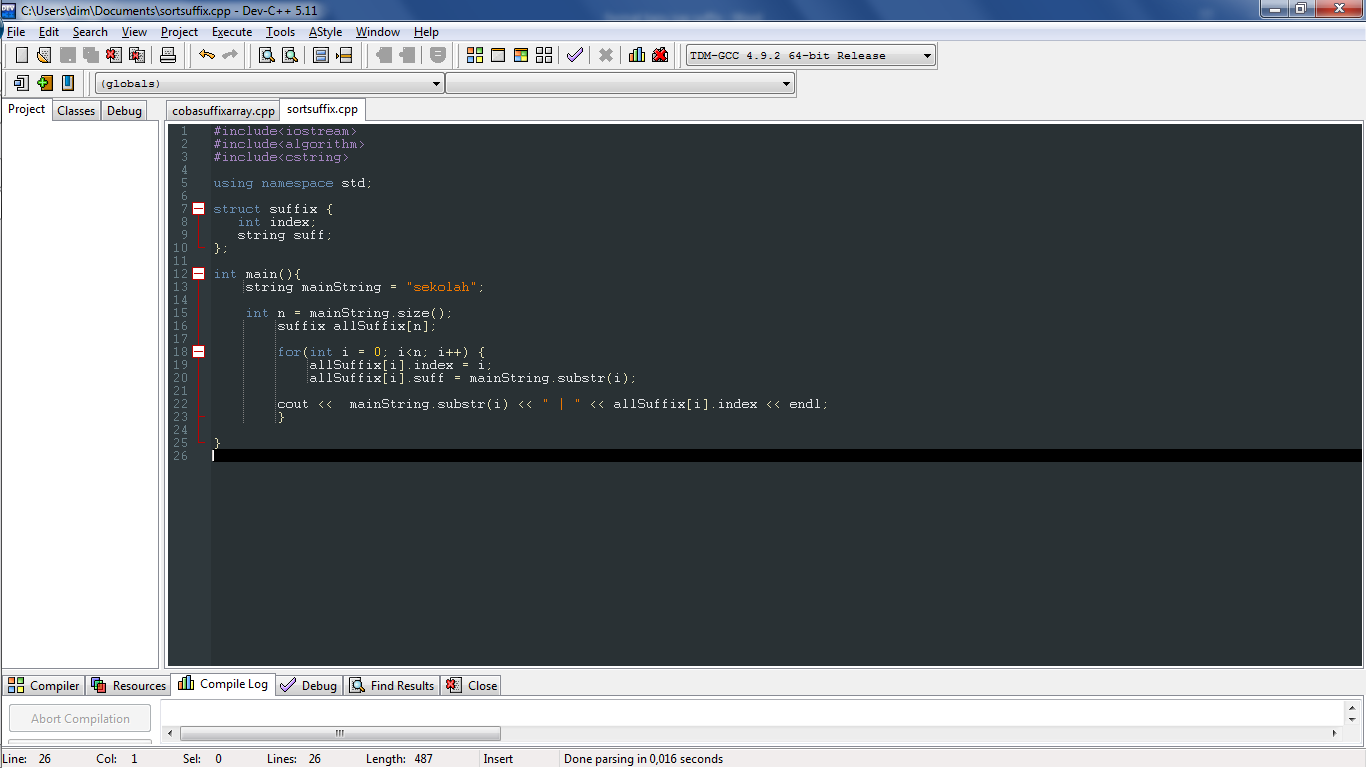
Conformity between the index and suffix can be observed in the image below.

Image 3.2 sort index point

* **C++**

Then, author apply into C++ language using this code

image 3.3 code sorting index c++

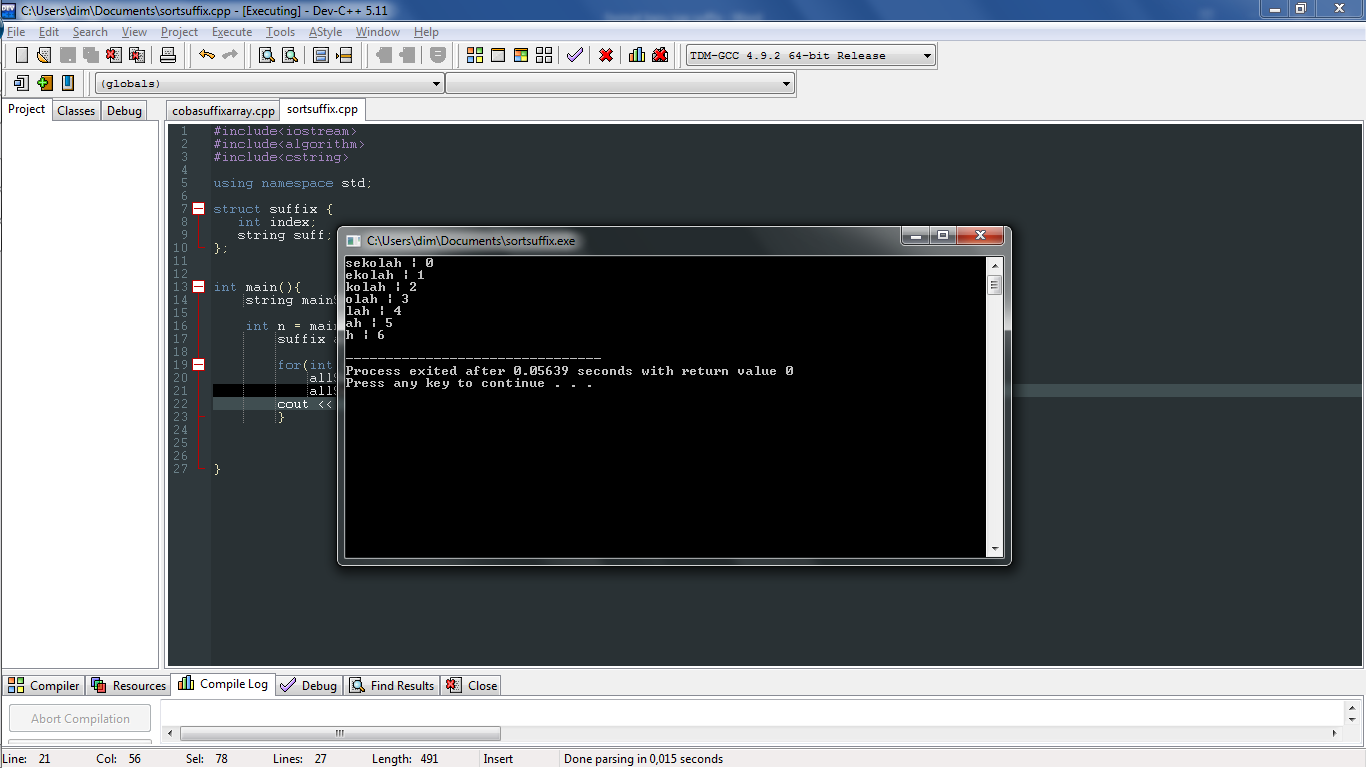
Let’s see in line 15 until 20. In line 15 until 16, that’s code for convert from sample text to suffix. Then, in line 19 until 20 is code for give the suffix index. The result in image below.

Image 3.4 result index c++

* **Java**

Then, this is code in java for using suffix

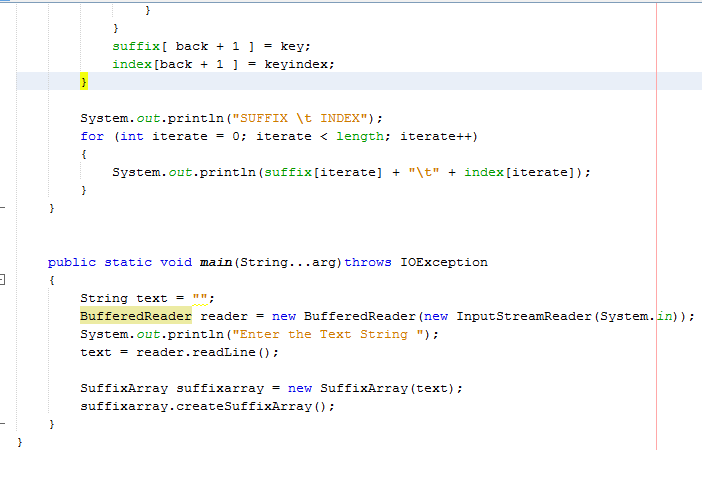


Image 3.5 sorting in Java

Then, if you run this a program you cna see out put “Enter the Text String” and you must input text in a prorgam like this, in the pictur a result for code java programing in suffix.

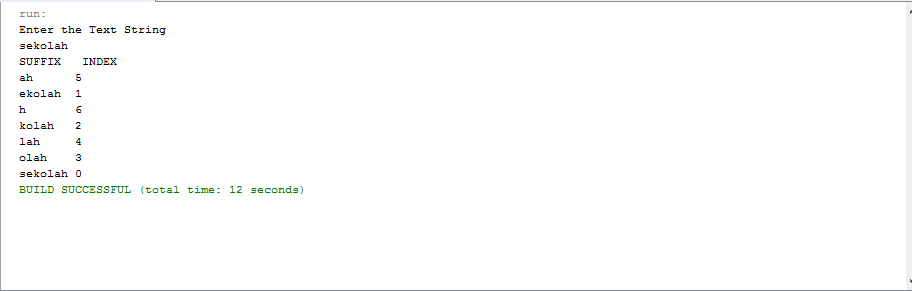


Image 3.6 result index Java

**III.1.3 Sort the suffix in alphabetical order as below.**

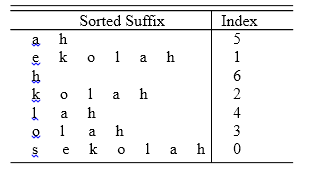
****After the suffix has been sorted, the suffix array from the "sekolah" sample is visible.

Image 3.7 sort suffix alphabetical

* C++

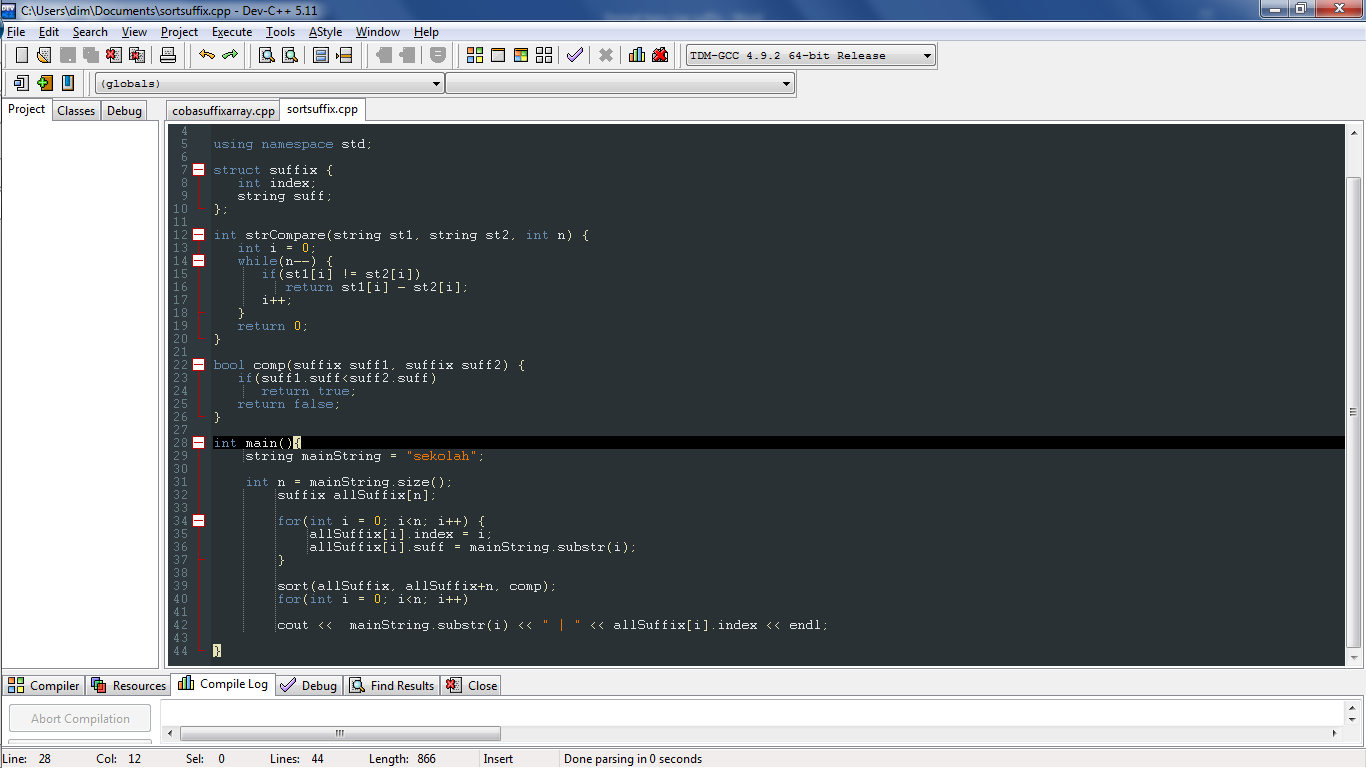
For sorting the suffix, create code for compare before the string. Readers can see the code in image below. Put this code before “*int main(){...}*”

Image 3.8 code compare C++

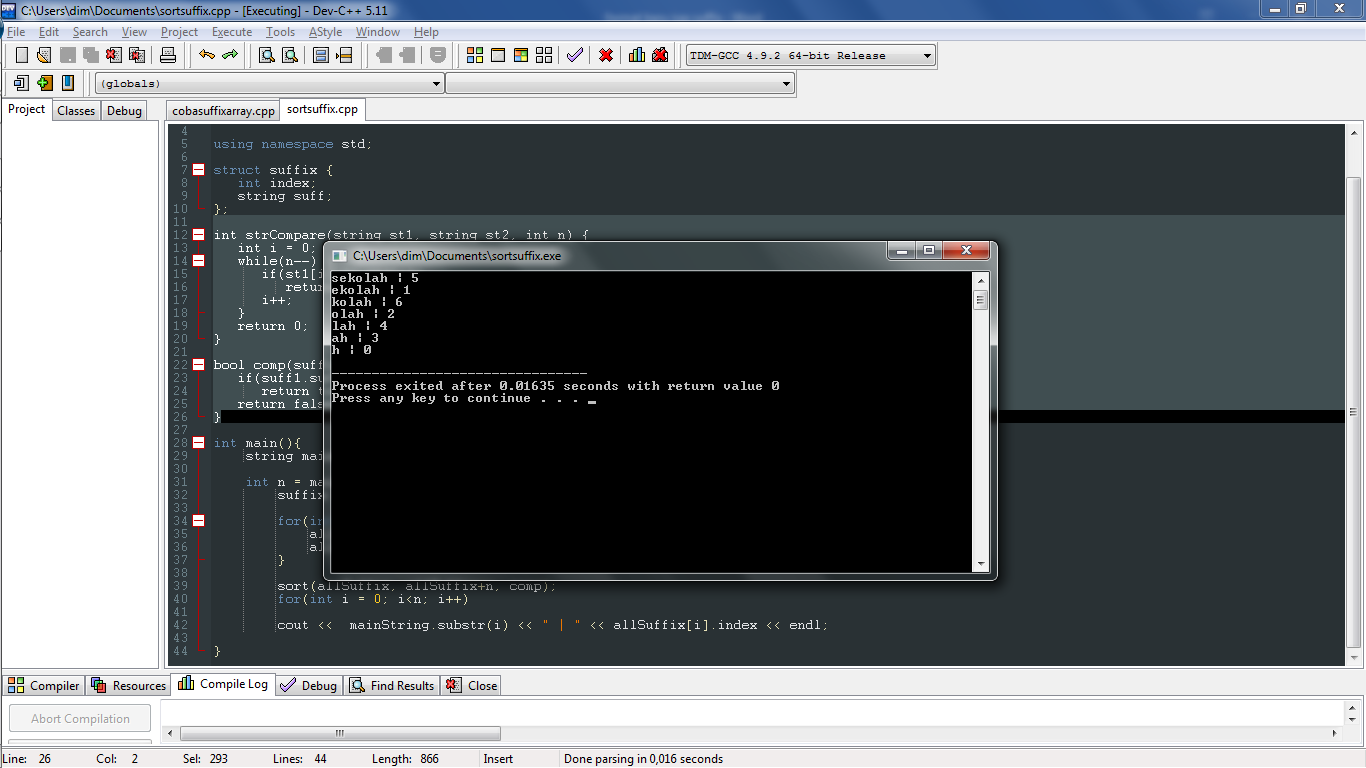
And the result like in image below.

Image 3.9 result sort suffix c++

* **Java**

For sorting the suffix, create code for compare before the string. Readers can see the code in image below.

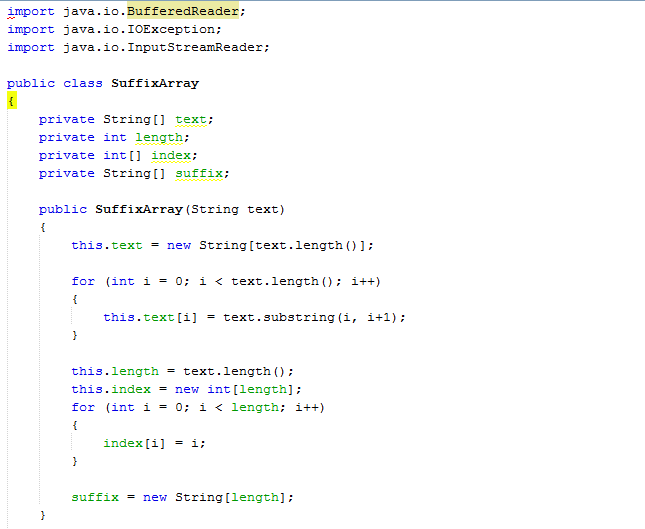


Image 3.10 Code Compare in Java

And this a result suffix array in java

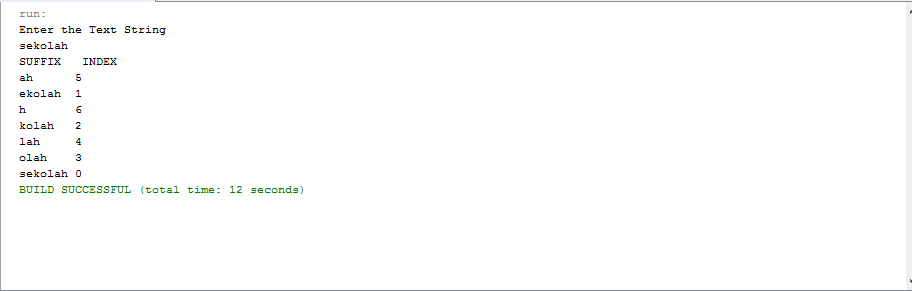


Image 3.11 result index java

**III.1.4 Final Result**

The index results from suffix sequencing become suffix arrays for "sekolah" sample text. This sorting index will later become the suffix array.

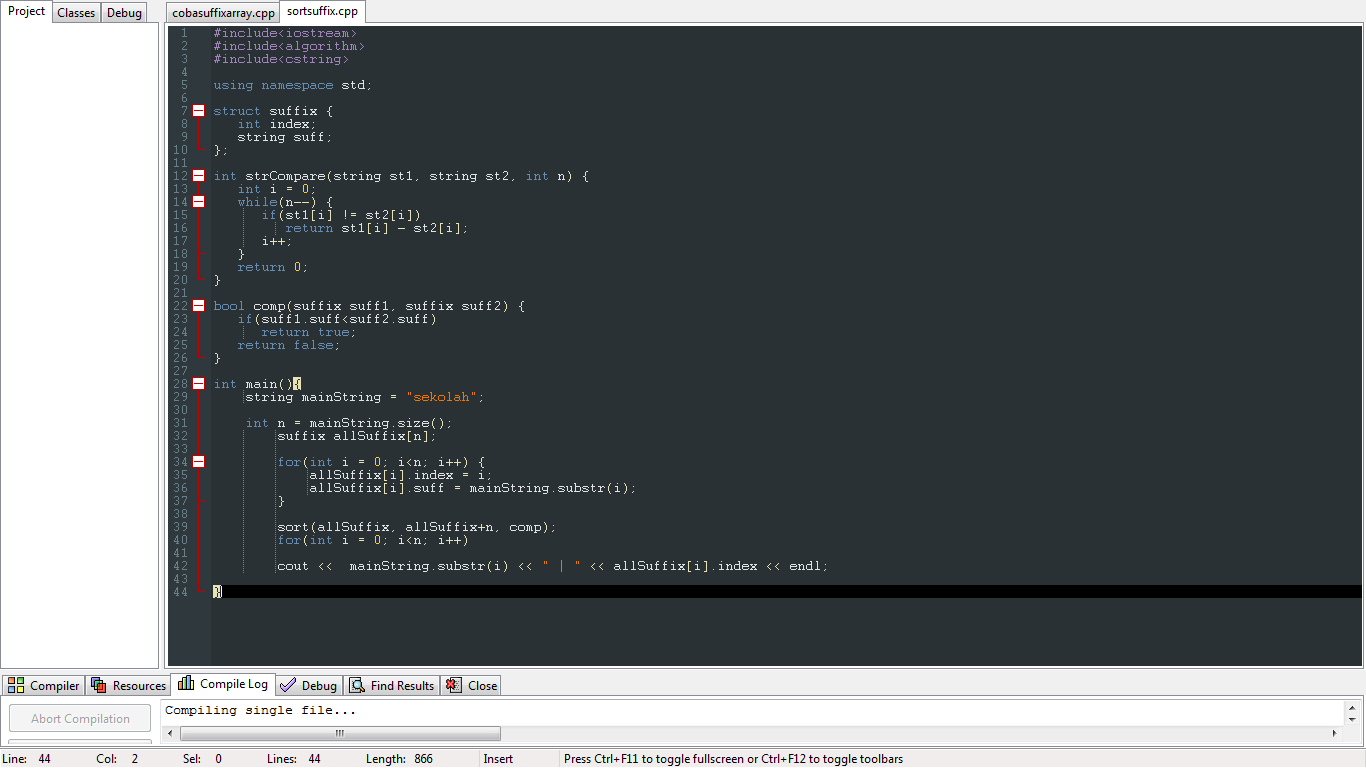
* C++
  + 1. Code.

Image 3.12 full code c++

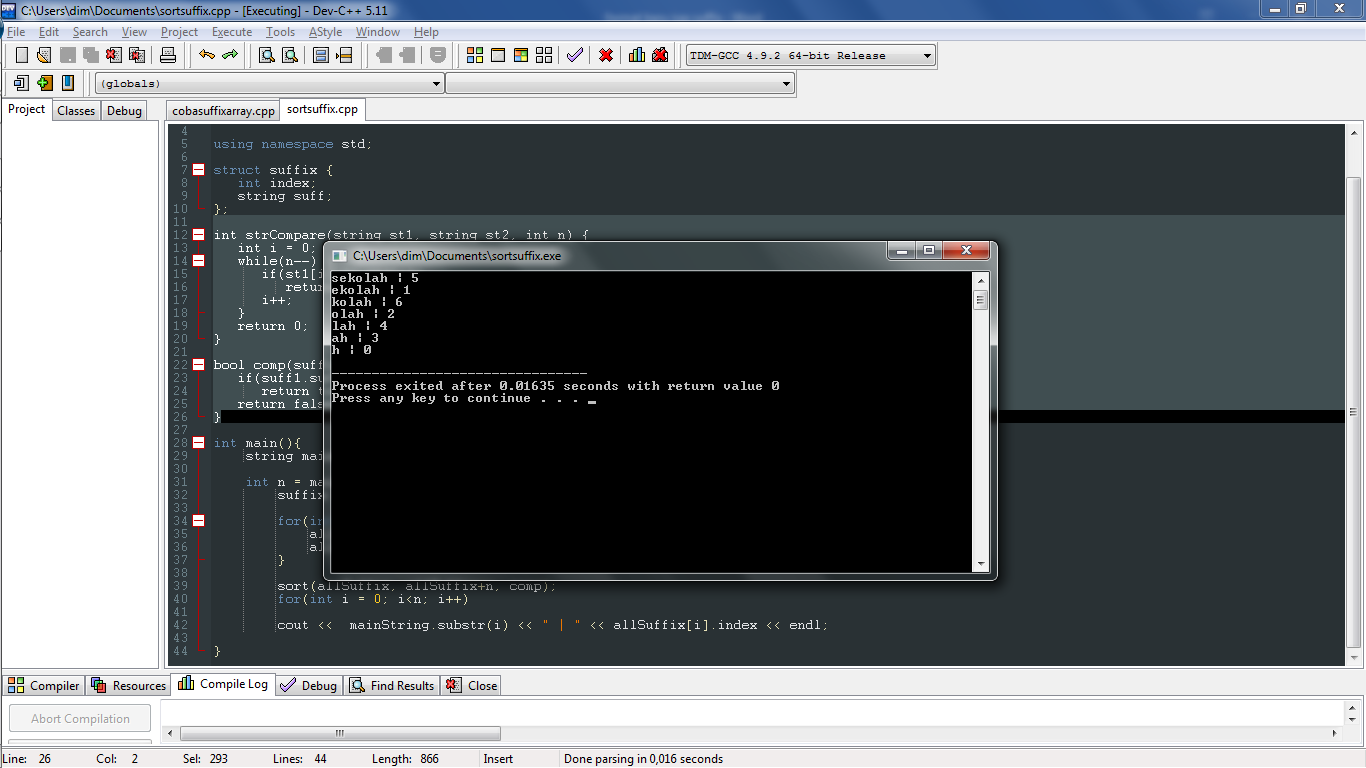
* + 1.  Program.

Image 3.13 result c++

* **Java**

1. Code.

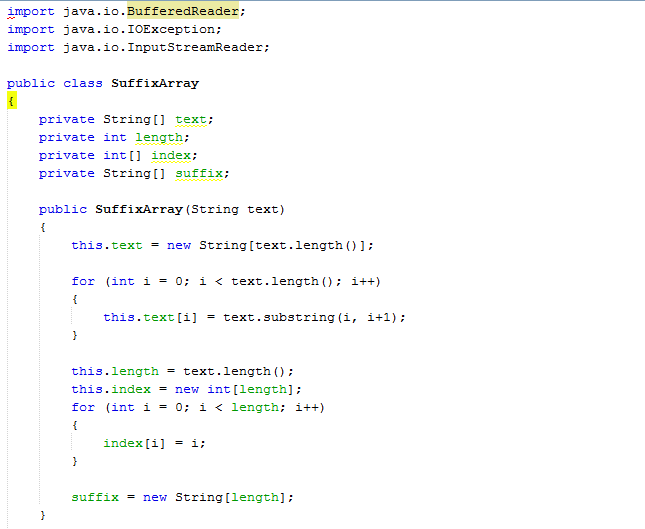


Image 3.14 full code Java

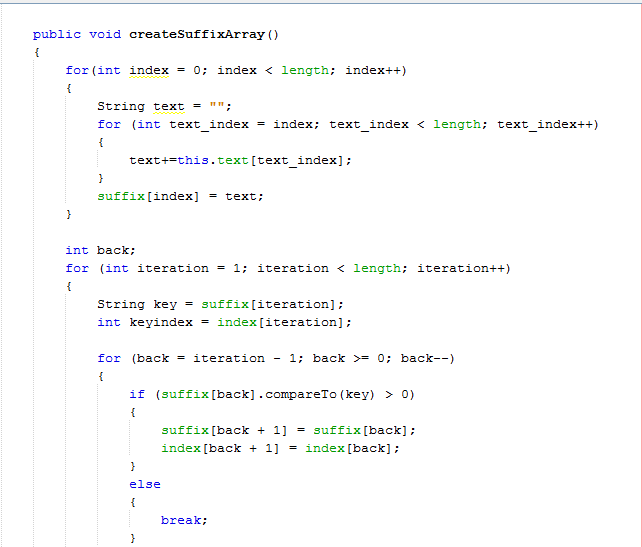


Image 3.15 full code Java

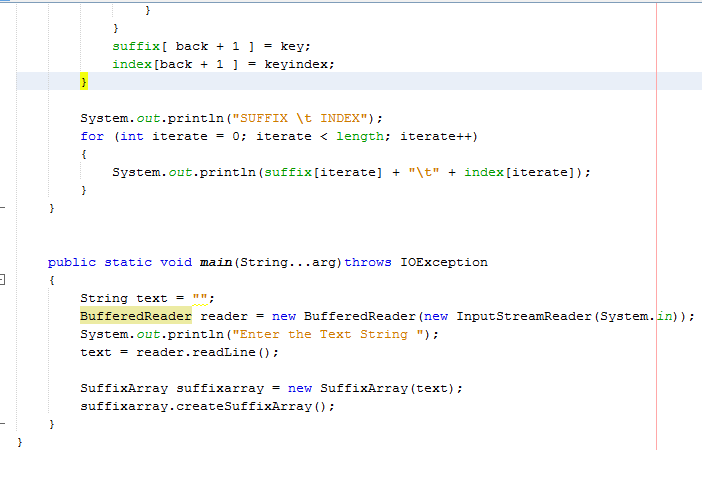


Image 3.16 full code Java

1. Program.

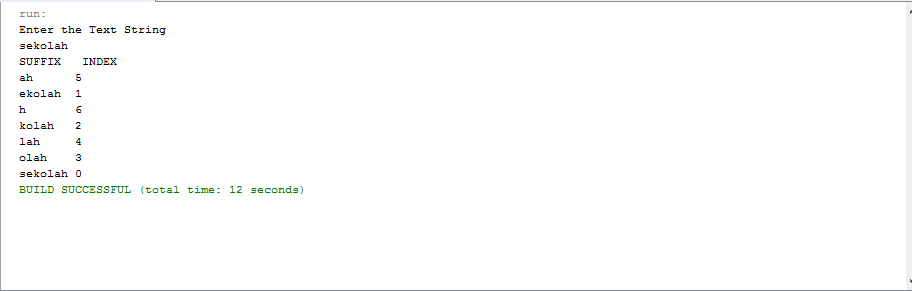


Image 3.17 result Java

Usually a text will be said to end if there is a marker. In this case, the marker is a symbol. The symbol usually used is "$". The addition of this symbol will affect the length of the text. The length of the text will increase by one from its original length, but with this marker, some processes will become easier.

By including the symbol "$" as a suffix, the sample text will become "sekolah$". So the suffix which has been sorted alphabetically is as below.

7 $

5 ah$

1 ekolah$

6 h$

2 kolah$

4 lah$

3 olah$

0 sekolah$

Thus, the suffix array for the "sekolah" text example above is {7,5,1,6,2,4,3,0}.

**III.2 IMPLEMENTATION SUFFIX ARRAY IN LIFE**

The use of suffix arrays is widely used to solve various string problems and is usually used in the field of biological research. Many algorithms can be used for this problem, however, these algorithms are difficult to develop to solve other types of problems that also involve strings, power strings, etc. And, most other algorithms besides suffix arrays, preprocessing on the pattern sought, not on the text, so preprocessing must be done many times if there are many searches. Preprocessing data is a process / step taken to make raw data into quality data. Suffix arrays can also be used to search letters or words in a dictionary.

So, according to the author, suffix arrays are more efficient, because they don't need to do a lot of preprocessing, and are suitable for various string problems.

**CHAPTER IV**

# CONCLUSION AND SUGGESTION

**IV.1 Conclusion**

Making searching proggram string using C++ and Java  
easy for both, But, there is deference in output, if you use C++ language output is only the index sought. While if you use Java lanuage index for sampel text will show all

**IV.2 Suggestion**

Author suggest, if we will looking for sufix proggram, we better use java language, because the display more completely and clearly.

**BIBLIOGRAPHY**

Suffix Tree dan Suffix Array.

<http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2009-2010/Makalah0910/MakalahStrukdis0910-085.pdf> **[05/03/2020]**

Pengertian Algoritma dalam Dunia Pemograman.

<https://pengertiandefinisi.com/pengertian-algoritma-dalam-dunia-pemrograman-komputer/> **[06/03/2020]**

Pengertian Algoritma dan Struktur Data.

<http://baiqlealeadyanahanesty.blogspot.com/2013/09/pengertian-algoritma-dan-struktur-data.html> **[06/03/2020]**

Suffix Array.

<https://www.tutorialspoint.com/Suffix-Array> **[07/06/2020]**

Java Program to Implement Suffix Array.

<https://www.sanfoundry.com/java-program-implement-suffix-array/> **[07/06/2020]**