****

**ISAS (Information Search and Analysis Skills)**

**“Knuth-Morris-Pratt Algorithm”**

Name :

Muhamad Dani Setiawan

Muhammad Dava Farrakhan

Natasha Syeikinah Marsauli

Class : 1SC9

Faculty : Mr. M. Hudya Ramadhana

**Continuing Education Program Center for Computing and Information Technology Faculty of Engineering  
University of Indonesia**

**PREFACE**

Thanks to Almighty God who has given the blessing for finishing this ISAS (Information Search and Analysis Skill) about Knuth-Morris-Pratt Algorithm. Also we wish to express our deep and sincere gratitude for those who have guided in completing this paper specially Mr. M. Hudya Ramadhana.

Our ISAS (Information Search and Analysis Skill) paper contains about Implementation of Knuth-Morris-Pratt on Dictionary’s Application. There are definition about

This paper may have some flaws and imperfections, we hope readers will give comments and suggestions in building this article also can expand the knowledge of the readers.

Depok, Feb 27th 2019

Author

**TABLE OF CONTENTS**

**PREFACE i**

**TABLE OF CONTENTS ii**

**TABLE OF FIGURES iv**

**CHAPTER I INTRODUCTION 1**

1.1 Background 1

1.2 Writing Objective 1

1.3 Problem Domain 1

1.4 Writing Methodology 2

1.5 Writing Framework 2

**CHAPTER II BASIC THEORY**

**CHAPTER III PROBLEM ANALYSIS**

**CHAPTER IV CONCLUSION AND SUGGESTION**

IV.1. Conclusion

IV.2. Suggestion

**BIBLIOGRAPHY**

**TABLE OF FIGURES**

**CHAPTER I  
INTRODUCTION**

1. **Background**

As the number of sites on the Web increased in the mid-to-late 90s, search engines started appearing to help people find information quickly. Search engines developed business models to finance their services, such as pay per click programs offered by Open Text in 1996 and then Goto.com in 1998. (Bullinaria, 2018)

Search Engine that serves to help users to search for information, in the form of articles, videos, images, etc. It can also be said that this program was created to look for data stored on the computer.

Some of the Search Engines also add features such as Word Suggestion, is the feature that gives the closest suggestion to the word we want to find. In making this feature a string matching algorithm is used. Examples of these String Matching algorithms include Brute Force, Knuth-Morris-Pratt, Boyer-Moore, Karp-Rabin, etc.

Several research related to this problem is research by Januardi, A that is about the Brute Force algorithm and Boyer Moore algorithm which can be applied to the Word Suggestion feature. The Brute Force algorithm in its application checks characters from left to right and Boyer Moore's algorithm checks characters from right to left. In this research both algorithms can be applied properly, but the best is the Boyer Moore algorithm because it does the fastest search.

Then further research is carried out by the Ecaputri, G.H. Sinaga. Y.A. She does a string search using the Knuth-Morris-Pratt algorithm in the Word Search application. The results of this study are the Knuth-Morris-Pratt (KMP) algorithm stores a number of information that is carried out to make a number of shifts and shifts further than Brute Force algorithm that only move one character. This causes the Knuth-Morris-Pratt (KMP) algorithm to shorten the matching time of the string.

Therefore, this ISAS will discuss how to apply string matching algorithms, namely Knuth-Morris-Pratt Algorithm. (Seymour, 2011)

1. **Writing Objective**

The purpose of writing this ISAS is to give better understanding about Structured Data and Algorithm especially Knuth-Morris-Pratt Algorithm. This paper will explain what is structured data, algorithm, and how it works.

1. **Problem Domain**

In this section writer will discuss a problem about :

1. Discuss about Structured Data
2. Discuss about Algorithm
3. Discuss about Knuth-Morris-Pratt (KMP) Algorithm
4. **Writing Methodology**

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

1. **Writing Framework**

The framework of the paper, are listed below:

**CHAPTER I: INTRODUCTION**

I.1 Background

I.2 Writing Objective

I.3 Problem Domain

I.4 Writing Methodology

I.5 Writing Framework

**CHAPTER II: BASIC THEORY**

**CHAPTER III: PROBLEM ANALYSIS**

**CHAPTER IV: CONCLUSION AND SUGGESTION**

IV.1 Conclusion

IV.2 Suggestion

**CHAPTER II  
BASIC THEORY**

**II.1 Data Structures**

**II.1.1 Data Structures Explanation**

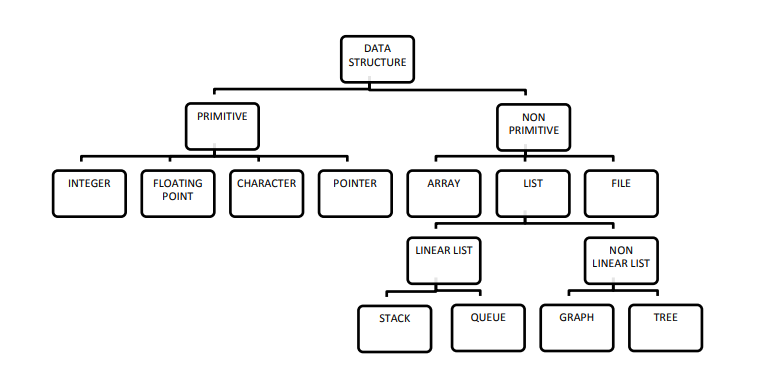
Structure means arrangement or level, and data means something symbol, letter, number symbol that states something.

Data structure is a procedure for storing, compiling, and analyzing data so the data can be used efficiently. Data structure is not only where the part is stored but how the relationship between one data and another.

In programming techniques, the data structure means the layout of data that contains columns of data, both columns that are visible to the user or columns that are only used for programming purposes that are not visible to the user. Each row of a collection of columns is called a record. The column width for data can change and vary. There are columns whose width changes dynamically according to the input of the user and there is also a column whose width is fixed.

Data structures can be used for database processing such as for financial data purposes, or for word processors whose columns change dynamically. Examples of data structures can be seen in databases, word processing, compressed images, and compressing files (compresses) with certain techniques that utilize data structures. (Jadeja, 2010).

**II.1.2 Classification of Data Structures**



**Figure 2.1 Classification of Data Structure (**[**http://www.darshan.ac.in/**](http://www.darshan.ac.in/)**)**

**II.1.3.1 Primitive Data Structure**

Primitive data structures are basic structures and are directly operated upon by machine instructions, it has different representations on different computers. Examples of Primitive Data Structure are Integers, Floats, Character, and Pointers.

1. Integer it is a data type which allows all values without fraction part. We can use it for whole numbers.
2. Float it is a data type which use for storing fractional numbers
3. Character it is a data type which is used for character values.

**II.1.3.2 Non-Primitive Data Structure**

Non-Primitive data structure is more sophisticated data structures, this is derived from primitive data structures. The non-primitive data structures emphasize on structuring of a group of homogeneous or heterogeneous data items. Examples of Non-primitive data type are Array, List, and File etc. A Non-primitive data type is further divided into Linear and Non-Linear data structure.

1. Array: An array is a fixed-size sequenced collection of elements of the same data type,
2. List: An ordered set containing variable number of elements is called as Lists.
3. File: A file is a collection of logically related information. It can be viewed as a large list of records consisting of various fields.

**II.1.3.3 Linear Data Structure**

A data structure is said to be Linear, if its elements are connected in linear fashion by means of logically or in sequence memory locations. There are two ways to represent a linear data structure in memory, static memory allocation and dynamic memory allocation. The possible operations on the linear data structure are: Traversal, Insertion, Deletion, Searching, Sorting and Merging. Examples of Linear Data Structure are Stack and Queue.

1. Stack: Stack is a data structure in which insertion and deletion operations are performed at one end only.
2. The insertion operation is referred to as ‘PUSH’ and deletion operation is referred to as ‘POP’ operation.

B. Stack is also called as Last in First out (LIFO) data structure.

1. Queue: The data structure which permits the insertion at one end and Deletion at another end, known as Queue.
2. End at which deletion is occurs is known as FRONT end and another end at which insertion occurs is known as REAR end.
3. Queue is also called as First in First out (FIFO) data structure.

**II.1.3.4 Non-Linear Data Structure**

Nonlinear data structures are those data structure in which data items are not arranged in a sequence.Examples of Non-linear Data Structure are Tree and Graph.

1. **Tree:** A tree can be defined as finite set of data items (nodes) in which data items are arranged in branches and sub branches according to requirement.

1**.** Trees represent the hierarchical relationship between various elements.

2**.** Tree consist of nodes connected by edge, the node represented by circle and edge lives connecting to circle.

**B. Graph:** Graph is a collection of nodes (Information) and connecting edges (Logical relation) between nodes.

1**.** A tree can be viewed as restricted graph.

2. Graphs have many types: Un-directed Graph, Directed Graph, Mixed Graph, Multi Graph, Simple Graph, Null Graph, Weighted Graph. (Bullinaria, 2018).

**II.2 Algorithm**

**II.2.1 Algorithm Explanation**

An essential aspect to data structures is algorithms. Data structures are implemented using algorithms. An algorithm is a procedure that you can write as a C function or program, or any other language. An algorithm states explicitly how the data will be manipulated.

**II.2.1 Algorithm Efficiency**

Some algorithms are more efficient than others. We would prefer to choose an efficient algorithm, so it would be nice to have metrics for comparing algorithm efficiency.The complexity of an algorithm is a function describing the efficiency of the algorithm in terms of the amount of data the algorithm must process.Usually there are natural units for the domain and range of this function. There are two main complexity measures of the efficiency of an algorithm. (Bullinaria, 2018)

**II.3 Knuth-Morris-Pratt Algorithm**

KMP algorithm (Knuth Morris Pratt) is an algorithm used to perform string matching process. This algorithm is a type of Exact String Matching Algorithm which is a precise string matching with the arrangement of characters in a matched string having the number and sequence of characters in the same string. In the KMP algorithm, the information used to do a further shift, not just one character like the Brute Force algorithm. This algorithm performs matching from left to right (C. S. Rao, 2013).

**CHAPTER III  
PROBLEM ANALYSIS**

**III.1 Knuth-Morris-Pratt Algorithm**

The Knuth-Morris-Pratt algorithm was developed by D. E. Knuth, along with J. H. Morris and V. R. Pratt (Jadhav, 2015) (El-Rayyes, 2012). The Knuth-Morris-Pratt algorithm is the development of the previous string search algorithm, the Brute Force algorithm. Brute-Force algorithm is the simplest basic algorithm in solving string matching problem that examines every position in the text between 0 and n-m, where n is the length of text/number of filenames stored on the computer and m is the character length of a pattern (the word to search) (Daptardar, 2006).

KMP algorithm (Knuth Morris Pratt) is an algorithm used to perform string matching process. This algorithm is a type of Exact String Matching Algorithm which is a precise string matching with the arrangement of characters in a matched string having the number and sequence of characters in the same string. In the KMP algorithm, the information used to do a further shift, not just one character like the Brute Force algorithm. This algorithm performs matching from left to right (C. S. Rao, 2013).

**III.2 Component of Knuth-Morris-Pratt Algorithm**

The prefix-function ᴨ :

It preprocesses the pattern to find matches of prefixes of the pattern with the pattern itself

It is defined as the size of the largest prefix of P[0..j − 1] that is also a suffix of P[1..j].

It also indicates how much of the last comparison can be reused if it fails.

It enables avoiding backtracking on the string ‘S’. (Mandumula, 2011)

m ← length[p]

a[1] ← 0

k ← 0

**for** q ← 2 to m **do**

**while** k > 0 and p[k + 1] ≠ p[q] **do**

k ← a[k]

**end while**

**if** p[k + 1] = p[q] **then**

k ← k + 1

**end if**

a[q] ← k

**end for**

return ᴨ

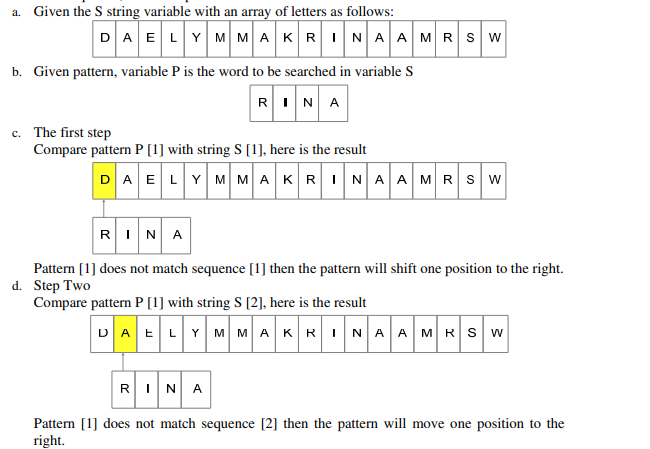
Here a = ᴨ

(Mandumula, 2011)

**III.3 Example How Knuth-Morris-Pratt Algorithm Works**

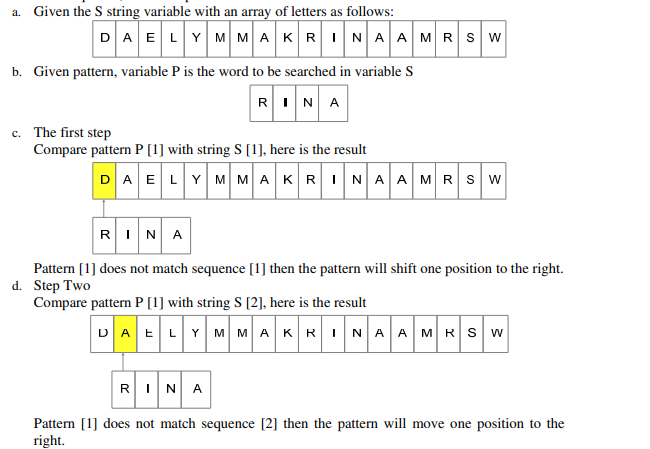
Here is the process of applying the Knuth Morris Pratt algorithm to search for a word in a sentence sequence, as for the process as follows:

1. Given the S string variable with an array of letters as follows:



**Figure 3.1 KMP Algorithm Works (Rahim, 2017)**

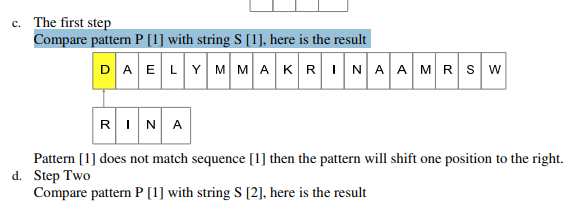
1. Given pattern, variable P is the word to be searched in variable S



**Figure 3.2 KMP Algorithm Works (Rahim, 2017)**

1. The first step

Compare pattern P [1] with string S [1], here is the result

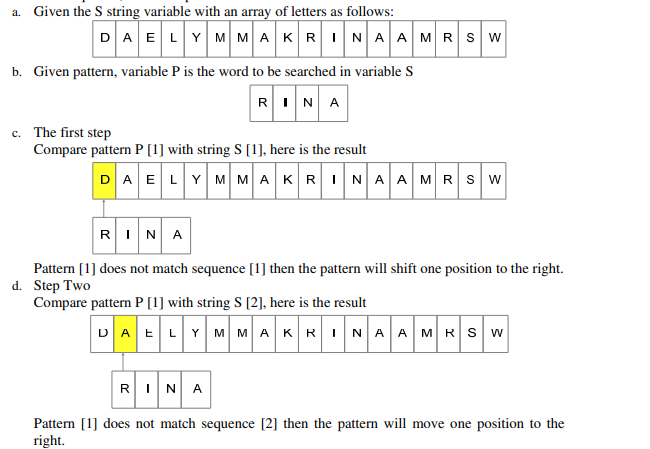


**Figure 3.3 KMP Algorithm Works (Rahim, 2017)**

Pattern [1] does not match sequence [1] then the pattern will shift one position to the right.

1. Step Two

Compare pattern P [1] with string S [2], here is the result

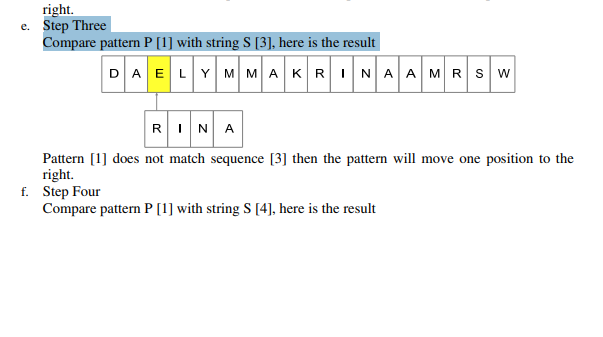


**Figure 3.4 KMP Algorithm Works (Rahim, 2017)**

Pattern [1] does not match sequence [2] then the pattern will move one position to the right.

1. Step Three

Compare pattern P [1] with string S [3], here is the result

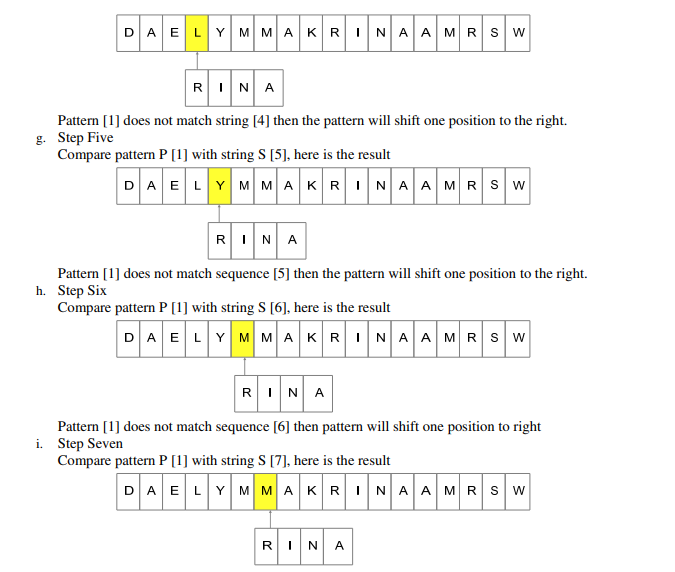


**Figure 3.5 KMP Algorithm Works (Rahim, 2017)**

Pattern [1] does not match sequence [3] then the pattern will move one position to the right.

1. Step Four

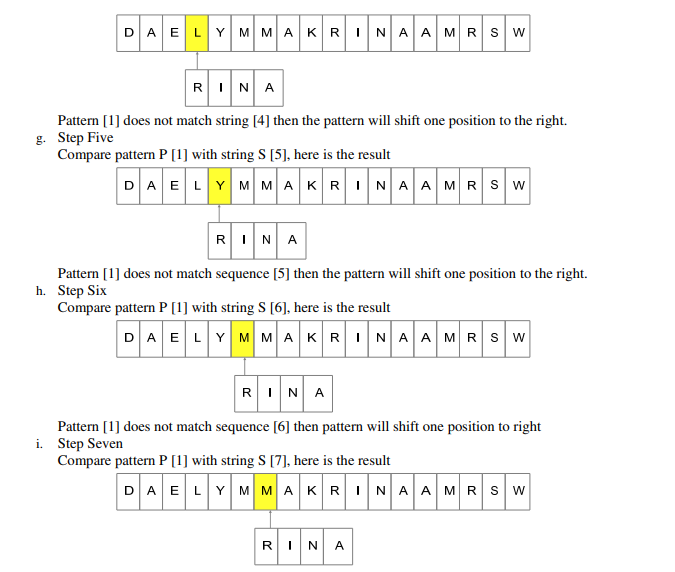
Compare pattern P [1] with string S [4], here is the result



**Figure 3.6 KMP Algorithm Works (Rahim, 2017)**

Pattern [1] does not match string [4] then the pattern will shift one position to the right.

1. Step Five Compare pattern P [1] with string S [5], here is the result

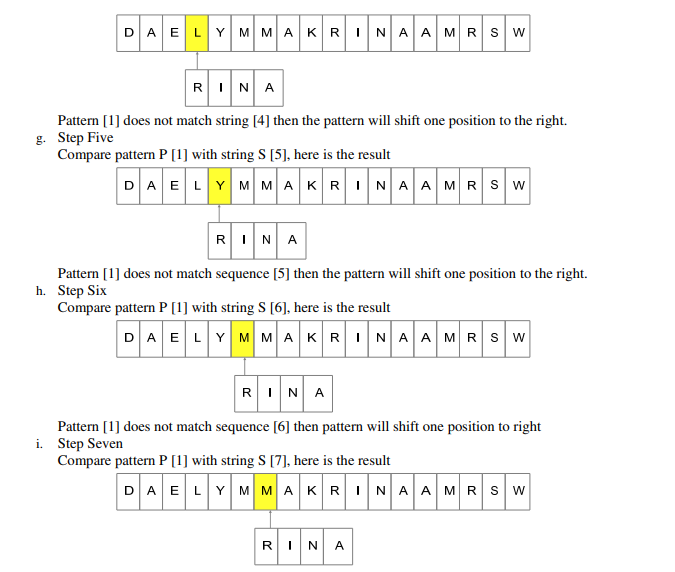


**Figure 3.7 KMP Algorithm Works (Rahim, 2017)**

Pattern [1] does not match sequence [5] then the pattern will shift one position to the right.

1. Step Six

Compare pattern P [1] with string S [6], here is the result

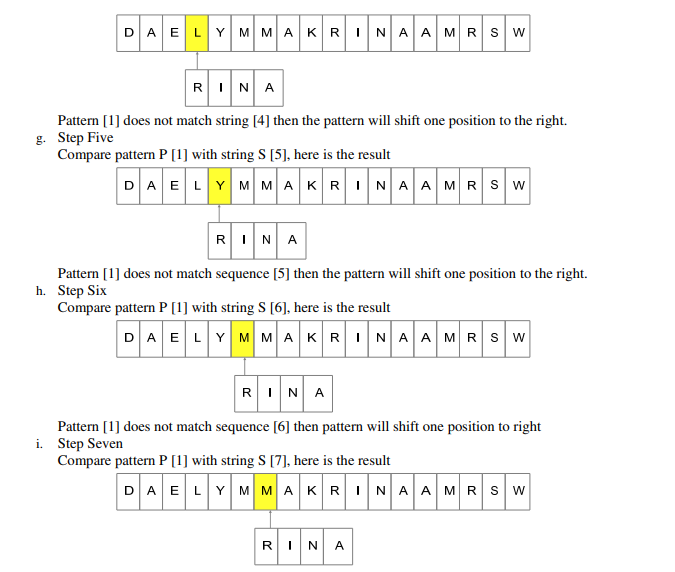


**Figure 3.8 KMP Algorithm Works (Rahim, 2017)**

Pattern [1] does not match sequence [6] then pattern will shift one position to right

1. Step Seven

Compare pattern P [1] with string S [7], here is the result

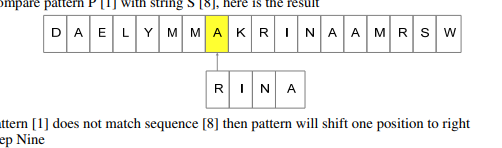


**Figure 3.9 KMP Algorithm Works (Rahim, 2017)**

Pattern [1] does not match string [7] then pattern will move one position to right

1. Step Eight

Compare pattern P [1] with string S [8], here is the result

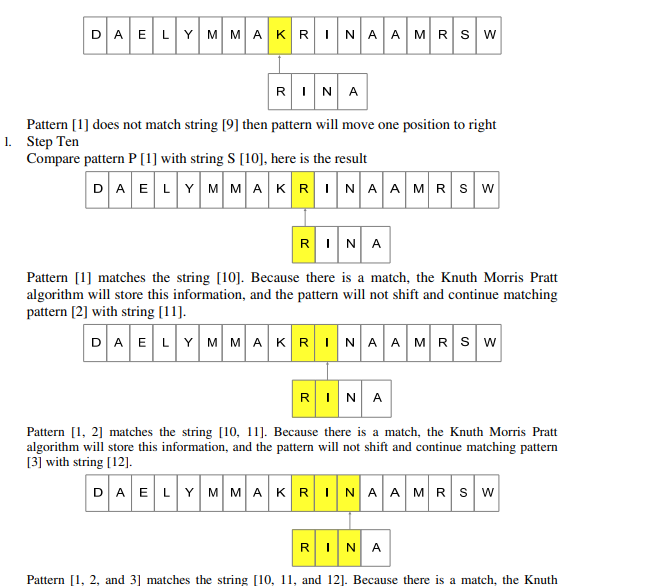


**Figure 3.10 KMP Algorithm Works (Rahim, 2017)**

Pattern [1] does not match sequence [8] then pattern will shift one position to right

1. Step Nine

Compare pattern P [1] with string S [9], here is the result

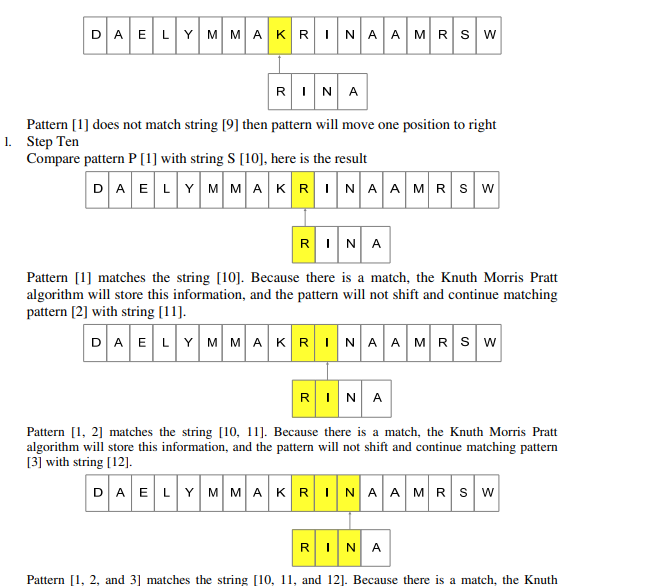


**Figure 3.11 KMP Algorithm Works (Rahim, 2017)**

Pattern [1] does not match string [9] then pattern will move one position to right

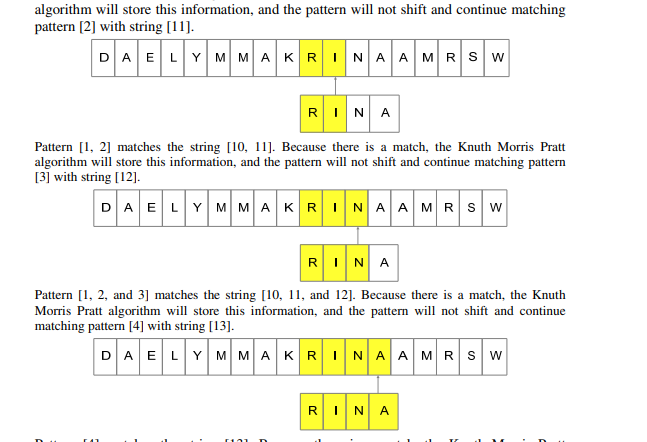
1. Step Ten

Compare pattern P [1] with string S [10], here is the result



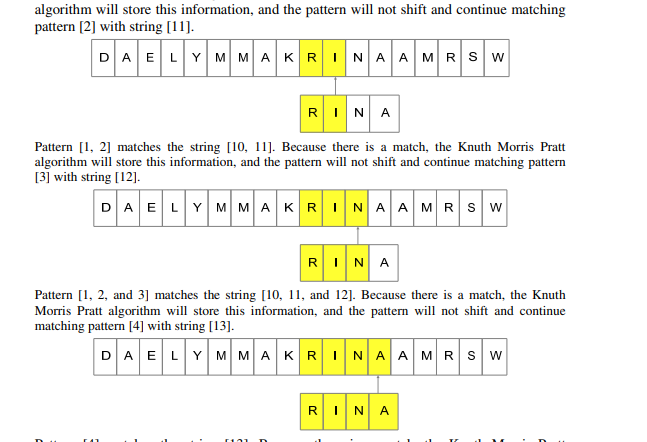
**Figure 3.12 KMP Algorithm Works (Rahim, 2017)**

Pattern [1] matches the string [10]. Because there is a match, the Knuth Morris Pratt algorithm will store this information, and the pattern will not shift and continue matching pattern [2] with string [11].



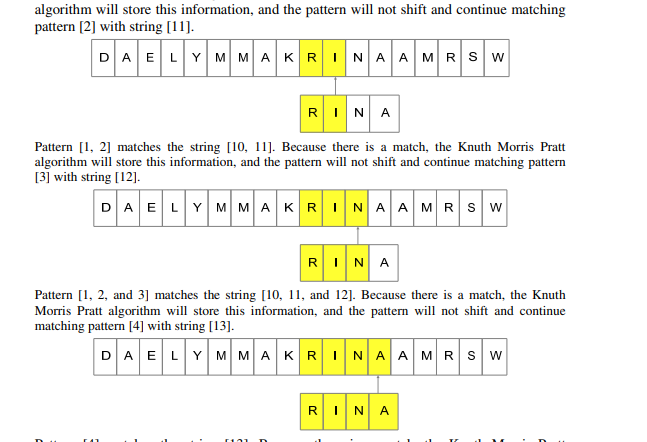
**Figure 3.13 KMP Algorithm Works (Rahim, 2017)**

Pattern [1, 2] matches the string [10, 11]. Because there is a match, the Knuth Morris Pratt algorithm will store this information, and the pattern will not shift and continue matching pattern [3] with string [12].



**Figure 3.14 KMP Algorithm Works (Rahim, 2017)**

Pattern [1, 2, and 3] matches the string [10, 11, and 12]. Because there is a match, the Knuth Morris Pratt algorithm will store this information, and the pattern will not shift and continue matching pattern [4] with string [13].



**Figure 3.15 KMP Algorithm Works (Rahim, 2017)**

Pattern [4] matches the string [13]. Because there is a match, the Knuth Morris Pratt algorithm will store this information, and the pattern will not shift and continue matching pattern [5] with string [14]. However, since the number of patterns is only four letters, the search will be stopped and the result is that the P pattern matches S-string by 100 percent. (Rahim, 2017)

**III.4 Run-Time Analysis**

1. O(m) - It is to compute the prefix function values.
2. O(n) - It is to compare the pattern to the text.
3. Total of O(n + m) run time. (Mandumula, 2011)

**III.5 Advantages and Disadvantages**

1. **Advantages**
   1. The running time of the KMP algorithm is optimal (O(m + n)), which is very fast.
   2. The algorithm never needs to move backwards in the input text T. It makes the algorithm good for processing very large files. (Mandumula, 2011)
2. **Disadvantages**

Doesn’t work so well as the size of the alphabets increases. By which more chances of mismatch occurs. (Mandumula, 2011)

**CHAPTER IV  
CONCLUSION AND SUGGESTION**

**IV.1 Conclusion**

The process visualization of the Knuth-Morris-Pratt (KMP) algorithm allows researchers or scholars to learn how the KMP algorithm works, and in the application development it will be easier to create a function for word search and can be implemented into many search processes.

**IV.2 Suggestion**

**BIBLIOGRAPHY**

Bullinaria, J. (2018). Data Structures and Algorithm. *Luctures*, 125.

C. S. Rao, K. B. (2013). "Parallel String Matching with Multi-Core Processors - A Comparative Study for Gene Sequences". *Global Journal of Computer Science and TechnologyHardware & Computation*, 26-41.

Daptardar, D. S. (2006). "Adapting the Knuth–Morris–Pratt algorithm for pattern matching in Huffman encoded texts," . *Elsevier: Information Processing and Management*, 429-439.

dqdqd. (wdw). fe. *wdw*, dwd.

El-Rayyes, I. M.-Z. (2012). "Parallel Search Using KMP Algorithm". *International Journal of Science and Technology*, 427-431.

Hondro, R. K. (2016). IMPLEMENTASI ALGORITMA KNUTH MORRIS PRATT PADA APLIKASI PENERJEMAHAN BAHASA MANDAILING-INDONESIA. *KMP Algorithm*, 53.

Jadeja, P. P. (2010). Data Structure. *Data Structure and Algorithm*, 131.

Jadhav, N. R. (2015). "A Composite Algorithm for String Matching". *International Journal of Modern Trends in Engineering and Research (IJMTER)*, 68-73.

Mandumula, K. K. (2011). "Knuth-Morris-Pratt Algorithm". *Implementation of Knuth-Morris-Pratt Algorithm*, 24-26.

Prajapati, G. P. (2016). "Applying Bi-Directional Search Strategy in Selected String Matching Algorithms," . *International Journal of Computer Applications*, 40-43.

R. Y. Tsarev, A. S. Chernigovskiy, E. A. Tsareva, V. V. Brezitskay, A. Y. Nikiforov and N. A. Smirnov. (2015). "Combined string searching algorithm based on Knuth-morrispratt and Boyer-Moore algorithms". *XIX International Scientific Conference*, 102-105.

Rahim, R. (2017). "A review: search visualization with Knuth Morris". *IOP Conference Series: Materials Science and Engineering*, 6-9.

Seymour, D. T. (2011). History Of Search Engine. *History Of Search Engine*, 171.

Vijayarani, R. J. (2016). "An Efficient Text Pattern Matching Algorithm for Retrieving Information from Desktop". *Indian Journal of Science and Technology*, 1-11.