“A COMPARATIVE STUDY OF STRING PATTERN MATCHING”

A

Synopsis Report

for

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

Name	**Roll No.**
Shreya Maheshwari	**R164218089**
Nishtha Sehgal	**R164218092**

under the guidance of

Dr. Ravi Tomar
Assistant Professor (S.G.)
Department of Informatics

Department of Systemics

School of Computer Science and Engineering

Bidholi, Via Prem Nagar, Dehradun, UK

September – 2020

ABSTRACT

Pattern Searching is a process of checking and finding a pattern from a string. Although there are huge numbers of searching algorithms available, but here our work intends to show an overview of comparison between three different types of searching algorithms. We have tried to cover some technical aspects of these three searching algorithms. This research provides a detailed study of how all the three algorithms work & give their performance analysis with respect to time complexity.

Data searching is the way of making our study, research process easy and fast. Our project deals with comparative study about few Pattern Matching Algorithms. These algorithms check the possibility of presence of a sequence of characters from a particular string. If the sequence of characters in found in the string, pattern matching is performed. We work on the time consumed and the space occupied on searching a particular pattern from the data and searches the best result from the entire set of characters as fast and in the most convenient manner.

So our study shows the comparison of the different pattern matching algorithms.

Keywords: Data, Matching, Searching, Memory and Time Complexity, Hash codes, C.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Data is a valuable asset that must be protected [1] [2]. Data is digital information transmitted through electrical signal [3] [4] It is divided into two types, writing, and sound. It may be manipulated or duplicated without the owner's knowledge. Plagiarism is an act of evil in the falsification of a work. It is done to increase the popularity of a plagiarist. With the advent of computer science technology, plagiarism can be avoided. Text mining is part of the science that discusses the processing of words.

Data Searching is a process which involves matching the data in a systematic order to make it easier to find, work and analyze. In computer science, string searching algorithms and also called string matching algorithms. String matching or searching algorithms search the pattern or alphabets from the array of elements.

Data Searching is a process which is most required when you have data in bulk and we have to find a particular item amongst hundreds, thousands or more items.

Rabin-Karp Algorithm:

Rabin Karp Algorithm works very similar to the brute force approach or the naive pattern matching algorithm, in which we traverse step by step while matching each character. Very similar to this is Rabin-Karp algorithm in which we once match the hash value of the substring if the hash value is matched then only we start with the matching of the characters individually. It algorithm uses hash functions and the rolling hash technique. A hash function is a function that maps the data of arbitrary size to fixed size values returning a values called hash values, hash codes, digests or hashes.

In this algorithm there is only one comparison taking place per text subsequence. The character searching and matching is done only when the hash values for the sequences match.

Steps for Rabin-Karp algorithm:

1. Find the hash function for the pattern for item.
2. Find the hash for sub patterns or string of the item (pattern length) .

3. Match and search the substring/pattern of text and its pattern from the block of data. Search the item if the hash of the sub pattern is equal to the pattern's hash.

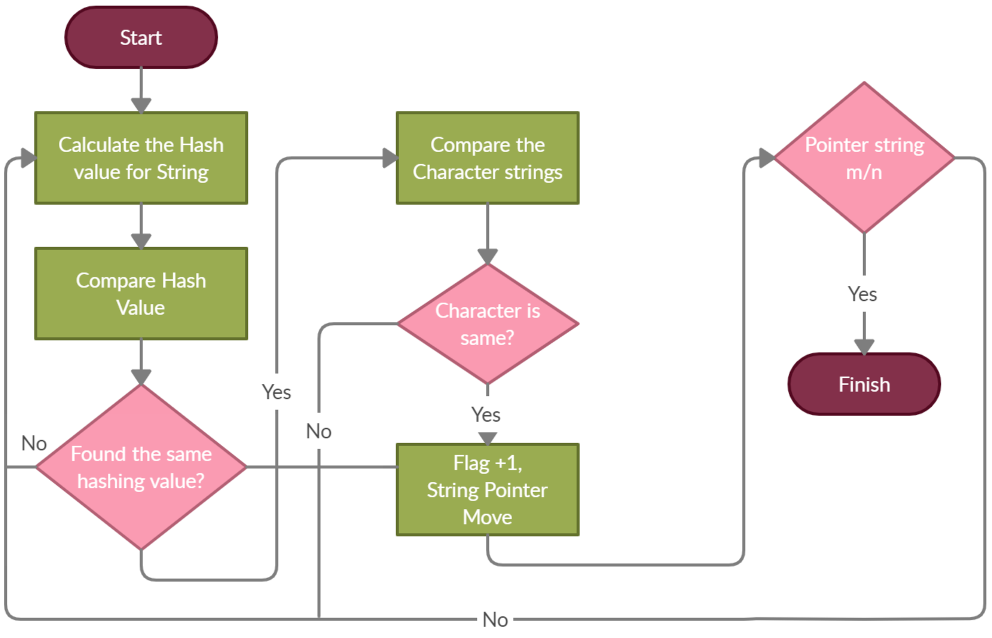

Start
Calculate the Hash value for String
Compare Hash Value
Found the same hashing value?
No
Yes
Compare the Character strings
Character is same?
No
Yes
Flag +1, String Pointer Move
Pointer string m/n
Yes
Finish
No

Figure 1Rabin Karp Algorithm

Knuth Morris Pratt Algorithm:

Knuth Morris Pratt Algorithm is based on the concept of generating a suffix-prefix table also known as the Pi table or the lps table. The pi table is generated using the substring (this is the pre-processing part). The way to generate the substring is the main part in this algorithm. The main concept behind this approach is we save the pattern. As soon as we detect the mismatch while searching for the pattern we already know a part of the pattern in the next window.

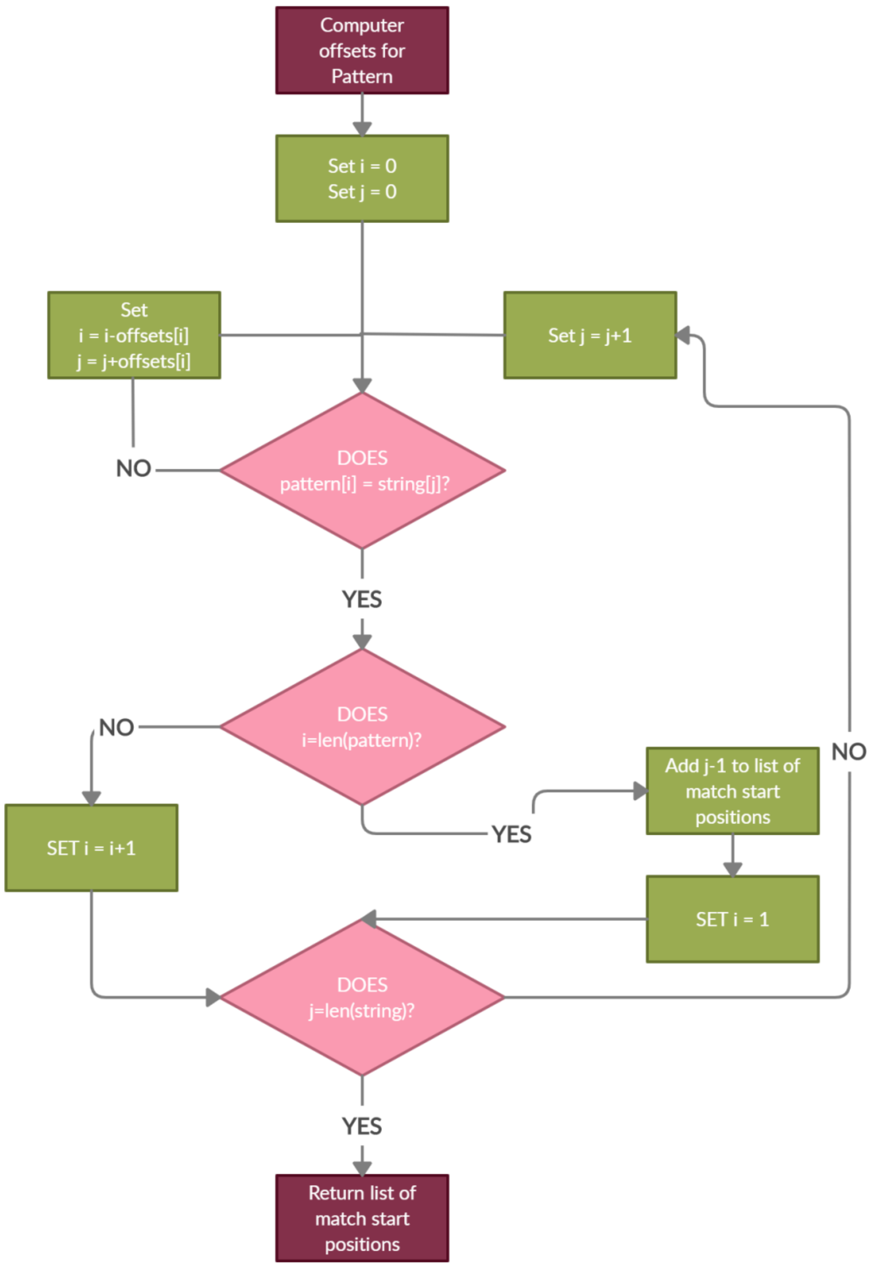


Figure 2 Knuth Morris Pratt Algorithm

Boyer-Moore Algorithm:

Boyer-Moore Algorithm unlike the other two algorithm starts matching from the last character of the pattern. In this algorithm we create two approaches , the bad character heuristic and the good suffix heuristic. In this approach we shift the character which is initially in the mismatch state and bring it to a position where a match is found. Upon comparing the last character of the pattern if the match is not found the entire pattern is shifted by the length of the pattern.

2. LITERATURE REVIEW

- Knuth, Morris and Pratt discovered first linear time string-matching algorithm by following a tight analysis of the naïve algorithm. Knuth-Morris-Pratt algorithm keeps the information that naïve approach wasted gathered during the scan of the text. [5] The algorithm was conceived by James H. Morris and independently discovered by Donald Knuth "a few weeks later" from automata theory. Morris and Vaughan Pratt published a technical report in 1970. [6]
- The Rabin Karp algorithm performs the matching by using hash function which was created by Richard M. Karp and Michael O. Rabin (1987). The Rabin–Karp algorithm is inferior for single pattern searching to Knuth–Morris–Pratt algorithm, Boyer–Moore string search algorithm. The hash function used here basically converts every string value into numeric value. [7]
- Until now Boyer-Moore is considered as the most efficient algorithm for pattern matching. It was developed by Robert S. Boyer and J Strother Moore in 1977. [8] This algorithm preprocesses the pattern that has to be searched in the string. There are two rules followed here. Two rules that are followed here are the good suffix rule and the bad character rule.

3. PROBLEM STATEMENT

There is numerous data which is analyzed and worked upon in every sector in the industry. Every time an item is to be searched an algorithm needs to be implemented to get the best results. A simple example is Google, its uses an algorithm to search for our searched item throughout its database matching the items and then showing us the output. It matches each word with the data and shows us any relevant data to it.
So this process needs to be as fast as possible for efficient searching.

- String matching is needed to search and retrieve items and important data from bulk of information which takes a lot of time if done manually.
- Working with large data and complex categories matching and searching, it becomes difficult and complex.
- Searching data becomes very time consuming and there is a need for better searching methods to make this task fast.
- Amongst all algorithms it’s important to compare and understand which algorithm is most suitable for the work.
- Best algorithm is one which takes the least time and space and gives the best output.
- String matching and searching can also be used for plagiarism check to match and compare the strings of Document 1 and Document 2.

When working with data in researches and studies, searching is a common method which helps to pick, update, delete and use data in an easier manner. Searching makes the study easy to understand and work on instead of going through all of it just to find one pattern or text. Working with patterns and matching of data also requires sorting. There are many ways to search data with different time and space complexities. Bases on these complexities and the following test cases we can decide the best algorithm for the data.

4. OBJECTIVES

This comparative study will help student to get the perfect idea of the algorithms they should use for different data projects. This study gives a clear view of the different functions of the pre existing algorithms with new updates. We also aim to lay a comparison between our generated algorithm and the previously compared algorithms.

Sub objective-

- This will help students to understand the searching algorithms and their best case scenarios.

5. SYSTEM REQUIREMENTS

Hardware Requirements:

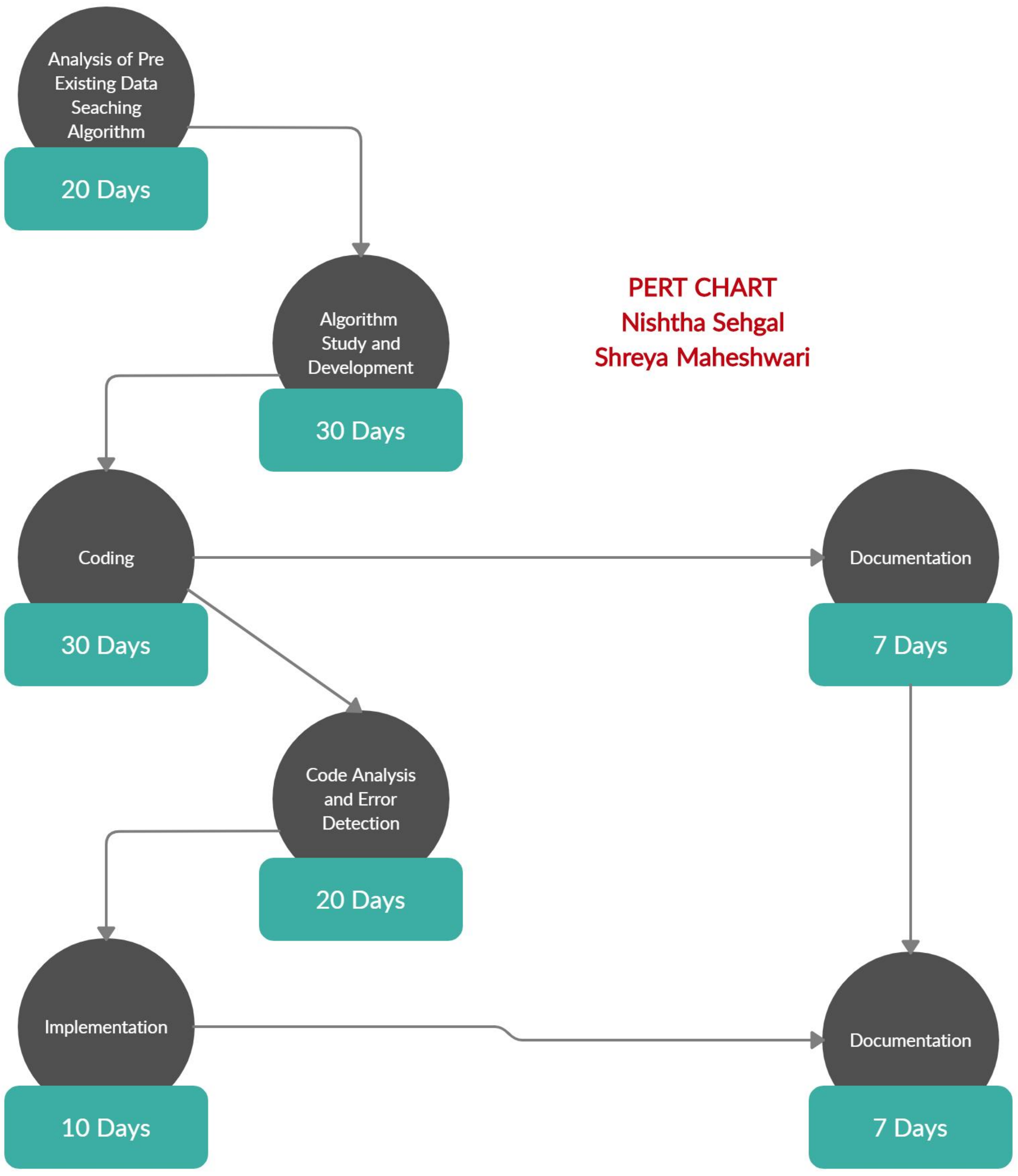
- 64 bits processor architecture supported by windows.
- The minimum RAM requirement for proper functioning is 4 GB.
- Required input as well as output devices.

Software Requirements:

- MinGW
- MS-word (for documentation)
- Command prompt
- Notepad ++

6. DESIGN

7.1 PERT CHART



Bibliography

[1] “. K. M. o. H. C. A. P. U. Siahaan and R. Rahim, "https://d1wqtxts1xzle7.cloudfront.net/56715682/Andysah_Putera_Utama_Siahaan_-_Combination_of_Levenshtein_Distance_and_Rabin-Karp.pdf?1528021851=&response-content-disposition=inline%3B+filename%3DCombination_of_Levenshtein_Distance_and.pdf&Expires=15984405," [Online].

[2] “. B. A. a. E. T. f. S. J. P. C. S. v. 1. n. 1. p. R. Rahim et al., "https://d1wqtxts1xzle7.cloudfront.net/56715682/Andysah_Putera_Utama_Siahaan_-_Combination_of_Levenshtein_Distance_and_Rabin-Karp.pdf?1528021851=&response-content-disposition=inline%3B+filename%3DCombination_of_Levenshtein_Distance_and.pdf&Expires=15984405," [Online].

[3] A. P. U. S. a. N. E. “. A. Z. Tharo, "https://d1wqtxts1xzle7.cloudfront.net/56715682/Andysah_Putera_Utama_Siahaan_-_Combination_of_Levenshtein_Distance_and_Rabin-Karp.pdf?1528021851=&response-content-disposition=inline%3B+filename%3DCombination_of_Levenshtein_Distance_and.pdf&Expires=15984405," [Online].

[4] “.-a.-W. A. T. Z. Ramadhan and A. P. U. Siahaan, "https://d1wqtxts1xzle7.cloudfront.net/56715682/Andysah_Putera_Utama_Siahaan_-_Combination_of_Levenshtein_Distance_and_Rabin-Karp.pdf?1528021851=&response-content-disposition=inline%3B+filename%3DCombination_of_Levenshtein_Distance_and.pdf&Expires=15984405," [Online].

[5] "KMP," [Online]. Available: http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/StringMatch/kuthMP.htm.

[6] "wikipidea," [Online]. Available: https://en.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm.

[7] "wikipidea," [Online]. Available: https://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_algorithm.

[8] "wikipidea," [Online]. Available: https://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_string-search_algorithm#:~:text=In%20computer%20science%2C%20the%20Boyer,J%20Strother%20Moore%20in%201977..

Synopsis verified by:

Dr. Ravi Tomar
(Project Guide)

Dr. Neelu Ahuja
(Program Head)