

# Application of String Matching in Auto Grading System

Akbar Suryowibowo Syam - 13511048  
Computer Science / Informatics Engineering Major  
School of Electrical Engineering & Informatics  
Bandung Institute of Technology, Jl. Ganessa 10 Bandung 40132, Indonesia  
13511048@std.stei.itb.ac.id

**Abstract**— Auto grading is a system that checks the submitted input and grade it based on the answer that's set by the administrator of the application beforehand. Auto grading system uses string matching as the algorithm to score the submission as the text of algorithm and the answers as the pattern to use in the algorithm. The system then grade submission by how many patterns that matches with the text. Using auto grading system, it would take less times and less resources to grade the submission.

String matching algorithms, or sometimes are called String searching algorithm are one of the algorithm with string as the base to work on. String matching algorithms are kind of algorithm that try to determine whether a pattern is exists in a text or not. Pattern is a string that is used in string matching algorithm to traverse the string in the text. A text is string (usually much larger than the pattern) that is used to be traversed by string matching algorithm using the pattern.

**Index Terms**— Auto Grading System, string matching, pattern, text.

## I. INTRODUCTION

Every school, university, and other educational institution must have assignments, projects, test, and every kind of task that are given to the students and to be checked by their teachers or lecturers. Up until now, teachers use manual checking for grading their students assignments. Unfortunately, manual grading takes a lot of times and resources especially if the teacher checks all of it by him/herself. Manual grading is very exhausting and time consuming. To overcome that problem, auto grading system was made.

Auto grading is a system that checks the submitted input and grade it based on the answer that's set by the administrator of the application beforehand. Auto grading system uses string matching as the algorithm to score the submission as the text of algorithm and the answers as the pattern to use in the algorithm. The system then grade submission by how many patterns that matches with the text. Using auto grading system, it would take less times and less resources to grade the submission.

## II. THEORIES

**String matching algorithms**, or sometimes are called

**String searching algorithm** are one of the algorithm with string as the base to work on. String matching algorithms are kind of algorithm that try to determine whether a pattern is exists in a text or not. Pattern is a string that is used in string matching algorithm to traverse the string in the text. A text is string (usually much larger than the pattern) that is used to be traversed by string matching algorithm using the pattern.

Based on the number of patterns that are used to search in the text, there are two types of string matching algorithm, **Algorithm with finite set of patterns** and **Algorithm with infinite set of patterns**. These are few of the known string matching algorithm with finite (one or more) pattern(s) to search [1] :

- Brute Force Algorithm
- Deterministic Finite Automaton Algorithm
- Karp-Rabin Algorithm
- Shift Or Algorithm
- Morris-Pratt Algorithm
- Knuth-Morris-Pratt Algorithm
- Simon Algorithm
- Colussi Algorithm
- Galil-Giancarlo Algorithm
- Apostolico-Crochemore Algorithm
- Not So Naive Algorithm
- Boyer-Moore Algorithm
- Turbo BM Algorithm
- Apostolico-Giancarlo Algorithm
- Reverse Colussi Algorithm
- Horspool Algorithm
- Quick Search Algorithm
- Tuned Boyer-Moore Algorithm
- Zhu-Takaoka Algorithm
- Berry-Ravindran Algorithm
- Smith Algorithm
- Raita Algorithm
- Reverse Factor Algorithm
- Turbo Reverse Factor Algorithm
- Forward Dawg Matching Algorithm
- Backward Nondeterministic Dawg Matching Algorithm
- Backward Oracle Matching Algorithm
- Galil-Seiferas Algorithm
- Two Way Algorithm

- String Matching on Ordered Alphabets Algorithm
- Optimal Mismatch Algorithm
- Maximal Shift Algorithm
- Skip Search Algorithm
- KMP Skip Search Algorithm
- Alpha Skip Search Algorithm

Only three algorithms that are widely known to use nowadays because of their efficiency or simplicity are discussed in this paper. Those three are **Brute Force Algorithm**, **Knuth-Morris-Pratt Algorithm**, and **Boyer-Moore Algorithm**.

The second type of string matching algorithm is the algorithm with infinite set of patterns. These kind of algorithm usually also called **String Matching Algorithm with Regular Expression**. Regular expression (or regex for short) is a special text string for describing a search pattern [2]. Using regular expression, string matching algorithm can be used more flexible because the possibilities of patterns could be infinite for one regular expression.

Below will be explained four string matching algorithm. That is Brute Force Algorithm, Knuth-Morris-Pratt Algorithm, Boyer-Moore Algorithm, and String Matching Algorithm with Regular Expression.

### A. Brute Force Algorithm

The most basic approach of string matching algorithm is through brute force. The principles of brute force string matching algorithm is very simple. First, the algorithm checks for a match between the first character of the pattern with the first character of the text. If they don't match, the algorithm will move forward to the second character of the text and compare the first character of the pattern with the second character of the text. If they don't match again the algorithm will keep moving forward until it gets a match or until it reaches the end of the text. In case the first character of pattern matches the text, the algorithm will move forward comparing the second character of the pattern with the new character in text and so on until it matches. If in the middle happened mismatch, then it will move forward and the search will restart from the first character in the pattern again.

Here is the visual representation of brute force string matching algorithm



Picture 2.1 visual representation of brute force string matching algorithm

The pseudo-code of brute force string matching algorithm is given below.

```

int BruteForce (string text, string
pattern)
{
    int m = pattern.length();
    int n = text.length();
    for (int i=0;i<n-m+1;++i)
    {
        int j = 0;
        bool check = true;
        while
((check)&&(j<pattern.length()))
        {
            if (text[j]!=pattern[i+j])
                check = false;
                j++;
            }
            if (check)
                // pattern found
                return i;
            }
        return -1;
    }
}

```

### B. Knuth-Morris-Pratt Algorithm

Knuth-Morris-Pratt (or for short, KMP) algorithm is one of the string matching algorithm with almost the same approach with brute force algorithm. KMP algorithm match pattern with text same as brute force algorithm but using smarter way to shift the pattern if mismatch happened. Let P be a pattern that is used in the algorithm and P[n] is character in pattern P with index number n. if mismatch happened in pattern P at P[j], the pattern will be shifted right (position of mismatch – border function(j)). Border function is a function that stores the number of largest prefix of P[1..j-1] that is the same with the suffix of P[1..j-1].

For example let pattern P = “abacabad” and mismatch happened in j=6 so that the substring that will be used in border function is “abacab”. Now the algorithm will try to find the largest suffix that is the same with the prefix. The answer is “ab”. So, the border function will give an output ‘2’. Let’s take a look what would table of border function be.

k	1	2	3	4	5	6	7	8
B(k)	0	0	1	0	1	2	3	0

Table 2.1 Border Function of pattern “abacabad”

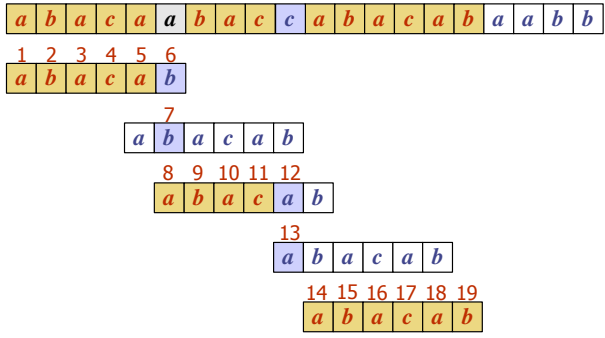
The pseudo-code of KMP algorithm is given below.

```

int KMP (string text, string pattern)
{
    int m = pattern.length();
    int n = text.length();
    // creating precompute table
    int table[m];
    table[0] = 0;
    int cnd = 0;
    int i = 1;
    while (i < m)
    {
        if (pattern[i]==pattern[cnd])
        {
            table[i] = cnd + 1;
            cnd = cnd + 1;
            i++;
        }
        else if (cnd > 0)
            cnd = table [cnd -1];
        else if (cnd ==0)
        {
            table[i] = 0;
            ++i;
        }
    }
    i = 0;
    int j = 0;
    bool check = true;
    while ((check)&&(i<n))
    {
        if (pattern[j]==text[i])
        {
            i++;
            j++;
        }
        if (j==m)
            return i;
        else if
        ((i<n)&&(pattern[j]!=text[i]))
        {
            if (j!=0)
                j = table[j-1];
            else
                i++;
        }
    }
}

```

Here is a visual representation of an example of KMP algorithm.



Picture 2.2 Visual Representation of KMP Algorithm

### C. Boyer-Moore Algorithm

Boyer-Moore (or for short, BM) algorithm searches differently from KMP and brute force string matching algorithm. BM algorithm checks the pattern from right to left. There are two unique characteristics of BM algorithm. The first one is BM algorithm search patter in text by moving backwards from end of pattern to start of pattern. The second one is if mismatch happened, BM algorithm move the pattern forward with three specific condition.

If KMP algorithm has Border function to help the algorithm, BM algorithm has Last Occurrence function. Last Occurrence function maps all letters in alphabet of the pattern to integer. The integer output is index that shows the last occurrence of the specified character in the pattern.

For example of pattern “abacabad”. Character ‘a’ has last occurrence function value of 7 because the last character of ‘a’ occupies the 7<sup>th</sup> index. For any character that is not exist in the pattern the index will be -1. The complete table of last occurrence function for “abacabad” is shown below :

x	a	b	c	d
L(x)	7	6	4	8

Table 2.2 Last Occurrence function of pattern “abacabad”

There are three specific condition if mismatch happened in position j in text T and position k in pattern P:

- If last occurrence function of T[j] shows an index m so that m < k, then the pattern will move forward so that P[k] would align with T[j].
- If last occurrence function of T[j] shows an index m so that m > k, then the pattern will move forward one step to the right.
- If last occurrence function of T[j] shows and index m so that m = -1 (meaning there is no character T[j] in pattern P), then the pattern will move forward as much as the length of pattern P.

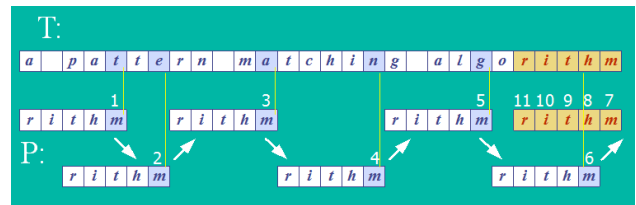
The Pseudo-code of BM algorithm is given below.

```

int BoyerMoore (string text, string
pattern)
{
    int m = pattern.length();
    int n = text.length();
    int table1[256];
    // creating the first char table
    for (int i=0;i<256;++i)
        table[i] = m;
    for (int i=0;i<m-1;++i)
        table[pattern[i]] = m - 1 - i;
    // creating the second offset table
    int table2[m];
    int last_prefix = m;
    for (int i=m-1;i>=0;--i)
    {
        bool check = true;
        int j = i+1, k = 0;
        while ((check)&&(j<m))
        {
            if (pattern[j]!=pattern[k])
                check = false;
            ++j;
            ++k;
        }
        if (check)
            last_prefix = i+1;
        table2[m-1-i] = last_prefix-j+m-1;
    }
    for (int i=0;i<m-1;++i)
    {
        int slen = 0;
        for (int j=i, k=m-1;j>=0 &&
pattern[j]==pattern[k]; --j, --k)
            slen +=1;
        table2[slen] = m - 1 - j
+slen;
    }
    // searching
    bool check = true;
    int i = m - 1, j = 0;
    while ((check)&&(i<text.length()))
    {
        j = m - 1;
        while ((check) && (pattern[j]
== pattern[i]))
        {
            if (j==0)
                // found
                check = false;
            --i;
            --j;
        }
        if (check)
            j+=max(table2[m-1-j],
table[text[i]]);
    }
    if (!check)
        return i;
}

```

Here is visual representation of an example of BM algorithm.



#### D. String Matching with Regular Expression

Regular expression (or regex for short) is a special text string for describing a search pattern. Any kind of string that wanted to be the pattern can be changed to a single regular expression. Below will be explained some important syntax of regular expression.

C	Description
.	Matches any single character.
[ ]	A bracket expression. Matches a single character that is contained within the brackets.
[^ ]	Matches a single character that is not contained within the brackets
^	Matches the starting position within the string
\$	Matches the ending position of the string or the position just before a string-ending newline.
( )	Defines a marked sub-expression. The string matched within the parentheses can be recalled later
\n	Matches what the <i>n</i> th marked sub-expression matched, where <i>n</i> is a digit from 1 to 9.
*	Matches the preceding element zero or more times.
{m,n}	Matches the preceding element at least <i>m</i> and not more than <i>n</i> times.

### III. IMPLEMENTATION IN AUTO GRADING SYSTEM

In an Educational Institution, auto grading system can be used to grade student's assignment and test. There are few advantages of using auto grading system for grading assignment and test. The first advantage is it would need less resources and less time to process than manual grading. The second advantage is it would be more objective to grade the submission because the pattern would always be the same and program have no subjectivity on them. The third advantage is that the administrator (in this case, teacher) can set beforehand how strict they want the answer should be. They can set how precise the submission to the real answer the teacher provided.

If teacher want the answer to be more flexible, they could have used string matching algorithm with regular expression because it means that there will be more option of what the answer could be. But string matching algorithm with regular expression doesn't have to used only if teacher want the answer to be flexible. It could also be used to make a strict option of answer, even only one option of answer.

Other ways to make the pattern for string matching algorithm is using finite patterns. Teachers can use KMP algorithm or BM algorithm as the string matching algorithm. There a condition where an algorithm is better than the other one and that condition is based on how many characters are in the alphabet that is used. If there are only few characters in the alphabet, then KMP algorithm is the better choice to use rather than BM. Otherwise, if there are a lot of characters in the alphabet, then BM algorithm would be better choice to use. BM algorithm is better to use when there are a lot of character in alphabet because the probability of the pattern will move forward as much as the length of the pattern is higher when the alphabet is wider.

Looking at the reason above, teacher can choose which algorithm is better to use depends on what kind of submission will be submitted later because the number of characters in the alphabet that are used will be the determinant to choose which algorithm to use.

For example if a teacher of programming course want his/her student to make a simple C program. The teacher expects the answer to be like the picture below.

```
#include <iostream>
using namespace std;

int main() {
    string name;
    cin >> name;
    cout << "Hello " + name;

    return 0;
}
```

Picture 3.1 Example to use in auto grading system

Because the number of characters in the alphabet is a lot, then KMP is not very efficient to use in this case. Then, the viable options are string matching with regular expression or BM algorithm. For example one of the list of pattern if the teacher want to use BM algorithm would be :

- string name;
- std::string name;
- cin >> name;
- #include <iostream>
- cout << "Hello " + name;
- return 0;
- int main()

based on the pattern above, the auto grading system then can score students submit. For example if a student submit like in picture below :

```
#include <iostream>
using namespace std;

int main() {
    cin >> name;
    cout << "Hello " + name;

    return 0;
}
```

As we can above, on the submission the student submitted, the student forgot to add declaration "string name;". based on the list of pattern the teacher set before, there are seven pattern the teacher listed. Therefore, the student that only correct six pattern get 6/7 score or 85.7 points.

#### IV. CONCLUSION

Auto grading system is better than manual grading system in educational institution because it would cost less resources and less time consuming. Auto grading should be used only if the pattern can be specified. If the pattern cannot be specified then auto grading cannot be used.

In selecting which string matching algorithm is better to choose, there are 3 options, string matching with regular expression, KMP algorithm, and BM algorithm. Regular expression should be used if you want more flexible patterns. KMP should be used if there are only few of characters in the alphabet and BM should be used if there are a lot of characters in the alphabet.

#### REFERENCES

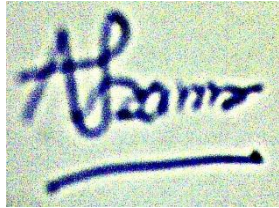
- Exact String Matching Algorithms, <http://www-igm.univ-mlv.fr/~lecroq/string> , accessed on 19.33, 18<sup>th</sup> December 2013
- Regular-expression info. <http://www.regular-expressions.info/> , accessed on 8.07, 19<sup>th</sup> December 2013
- Brute Force Algorithm visual representation, <http://www.compbio.biosci.uq.edu.au> , accessed on 10.40, 19<sup>th</sup> December 2013
- M.Rinaldi , PatternMatching.ppt, accessed on 9.30, 20<sup>th</sup> December 2013
- Pattern Matching , <http://c2.com/cgi/wiki?PatternMatching> , accessed on 9.44 , 20<sup>th</sup> December 2013.

## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Desember 2013

ttd

A square image showing a handwritten signature in blue ink. The signature appears to be 'Akbar' followed by a horizontal line.

Akbar Suryowibowo Syam - 13511048