String Matching Analysis in Antivirus Software

Bethea Zia Davida 13515084¹
Informatics Engineering Study Program
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
¹betheazdavida@gmail.com

Abstract—Antivirus software is a class of program that will prevent, detect and remediate malware infections on individual computing devices and IT systems. It uses string matching to do its job. String matching consists of finding one or more generally, all of the occurrences of a pattern in a text.

Keywords—Antivirus Software, String Matching, Knuth-Morris-Pratt Algorithm, Boyer-Moore Algorithm,

I. INTRODUCTION

Recently, all medias reported about cyber attack because of virus that named as ransomware virus. Ransomware prevents users from accessing their devices and data till a certain amount is paid to its creator as ransom. Ransomware usually locks computers, encrypts the data on it and prevents other software and apps from running. The ransomware is WannaCry.

II. THEORY

A. Antivirus Software

Wannacry (or WannaCryp) is a ransomware computer worm that targets the Microsoft Windows operating system. The virus was used to launch the WannaCry ransomware attack on Friday, 12 May 2017. A lot of people feel unsafe due to the existence and government announced about how to protect our computers from the virus.
Antivirus Software is a software that is made to prevent, detect, quarantine, and remove viruses and other malicious things like worms, trojans, and more. Antivirus software runs in the background on computer, checking every file opened. This is generally known as on-access scanning, background scanning, resident scanning, real-time protection, or something else.

Antivirus software was originally developed to detect and remove computer viruses. However, with the proliferation of other kinds of malware, antivirus software started to provide protection from other computer threats. In particular, modern antivirus software can protect from: malicious browser helper objects (BHOs), browser hijackers, ransomware, keyloggers, backdoors, rootkits, trojan horses, worms, malicious LSPs, dialers, fraudtools, adware and spyware.

B. String Matching

String matching problem is defined as follows: given two strings which are a text and a pattern, and determining whether the pattern appears in the text or not.

This problem also known as “the needle in a haystack problem”. Examples of application in string matching are web search engine, bioinformatics, and searching in text editor.

There are some ways to overcome this problem, those are:

1. “Naive” method or Brute Force Algorithm
   Brute force is a straightforward way to find the solution. For every position in the text, consider it is a starting position of the pattern and find out if you get a match. Brute force algorithm is easy to understand and implement but it can be too slow in some cases. If the length of the text is n and the length of the pattern m, the worst case that it may take is $O(n \cdot m)$.

   *Pattern*: NOT
   *Text*: NOBODY NOTICED HIM

   ![Pattern and Text](http://4.bp.blogspot.com/_39NGKVWYg3o/TQ4Q1Tmi7X I/AAAAAAAABZo/-Dz96d48 6o/s1600/Screen%252Bshot%252B2010-12-19%252BBat%252B8.08.56%252BAM.png)

   ```
   function brute_force(text[], pattern[]){
     for(i = 0; i < n; i++) {
       for(j = 0; j < m && i + j < n; j++)
         if(text[i + j] != pattern[j])
           break;
       if(j == m) // match found
     }
   }
   ```

2. Knuth-Morris-Pratt Algorithm
   The Knuth-Morris-Pratt (KMP) algorithm looks for the pattern in the text in a left-to-right order. The algorithm was conceived in 1970 by Donald Knuth and Vaughan Pratt, and independently by James H. Morris. The three published it jointly in 1977. The automaton used in KMP is just an array of “pointers” and separate “external” pointer to some index of that array. It is like brute force algorithm, but it shifts the pattern more intelligently than the brute force algorithm.
In order to build the KMP automation (or the so called KMP “failure function”), it is needed to initialize an integer array \( F[] \). The indexes represent the numbers under which the consecutive prefixes of the pattern are listed in the “list of prefixed”.

Notice that after initialization, \( F[i] \) contains information not only about the largest next partial match for the string under index \( i \), but also about every partial match of it.

In terms of pseudocode, the initialization of the array \( F[] \) (the “failure function”) may look like this:

```plaintext
function build_failure_function(pattern[])
{
    // let m be the length of the pattern
    F[0] = F[1] = 0; // always true
    for(i = 2; i <= m; i++) {
        // j is the index of the largest next partial match
        // (the largest suffix/prefix) of the string under
        // index i - 1
        j = F[i - 1];
        for( ; j > 0) {
            // check to see if the last character of string i - 1
            // "expands" the current "candidate"
            // best partial match - the prefix under index j
            if(pattern[j] == pattern[i - 1]) {
                F[i] = j + 1; break;
            }
            // if we cannot "expand" even the empty string
            if(j == 0) { F[i] = 0; break; }
            // else go to the next best "candidate" partial match
            j = F[j];
        }
    }
}
```

The automaton consists of the initialized array \( F[] \) (“internal rules”) and a pointer to the index of the prefix of the pattern that is the best (largest) partial match that ends at the current position in the text (“current state”).

```plaintext
function Knuth_Morris_Pratt(text[], pattern[])
{
    // let n be the size of the text, m the
    // size of the pattern, and F[] - the
    // "failure function"
    build_failure_function(pattern[]);
    i = 0; // the initial state of the automaton is
    \( \text{the empty string} \)
    j = 0; // the first character of the text
    for( ; ; ) {
        if(j == n) break; // we reached the end of the text
        // if the current character of the text "expands" the
        // current match
        if(text[j] == pattern[i]) {
            i++; // change the state of the automaton
            j++; // get the next character from the text
            if(i == m) // match found
                break;
        }
        // if the current state is not zero (we have not
```
3. Boyer-Moore Algorithm

The Boyer–Moore string searching algorithm is an efficient string searching algorithm that is the standard benchmark for practical string search literature. It was developed by Robert S. Boyer and J Strother Moore in 1977. The Boyer-Moore pattern matching algorithm is based on two techniques.

1. The looking-glass technique
   
   find P in T by moving backwards through P, starting at its end

2. The character-jump technique
   
   when a mismatch occurs at T[i] == x, the character in pattern P[j] is not the same as T[i]

Before we process the string, we need to preprocess the string first:

```c

// The preprocessing function for Boyer Moore's bad character heuristic
void badCharHeuristic( char *str, int size, int badchar[NO_OF_CHARS])
{
    int i;
    // Initialize all occurrences as -1
    for (i = 0; i < NO_OF_CHARS; i++)
        badchar[i] = -1;
    // Fill the actual value of last occurrence of a character
    for (i = 0; i < size; i++)
        badchar[(int) str[i]] = i;
}
```

And then the algorithm:

```c

void search( char *txt, char *pat) {
    int m = strlen(pat);
    int n = strlen(txt);
    int badchar[NO_OF_CHARS];
    /* Fill the bad character array by calling the preprocessing function badCharHeuristic() for given pattern */
    badCharHeuristic(pat, m, badchar);
    int s = 0; // s is shift of the pattern with respect to text
    while(s <= (n - m))
    {
        int j = m-1;
        /* Keep reducing index j of pattern while characters of pattern and text are matching at this shift s */
        while(j >= 0 && pat[j] == txt[s+j])
            j--;
        /* If the pattern is present at current shift, then index j
```
will become -1 after the above loop */

if (j < 0)
{
    printf("\n pattern occurs at shift = %d", s);

    /* Shift the pattern so that the next character in text
     aligns with the last occurrence of it in pattern.
     The condition s+m < n is necessary for the case when
     pattern occurs at the end of text */
    s += (s+m < n)? m-badchar[txt[s+m]] : 1;
}
else

    /* Shift the pattern so that the bad character in text
     aligns with the last occurrence of it in pattern. The
     max function is used to make sure that we get a positive
     shift. We may get a negative shift if the last occurrence
     of bad character in pattern is on the right side of the
     current character. */
    s += max(1, j - badchar[txt[s+j]]);
}

III. ANTIVIRUS SOFTWARE STRING MATCHING ANALYSIS

A. How Antivirus Software Work

Antivirus software scans files in your computer for certain pattern that may indicate a malicious things. It looks for pattern based on the virus signatures or definition of known malware that saved in the antivirus software. Antivirus makers have updated malware data every day, so it is necessary to update and have the lastest antivirus software installed.

When it is installed, it should scan computers periodically, so the computers will be safe from harming things. In every antivirus softwares, it can scan computers in two ways, those are:

1. Automatic scans

    Most antivirus software can be configured to do this kind of scan.

2. Manual scans

    If the antivirus software can’t do the automatic scans, so it have to be asked to scan manually.

Antivirus software will alert computer’s users if it has found malware and ask for the opinion whether want to clean the file or the other options that provide by the antivirus software. In some cases, the antivirus software will attempt to remove the malware without asking first.

![Select method of processing malware](https://46qasb3uw5yn639ko4bz2ptr8u-wpengine.netdna-ssl.com/files/2016/10/malanov3.png)

There are many vendors that produce antivirus software. The example of antivirus softwares are smadav, avg, etc. By installing antivirus software, it will increase the level of protection in the computer.

Antivirus works by matching the files that were scanned with the virus signatures that it have. If there is matching part of the virus signature and the file, the file will be recognize as a virus. So, it is important to update the virus signature in the antivirus software.

https://46qasb3uw5yn639ko4bz2ptr8u-wpengine.netdna-ssl.com/files/2016/10/malanov3.png
Virus Signature

Virus signature is a unique string of bits, or the binary pattern, of a virus. The virus signature is like a fingerprint in that it can be used to detect and identify specific viruses. Anti-virus software uses the virus signature to scan for the presence of malicious code. A virus signature is a continuous sequence of bytes that is common for a certain malware sample. That means it’s contained within the malware or the infected file and not in unaffected files.

Example of virus signature:

1. 1024–PrScr
   #1=8cc0488ec026a103002d800026a303 00
2. 1024–PrScr
   #2=a172041f3df0f07505a10301cd0526 a1
3. 1024–PrScr
   #3=00012ea30300b4400e1fba0004b900 04e8e8007230
4. 1024–PrScr
   #4=babf00b82125cd2133c08ec08f0f0 26
5. 1210–Prudent=2f040175d00e0e1f07bed3042 bc92e8a0446410ac0
6. Etc

Antivirus will use the virus signature as a pattern and search for the pattern in files that it scanned.

B. String Matching in Antivirus Software

As mentioned in part A, the antivirus software find candidate of viruses with string matching.

1. Brute Force Algorithm

```
1 0 2 4 - P r S c
0 2 4 - P r S c
1 0 2 4 - P r S c
```

2. KMP Algorithm

```
1 0 2 4 - P r S c
0 2 4 - P r S c
1 0 2 4 - P r S c
```

3. BM Algorithm

```
e 1 0 2 4 - P r S c
1 0 2 4 - P r S c
1 0 2 4 - P r S c
```

Picture 9. Antivirus Software

[link to image]

Picture 10. Antivirus Software

[link to image]
IV. CONCLUSION

String matching is one of algorithm behind antivirus software. It used to search for virus in bunch of files. The antivirus software will scan every matching files with the pattern in virus signature and quarantine malicious files.

ACKNOWLEDGMENT

First and above all, the writer praises God, the almighty for providing me this opportunity and granting me the capability to proceed successfully this paper. In performing this paper, the writer had to take the help and guideline of some respected persons and resources. The writer would like to show her gratitude to Mr. Rinaldi Munir, Mrs. Nur Ulfa M., and Mrs. Masayu L., lecturers of IF2211 Strategi Algoritma, Institut Teknologi Bandung, for giving the writer a good guideline for the paper. The writer would also like to expand deepest gratitude to all those who have directly and indirectly guided the writer in writing this paper.

REFERENCES

tringMatch/stringMatchIntro.htm accessed May 17, 2017
technology/wannacry-everything-you-need-to-know-about-the-biggest-
ransomware-attack/ accessed May 17, 2017
[5] https://www.howtogeek.com/125650/htg-explains-how-antivirus-
software-works accessed May 17, 2017
algorithm-bad-character-heuristic/ accessed May 18, 2017

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, 17 Mei 2017

Beth Zia Davida 13515084

Makalah IF2211 Strategi Algoritma, Semester II Tahun 2016/2017