Penerapan Algoritma *String Matching* Dalam Pengecekan Plagiarisme

Suryanto - 13520059
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520059@std.stei.itb.ac.id

Abstract*—Plagiarisme adalah suatu tindakan penjiplakan yang melanggar hak cipta. Salah satu upaya dalam mengurangi plagiarisme adalah *plagiarism checker*, yaitu program yang melakukan pemeriksaan secara otomatis terhadap tulisan dan membandingkannya dengan tulisan-tulisan lain. Salah satu algoritma yang dapat digunakan pada program ini adalah *string matching* seperti *Brute Force*, algoritma Knuth-Morris-Pratt (KMP), dan algoritma Boyer-Moore. Makalah ini akan membahas tentang implementasi ketiga algoritma tersebut pada program *plagiarism checker.

Keywords—Plagiarisme, plagiarism checker, algoritma string matching

I. Pendahuluan

Teknologi yang kian berkembang memberikan kemudahan dalam mengakses informasi dari berbagai sumber. Kemudahan ini seringkali disalahgunakan oleh beberapa pihak dengan menggunakan suatu informasi tanpa menyantumkan sumbernya atau bahkan mengeklaim bahwa informasi tersebut adalah miliknya. Tindakan ini dikenal sebagai plagirisme.

Menurut KBBI, plagirisme adalah suatu tindakan penjiplakan yang melanggar hak cipta. KBBI juga menjelaskan pengertian plagiat sebagai pengambilan karangan (pendapat dan sebagainya) orang lain dan menjadikannya seolah-olah karangan sendiri. Melalui definisi tersebut, dapat disimpulkan plagiarisme adalah tindakan salah yang tentunya dapat merugikan para pencetus karangan, ide, gagasan, pendapat aslinya.

Salah satu bidang yang memiliki kecenderungan dalam tindakan plagiarisme adalah bidang pendidikan. Hal ini dapat dilihat dari banyaknya siswa yang menuliskan pekerjaan mereka (makalah, laporan, poster, dll) tanpa mencantumkan sumbernya. Tak jarang tindakan ini dilakukan juga oleh para akademisi setingkat guru dan dosen yang menggunakan ide atau gagasan orang lain tanpa memberikan *credit* kepada pencentus ide tersebut.

Salah satu cara untuk mengatasi masalah ini adalah dengan melakukan pemeriksaan terhadap suatu karya. Namun, pemeriksaan secara manual akan membutuhkan waktu yang sangat lama karena luasnya sumber informasi yang tersedia. Oleh karena itu telah dibuat suatu sistem yang dapat melakukan pengecekan plagirisme secara otomatis yang dikenal sebagai *plagiarism checker*. Oleh karena itu, penggunaan algoritma *string matching* pada pengecekan plagiarisme secara sederhana akan menjadi topik dari makalah ini.

II. Landasan Teori

A. Pencocokan String

Pencocokan string (*string/pattern matching)* merupakan algoritma yang digunakan untuk mencari apakah suatu pola *string* terdapat pada suatu teks. Secara umum, algoritma ini akan mengembalikan indeks dari teks yang merupakan lokasi pertama pola ditemukan. Definisi umum yang digunakan pada *string matching* adalah sebagai berikut :

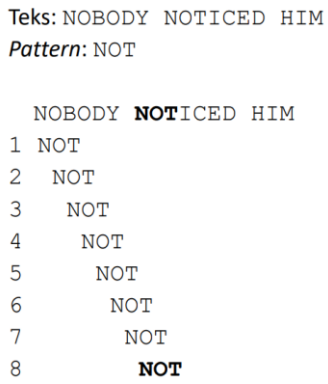
1. *T* (*text)* yang merupakan suatu *string* sepanjang *n* karakter.
2. *P (Pattern)* adalah suatu *string* dengan panjang *m* karakter ($m<n$) yang akan diperiksa kemunculannya pada teks.

Beberapa algoritma *string matching* yang akan dibahas pada makalah ini adalah sebagai berikut.

1. Algoritma *Brute Force*
2. Algoritma Knuth-Morris-Pratt (KMP)
3. Algoritma Boyer-Moore

B. Algoritma Brute Force

Algoritma ini adalah pendekatan secara *straighforward* untuk memecahkan suatu persoalan, termasuk pada *string matching*. Pada Pencocokan *string,* pemeriksaan *pattern* akan dilakukan satu persatu karakter secara terurut terhadap karakter pada teks. Apabila karakter cocok, maka pemeriksaan akan bergeser sebesar satu karakter terhadap masing-masing indeks teks dan *pattern.* Namun, apabila karakter tidak sama (*mismatch)*, maka indeks teks yang diperiksa akan bergeser sebesar satu karakter, sedangkan pemeriksaan *pattern* diulang dari karakter pertama.



Gambar 1. Pencocokan String dengan Algoritma *Brute Force*

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Berikut adalah *pseudocode* dari algoritma *brute force.*

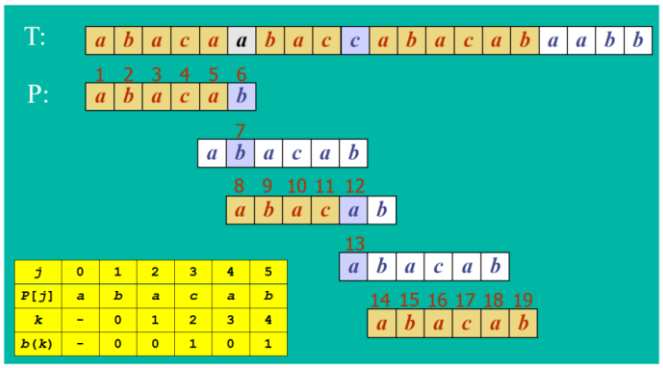
```
Function BruteForceMatch (T : string, P : string) → boolean
KAMUS
   i,j : integer
   found : Boolean
ALGORITMA
   found ← false
   i ← 0
   while (i < T.length and not found) do
        j ← 0
        while (j < P.length and P[j] = T[i+j]) do
               j ← j + 1
        endwhile

        if ( j == P.length) then
               found ← true
        endif
        i ← i + 1
   endwhile
→found
```

C. *Algoritma Knuth-Morris-Pratt*

Algoritma Knuth-Morris-Pratt (KMP) adalah suatu algoritma pencocokan *string* yang memiliki pendekatan mirip dengan algoritma *brute force*. Pencocokan dilakukan dari kiri ke kanan dengan pergeseran yang ditentukan dengan *border function*, sehingga algortima ini dapat bekerja secara lebih efektif dibandingkan dengan *brute force*.

Pada pencocokkan pattern P pada text T, apabila terjadi mismatch pada posisi karakter T[i] dan P[j], maka algoritma ini akan melakukan pergeseran seukuran panjang dari prefix terpanjang dari P[0..j-1] yang juga merupakan suffix dari P[1..j-1]. Oleh karena itu, terdapat digunakan sebuah fungsi yang dikenal sebagai border/failure function yang digunakan untuk menentukan ukuran pergeseran untuk tiap kejadian mismatch pada indeks P.

Misal akan dilakukan pencocokkan string pada text T = abacaabaccabacabaabb dengan pattern P = abacab. Pada mismatch pertama (T[5] dan P[5]), maka pencocokkan selanjutnya akan dilakukan pada T[5] dan P[b(5-1)]. Dengan nilai b(4) = 1, maka P[0] tidak perlu diperiksa lagi karena pencocokkan akan dimulai pada P[1]. Pencocokkan akan terus dilakukan hingga tidak terjadi mismatch pada P[j] = T[i] dengan j adalah indeks terakhir dari P. Kasus lainnya adalah saat T telah mencapai indeks terakhir tanpa menemukan match dengan P, yang berarti pattern P tidak ditemukan pada text T.



Gambar 2. Pencocokan String dengan Algoritma *KMP*

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Berikut adalah *pseudocode* dari algoritma KMP beserta *border function* yang digunakan.

```
function computeFail(pattern : string) → array of integer
KAMUS
   i : integer
   fail : array of integer
ALGORITMA
   fail[0] ← 0
   fail[1] ← 0
   for (i = 2 to (pattern.length – 1) ) do
        {temp = panjang prefix terbesar dari p[0..i] yang juga suffix dari p[1..i] }
        fail[i] ← temp
   endfor
→fail


function kmpMatch(text : string pattern : string) →
```

```
boolean
KAMUS
   i,j : integer
   fail : array of integer
ALGORITMA
   fail ← computeFail(pattern)
   i ← 0
   j ← 0
   while (i < text.length) do
        if (pattern[j] = text[i] then
               if ( j = pattern.length -1) then
                      → true
               endif
               i ← i + 1
               j ← j + 1
        else if (j > 0) then
               j ← fail[j-1]
        else
               i ← i + 1
        endif
   endwhile
→false
```

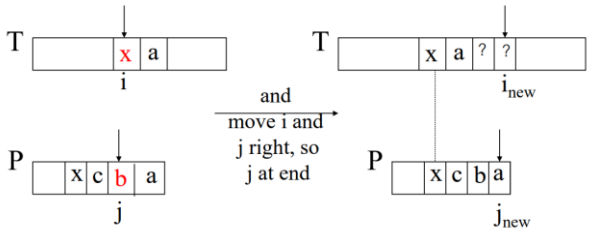
D. Algoritma Boyer-Moore

Algoritma Boyer-Moore (BM) adalah suatu algoritma pencocokkan string yang menggunakan dua buah teknik, yaitu *looking-glass technique* dan *character-jump technique*. *Looking-glass technique* adalah teknik mencari *pattern* P pada *text* T dengan pencocokkan mundur melalui *pattern* P, dimulai dari akhirnya. Sedangkan *character-jump technique* adalah teknik yang digunakan untuk melakukan "pergeseran" *pattern* P pada saat ditemukan ketidakcocokan saat melakukan *looking-glass technique*.

Terdapat 3 kemungkinan kasus untuk *character-jump technique*, dengan urutan pengecekan kasus dari terawal ke terakhir:

1. Kasus 1

   Misal ditemukan ketidakcocokan karakter pada indeks *i* pada *text* dan indeks *j* pada *pattern* dan karakter pada indeks *i* pada *text* adalah *x*. Jika *pattern* mengandung *x* pada indeks sebelah kiri *j* (lebih kecil dari *j*), *pattern* digeser ke kanan sampai karakter *x* pada *pattern* "sejajar" dengan karakter *x* pada *text* yang berada pada indeks *i* (karakter *x* pada *pattern* yang digunakan adalah yang terakhir (paling kanan), *T*[*i*] disejajarkan dengan karakter *x* terakhir pada *pattern*). Indeks *j* lalu dipindahkan ke akhir *pattern* dan *i* disejajarkan dengan *j*.

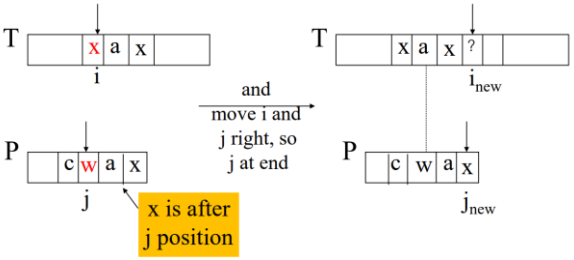


Gambar 3. Ilustrasi Pencocokan string dengan Algoritma Boyer-Moore (1)

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

2. Kasus 2

   Misal ditemukan ketidakcocokan karakter pada indeks *i* pada *text* dan indeks *j* pada *pattern* dan karakter pada indeks *i* pada *text* adalah *x*. Jika *pattern* mengandung *x* pada indeks sebelah kanan *j* (lebih besar dari *j*), *pattern* digeser ke kanan sejauh satu karakter (*P*[*j*] disejajarkan dengan *T*[*i*+*1*]). Indeks *j* lalu dipindahkan ke akhir *pattern* dan *i* disejajarkan dengan *j*.

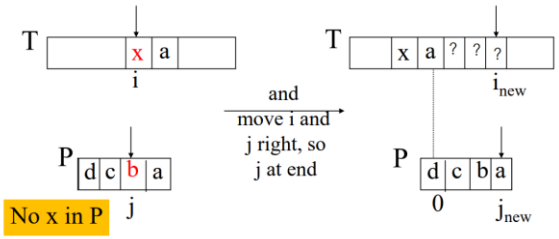


Gambar 4. Ilustrasi Pencocokan string dengan Algoritma Boyer-Moore (2)

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

3. Kasus 3

   Misal ditemukan ketidakcocokan karakter pada indeks *i* pada *text* dan indeks *j* pada *pattern* dan karakter pada indeks *i* pada *text* adalah *x*. Jika *pattern* tidak mengandung *x*, *pattern* digeser ke kanan sampai karakter pertama pada *pattern* sejajar dengan karakter yang berada di sebalah kanan *x* (*P*[*0*] disejajarkan dengan *T*[*i*+*1*]). Indeks *j* lalu dipindahkan ke akhir *pattern* dan *i* disejajarkan dengan *j*.



Gambar 5. Ilustrasi Pencocokan String dengan Algoritma Boyer-Moore (3)

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Setelah melakukan *character-jump technique*, algoritma akan kembali melakukan *looking-glass technique*. Hal ini akan terus dilakukan sampai *pattern* ditemukan pada *text* atau *pattern* tidak ditemukan sampai akhir *text*.

Algoritma Boyer-Moore juga menggunakan suatu fungsi yang disebut *last occurrence function* (dilambangkan dengan *L*()). *L*() memetakan semua kemungkinan karakter pada *pattern* dan *text* menjadi sebuah bilangan bulat (*integer*). *L*(*x*) didefinisikan sebagai indeks *i* terbesar pada *pattern* dengan *P*[*i*] sama dengan *x* atau -1 jika karakter tidak ditemukan pada *pattern*. Fungsi ini biasanya disimpan dalam struktur data yang berbentuk tabel, seperti *array*. Fungsi ini dikalkulasikan saat algoritma pertama kali membaca *pattern* P.

Berikut adalah *pseudocode* dari algoritma Boyer-Moore.

```
function buildLast(pattern : string) → array of integer
KAMUS
   i : integer
   last : array of integer
ALGORITMA
   for (i = 0 to pattern.length):
        last[pattern[i]] = i
   endfor
→last


function bmMatch(text : string pattern : string) → boolean
KAMUS
   i,j : integer
   last : array of integer
ALGORITMA
   last ← buildLast(pattern)
   i ← pattern.length - 1
   j ← pattern.length – 1
   while (i < text.length - 1) do
        if (pattern[j] = text[i] then
               if ( j = 0) then
                      → true
               endif
               i ← i - 1
               j ← j - 1
        else
               lo ← last[text[i]]
               i ← i + pattern.length – min(k, 1 + lo)
               j ← m – 1
        endif
   endwhile
→false
```

E. Plagiarisme

Menurut KBBI, Plagiat adalah suatu tindakan pengambilan karangan orang lain dan mengklaim karangan tersebut merupakan buah pemikirannya sendiri. Tindakan ini seringkali terjadi pada dunia pendidikan, khususnya oleh para siswa/mahasiswa yang secara sengaja/tidak sengaja mengutip ide atau gagasan tanpa menuliskan sumbernya.

Berdasarkan salah satu artkel pada perpustakaan UGM, terdapat beberapa kriteria untuk menentukan apakah suatu karangan terlibat dalam plagiarisme atau bukan. Kriteria tersebut diuraikan sebagai berikut:

1. Penggunaan kalimat orang lain (termasuk parafrase) tanpa menyebutkan identitasnya.
2. Penggunaan fakta, data, gagasan, ide, teori, dan pandangan orang lain tanpa menyebutkan sumbernya.
3. Pengakuan atas tulisan atau karya orang lain sebagai milik sendiri

Pada artikel yang sama juga dijelaskan terkait tipe-tipe dari plagiarisme. Menurut Soelistyo (2011) beberapa tipe plagiarisme:

1. Plagiarisme kata demi kata. Plagiarisme tipe ini termasuk dalam kriteria pertama, yaitu ketika seseorang menuliskan kalimat secara persis tanpa menuliskan sumbernya
2. Plagiasirme terhadap sumber. Tindakan yang didasari kurang jelasnya penulisan akan sumber terhadap suatu gagasan atau ide.
3. Plagiarisme terhadap diri sendiri. Tindakan yang terjadi apabila seorang penulis mempublikasikan karangannya lebih dari satu kali tanpa perubahan yang berarti
4. Plagiarisme kepengarangn, yaitu pengakuan atas karya atau karangan orang lain.

III. Implementasi dan Pembahasan

A. Implementasi Program

Implementasi pengecekan plagiarisme dilakukan dengan memanfaatkan bahasa pemrograman Python. Program akan memanfaatkan ketiga algoritma *string matching* yang telah dibahas dan akan dianalisis pula algoritma yang cocok untuk aplikasi pengecekan plagiarisme.

Secara sederhana, program akan menerima masukan dua buah file, yaitu sebuah file *S* sebagai sumber dan file *U* yang diduga melakukan plagiarisme. Selanjutnya program akan membagi keseluruhan teks pada file *U* perbarisnya yang dianggap sebagai *pattern P* yang akan dicek pada file *S*. Apabila ditemukan kecocokan antara *P* dan *S*, maka *counter* akan bertambah satu. Pada akhir program akan ditampilkan besar dari rasio counter baris yang sama dengan jumlah baris dari *U*. Berikut adalah *pseudocode* dari program utama yang diimplementasikan :

```
{Program Utama Pengecekan Plagiarisme}
KAMUS
nrow,counter : integer
ALGORITMA
nrow ← 0
counter ← 0
{pembacaan file S dan U}
foreach row in U do
   if (row.length) > 0 then
        nrow ← nrow+ 1
        {Bagian ini menyesuaikan algoritma yang digunakan}
        counter ← bruteForceSearch(S, row)
   endif
endfor

output(counter/nrow * 100%)
```

B. Pengujian

Pengujian program dilakukan pada beberapa kasus uji. Seluruh pengujian akan dilakukan dengan menggunakan beberapa file uji dan satu file acuan sumber.

```
The template is designed so that author affiliations are not
repeated each time for multiple authors of the same affiliation.
Please keep your affiliations as succinct as possible (for example,
do not differentiate among departments of the same organization).
This template was designed for two affiliations.
1)  For author/s of only one affiliation (Heading 3):
To change the default, adjust the template as follows.
a)  Selection (Heading 4): Highlight all author and affiliation lines.
b)  Change number of columns: Select the Columns icon from the MS Word
Standard toolbar and then select 1 Column from the selection palette.
c)  Deletion: Delete the author and affiliation lines for the second
affiliation.
2)  For author/s of more than two affiliations: To change the default,
adjust the template as follows.
a)  Selection: Highlight all author and affiliation lines.
b)  Change number of columns: Select the Columns icon from the MS Word
Standard toolbar and then select 1 Column from the selection palette.
c)  Highlight author and affiliation lines of affiliation 1 and copy this selection.
d)  Formatting: Insert one hard return immediately after the last character
of the last affiliation line. Then paste down the copy of affiliation 1.
Repeat as necessary for each additional affiliation.
e)  Reassign number of columns: Place your cursor to the right of the last
character of the last affiliation line of an even numbered affiliation
(e.g., if there are five affiliations, place your cursor at end of fourth affiliation).
Drag the cursor up to highlight all of the above author and affiliation lines.
Go to Column icon and select 2 Columns. If you have an odd number of affiliations,
the final affiliation will be centered on the page; all previous will be in two columns.
```

Gambar 6. File Acuan Sumber

1. Pengujian 1

   Pada pengujian pertama, file yang digunakan berisi potongan teks dari file sumber yang digunakan. Dengan begini, program akan menghasilkan tingkat kemiripan sebesar 100% alias file terindikasi melakukan plagiarisme total.

```
1)  For author/s of only one affiliation (Heading 3):
To change the default, adjust the template as follows.
a)  Selection (Heading 4): Highlight all author and affiliation lines.
b)  Change number of columns: Select the Columns icon from the MS Word
Standard toolbar and then select 1 Column from the selection palette.
c)  Deletion: Delete the author and affiliation lines for the second
affiliation.
2)  For author/s of more than two affiliations: To change the default,
adjust the template as follows.
a)  Selection: Highlight all author and affiliation lines.
b)  Change number of columns: Select the Columns icon from the MS Word
Standard toolbar and then select 1 Column from the selection palette.
```

Gambar 7. Teks yang Digunakan pada Pengujian 1

```
Waktu yang dibutuhkan algoritma Brute Force  :  0.004998207092285156
Waktu yang dibutuhkan algoritma KMP :  0.003973484039306641
Waktu yang dibutuhkan algoritma Boyer-Moore :  0.0009958744049072266
Tingkat kemiripan = 100.0%
```

Gambar 8. Hasil Eksekusi pada Pengujian 1

2. Pengujian 2

   File kedua merupakan gabungan dari teks pada file sumber dan beberapa teks yang tidak berhubungan sekali. Program mendapati terdapat kemiripan antara file uji dan sumber sebesar 71,43%.

```
TestCase Kedua Dimana saya menambahkan sebuat tulisan yang tidak terdapat
pada file sumber.
1)  For author/s of only one affiliation (Heading 3):
To change the default, adjust the template as follows.
a)  Selection (Heading 4): Highlight all author and affiliation lines.
b)  Change number of columns: Select the Columns icon from the MS Word
Standard toolbar and then select 1 Column from the selection palette.
c)  Deletion: Delete the author and affiliation lines for the second
affiliation.
2)  For author/s of more than two affiliations: To change the default,
adjust the template as follows.
a)  Selection: Highlight all author and affiliation lines.
TestCase Kedua Dimana saya menambahkan sebuat tulisan yang tidak terdapat
pada file sumber.
```

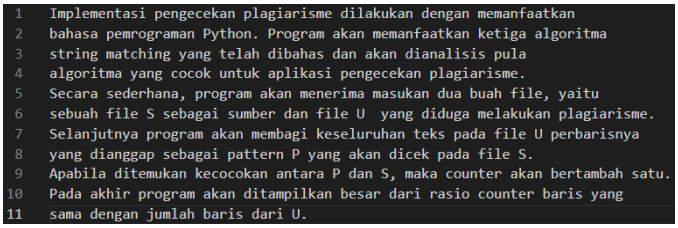
Gambar 9. Teks yang Digunakan pada Pengujian 2

```
Waktu yang dibutuhkan algoritma Brute Force  :  0.005993843078613281
Waktu yang dibutuhkan algoritma KMP :  0.003999948501586914
Waktu yang dibutuhkan algoritma Boyer-Moore :  0.0010027885437011719
Tingkat kemiripan = 71.42857142857143%
```

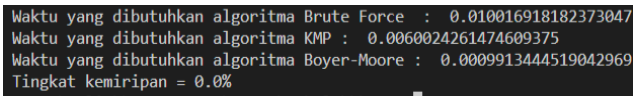
Gambar 10. Hasil Eksekusi pada Pengujian 2

3. Pengujian 3

   Pengujian ketiga dilakukan dengan menggunakan teks uji yang merupakan potongan teks pada makalah ini. Program mendapati bahwa teks uji memiliki tingkat kemiripan 0% dengan file sumber. Dari Hasil yang didapatkan, dapat disimpulkan bahwa file uji 3 bukan merupakan hasil plagiasi terhadap file acuan sumber.



Gambar 11. Teks yang Digunakan pada Pengujian 3



Gambar 12. Hasil Eksekusi pada Pengujian 3

C. *Pembahasan*

Melalui ketiga pengujian di atas, didapatkan data waktu eksekusi untuk tiap algoritma sebagai berikut:

TABLE I. HASIL EKSEKUSI TIAP ALGORITMA

Algoritma	Waktu Eksekusi (s)	Rerata (s)
Brute Force	0.004998207092285156	0.007002989
	0.005993843078613281	
	0.010016918182373047	
KMP	0.003973484039306641	0.004658465
	0.003999948501586914	
	0.0060024261474609375	
Boyer-Moore	0.0009958744049072266	0.000996669
	0.0010027885437011719	
	0.0009913444519042969	

Dari Tabel di atas, dapat dilihat bahwa algoritma *Brute Force* memiliki rerata waktu eksekusi terlama. Hal ini disebabkan karena algoritmanya yang terbilang naif . Selanjutnya, algoritma KMP dengan waktu eksekusi yang lebih baik, yang sebenarnya disebabkan oleh algoritmanya yang terbilang lebih cerdas dibanding *Brute Force.* Kekurangan dari KMP juga disebabkan oleh algoritmanya yang lebih cocok digunakan pada kasus string yang banyak mengalami perulangan karakter seperti pada pengecekan rantai sekuens DNA.

Algoritma Boyer-Moore memiliki kecepatan dan keefisienan yang lebih baik dibandingkan kedua algoritma yang lainnya. Hal ini sesuai dengan karakteristik dari Boyer Moore yang lebih baik pada pengecekan string yang mencakup karakter yang lebih luas dan natural seperti teks pada bahasa inggris.

IV. KESIMPULAN

Algoritma *String Matching* dapat digunakan untuk memeriksa tingkat plagiarisme secara sederhana. Dari hasil pengujian pun didapatkan bahwa ketiga algoritma *string matching* yang digunakan berhasil mencapai tujuan dari pembuatan makalah ini. Dengan catatan algortima Boyer-Moore menjadi algoritma yang paling cocok digunakan pada pengaplikasian pengecekan plagiarisme pada suatu teks.

VIDEO LINK AT YOUTUBE

https://www.youtube.com/watch?v=i4IExT9AqCs

UCAPAN TERIMA KASIH

Puji syukur kepada Tuhan yang Maha Esa karena berkat rahmat-Nya, penulis dapat menyelesaikan makalah ini secara tepat waktu. Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada segala pihak atas dukungan dan bantuan yang telah diberikan dalam pembuatan makalah "Penerapan Algoritma *String Matching* dalam Pengecekan Plagiarisme" ini. Selain itu, penulis secara khusus mengucapkan terima kasih kepada Ibu Dr. Nul Ulfa Maulidevi, S.T., M.Sc, Ibu Dr. Masayu Leylia Khodra, S.T., M.T serta Bapak Dr. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma Tahun 2021/2022.

REFERENSI

[1] Munir, Rinaldi. Pencocokan String. Diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf pada 19 Mei 2022 pukul 18.00 WIB

[2] Munir, Rinaldi. Algoritma Brute Force. Diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf pada 19 Mei 2022 pukul 18.00 WIB

[3] Ayum A, Muhammad. Penerapan Algoritma Knuth-Morris-Pratt dan Boyer-Moore dalam Mendeteksi Plagiarisme pada Artikel Blog.

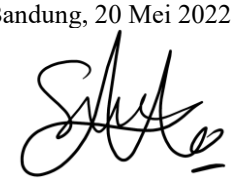
[4] Perpustakaan UGM. Panduan Anti Plagiarism. Diakses melalui https://lib.ugm.ac.id/ind/?page_id=327 pada 20 Mei 2022 pukul 18.00 WIB

[5] Stack Overflow. What are the main differences between the Knuth-Morris-Pratt and Boyer-Moore search algorithms? . Diakses melalui https://stackoverflow.com/questions/12656160/what-are-the-main-differences-between-the-knuth-morris-pratt-and-boyer-moore-sea pada 20 Mei 2022 pukul 16.00 WIB

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 Mei 2022



Suryanto 13520059