Aplikasi Algoritma KMP(Knutt-Morris-Pratt) dalam Pengecekan Plagiarisme

Sulthan Dzaky Alfaro - 13521159
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521159@std.stei.itb.ac.id

***Abstract*—Plagiarisme merupakan tindakan yang tidak etis, yaitu menulis fakta,kutipan atau pendapat yang merupakan milik orang lain tanpa menyebut sumber yang telah diambil. Plagiarisme merupaka tindakan kriminal karena tindakan ini dapat merugikan orang lain. Yang dimaksud orang lain ini orang yang diambil karangannya. Oleh karena itu, diperlukan pengecekan untuk mendeteksi apakah suatu karangan merupakan plagiat dari sebuah karangan orang lain. Pada makalah ini, akan membahas cara pengecekan plagiarisme dengan menggunakan string matching. Ada beberapa algoritma dalam penggunaan string matching, yaitu Boyer-Moore,Brute Force, dan KMP. Dalam makalah ini, pengecekan plagiarisme menggunakan algoritma KMP atau Knutt-Morris-Pratt.**

Keywords— Plagiarsime,KMP,String Matching

I. PENDAHULUAN

Pada zaman ini, perkembangan teknologi berkembang sangatlah cepat. Banyak informasi berterbangan di dalam teknologi kita. Kita dapat mendapatkan berbagai informasi dari teknologi yang kita pakai. Informasi yang dimaksud yaitu, artikel, karangan, buku, film, video, dan lain sebagainya. Informasi-informasi ini tentunya bukan tiba-tiba muncul dalam teknologi kita. Informasi-informasi ini tentunya dibuat oleh seseorang. Kita bisa mendapatkan informasi-informasi tersebut dengan percuma. Namun, hal ini dapat berbahaya apabila kita yang bukan pembuat informasi-informasi ini, misalkan sebuah artikel menyalin isi dari artikel ini dan mengaku artikel ini punya kita. Hal ini merupakan hal yang sangat merugikan bagi si pembuat artikel. Tindakan ini yang biasa kita sebut *Plagiarisme*.

Gambar 1.1 Menyontek Salah Satu Contoh Plagiarisme

Sumber:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fggwp.id%2Fmedia%2Fhiburan%2Flucu%2Fkreatifitas-saat-nyontek&psig=AOvVaw1H_uRQwIH1ddzrcnUSqJSC&ust=1684679617909000&source=images&cd=vfe&ved=0CBEQjRxqFwoTCMiPveKOhP8CFQAAAAAdAAAAABAZ

Menurut Silverman, plagiarisme merupakan menulis fakta, kutipan, atau pendapat yang didapat dari orang lain atau buku, makalah, film, televisi, atau tape tanpa menyebutkan sumbernya. Plagiarisme merupakan tindakan kriminal yang dapat merugikan seseorang. Plagarisme ini bisa dibilang merupakan tindakan mencuri karya milik orang lain. Oleh karena itu, plagiarisme ini harus dihindari dan harus dijauhi, agar dapat menjaga originalitas karya seseorang.

Untuk mencegah terjadinya plagiarisme, diperlukan alat untuk mendeteksi plagiarisme pada sebuah dokumen. Alat digunakan untuk mendeteksi kesamaan antara sebuah dokumen dengan dokumen lain. Banyak cara untuk mendeteksi kesamaan antara dokumen dokumen ini. Ada yang dengan cara membandingkan dengan sumber teks yang ada, menganalisis struktur dari kalimat yang dibuat, menggunakan string matching, dan menghitung presentase kemiripan. Namun alat ini merupakan alat bantu untuk mengecek sebuah dokumen. Ini tidak menjadikan alat ini pengganti penilaian manusia. Hasil yang diberikan oleh alat pengecekan ini juga perlu dievaluasi apakah plagiarisme benar benar terjadi atau tidak.

Pada makalah ini, penulis akan membahas penggunaan konsep String Matching / pencocokan string pada pengecekan plagiarisme. Penggunaan String Matching pada makalah kali ini, penulis akan menggunakan Algoritma KMP atau Knutt-Morris-Pratt pada pengecekan plagiarisme ini.

II. DASAR TEORI

A. *String*

String merupakan bentuk data yang dipakai dalam bahasa pemrograman untuk keperluan menampung dan memodifikasi suatu kalimat pada sebuah teks. Isi dari string dapat berupa huruf, angka, maupun simbol. Dalam pemrograman, mayoritas string dianggap sebagai sebuah array dari sebuah char. Contoh dari string yaitu “makan”. Dalam bahasa pemrograman, dapat di inisialisasi menjadi:

"makan" =

m	a	k	a	n
0	1	2	3	4

Bisa dilihat, indeks pada sebuah string dimulai dari 0 sampai panjang kata - 1.

Salah satu konsep dari string ini, misal panjang dari sebuah string (S) yaitu m. Prefix dari sebuah string adalah substring S[0..k] dan suffix dari S yaitu substring S[k..m-1] dengan k diantara 0 dan m-1. Contoh dari prefix dari sebuah string, ambil contoh yang diatas. Ambil k = 2, didapat prefix dan suffix sebagai berikut.

Prefix:

m	a	k

Suffix:

k	a	n

B. *String Matching*

String Matching merupakan pencarian semua kemunculan pattern pada sebuah teks. Pattern merupaka sebuah string kecil yang nantinya akan dicari pada sebuah teks. Persoalan pada string matching yaitu:

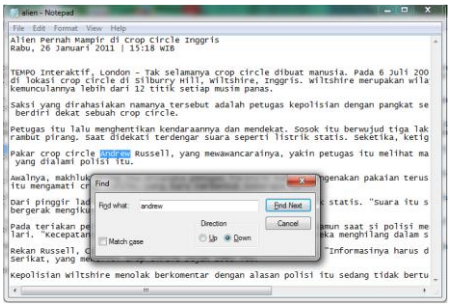
1. T: teks(text), yaitu (long) string yang panjangnya n karakter
2. P: pattern yaitu string dengan panjang m karakter (asumsi m<<n) yang akan dicari didalam sebuah teks

Kita diperintahkan untuk mencari lokasi pertama/semua kemungkinan lokasi di dalam teks yang bersesuaian dengan pattern. Pada string matching ini, dapat diklasifikasikan menjadi 2 bagian menurut arah pencariannya, yaitu:

1. Dari arah kiri ke kanan, yang merupakan arah untuk membaca. Contoh algoritma yang termasuk kategori ini yaitu algoritma Brute Force dan KMP(Knuth,Morris,Pratt)
2. Dari arah kanan ke kiri, arah yang biasanya menghasilkan hasil yang terbaik secara praktikan. Contoh algoritma yang temasuk kategori ini yaitu algoritma BM(Boyer-Moore).

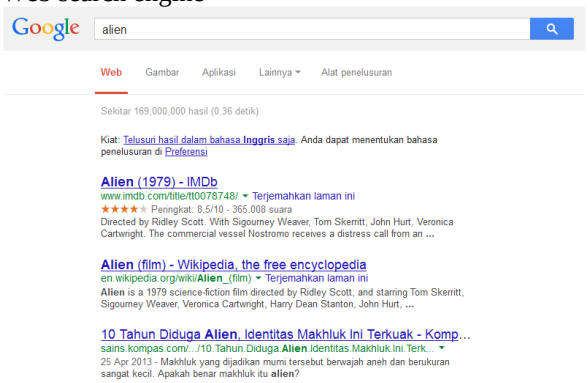
Penggunaan string matching ini sangat bermacam-macam. Baik dalam bidang teknologi maupun kesehatan. Berikut beberapa contoh penggunaan string matching pada kehidupan sehari hari.

1. Pencarian didalam editor teks



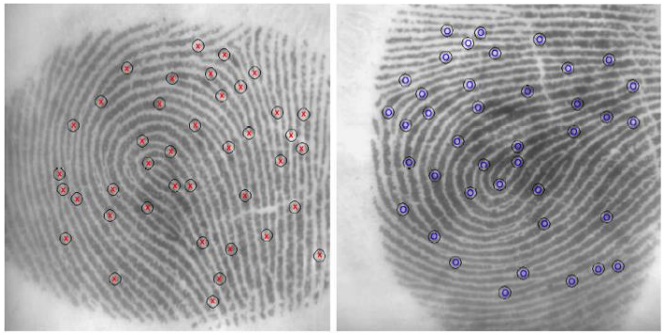
Gambar 2.1 Contoh String Matching pada Teks Editor

2. Web search engine



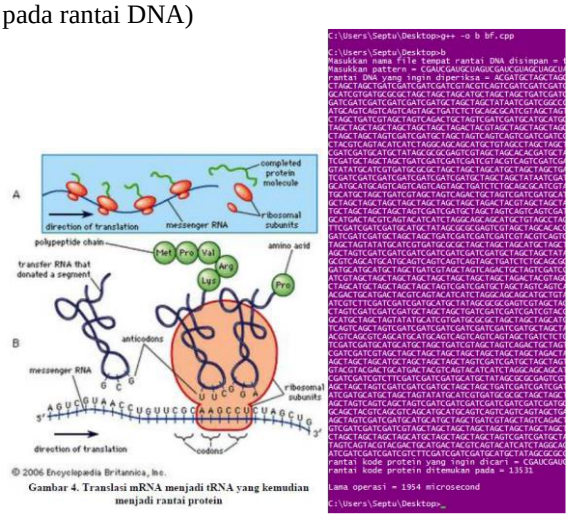
Gambar 2.2 Contoh Penggunaan String Matching pada Search Engine

3. Analisis Citra



Gambar 2.3 Penggunaan String Matching pada Analisis Citra

4. Bioinformatics (Pencocokan Rantai Asam Amino pada rantai DNA)



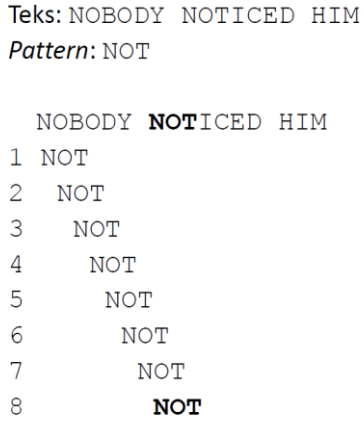
Gambar 4. Translasi mRNA menjadi tRNA yang kemudian menjadi rantai protein

Gambar 2.4 Penggunaan String Matching pada Pencocokan Rantai Asam Amino pada DNA

C. *Algoritma Brute Force*

Algoritma Brute Force pada string matching merupakan algoritma yang termasuk lama dibandingkan algoritma lain. Algoritma ini menjadikannya lama karena pada saat pengecekan string pattern pada sebuah teks, pattern dicek secara huruf per huruf dan apabila ada huruf yang miss match,

pengecekan akan maju satu huruf pada teks dan pengecekan pattern dimulai dari awal. Untuk lebih jelasnya, dapat dilihat pada gambar berikut.



Gambar 2.5 Ilustrasi String Matching dengan Algoritma Brute Force

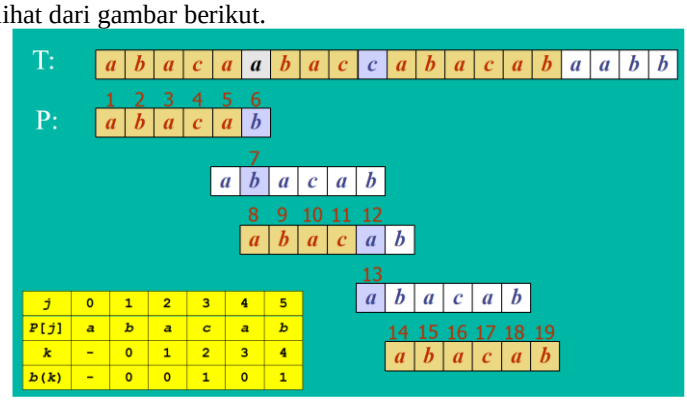
Berikut contoh dari pseudocode Algoritma Brute Force

```
Function BruteForce(teks : string,pattern :
string)
Kamus
       i,j = integer
       indeks = arrays of integer
Algoritma

Begin

for (i = 0 to length(teks)-1) do
    for (j = 0 to length(pattern)-1)do
        if(pattern[j] != teks[i+j]) do
            endfor
        else
          if(pattern[j] == teks[i+j]) do
            if(j==length(pattern)-1) do
                indeks.append(i)
            else
                continue
endfor
->indeks

End
```

Berdasarkan contoh pseudocode diatas, kita bisa lihat bahwa pengulangan terjadi 2 kali, yaitu pada teks dan pattern untuk pengecekan. Sehingga Algoritma Brute Force ini memiliki kompleksitas waktu yang sangat besar yaitu worst case O(mn) dan best case O(n).

D. Algoritma KMP (Knuth-Morris-Pratt)

Algoritma KMP merupakan salah satu algoritma string matching yang prinsipnya mirip dengan Algoritma Brute Force. Namun Algoritma KMP ini memiliki perbedaan pada cara pengecekannya. Pada algoritma ini, pengecekan string pattern tidak selalu pada awal huruf pattern. Apabila ada missmatch pada string pattern, indeks pengecekan pada teks saat sebelum missmatch tetap sama, namun pengecekan pada pattern tidak dimulai dari awal lagi. Supaya lebih jelas, dapat dilihat dari gambar berikut.



Gambar 2.6 Ilustrasi String Matching dengan Algoritma KMP

Sebelum pengecekan dimulai, terlebih dahulu membuat Border Function pada pattern yang akan dicek. Border Function ini digunakan sebagai acuan apabila missmatch pada pattern, pengecekan tidak mulai dari awal.

Untuk mendapatkan border function pada sebuah pattern, yaitu dengan cara kita lihat satu satu indeks dari 0 sampai length(pattern)-1,misalkan pengecekan pada indeks i. Pengecekannya dengan cara ambil jumlah huruf tertinggi dari prefix dan suffix yang sama. Prefix disini diambil dari pattern[0..i] dan Suffix diambil dari pattern[1..i]. Sebagai contoh, kita ambil contoh yang diatas “abacab”. Misal kita cek pada indeks yang ke 3. Untuk nilai k disini yaitu nilai j-1. Kita cek apakah ada prefix dari [0..k] yang sama dengan suffix dari [1..k]. Kita mulai dari prefix. Prefix yang didapat pertama adalah “a” dan suffix adalah “a”. Karena sama, berarti kita bisa simpan pada k = j -1 = 2 sementara adalah 1. Lanjut dengan prefix [0..1] dan suffix [k-1..k], yaitu “ab” dan “ba”. Karena tidak cocok, maka hasil ketika 2 huruf tidak disimpan. Karena suffix sudah mencapai angka 1, maka kita stop pengecekan, sehingga dihasilkan untuk k = 2 dengan indeks 3, didapat border function adalah 1.

Untuk proses algoritma KMP ini, pertama tama tentunya mencari border function untuk pattern yang akan dicari(misal border). Lalu selanjutnya mulai pengecekan string dari awal teks. Pengecekan satu satu seperti Brute Force. Apabila ada missmatch, dan missmatch terjadi pada indeks j pada pattern, selanjutnya j digeser menjadi border[j-1]. Apabila j berada pada indeks 0, maka pattern digeser 1 huruf terhadap teks, seperti algoritma Brute Force. Untuk ilustrasi cara kerja algoritma KMP dapat dilihat pada gambar di atas. Berikut contoh dari pseudocode dari Algoritma KMP.

```
Function borderFunction(pattern:string)
Kamus
    i = integer
    simpan = arrays of integer
Algoritma
Begin

for (i=2 to length(pattern)-1) do
   {cari prefix dan suffix yang sama seperti
    cara diatas}
   simpan.append(hasilnya)
```

```
endfor
->simpan
End

Function KMP (pattern:string,text:string)
Kamus
    i,j = integer
    indeks = arrays of integer
Algoritma
i <- 0
j <- 0
border <- borderFunction(pattern)
while(i<length(teks)) do
    if(pattern[j]==teks[i]) do
        if(j==length(pattern)-1) do
            indeks.append(i-j)
            j = border[j-1]
        i++
        j++
    elif(pattern[j]!=teks[i]) do
        if(j!=0) do
           j = border[j-1]
        else
           i+=1
endwhile
 ->indeks
```

Algoritma KMP terbilang lebih baik dari pada Brute Force. Kempleksitas waktu yang dihasilkan pun lebih cepat dari pada Brute Force, yaitu untuk mencari border function O(m) dan pengecekan string O(n) jadi kompleksitasnya O(m+n).

E. Algoritma BM (Boyer-Moore)

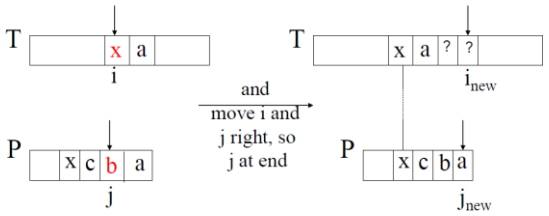
Algoritma BM atau Boyer-Moore juga merupakan salah satu algoritma pada string matching. Algoritma ini merupakan algoritma yang unik, karena tidak seperti algoritma Brute Force maupun KMP yang pengecekan string dimulai dari kiri ke kanan, BM ini pengecekan string dimulai dari akhir pattern ke kiri. Dalam pengecekan pattern pada string dengan menggunakan algoritma BM ini bisa terjadi 3 kasus.

Sebelum pengecekan dimulai, kita perlu mencari last occurrence. Last Occurrence merupakan semua kemungkinan huruf pada teks yang muncul terakhir pada sebuah pattern. Sebagai contoh teks yang ingin digunakan yaitu “abbacdaabacab”. Dalam teks ini terdiri dari 4 huruf yang menyusun string tersebut yaitu {a,b,c,d}. Lalu untuk contoh pattern yang ingin dicocokkan yaitu “abacab”. Untuk mencari last occurrence, kita lihat huruf huruf yang menyusun pada teks di pattern terkahir berada di mana. Misal huruf “b”. Huruf ini terakhir terlihat pada pattern pada indeks 5, sehingga last occurrence nya 5. Begitu juga “a”, huruf ini terlihat terakhir berada pada indeks 4, sehingga last occurrence nya 4. Untuk “d”, karena tidak ada pada pattern, sehingga last occurrencenya -1. Berikut tabel last occurrence untuk pattern tersebut

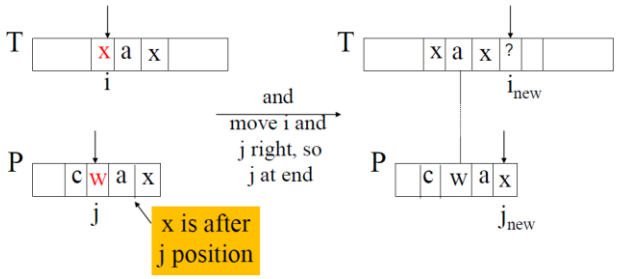
a	b	c	d
4	5	3	-1

Selanjutnya, seperti yang sebelumnya sudah ditulis, ada 3 kemungkinan yang mungkin dalam proses pengecekan dengan algoritma BM. Berikut kasus kasus yang mungkin.

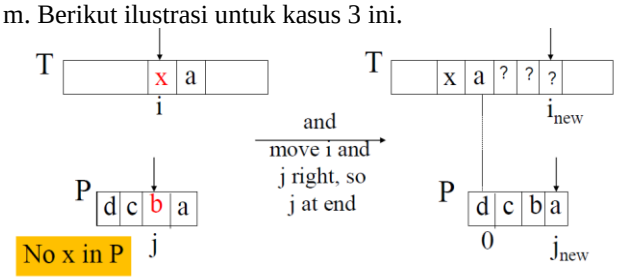
1. Pada saat pengecekan pattern dengan teks, misalkan pengecekan pada teks i dan pada pattern j, jika ada missmatch antara i dan j dan misal huruf yang missmatch pada teks adalah x, dan x yang ada di pattern berada disebelah kiri teks yang missmatch (lebih kecil dari j), maka pattern digeser sampai pattern huruf x sejajar dengan x yang ada di teks. Adapun rumus untuk menentukan indeks i selanjutnya, yaitu i = i + m – (lo+1) dengan m yaitu panjang pattern dan lo merupakan last occurrence missmatch. Berikut ilustrasi untuk kasus ini.



2. Pada saat pengecekan string, missmatch pada i dan j dengan huruf pada teks misal x, dan x yang ada di pattern berada di sebelah kanan missmatch. Dalam kasus ini, untuk melanjutkan pengecekan, i digeser menjadi i baru dengan rumus i = i + m – j dengan m panjang pattern dan j adalah indeks pattern pada saat missmatch. Berikut ilustrasi pada kasus 2 ini.



3. Pada saat pengecekan string, missmatch pada i dan j dengan huruf pada teks misal x, dan x tidak ada pada pattern, sehingga i digeser sejauh m. Dapat dirumuskan, i baru untuk pengecekan selanjutnya i = i + m. Berikut ilustrasi untuk kasus 3 ini.



Berikut pseudocode untuk algoritma BM.

```
Function lastOcc(teks:string,pattern:string)
Kamus
    i,j,maks = integer
    lo = map(char,integer)
    huruf = arrays of char
```

```
Algoritma
Begin
huruf <- {semua huruf yang ada di teks}
for (i = 0 to length(huruf)-1) do
    maks <- -1
    for (j = 0 to legth(pattern)-1) do
        if(huruf[i] == pattern[j]) do
            maks <- j
    lo.add(huruf[i],maks)
-> lo

Function BM(teks:string, pattern:string)
Kamus
    i,j = integer
    lo = map(char,integer)
    indeks = arrays of integer
Algoritma
lo <- lastOcc(teks,pattern)
i <- length(pattern) - 1
j <- length(pattern) - 1
while(i<length(teks)) do
    if(teks[i]==pattern[j]) do
        if(j==0) do
            indeks.append(i)
        else
            i--
            j--
    else
         last = lo.value(teks[i])
         i = i + length(pattern) - min(lo+1,j)
         j = length(pattern)-1
endwhile
->indeks
End
```

F. Plagiarisme

Menurut Silverman, plagiarisme merupakan menulis fakta, kutipan, atau pendapat yang didapat dari orang lain atau buku, makalah, film, televisi, atau tape tanpa menyebutkan sumbernya. Plagiarisme merupakan tindakan kriminal yang dapat merugikan seseorang. Plagiarisme sering terjadi belakangan ini dan mayoritas berada pada bidang pendidikan.

Pada artikel dari Binus, plagiarisme dibagi menjadi 5 jenis yaitu:

1. Plagiarisme Verbatim
   Plagiarisme ini yang pelanggarannya sangat berat, yaitu menyalin semua isi dari sebuah karya orang lain sama persis dengan memberi kesan sebagai karya pribadi pelaku plagiarisme
2. Plagiarisme Kain Perca (Patchwork)
   Plagiarisme ini mengambil karya orang lain tanpa menyebutkan sumber yang dia ambil. Pelaku mengambil beberapa bagian karya seseorang lalu disusun dan dikesankan sebagai karya milik si pelaku
3. Plagiarisme Parafrasa
   Plagiarisme ini mengubah kalimat milik karya orang lain menjadi kalimat baru yang pelaku buat. Jika pengutip jujur, maka pengutip akan mencantumkan sumbernya pada karya yang dia buat. Namun pelaku plagiarisme akan mengambil kutipan karya orang lain dan menampilkannya sebagai kutipan tidak langsung, tanpa menyebutkan sumbernya. Hal ini juga berlaku apabila berbeda bahasa.
4. Plagiarisme Kata Kunci / Frasa Kunci
   Plagiarisme ini hanya mengambil beberapa kata kunci atau frasa kunci dari karya orang lain. Selanjutnya, pelaku akan membuat ulang kalimat-kalimat dalam karyanya, tetapi tetap memasukkan kata kunci ata frasa kunci dari karya orang lain. Dan selain itu, pelaku juga tidak menyebutkan sumber yang dia ambil.
5. Plagiarisme Struktur Gagasan
   Plagiarisme ini merupakan jenis yang sulit untuk dilacak. Pelaku plagiarisme menyalin karya orang lain lalu gagasan ini dibuat kembali menjadi rangkaian kalimat dengan kata kunci atau frasa kunci yang berbeda. Walaupun kata kunci dan frasa kunci dari karya orang lain memang sudah tidak dipakai lagi, tetapi struktur gagasannya sama.

III. Implementasi dan Pembahasan

A. Implementasi Program

Implementasi program pada makalah ini, yaitu pengecekan plagiarisme dengan menggunakan bahasa pemrograman python. Program ini hanya menggunakan 1 algoritma diantara 3 algoritma string matching, yaitu KMP (Knuth-Morris-Pratt). Program ini juga menggunakan perhitungan presentase kesamaan antara 2 teks dalam pengaplikasiannya.

Penjelasan untuk program ini, pertama program akan meminta input 2 file teks, misal teks1 dan teks2. Nantinya, program akan mengambil semua kata kata yang pernah muncul pada teks 1 dan disimpan pada sebuah array, begitu juga pada teks 2. Lalu untuk array kata teks 1, program akan mencari berapa banyak kata kata yang ada di teks1 muncul pada teks1 dan teks2 dengan menggunakan algoritma KMP. Setelah mendapatkan berapa kali kata teks1 muncul pada teks1 dan 2, kita bandingkan hasilnya dan hitung berapa persen kesamaan dari banyak kata pada teks1 dengan teks2. Setelah semua kata mendapatkan persentase kesamaannya, kita rata rata semua persentase tadi dan didapat hasil persentase kesamaan untuk teks1 terhadap teks2. Begitu juga untuk teks2, ambil semua kata yang pernah muncul di teks2, cari total kemunculan kata kata pada teks2 dan teks1, lalu hitung persen kesamaan dari total kemunculan tersebut lalu hitung rata-ratanya dan didapat hasil persentase kesamaan dari teks2 terhadap teks1. Setelah mendapat hasil dari teks1 dan teks2, lalu dihitung rata rata hasil tersebut dan hasil akhir didapat. Berikut secara sederhana pseudocode dari programnya.

```
Function getPercentage(teks1:string,teks2:string)
Kamus
    i,j,hasil = integer
    hasilkmp1, hasilkmp2= arrays of integer
Algoritma
Begin
```

```
kata1 <- {semua kata yang pernah muncul di teks1}
kata2 <- {semua kata yang pernah muncul di teks2}
for (i=0 to length(kata1)) do
    hasilkmp11 <- KMP (kata1[i], teks1)
    hasilkmp12 <- KMP (kata1[i], teks2)
    minim = min(length(hasilkmp11),
            length(hasilkmp12))
    maks = max(length(hasilkmp11),
           length(hasilkmp12))
    hasilkmp1.append(minim/maks*100)
endfor
for (j=0 to length(kata2)) do
    hasilkmp21 <- KMP (kata2[i], teks1)
    hasilkmp22 <- KMP (kata2[i], teks2)
    minim = min(length(hasilkmp21),
            length(hasilkmp22))
    maks = max(length(hasilkmp21),
           length(hasilkmp22))
    hasilkmp2.append(minim/maks*100)
endfor
hasil <- (rataArr(hasilkmp1)+rataArr(hasilkmp2))/2
#rataArr merupakan fungsi untuk mendapatkan rata
rata dari sebuah array
->hasil
End
```

B. Pengujian

Pengujian program dilakukan dengan beberapa kasus uji. Test pengujian teks dilakukan dengan menggunakan beberapa file txt dengan berisi artikel singkat. Artikel didapat dari bantuan ChatGPT untuk uji coba.

1. Pengujian 1

   Pada pengujian pertama, file yang digunakan berisi sama persis dengan file lain yang akan diuji. Program akan mengasilkan persentase 100% karena file berisi sama dengan file lain. File test bernama test1.txt dan test1sama.txt.

Kata test1sama.txt	KMP test1sama.txt	KMP test1.txt	Presentase Kesamaan(%)
Di	6	6	100.0
zaman	1	1	100.0
digital	1	1	100.0
yang	2	2	100.0
terus	1	1	100.0
berkembang	1	1	100.0
teknologi	2	2	100.0
telah	3	3	100.0
mengubah	1	1	100.0
cara	1	1	100.0
kita	4	4	100.0
berinteraksi	1	1	100.0
bekerja	1	1	100.0
dan	3	3	100.0
berkomunikasi	1	1	100.0
Internet	1	1	100.0
menjadi	1	1	100.0
bagian	1	1	100.0
tak	1	1	100.0
terpisahkan	1	1	100.0
dari	1	1	100.0
kehidupan	2	2	100.0
sehari-hari	1	1	100.0
memungkinkan	1	1	100.0
untuk	1	1	100.0
dengan	2	2	100.0
cepat	1	1	100.0
mengakses	1	1	100.0
informasi	1	1	100.0
terhubung	1	1	100.0
orang-orang	1	1	100.0
di	6	6	100.0
seluruh	1	1	100.0
dunia	1	1	100.0
Smartphone	1	1	100.0
aplikasi	1	1	100.0
mobile	1	1	100.0
juga	1	1	100.0
semakin	1	1	100.0
memudahkan	1	1	100.0
dalam	2	2	100.0
melakukan	1	1	100.0
berbagai	1	1	100.0
aktivitas	1	1	100.0
seperti	1	1	100.0
berbelanja	1	1	100.0
online	1	1	100.0
memesan	1	1	100.0
makanan	1	1	100.0
atau	1	1	100.0
mengatur	1	1	100.0
keuangan	1	1	100.0
pribadi	1	1	100.0
Oleh	1	1	100.0
karena	1	1	100.0
itu	1	1	100.0
tidak	1	1	100.0
dapat	1	1	100.0
disangkal	1	1	100.0
bahwa	1	1	100.0
perkembangan	1	1	100.0
memiliki	1	1	100.0
dampak	1	1	100.0
signifikan	1	1	100.0
modern	1	1	100.0

Gambar 3.1 Detail Persentase test1sama.txt Dengan test1.txt

Bisa dilihat semua kata yang ada pada 2 file tersebut memiliki jumlah masing masing kemunculan kata yang sama, sehingga persentase kesamaan memiliki nilai 100%.

2. Pengujian 2

Pada pengujian kedua, file yang digunakan berisi teks yang hampir mirip dengan file lain yang akan diuji. Hampir mirip disini maksudnya tema, kata-kata yang digunakan hampir sama. File yang digunakan yaitu file1.txt dengan file2.txt.

```
Masukkan file teks 1 = TugasMakalahIF2211_13521159\test\test2.txt
Masukkan file teks 2 = TugasMakalahIF2211_13521159\test\test1.txt
Presentase plagiarisme antara test2.txt dan test1.txt sekitar = 67.49465811965811
```

Kata test2.txt	KMP test2.txt	KMP test1.txt	Presentase Kesamaan(%)
Dalam	2	2	100.0
era	2	1	50.0
digital	1	1	100.0
yang	2	2	100.0
terus	1	1	100.0
berubah	1	0	0.0
teknologi	2	2	100.0
telah	3	3	100.0
mengubah	1	1	100.0
cara	1	1	100.0
kita	5	4	80.0
berinteraksi	1	1	100.0
bekerja	1	1	100.0
dan	4	3	75.0
berkomunikasi	1	1	100.0
Internet	1	1	100.0
menjadi	1	1	100.0
bagian	1	1	100.0
tak	2	1	50.0
terpisahkan	1	1	100.0
dari	2	1	50.0
kehidupan	2	2	100.0
sehari-hari	1	1	100.0
memberikan	1	0	0.0
akses	1	1	100.0
terbatas	1	0	0.0
kepada	1	0	0.0
informasi	1	1	100.0
menghubungkan	1	0	0.0
dengan	4	2	50.0
orang-orang	1	1	100.0
di	4	6	66.66666666666666
seluruh	1	1	100.0

dunia	1	1	100.0
Selain	1	0	0.0
itu	1	1	100.0
adanya	1	0	0.0
smartphone	1	1	100.0
aplikasi	1	1	100.0
mobile	1	1	100.0
dapat	1	1	100.0
mudah	1	1	100.0
melakukan	1	1	100.0
berbagai	1	1	100.0
aktivitas	1	1	100.0
mulai	1	0	0.0
berbelanja	1	1	100.0
online	1	1	100.0
hingga	1	0	0.0
mengatur	1	1	100.0
keuangan	1	1	100.0
pribadi	1	1	100.0
Dengan	4	2	50.0
demikian	1	0	0.0
perkembangan	1	1	100.0
membawa	1	0	0.0
perubahan	1	0	0.0
besar	1	0	0.0
dalam	2	2	100.0
modern	1	1	100.0

Rata-rata presentase kesamaan = 72.8611111111111

Gambar 3.2 Detail Persentase Kesamaan test2.txt terhadap test1.txt

Kata test1.txt	KMP test2.txt	KMP test1.txt	Presentase Kesamaan(%)
Di	4	6	66.66666666666666
zaman	0	1	0.0
digital	1	1	100.0
yang	2	2	100.0
terus	1	1	100.0
berkembang	0	1	0.0
teknologi	2	2	100.0
telah	3	3	100.0
mengubah	1	1	100.0
cara	1	1	100.0
kita	5	4	80.0
berinteraksi	1	1	100.0
bekerja	1	1	100.0
dan	4	3	75.0
berkomunikasi	1	1	100.0
Internet	1	1	100.0
menjadi	1	1	100.0
bagian	1	1	100.0
tak	2	1	50.0
terpisahkan	1	1	100.0
dari	2	1	50.0
kehidupan	2	2	100.0
sehari-hari	1	1	100.0
memungkinkan	0	1	0.0
untuk	0	1	0.0
dengan	4	2	50.0
cepat	0	1	0.0
mengakses	0	1	0.0
informasi	1	1	100.0
terhubung	0	1	0.0
orang-orang	1	1	100.0
di	4	6	66.66666666666666
seluruh	1	1	100.0

dunia	1	1	100.0
Smartphone	1	1	100.0
aplikasi	1	1	100.0
mobile	1	1	100.0
juga	0	1	0.0
semakin	0	1	0.0
memudahkan	0	1	0.0
dalam	2	2	100.0
melakukan	1	1	100.0
berbagai	1	1	100.0
aktivitas	1	1	100.0
seperti	0	1	0.0
berbelanja	1	1	100.0
online	1	1	100.0
memesan	0	1	0.0
makanan	0	1	0.0
atau	0	1	0.0
mengatur	1	1	100.0
keuangan	1	1	100.0
pribadi	1	1	100.0
Oleh	0	1	0.0
karena	0	1	0.0
itu	1	1	100.0
tidak	0	1	0.0
dapat	1	1	100.0
disangkal	0	1	0.0
bahwa	0	1	0.0
perkembangan	1	1	100.0
memiliki	0	1	0.0
dampak	0	1	0.0
signifikan	0	1	0.0
modern	1	1	100.0

Rata-rata presentase kesamaan = 62.128205128205124

Gambar 3.3 Detail Persentase Kesamaan test1.txt terhadap test2.txt

Seperti yang terlihat, banyak kata kata yang kemunculannya sama antara 2 file tersebut. Walaupun ada beberapa kata yang di satu file ada dan di file lain tidak ada, tetapi mayoritas persentase kesamaan di nilai yang besar, sehingga persentase kesamaan antara 2 file menjadi besar. Bisa dikatakan kedua file terindikasi plagiarisme.

3. Pengujian 3

Pada pengujian 3, file yang digunakan berisi teks yang sangat berbeda dengan file teks yang lain. Karena isi dari teks berbeda, persentase kesamaan antara kedua teks akan semakin kecil. File yang digunakan adalah file test2.txt dan test3.txt

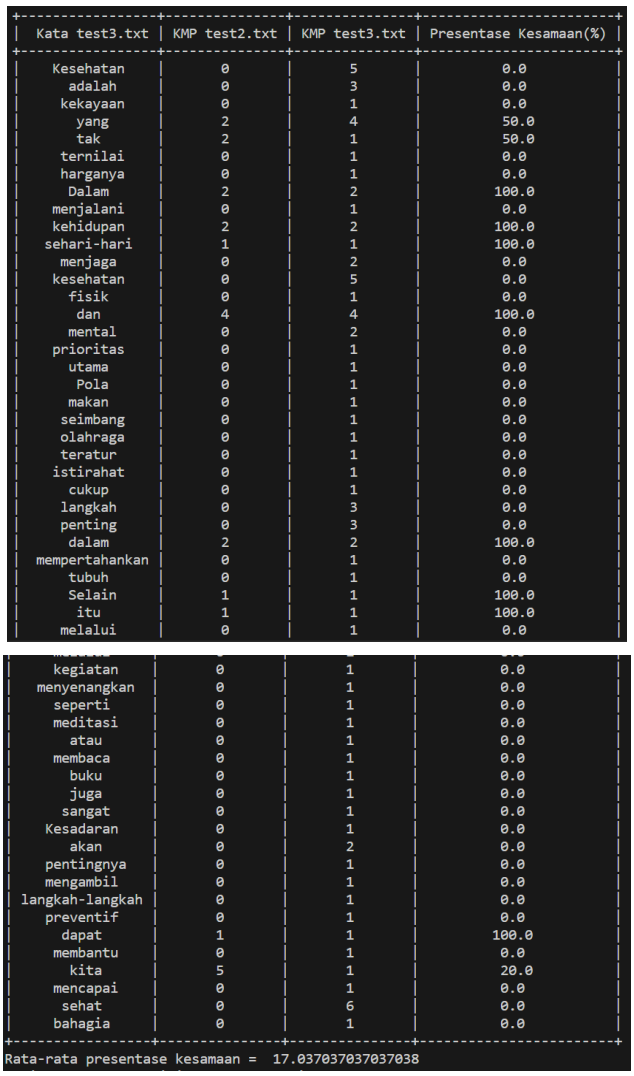
```
Masukkan file teks 1 = TugasMakalahIF2211_13521159\test\test2.txt
Masukkan file teks 2 = TugasMakalahIF2211_13521159\test\test3.txt
Presentase plagiarisme antara test2.txt dan test3.txt sekitar = 16.810185185185183
```

Kata test2.txt	KMP test2.txt	KMP test3.txt	Presentase Kesamaan(%)
Dalam	2	2	100.0
era	2	1	50.0
digital	1	0	0.0
yang	2	4	50.0
terus	1	0	0.0
berubah	1	0	0.0
teknologi	2	0	0.0
telah	3	0	0.0
mengubah	1	0	0.0
cara	1	0	0.0
kita	5	1	20.0
berinteraksi	1	0	0.0
bekerja	1	0	0.0
dan	4	4	100.0
berkomunikasi	1	0	0.0
Internet	1	0	0.0
menjadi	1	0	0.0
bagian	1	0	0.0
tak	2	1	50.0
terpisahkan	1	0	0.0
dari	2	0	0.0
kehidupan	2	2	100.0
sehari-hari	1	1	100.0
memberikan	1	0	0.0
akses	1	0	0.0
terbatas	1	0	0.0
kepada	1	0	0.0
informasi	1	0	0.0
menghubungkan	1	0	0.0
dengan	4	0	0.0
orang-orang	1	0	0.0
di	4	1	25.0
seluruh	1	0	0.0

dunia	1	0	0.0
Selain	1	1	100.0
itu	1	1	100.0
adanya	1	0	0.0
smartphone	1	0	0.0
aplikasi	1	0	0.0
mobile	1	0	0.0
dapat	1	1	100.0
mudah	1	0	0.0
melakukan	1	0	0.0
berbagai	1	0	0.0
aktivitas	1	0	0.0
mulai	1	0	0.0
berbelanja	1	0	0.0
online	1	0	0.0
hingga	1	0	0.0
mengatur	1	0	0.0
keuangan	1	0	0.0
pribadi	1	0	0.0
Dengan	4	0	0.0
demikian	1	0	0.0
perkembangan	1	0	0.0
membawa	1	0	0.0
perubahan	1	0	0.0
besar	1	0	0.0
dalam	2	2	100.0
modern	1	0	0.0

Rata-rata presentase kesamaan = 16.583333333333332

Gambar 3.4 Detail Persentase Kesamaan test2.txt terhadap test3.txt



Kata test3.txt	KMP test2.txt	KMP test3.txt	Presentase Kesamaan(%)
Kesehatan	0	5	0.0
adalah	0	3	0.0
kekayaan	0	1	0.0
yang	2	4	50.0
tak	2	1	50.0
ternilai	0	1	0.0
harganya	0	1	0.0
Dalam	2	2	100.0
menjalani	0	1	0.0
kehidupan	2	2	100.0
sehari-hari	1	1	100.0
menjaga	0	2	0.0
kesehatan	0	5	0.0
fisik	0	1	0.0
dan	4	4	100.0
mental	0	2	0.0
prioritas	0	1	0.0
utama	0	1	0.0
Pola	0	1	0.0
makan	0	1	0.0
seimbang	0	1	0.0
olahraga	0	1	0.0
teratur	0	1	0.0
istirahat	0	1	0.0
cukup	0	1	0.0
langkah	0	3	0.0
penting	0	3	0.0
dalam	2	2	100.0
mempertahankan	0	1	0.0
tubuh	0	1	0.0
Selain	1	1	100.0
itu	1	1	100.0
melalui	0	1	0.0
kegiatan	0	1	0.0
menyenangkan	0	1	0.0
seperti	0	1	0.0
meditasi	0	1	0.0
atau	0	1	0.0
membaca	0	1	0.0
buku	0	1	0.0
juga	0	1	0.0
sangat	0	1	0.0
Kesadaran	0	1	0.0
akan	0	2	0.0
pentingnya	0	1	0.0
mengambil	0	1	0.0
langkah-langkah	0	1	0.0
preventif	0	1	0.0
dapat	1	1	100.0
membantu	0	1	0.0
kita	5	1	20.0
mencapai	0	1	0.0
sehat	0	6	0.0
bahagia	0	1	0.0

Rata-rata presentase kesamaan = 17.037037037037038

Gambar 3.5 Detail Persentase Kesamaan test3.txt terhadap test2.txt

Bisa terlihat pada detail persentase kesamaan diatas, banyak kata kata yang ada di suatu file tetapi tidak ada pada file lain, sehingga mayoritas persentase kesamaan pada kata kata tersebut 0%. Oleh karena itu, hasil akhir persentase kesamaannya sangat kecil. Bisa dikatakan kedua file tidak terindikasi plagiarisme.

IV. Kesimpulan

String Matching merupakan salah satu cara untuk mendapatkan lokasi suatu pattern pada sebuah teks. String Matching memiliki beberapa algoritma dalam penyelesaiannya, diantaranya yaitu algoritma Brute Force, KMP, dan BM. Konsep dari String Matching ini dapat diaplikasikan ke berbagai hal, salah satunya yaitu pengecekan plagiarisme. Plagiarisme merupakan tindakan kriminal berupa mengambil karya orang lain tanpa menyebutkan sumber yang diambilnya. Pada program ini, program telah berhasil menentukan berapa persen kemiripan antara 2 teks. Program ini tidaklah menjadi patokan, karena bisa jadi hasil yang diberikan sedikit melenceng. Kemungkinan hal ini bisa disebabkan adanya duplikasi pada saat pengecekan dengan menggunakan KMP. Tetapi untuk ukuran pengecekan plagiarisme sederhana bisa terbilang cukup dalam mendeteksi adanya plagiarisme.

Ucapan Terimakasih

Segala puji bagi Allah SWT yang telah memberikan kemudahan serta kelancaran kepada penulis sehingga dapat menyelesaikan tugas makalah dengan tepat waktu. Penulis mengucapkan terima kasih banyak kepada dosen-dosen Strategi Algoritma IF2211, terutama kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengajar kelas K1 yang telah memberikan materi sehingga penulis dapat mengaplikasikan materi yang diberikan untuk memenuhi tugas makalah ini. Tidak lupa juga penulis mengucapkan terima kasih kepada kedua orang tua dan teman-teman yang telah memberi semangat dan dukungan selama pengerjaan makalah ini. Akhir kata, penulis menyadari masih terdapat kekurangan dalam pengerjaan makalah ini. Dan penulis juga berharap makalah ini dapat digunakan dan dimanfaatkan sebaik-baiknya untuk masyarakat sekitar.

References

[1] https://teknik.unpas.ac.id/blogs/apa-itu-plagiarisme/
Diakses tanggal 19/05/2023 pukul 18.49

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf
Diakses tanggal 20/05/2023 pukul 08.12

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf
Diakses tanggal 20/05/2023 pukul 09.04

[4] https://yuliana.lecturer.pens.ac.id/Konsep%20Pemrograman/Teori/T10-String(1).pdf
Diakses tanggal 20/05/2023 pukul 08.21

[5] https://www.ilmuskripsi.com/2016/05/algoritma-pencarian-string-string.html
Diakses tanggal 20/05/2023 pukul 09.04

[6] https://business-law.binus.ac.id/2015/04/01/plagiarisme-jenis-jenisnya-bagian-2-dari-3-tulisan/
Diakses tanggal 20/05/2023 pukul 21.07

LINK GITHUB

https://github.com/SulthanDA28/TugasMakalahIF2211_13521159.git

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, 21 Mei 2023

Sulthan Dzaky Alfaro - 13521159