Penerapan Konsep Pencocokan String Pada Program Pendeteksi Plagiarisme Sederhana

Nabila Hannania - 13519097
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519097@std.stei.itb.ac.id

***Abstract*—Plagiarisme merupakan suatu tindakan kriminal yang harus ditindak lanjuti karena merugikan orang lain dan melanggar moral yang berlaku di masyarakat. Maka dari itu, diperlukan suatu tindak pencegahan dan penangganan terhadap plagiarisme ini, salah satunya dengan membuat suatu aplikasi yang dapat mendeteksi plagiarisme. Pada makalah ini, akan dibahas mengenai program sederhana yang dapat mendeteksi plagiarisme antara suatu dokumen dengan lainnya (ada 2 dokumen yang berbeda) dengan menerapkan konsep pencocokan string. Pencocokan string merupakan suatu algoritma yang digunakan untuk mencari lokasi pertama dalam teks yang bersesuaian dengan pattern. Ada beberapa macam algoritma pencocokan string, yaitu algoritma Brute Force, algoritma Knuth-Morris-Pratt (KMP), algoritma Boyer Moore, dan Regular Expression. Pada program sederhana yang telah dibuat, pencocokan string adalah dengan algoritma Boyer Moore. Untuk menilai program yang telah dibuat, dilakukan beberapa pengujian dengan beberapa dokumen yang berbeda.**

Kata kunci - Pencocokan string, Plagiarisme, Mendeteksi

I. PENDAHULUAN

Teknologi pada era digital seperti sekarang berkembang dengan sangat pesat. Berbagai informasi dapat diperoleh dengan sangat mudah. Hal ini tentu membawa keuntungan bagi kita, namun selain keuntungan, kita juga dapat memperoleh kerugian, salah satunya adalah plagiarisme. Plagiarisme adalah suatu tindak kejahatan akademik karena di dalamnya terdapat unsur pencurian berupa pencurian ide-ide dan gagasan tanpa mencantumkan sumber aslinya. Hal ini sangatlah bertentangan dengan prinsip pendidikan yang ingin menciptakan sumber daya manusia yang berilmu dan berakhlak mulia.

Untuk mengatasi hal ini, dibuatlah suatu aplikasi, baik itu *desktop application* maupun *web based application*, yang dapat mendeteksi plagiarisme dari suatu dokumen serta menampilkan tingkat plagiarisme dari dokumen tersebut. Adapun yang dilakukan aplikasi ini adalah mencocokkan setiap kalimat yang terdapat pada dokumen dengan kalimat pada dokumen - dokumen lain yang telah / pernah di *publish*. Pada saat ini sudah banyak aplikasi yang dapat mendeteksi plagiarisme ini, contohnya : Copyscape, DupliChecker, Dustball, PlagTracker, dan plagiarismchecker.co.

Pada makalah ini, penulis akan membahas mengenai konsep Pencocokan String (String Matching) dan penerapannya pada aplikasi pendeteksi plagiarisme sederhana, yang mana nantinya konsep - konsep yang telah dijelaskan akan digunakan untuk melakukan pencocokan kata / kalimat yang terdapat pada dokumen yang akan dideteksi. Pada makalah ini, penulis juga akan menjelaskan sedikit tentang program yang telah dibuat beserta uji coba yang telah dilakukan.

II. DASAR TEORI

A. String

String adalah rangkaian atau larik dengan setiap elemennya dalah karakter. Dalam pemograman, string juga dapat disebut sebagai sebuah deret simbol. String merupakan suatu jenis tipe data yang menyimpan barisan karakter, yaitu dapat berupa huruf, spasi, tanda baca, maupun angka.

Misalkan terdapat suatu string S dengan panjang string S adalah sepanjang m satuan. String S tersebut dapat direpresentasi sebagai berikut.

$$S = x_0 x_1 \ldots x_{m-1}$$

Prefix (awalan) dari string S tersebut adalah sebuah himpunan bagian atau substring dengan S[0. . K]. Suffix (akhiran) dari string tersebut adalah sebuah himpunan bagian atau substring dengan S[K. . m−1]. K menunjukkan suatu indeks pada string S yang bernilai antara 0 sampai m-1. Karena prefix dan suffix merupakan himpunan bagian (*inproper subset*) dari S, maka prefix dan suffix ini dapat berupa himpunan kosong ataupun berupa string S itu sendiri.

Sebagai contoh, misal terdapat sebuah string S = “NABILA”.

- S (string) : “NABILA”
- Prefix : “N” , “NA”, “NAB”, “NABI”, “NABILA”, “NABILA”
- Sufix : “A”, “LA”, “ILA”, “BILA”, “ABILA”, “NABILA”

B. Pencocokan String

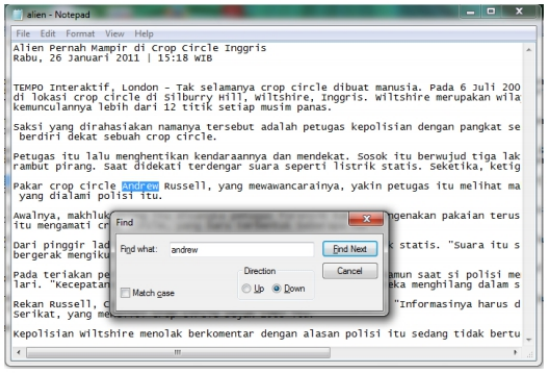
Pencocokan string (*string matching*) adalah suatu algoritma yang digunakan untuk mencari lokasi pertama dalam teks yang bersesuaian dengan pattern. Berdasarkan pengertian yang telah disebutkan, terdapat beberapa istilah, yaitu

- Teks (T) adalah sebuah string yang panjangnya n karakter. Contoh : T: the rain in spain stays mainly on the plain

- Pattern (P) adalah sebuah string yang panjangnya m karakter yang akan dicari di dalam teks, dimana nilai m lebih kecil dibandingkan dengan n (m <<< n). Contoh : P: main

Pencocokan string dapat digunakan dalam pengaplikasian berbagai hal, diantaranya:

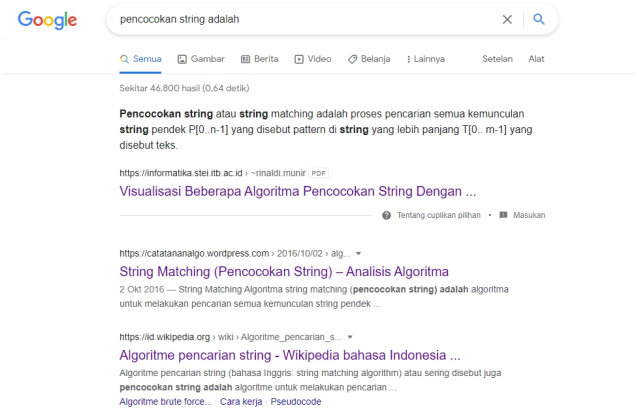
- Pencarian dalam text editor



Gambar 1. Pencarian dalam text editor

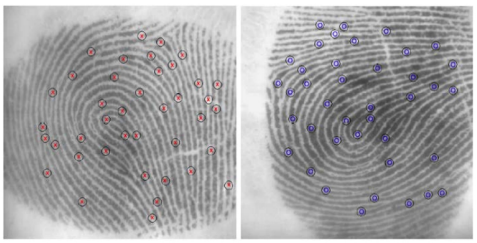
Sumber :https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

- Web search engine. Contohnya Google.



Gambar 2. Pencarian pada Google

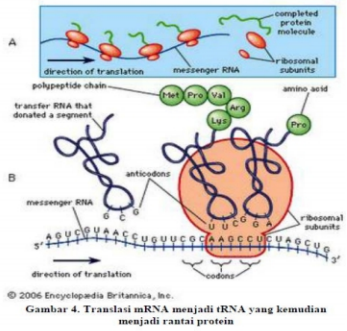
- Analisis Citra (*Fingerprint Recognation*)



Gambar 3. Analisis Citra

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

- Bioinformatics. Contohnya Pencocokan Rantai Asam Amino pada rantai DNA



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

Gambar 4. Pencocokan Rantai Asam Amino

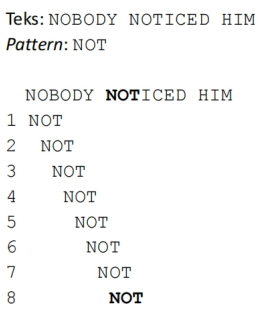
Sumber :https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

C. Algoritma Brute Force

Pencockan string dengan menggunakan algoritma Brute Force akan melakukan pemeriksaan satu - persatu karakter yang terdapat pada Teks T apakah karakter tersebut sesuai (*match*) dengan karakter yang terdapat pada Pattern P, dimulai dari awal karakter Pattern P. Apabila karakter tersebut tidak sama (*missmatch*), maka akan dilakukan pergeseran 1 karakter ke kanan di Teks T dan akan dimulai pencarian lagi dari awal karakter Pattern P.

Adapun tahapan penerapan algoritma Brute Force pada pencocokan string untuk Teks T yang berada di dalam array T[0..n-1] dan Pattern P yang berada di dalam array P[0..m-1] adalah sebagai berikut.

1. Pertama, dilakukan pencocokan T[0] dengan P[0]. Pencocokkan akan dilakukan dari kiri ke kanan.
2. Apabila kedua karakter tersebut sama (*match*) maka akan dilakukan pencocokan untuk karakter berikutnya dengan menambahkan indeks pada T dan P lalu dilakukan pencocokan kembali untuk indeks selanjutnya. Namun, jika kedua karakter tersebut berbeda, maka akan dilakukan pergeseran 1 karakter ke kanan pada Teks T dengan menambahkan indeks pada T dan dicocokkan dengan pattern P dengan indeks 0.
3. Pencocokan ini dilakukan berulang sampai ditemukan pattern P yang cocok dengan teks T atau sudah dilakukan pencocokan sampai dengan indekx terakhir dari teks T (tidak ditemukan pattern P yang cocok pada teks T).



Gambar 5. Contoh Penerapan Algoritma Brute Force

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

Pseudocode untuk melakukan pencocokan string dengan algoritma Brute Force adalah sebagai berikut ini:

```
Procedure BruteForceSearch(
          input m, n : integer,
          input P : array[0..n-1] of char,
          input T : array[0..m-1] of char,
          output ketemu : array[0..m-1] of boolean
        )

Deklarasi:
     i, j: integer

Algoritma:
     for (i:=0 to  m-n) do
          j:=0
          while (j < n and T[i+j] = P[j]) do
               j:=j+1
          endwhile
          if(j >= n) then
                ketemu[i]:=true
          endif
      endfor
```

Untuk kasus terburuk (*worst case*), yaitu dimana karakter awal (prefix) dari Patter P sama (*match*) dengan karakter yang terdapat pada Teks T. Jumlah perbandingannya : m(n - m + 1), sehinga diperoleh kompleksitas waktu : O(mn). Contoh :

- T : aaaaaaaaaaaaaaaaaaaaaaaaah
- P : aaah

Untuk kasus terbaik (*best case*), yaitu ketika karakter pertama dari Pattern P tidak pernah sama dengan karakter yang terdapat pada Teks T, selain dari hasil. Maka, akan diperoleh kompleksitas waktu : O(n). Contoh :

- T : String ini berakhir dengan zzz
- P : zzz

Untuk kasus rata - rata (*average case*), yaitu untuk pencarian yang biasa kita lakukan. Kompleksitas waktu : O(m+n). Contoh :

- T : a string searching example is standard
- P : store

Pendekatan dengan algoritma Brute Force cukup cepat ketika katakter pada T dan P beragam. Contoh : A..Z, a..z, 1..9, dsb. Namun, akan lambat ketika jenis karakter tidak beragam. Contoh : 0,1 (binary files, images files, dsb)

D. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) merupakan salah satu algoritma pencocokan string yang melakukan pencocokan dari kiri ke kanan mirip, seperti algoritma pada pencarian string Brute Force. Namun, terdapat sedikit perbedaan, yaitu pada algoritma Knuth-Morris-Pratt pergeseran yang dilakukan lebih cerdas ketika terdapat perbedaan karakter (*missmatch*). Pergeseran karakternya memanfaatkan konsep suffix dan prefix dengan mempertimbangkan sebanyak mungkin pergeseran yang dapat dilakukan sehingga tidak mengulang pemeriksaan yang sama. Maka dari itu, algoritma KMP lebih efisien daripada algoritma pada Brute Force.

Pada algoritma KMP sebelum dilakukan pencocokan string, KMP melakukan preproses terhadap Pattern P yang akan dicari kecocokannya dengan Teks T, yaitu dengan melakukan proses kalkulasi fungsi pinggiran (The border function). Fungsi pinggiran ini mengindikasikan pergeseran terbesar yang mungkin terjadi dengan menggunakan perbandingan yang dibentuk sebelum pencarian string. Fungsi pinggiran b(k) dapat diperoleh dengan mencari semua nilai prefix terbesar pada P[0..k] yang sama nilainya dengan suffixpada P[1..k]. Nilai k merupakan posisi pada Pettern P sebelum posisi missmatch, misalkan terjadi missmatch pada index j, maka nilai k = j - 1. Fungsi pinggiran (the border function) bisa juga disebut sebagai failure function. Penghitungan border function ini dilakukan ketika program menerima inputan Pattern P.

Setelah membuat fungsi pinggiran, barulah dilakukan prncocokan string antara Pattern P dan Teks T, prosesnya mirip dengan Brute Force. Namun, disaat menemukan mismatch atau ketidakcocokan antara katakter pada Pattern P dan karakter pada Teks T dilakukan pergeseran sejumlah yang sesuai pada fungsi pinggiran yang telah dibuat. Jika nilai fungsi pinggitan b(k) = -, maka akan dilakukan pergeseran 1 karakter ke kanan, seperti pada algoritma Brute Force.

Pseudocode untuk mencari nilai fungsi pinggiran pada algoritma KMP adalah sebagai berikut ini:

```
procedure preKMP(
    input P : array[0..n-1] of char,
    input n : integer,
    input/output kmpNext : array[0..n] of integer
)
Deklarasi:
i,j: integer

Algoritma
  i := 0;
  j := kmpNext[0] := -1;
  while (i < n) {
     while (j > -1 and not(P[i] = P[j]))
        j := kmpNext[j];
     i:= i+1;
     j:= j+1;
     if (P[i] = P[j])
        kmpNext[i] := kmpNext[j];
     else
        kmpNext[i] := j;
     endif
  endwhile
```

Kemudian untuk melakukan pencocokan string pada pattern P dan Teks T, pseudocode-nya :

```
procedure KMPSearch(
    input m, n : integer,
    input P : array[0..n-1] of char,
    input T : array[0..m-1] of char,
    output ketemu : array[0..m-1] of boolean
)

Deklarasi:
i, j,next: integer
kmpNext : array[0..n] of interger

Algoritma:
preKMP(n, P, kmpNext)
i:=0
while (i<= m-n) do
    j:=0
    while (j < n and T[i+j] = P[j]) do
       j:=j+1
    endwhile
    if(j >= n) then
       ketemu[i]:=true;
    endif
    next:= j - kmpNext[j]
    i:= i+next
endwhile
```

Berikut ini contoh penerapan algoritma KMP :

- T : abacaabaccabacabaabb
- P : abacab

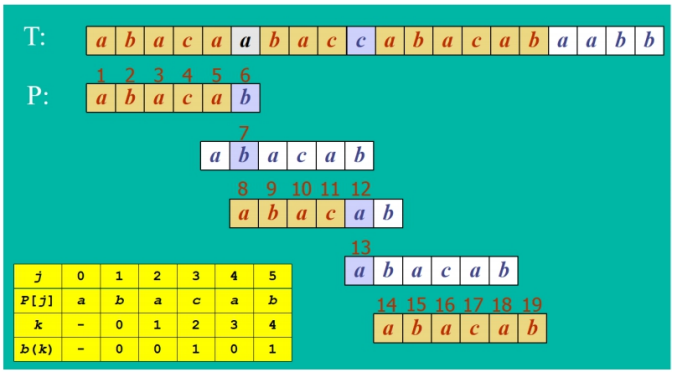
Pertama, dilakukan perhitungan fungsi pinggiran terhadap Pattern P. Pada contoh akan diperoleh nilai fungsi pinggiran sesuai pada tabel berikut.

j	0	1	2	3	4	5
P[j]	a	b	a	c	a	b
k	-	0	1	2	3	4
b(k)	-	0	0	1	0	1

Gambar 6. Nilai Border Function

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

Kemudian, barulah dilakukan pencocokan string antara Pattern P dan Teks T. Pada contoh akan dihasilkan proses pencocokan sebagai berikut.



Gambar 7. Contoh Penerapan Algoritma KMP

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

Kompleksitas waktu menghitung fungsi pinggiran adalah O(m) dan untuk pencocokan string adalah O(n). Maka dari itu, diperolehlah kompleksitas waktu algoritma Knuth-Morris-Pratt (KMP) adalah O(m+n).

Keuntungan algoritma KMP adalah tidak pernah bergerak "mundur" (mengulang pemeriksaan) seperti pada algoritma Brute Force sehingga algoritma KMP ini cocok untuk file berukuran besar.

Kerugian algoritma KMP adalah ketika variasi karakter Teks beragam, akan sering terjadi missmatch sehingga pencariannya akan mirip seperti algoritma Brute Force.

E. Algoritma Boyer Moore

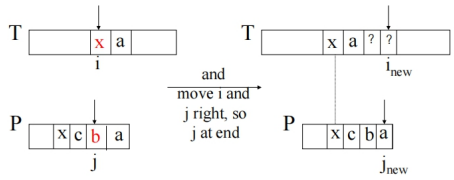
Berbeda dengan algoritma KMP dan Brute Force, Algoritma Boyer Moore merupakan salah satu algoritma pencocokan string yang melakukan pencocokan dari kanan ke kiri. Terdapat dua teknik yang mendasari pencocokan string dengan algoritma Boyer Moore antara lain :

1. The Looking-glass Technique
   Teknik ini memeriksa kecocokan Pattern P dengan Teks T dimulai dari index terakhir (kanan) P. Pemeriksaan terhadap Teks T tetap dari awal, index I akan dimulai pada index nilai m - 1 (m adalah panjang Pattern P).
2. The character-jump technique
   Teknik ini diterapkan ketika terjadi missmatch (P[j] ≠ T[i] dengan T[i] = x).

   Ada 3 kasus yang mungkin antara lain sebagai berikut:

1. Kasus I
   Jika terdapat 'x' di P dengan index yang lebih kecil dari j (di kiri), maka seolah - olah geser P ke kanan agar posisi 'x' di T[i] sejajar dengan posisi kemunculan terakhir (lo) 'x' di P.

$$i_{new} = i + (m - 1) - lo$$



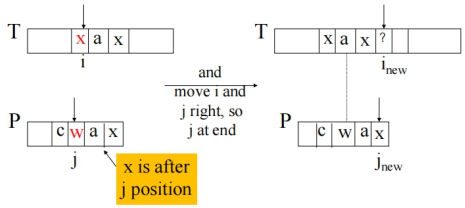
Gambar 8. Case 1 Character-jump Technique

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

2. Kasus II

Jika terdapat 'x' di P, pada posisi index lebih besar dari j (di kanan) maka seolah - olah digeser P sebanyak 1 karakter ke kanan, agar posisi index terakhir P sejajar dengan sehingga sejajar dengan T[i+1].

$$i_{new} = i + m - j$$



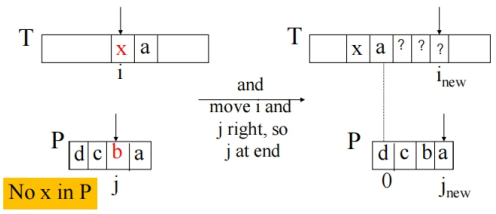
Gambar 9. Case 2 Character-jump Technique

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

3. Kasus III

Jika kasus satu dan dua tidak muncul saat missmatch ('x' tidak terdapat pada P), maka seolah - olah geser P agar posisi index awal P (P[0]) sejajar dengan T[i+1].

$$i_{new} = i + m$$



Gambar 10. Case 3 Character-jump Technique

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

Pseudocode untuk melakukan pencocokan string dengan algoritma Boyer Moore adalah sebagai berikut ini:

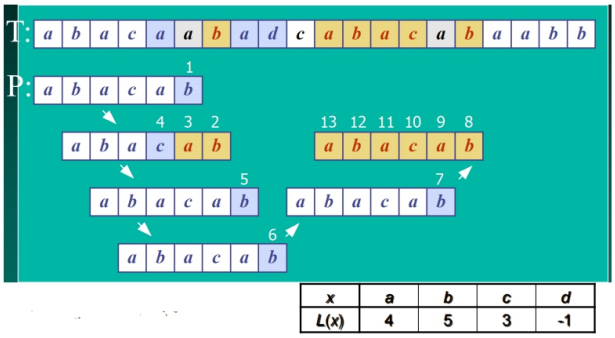
```
Procedure preBmBc(
   input P : array[0..n-1] of char,
   input n : integer,
   input/output bmBc : array[0..ASIZE-1] of integer
)

Deklarasi:
   i: integer

Algoritma:
   for (i := 0 to ALPHABETSIZE-1)
      bmBc[i] := n
   endfor
   for (i := 0 to n - 2)
      bmBc[P[i]] := n - i - 1
   endfor
```

```
Procedure BoyerMooreSearch(
   input m, n : integer,
   input P : array[0..n-1] of char,
   input T : array[0..m-1] of char,
   output ketemu : array[0..m-1] of boolean
)

Deklarasi:
   i, j, shift, bmBcShift, bmGsShift: integer
   BmBc : array[0..ALPHABETSIZE] of integer
   BmGs : array[0..n-1] of integer

Algoritma:
   preBmBc(n, P, BmBc)
   preBmGs(n, P, BmGs)
   i:=0
   while (i<= m-n) do
      j:=n-1
      while (j >=0 n and T[i+j] = P[j]) do
         j:=j-1
      endwhile
      if(j < 0) then
         ketemu[i]:=true
         shift := bmGs[0]
      else
         bmBcShift:= BmBc[chartoint(T[i+j])]-n+j+1
         bmGsShift:= BmGs[j]
         shift:= max(bmBcShift, bmGsShift)
      endif
      i:= i+shift
   endwhile
```

- Last Occurrence Function (L(x))

Pada algoritma Boyer Moore, juga dilakukan preproses terlebih dahulu sebelum melakukan pencocokan string. Preprosesnya yaitu menghitung fungsi Last Occurrence dengan menentukan posisi kemunculan terakhir semua karakter yang ada pada Teks T di dalam Pattern P. Jika suatu karakter pada T tidak muncul pada P, maka nilai fungsi L(x) untuk karakter tersebut adalah -1.

Berikut ini contoh penerapan algoritma Boyer Moore :



Gambar 11. Contoh Penerapan Algoritma Boyer Moore

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

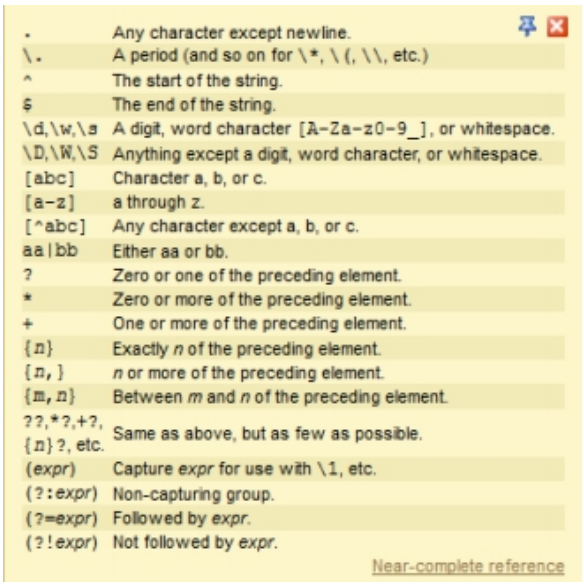
Kompleksitas waktu menghitung fungsi kemunculan terakhir (last occurrence) adalah O(A). Kompleksitas waktu pencocokan string dengan algoritma Boyer Moore dengan kasus terburuk adalah O(nm). Maka dari itu, diperolehlah kompleksitas waktu total pencocokan string dengan algoritma Boyer Moore adalah O(nm + A).

F. Regular Expression

Regular Expression atau sering disingkat sebagai Regex merupakan sekumpulan karakter yang digunakan untuk mendefinisikan Pola (pattern). Pattern ini digunakan untuk pencarian suatu informasi dari suatu text atau kalimat maupun untuk melakukan manipulasi data.

Terdapat beberapa manfaat dari Regular Expression ini dalam pemrograman, antara lain :

- Regular Expression untuk validasi data
- Regular Expression untuk pencarian
- Regular Expression untuk find and replace



.	Any character except newline.
\.	A period (and so on for *, \(, \\, etc.)
^	The start of the string.
$	The end of the string.
\d,\w,\s	A digit, word character [A-Za-z0-9_], or whitespace.
\D,\W,\S	Anything except a digit, word character, or whitespace.
[abc]	Character a, b, or c.
[a-z]	a through z.
[^abc]	Any character except a, b, or c.
aa|bb	Either aa or bb.
?	Zero or one of the preceding element.
*	Zero or more of the preceding element.
+	One or more of the preceding element.
{n}	Exactly *n* of the preceding element.
{n,}	*n* or more of the preceding element.
{m,n}	Between *m* and *n* of the preceding element.
??,*?,+?, {n}?, etc.	Same as above, but as few as possible.
(expr)	Capture *expr* for use with \1, etc.
(?:expr)	Non-capturing group.
(?=expr)	Followed by *expr*.
(?!expr)	Not followed by *expr*.

Near-complete reference

Gambar 12. Syntax Regular Expression

Sumber :https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf

G. Plagiarisme

Berdasarkan Kamus Besar Bahasa Indonesia (KBBI), plagiarisme dapat diartikan sebagai penjiplakan yang melanggar hak cipta. Plagiarisme atau sering disebut plagiat adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah - olah karangan dan pendapat sendiri. Plagiarisme merupakan tindak pidana karena mencuri hak cipta orang lain. Di dalam dunia pendidikan, pelaku plagiarisme ini akan mendapatkan hukuman berat, seperti dikeluarkan dari sekolah/universitas. Pelaku plagiat disebut sebagai plagiator. Plagiarisme tidak hanya mengacu pada hasil karya tulisan saja, namun juga hasil karya musik, desain, dsb.

Plagiarisme dalam literatur terjadi ketika seseorang mengaku sebagai penulis asli suatu naskah yang milik orang lain, atau mengambil mentah-mentah hasil tulisan atau karya orang lain atau karya sendiri secara keseluruhan maupun sebagian, tanpa mencantumkan sumber karya tersebut.

Berdasarkan kutipan pada lib.ugm.ac.id, menurut Soelistya (2011) plagiarisme ada beberapa jenis, diantaranya :

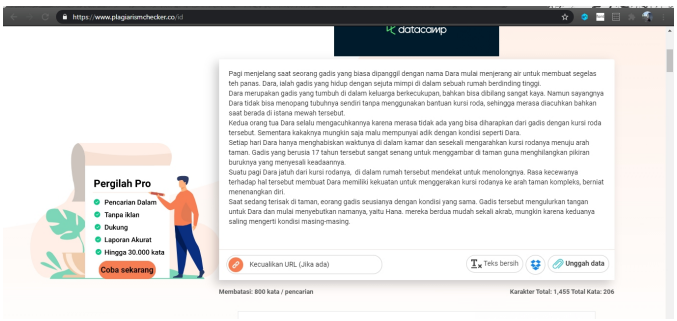
1. Plagiarisme Kata (Word for word Plagiarism)
   Ketika seseorang menggunakan kata-kata penulis lain (persis) tanpa menyebutkan sumbernya.
2. Plagiarisme Sumber (Plagiarism of Source)
   Ketika seseorang menggunakan gagasan orang lain tanpa memberikan pengakuan yang cukup (tanpa menyebutkan sumbernya secara jelas).
3. Plagiarisme Kepengarangan (Plagiarism of Authorship)
   Ketika seseorang mengaku sebagai pengarang karya tulis karangan orang lain.
4. Self Plagiarism
   Ketika seseorang mempublikasikan satu artikel pada lebih dari satu redaksi publikasi, kemudian mendaur ulang karya tulis/ karya ilmiah tersebut.

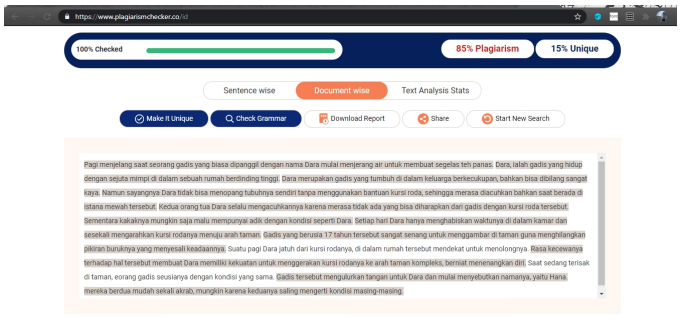
III. IMPLEMENTASI DAN UJI KASUS

Pada saat sekarang sudah banyak aplikasi yang dapat digunakan untuk mendeteksi plagiarisme, baik itu berupa *desktop application* maupun *web based application*. Adapun beberapa contoh aplikasi tersebut, yaitu Copyscape, DupliChecker, Dustball, PlagTracker, plagiarismchecker.co, dsb.

Penggunaan plagiarismchecker.co untuk mendeteksi tingkat plagiarisme suatu dokumen dapat dilakukan dengan beberapa tahapan :

1. Masukkan file yang akan di cek pada kolom yang telah di sediakan
2. Tekan tombol “Cek Plagiat”
3. Kemudian akan ditampilkan tingkat plagiarisme dari file yang telah kita inputkan, serta akan ditampilkan isi file tadi yang telah ditandai bagian yang merupakan plagiat.





Gambar 12. Contoh Pengecekan Plagiarisme Pada plagiarismchecker.co

Sumber : https://www.plagiarismchecker.co/id

Pada makalah ini, saya membuat suatu program plagiarism checker yang sangat sederhana, dimana program ini akan membandingkan 2 buah dokumen Teks. Untuk setiap kalimat yang terdapat pada kedua dokumen, akan dimasukkan ke dalam suatu array. Untuk melakukan pengecekan plagiarisme, dilakukan iterasi array dokumen1 dan array dokumen2, dimana setiap kalimat yang ada pada dokumen1 akan dicek kecocokannya dengan setiap kalimat yang pada dokumen2. Pengecekan ini menggunakan algoritma pencocokan string Boyer Moore.

Pertama - tama, program akan meminta input nama file 1 dan file 2. File 1 adalah file yang akan dicek tingkat plagiarismenya, sedangkan file 2 adalah file pembanding dari file 1.

```
PS D:\KULIAH\Stima\tubes3> python -u "d:\KULIAH\Stima\tubes3\checker.py"
Masukkan nama file 1 : file1.txt
Masukkan nama file 2 : file2.txt
```

Gambar 13. Contoh Input File pada Program

Setelah itu, program akan membaca isi kedua file tersebut. Kemudian, semua kalimat yang ada pada file akan dimasukkan ke dalam sebuah string yang berbeda. Lalu, setiap kalimat yang ada pada string tersebut akan dipisahkan dan dimasukkan ke dalam suatu array. Berikut ini potongan programnya :

```
# Read file
file1=open(namafile1,"r")
text1=file1.readlines()

file2=open(namafile2,"r")
text2=file2.readlines()

# Convert list to string
str1=''.join(text1)
str2=''.join(text2)

# Split the string
sent_text1=str1.split('.')
sent_text2=str2.split('.')
```

Gambar 14. Potongan Program

Untuk menghitung tingkat plagiarisme dari dokumen 1 terhadap dokumen 2 digunakan fungsi tingkatPlagiarisme yang menerima array kalimat dokumen 1 (doc1) dan array hasil pencocokan string (final_list). Tingkat plagiarisme ini didapat dengan membagi panjang array final_list dengan panjang array doc1, kemudian dikali dengan 100.

```
def tingkatPlagiarisme(doc1, final_list) :
    lenArr = len(doc1)
    #print(lenArr)
    lenFinal = len(final_list)
    #print(lenFinal)
    hasil = (lenFinal/lenArr) * 100
    return hasil
```

Gambar 15. Fungsi Mencari Tingkat Plagiarisme

Pada akhir program, akan ditampilkan tingkat plagiarisme dari dokumen 1 beserta isi dokumen 1 yang telah ditandai bagian yang merupakan plagiat, bagian yang plagiat ditulis dengan huruf tebal (bold).

Kode lengkap dari program pengecekan plagiarisme sederhana yang dibuat ini, dapat diakses pada link berikut ini : https://github.com/nabilahann/PlagiarismChecker

A. Uji Coba 1

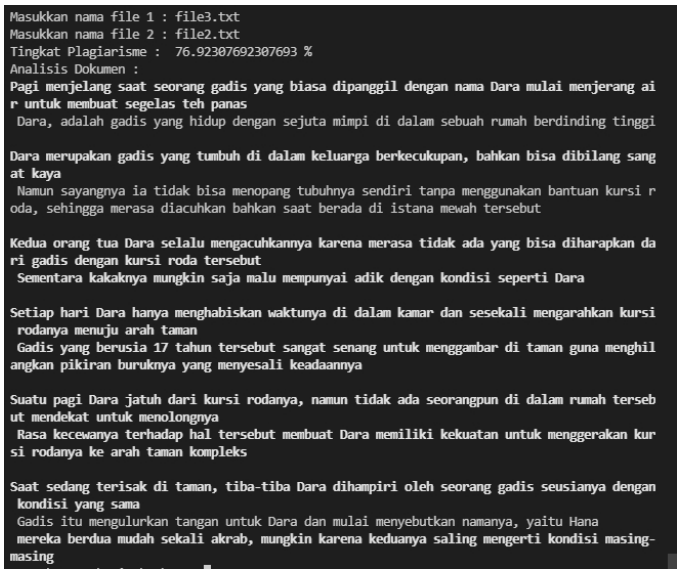
Pada uji coba 1 ini terdapat 2 buah file Teks yang identik (sama), yaitu file1.txt dan file2.txt. Pada saat kedua program ini dimasukkan ke dalam program Plagiarism Checker, akan menghasilkan tingkat plagiarisme 100% dan semua kalimatnya ditulis dengan huruf tebal (kalimat yang merupakan plagiat). Hal ini dapat dilihat pada gambar di bawah.

```
Masukkan nama file 1 : file1.txt
Masukkan nama file 2 : file2.txt
Tingkat Plagiarisme :  100.0 %
Analisis Dokumen :
Pagi menjelang saat seorang gadis yang biasa dipanggil dengan nama Dara mulai menjerang ai
r untuk membuat segelas teh panas
 Dara, ialah gadis yang hidup dengan sejuta mimpi di dalam sebuah rumah berdinding tinggi

Dara merupakan gadis yang tumbuh di dalam keluarga berkecukupan, bahkan bisa dibilang sang
at kaya
 Namun sayangnya Dara tidak bisa menopang tubuhnya sendiri tanpa menggunakan bantuan kursi
 roda, sehingga merasa diacuhkan bahkan saat berada di istana mewah tersebut

Kedua orang tua Dara selalu mengacuhkannya karena merasa tidak ada yang bisa diharapkan da
ri gadis dengan kursi roda tersebut
 Sementara kakaknya mungkin saja malu mempunyai adik dengan kondisi seperti Dara

Setiap hari Dara hanya menghabiskan waktunya di dalam kamar dan sesekali mengarahkan kursi
 rodanya menuju arah taman
 Gadis yang berusia 17 tahun tersebut sangat senang untuk menggambar di taman guna menghil
angkan pikiran buruknya yang menyesali keadaannya
```

Gambar 16. Uji Coba 1

B. Uji Coba 2

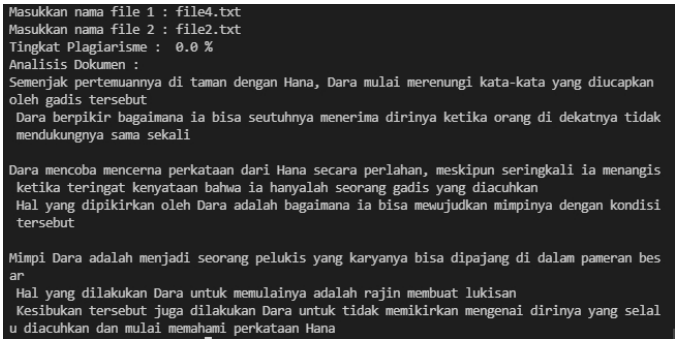
Pada uji coba 2 terdapat 2 buah file Teks yang memiliki sedikit perbedaan, yaitu file3.txt dan file2.txt. Pada saat kedua program ini dimasukkan ke dalam program Plagiarism Checker, akan menghasilkan tingkat plagiarisme kurang dari 100%, yaitu 76.9%, dan sebagian besar kalimatnya ditulis dengan huruf tebal (kalimat yang merupakan plagiat), namun juga ada kalimat yang tidak ditulis dengan huruf tebal (kalimat yang bukan plagiat). Hal ini dapat dilihat pada gambar di bawah.



Gambar 17. Uji Coba 2

C. Uji Coba 3

Pada uji coba 3 terdapat 2 buah file Teks yang berbeda semua kalimatnya, yaitu file4.txt dan file2.txt. Pada saat kedua program ini dimasukkan ke dalam program Plagiarism Checker, akan menghasilkan tingkat plagiarisme 0% dan semua kalimatnya tidak ditulis dengan huruf tebal (kalimat bukan plagiat). Hal ini dapat dilihat pada gambar di bawah.



Gambar 18. Uji Coba 3

IV. KESIMPULAN

Pencocokan string merupakan suatu algoritma yang digunakan untuk mencari lokasi pertama dalam teks yang bersesuaian dengan pattern. Ada beberapa macam algoritma pencocokan string, yaitu algoritma Brute Force, algoritma Knuth-Morris-Pratt (KMP), algoritma Boyer Moore dan Regular Expression. Konsep pencocokan string ini dapat diterapkan dalam berbagai hal, salah satunya adalah untuk mendeteksi plagiarisme. Plagiarisme adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah - olah karangan dan pendapat sendiri. Pada program *simple plagiarism checker* yang telah dibuat, program telah berhasil mendeteksi tingkat plagiarisme dari suatu dokumen terhadap dokumen lainnya, serta berhasil menampilkan tingkat plagiarismenya dan analisis kalimat yang merupakan plagiat.

VIDEO LINK AT YOUTUBE

https://youtu.be/TPSmMYbwfyg

UCAPAN TERIMAKASIH

Segala puji bagi Allah SWT yang telah memberikan penulis kemudahan sehingga dapat menyelesaikan makalah ini dengan tepat waktu. Penulis mengucapkan terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc. selaku dosen mata kuliah Strategi Algoritma kelas K02 yang telah memberi materi untuk penulisan makalah. Tidak lupa juga penulis mengucapkan terima kasih kepada kedua orang tua serta teman-teman yang telah memberi dukungan selama pengerjaan makalah ini. Akhir kata, penulis menyadari masih terdapat kekurangan dan kesalahan kata dalam makalah ini, penulis berharap makalah ini dapat digunakan sebaik-baiknya dan dikembangkan sehingga lebih menghasilkan manfaat bagi masyarakat luas

REFERENCES

[1] Munir, Rinaldi. Pencocokan String. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Diakses tanggal 10 Mei 2021, pukul 13.00 GMT+7.

[2] Munir, Rinaldi. String Matching dengan Regular Expression. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf. Diakses tanggal 10 Mei 2021, pukul 13.00 GMT+7.

[3] https://penerbitdeepublish.com/plagiarisme-adalah/. Diakses tanggal 11 Mei 2021, pukul 09.00 GMT+7.

[4] https://www.plagiarismchecker.co/id. Diakses tanggal 11 Mei 2021, pukul 14.00 GMT+7.

[5] https://kbbi.kemdikbud.go.id/entri/plagiarisme. Diakses tanggal 10 Mei 2021, pukul 10.00 GMT+7.

[6] Stepchyshyn, Vera; Nelson, Robert S. (2007). Library plagiarism policies. Assoc. of College & Resrch Libraries. hlm. 65. ISBN 0838984169.

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.

Padang, 11 Mei 2021

Nabila Hannania - 13519097