Penerapan Algoritma String Matching pada Proses Penyaringan Teks

Avisenna Abimanyu/13517010
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13517010@std.stei.itb.ac.id

***Abstract*—Penerapan algoritma string matching sangatlah luas. Manfaatnya dapat kita rasakan baik secara langsung maupun tidak. Aplikasi-aplikasi yang kita temukan rasanya sudah sangat sulit untuk terlepas dari proses pencocokan kata.**

Keywords—Boyer-Moore,KMP, Algorithm, patter, string

I. PENDAHULUAN

Pada zaman ini teknologi merupakan sesuatu yang sudah bergerak hampir diseluruh aspek kehidupan. Terutama dalam bidang informasi dan komunikasi. Dalam bidang informasi dan komunikasi, teknologi mengubah waktu dan proses dalam menangani informasi-informasi yang tersedia. Efek yang ditimbulkan ini mempunyai dampak positif dan dampak negatif. *Hate speech, spam, inappropriate content*, kata-kata yang tidak pantas merupakan contoh efek negatif dari perkembangan teknologi dan informasi. Untuk itu perlu dilakukan pencegahan dan penanganan untuk menangani efek negatif tersebut.

Proses untuk mengoreksi isi dari suatu konten tertentu secara manual atau menggunakan tenaga manusia memerlukan waktu yang lama, sehingga diperlukan teknik pemecahan masalah dengan pendekatan khusus. Algoritma *Boyer-More* yang selanjutnya disingkat menjadi BM dan algoritma *KMP* merupakan algoritmoa pencocokan string yang berfungsi untuk mencari *pattern* tertentu di dalam teks yang tersedia. Pada proses pembangunannya, algoritma ini banyak dikembangkan lebih lanjut oleh masing-masing institusi untuk memaksimalkan potensi algoritma ini dan menyesuaikan dengan kebutuhan masing-masing.

II. LANDASAN TEORI

A. Pencarian String

Algoritma pencarian string merupakan algoritma pencocokan string. Sifat algortima string matching adalah mencari sebuat string yang terdiri dari beberapa karakter (yang biasa disebuat pattern) di dalam suatu teks. Dengan hal tersebut pencocokan string merupakan permasalahan paling sederhana dari semua permasalahan string lainnya, dan merupakan bagian dari pemrosesan data, pengkompresian data, *lexical analysis*, dan temu balik informasi. Teknik untuk menyelesaikan permasalahan pencocokan string biasanya akan menghasilkan implikasi langsung ke aplikasi string lainnya. Prinsip kerja algoritma *string matching* adalah sebagai berikut:

1. Memindai teks dengan bantuan sebuah window yang ukurannya sama dengan panjang pattern.
2. Menempatkan window pada awal teks.
3. Membandingkan karakter pada window dengan karakter dari pattern. Setelah pencocokan, window di *shift* ke kanan sesuai dengan aturan algoritma *Boyer-More* atau *KMP*. Prosedur ini dilakukan berulang-ulang sampai window berada pada akhir teks. Mekanisme ini disebut mekanisme *sliding-window*.

B. Algoritma Boyer-More

Algoritma Boyer-Moore adalah salah satu algoritme pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977.

Algoritme ini dianggap sebagai algoritme yang paling efisien pada aplikasi umum. Tidak seperti algoritme pencarian string yang ditemukan sebelumnya, algoritme Boyer-Moore mulai mencocokkan karakter dari sebelah kanan pattern. Ide di balik algoritme ini adalah bahwa dengan memulai pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Pada algoritma Boyer-more, pencarian pattern dibagi menjadi 3 kasus, yakni:

1. Huruf teks tidak sesuai dengan huruf pattern, namun huruf teks tersebut terdapat pada pattern pada posisi sebelum huruf teks. Pada kasus ini, dilakukan pergeseran pattern ke kanan sehingga huruf attern tersebut sejajar dengan huruf teks yang terdapat dalam pattern.
2. Huruf teks tidak sesuai dengan huruf pattern dan huruf pattern terdapat pada teks tersebut, namun pergeseran ke kanan tidak dimungkinkan karena posisi huruf pattern terdapat setelah huruf teks tersebut. Pada kasus ini, pattern digeser sebanyak 1 huruf ke kanan.
3. Ketika 2 kasus tersebut tidak berlaku, geser pattern sehingga huruf awal pattern terdapat pada huruf setelah huruf teks yang tidak sesuai dengan pattern.

Untuk mempermudah pergeseran, algoritma Boyer-Moore memiliki sebuah fungsi yang menghitung kemunculan terakhir sebuah kata dalam pattern, yang dinotasikan dalam *L(x)*, dengan x merupakan huruf dalam pattern. Fungsi ini didasarkan oleh alphabet yang disediakan saat pencarian, dengan alphabet yang tidak ada dalam pattern diberikan nilai -1.

Pseudocode algoritma Boyer-Moore pada fase pra-pencarian

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

Algoritme:
 for (i := 0 to ASIZE-1)
   bmBc[i] := m;
 endfor
 for (i := 0 to m - 2)
   bmBc[P[i]] := m - i - 1;
 endfor
procedure preSuffixes(
  input P : array[0..n-1] of char,
  input n : integer,
  input/output suff : array[0..n-1] of integer
)

Deklarasi:
 f, g, i: integer

Algoritme:
 suff[n - 1] := n;
 g := n - 1;
 for (i := n - 2 downto 0) {
   if (i > g and (suff[i + n - 1 - f] < i - g))
     suff[i] := suff[i + n - 1 - f];
   else
     if (i < g)
       g := i;
     endif
     f := i;
     while (g >= 0 and P[g] = P[g + n - 1 - f])
       --g;
     endwhile
     suff[i] = f - g;
   endif
 endfor
```

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

 preSuffixes(x, n, suff);

 for (i := 0 to m-1)
   bmGs[i] := n
 endfor
 j := 0
 for (i := n - 1 downto 0)
   if (suff[i] = i + 1)
     for (j:=j to n - 2 - i)
       if (bmGs[j] = n)
         bmGs[j] := n - 1 - i
       endif
     endfor
   endif
 endfor
 for (i = 0 to n - 2)
   bmGs[n - 1 - suff[i]] := n - 1 - i;
 endfor
```

Pseudocode algoritma Boyer-Moore pada fase pencarian

```
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..255] of interger
BmGs : array[0..n-1] of interger

Algoritme:
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;
   endif
   bmBcShift:= BmBc[chartoint(T[i+j])]-n+j+1
   bmGsShift:= BmGs[j]
   shift:= max(bmBcShift, bmGsShift)
   i:= i+shift
```

C. Algoritma KMP

Algoritma Knuth-Morris-Pratt adalah salah satu algoritme pencarian string, dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, namun keduanya mempublikasikannya secara bersamaan pada tahun 1977.

Jika kita melihat algoritme brute force lebih mendalam, kita mengetahui bahwa dengan mengingat beberapa perbandingan yang dilakukan sebelumnya kita dapat meningkatkan besar pergeseran yang dilakukan. Hal ini akan menghemat perbandingan, yang selanjutnya akan meningkatkan kecepatan pencarian.

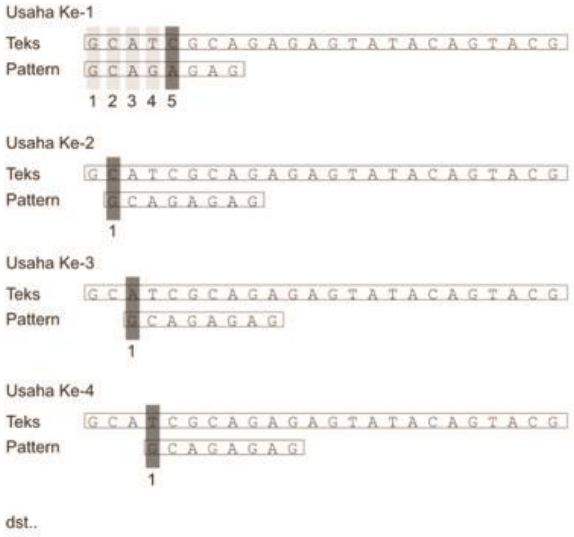
Pseudocode algoritma KMP pada fase pra-pencarian

```
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

Algoritme
 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
```

Pseudocode algoritma KMP pada fase pencarian

```
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
```

```
Algoritme:
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
```



Gambar 1: algoritma brute force

PseudoCode Algoritma Brute Force

```
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

Algoritme:
    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
```

D. Brute Force

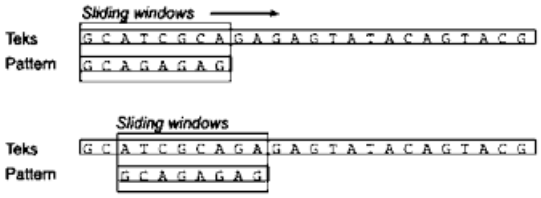
Algoritma bruteforce adalah algoritma pattern matching yang paling sederhana, dan pencarian bruteforce bersifat left-to-right, yakni dari kiri ke kanan. Sebuah pattern akan dicocokkan huruf per huruf dalam sebuah teks, dan ketika sebuah huruf dalam pattern yang ingin dicocokkan tidak sesuai dengan salah satu huruf dalam teks, maka pencarian akan diulang dan dimulai pada huruf teks selanjutnya.

Secara sistematis, langkah-langkah yang dilakukan algoritme brute force pada saat mencocokkan string adalah:

1. Algoritme brute force mulai mencocokkan pattern pada awal teks.
2. Dari kiri ke kanan, algoritme ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
    1. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
    2. Semua karakter di pattern cocok. Kemudian algoritme akan memberitahukan penemuan di posisi ini.
3. Algoritme kemudian terus menggeser pattern sebesar satu ke kanan, dan mengulangi langkah ke-2 sampai pattern berada di ujung teks.

Contoh ilustrasi pattern matching dengan algoritma bruteforce:

III. PERUMUSAN FILTER STRING DI DALAM TEKS



Gambar 2.1: sliding windows

Ada beberapa tahap yang perlu dilakukan sebelum memulai proses filter string terhadap suatu konten tertentu. Diperlukan list kata-kata yang mengandung unsur negatif, dan list tersebut disimpan didalam database. Kemudian teks yang telah tersedia dibaca oleh program dan kemudian program menjalankan algoritma Boyer-Moore atau KMP untuk melakukan pengecekan terhadap teks tersebut. Apabila ditemukan kata-kata yang disimpan di dalam database ataupun sinonimnya, maka program akan menghapus kata tersebut. Program juga menyimpan jumlah kata yang telah dihapusnya.

A. Database

Database menyimpan kata-kata yang akan dihapus dari teks. Dalam proses pembangunannya dapat menggunakan beragam DML. Untuk aplikasi yang tidak berskala besar, tempat penyimpanan database bisa di file eksternal.

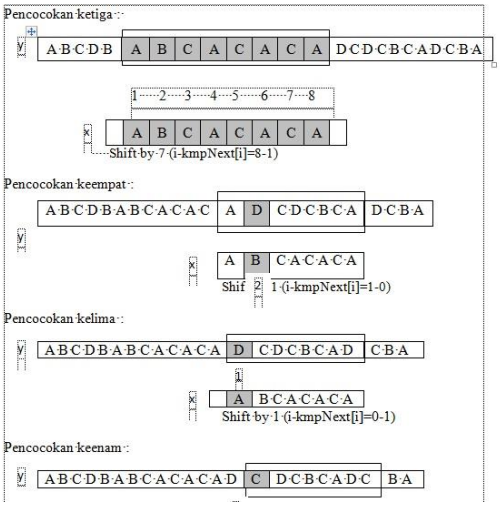
ID	KATA

B. Program and Algorithm

Algoritma akan menerima input teks dan kemudian melakukan pencocokan string dengan pattern yang berasal dari dalam database beserta sinonimnya.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Teks	A	C	T	C	T	C	C	A	T	G	A	T	T	A	G	T	C	A	C	T	C	C	A	C	T	A	T	C	C	T	A
i=8	T	A	G	T	C	A	C	T	C																						
i=9		T	A	G	T	C	A	C	T	C																					
i=15								T	A	G	T	C	A	C	T	C															
i=16									T	A	G	T	C	A	C	T	C														
i=20													T	A	G	T	C	A	C	T	C										

Gambar 1: proses algoritma boyer-moore



Gambar 2.2: proses algoritma KMP

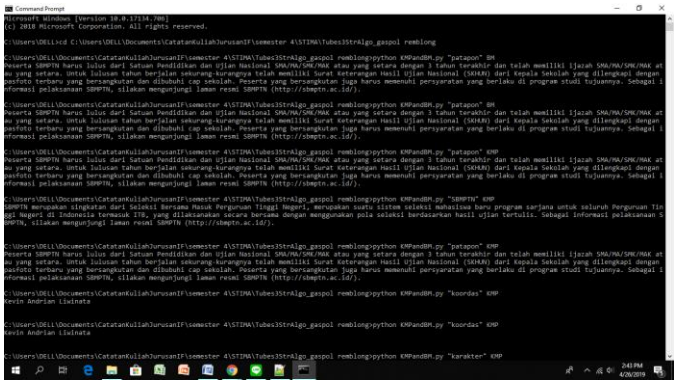
IV. PENYELESAIAN FILTER TEKS DENGAN PROGRRAM

Program akan membaca input teks dan membaca input pattern dari database. Kemudian melakukan pencocokan string. Cara program mencocokan string berbeda tergantung dari algoritma yang dipakainya.

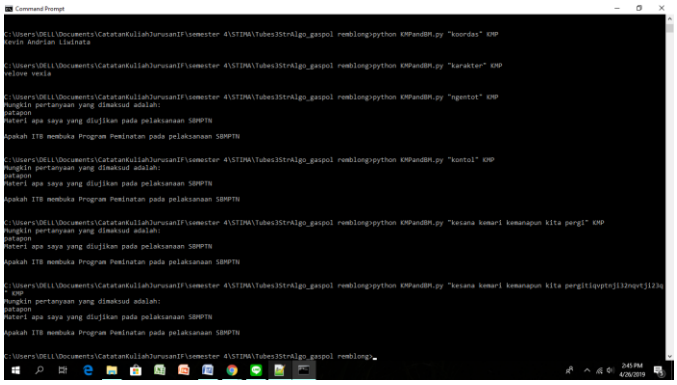
Apabila program menggunakan algoritma KMP maka program akan melakukan pergeseran dengan berdasar pada prefix dan suffix dari pattern untuk pergeserannya. Apabila kata tertentu ditemukan, maka kata tersebut akan ditandai dan kemudian dicatat ke dalam variable ‘jumlahkata’. Setelah pengecekan berakhir apabila ada kata yang ditandai, maka kata tersebut akan dihapus.

Apabila program menggunakan algoritma BM maka program akan melakukan pergeseran dengan berdasar pada kejadian ketika huruf tidak sama di teks dan pattern, terhadap keberadaan huruf teks yang sedang di cek di dalam pattern.

Berikut adalah contoh penggunaan algoritma KMP dan BM dalam pencarian pattern di dalam suatu teks.



Gambar 3:contoh eksekusi program



Gambar 4: contoh eksekusi program

References

[1] Diktat Rinaldi Munir

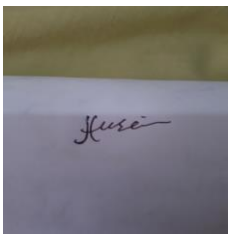
[2] Researchgate,net

[3] Stei.kuliah.itb.ac.id

[4] Informatika.stei.itb.ac.id

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, 29 April 2012
Avisenna Abimanyu