Penerapan Pencocokan String pada Aplikasi Kamusku Indonesia

Reno Rasyad - 13511045[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*reno.rasyad@students.itb.ac.id*

***Abstrak*—Di era *digital* seperti sekarang ini, banyak sekali informasi yang berlalu-lalang. Kita sebagai masyarakat harus bisa memilah secara cerdas informasi yang kita dapatkan dan tidak boleh langsung percaya pada satu sumber. Begitu pula dengan isi dari informasi tersebut, apabila terdapat suatu istilah yang khusus, sangat dianjurkan untuk membuka kamus agar kita tidak salah mengartikan. Di zaman sekarang sudah banyak tersedia kamus *digital* sehingga kita tidak perlu repot-repot membawa buku kamus yang tebal. Untuk mempercepat pencarian kata di kamus *digital*, tentu pengembang kamus tersebut harus menggunkan algoritma String Matching**

***Kata Kunci*—String, Matching, Kamus, Pencarian.**

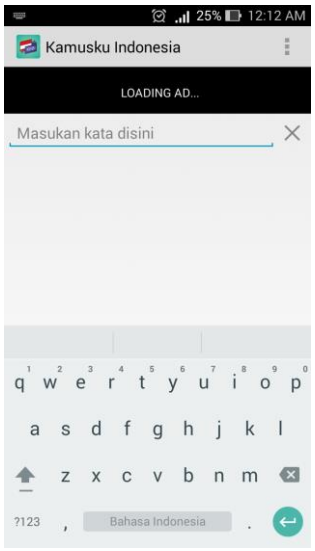
I. PENDAHULUAN

Informasi yang demikian banyak dan disertai dengan istilah-istilah khusus sudah menjadi makanan sehari-hari masyarakat sekarang, sayangnya banyak dari kita yang bersikap reaktif dan menyebarkan berita tanpa tahu substansi dan kebenaran berita tersebut seperti apa.

Saat ini, hampir semua orang memiliki *smartphone* masing-masing dan berbagai aplikasi dapat di pasang pada *smartphone* tersebut. Begitu pula aplikasi kamus yang banyak dikembangkan oleh para *mobile developer* untuk memudahkan pengguna *smartphone* mencari definisi dari sebuah kata yang tidak dimengerti.

1.1 Aplikasi Kamusku Indonesia

Kamusku Indonesia merupakan aplikasi yang berjalan pada sistem operasi Android. Aplikasi ini merupakan Kamus Besar Bahasa Indonesia versi *digital* yang bisa digunakan meskipun pengguna sedang dalam keadaan *offline* karena mempunyai basis data local. Aplikasi ini dikembangkan oleh Kodelokus, pengembang asal Kota Bandung.



Gambar 1: Aplikasi Duolingo saat dijalankan di platform Android

Dalam makalah ini akan dibahas penerapan pencocokan string pada aplikasi Kamusku Indonesia ini

II. DASAR TEORI

Banyak sekali algoritma yang bisa dipakai untuk mencocokkan sebuah string, seperti Algoritma Aho-Corasick, Rabin-Karp, Boyer-Moore-Horspool, Knuth-Morris-Pratt, dan banyak lainnya. Akan tetapi, algoritma yang akan dibahas di makalah ini adalah algoritma yang diajarkan pada saat perkuliahan, yaitu Brute Force, Boyer-Moore dan Knuth-Morris-Pratt.

2.1 Algoritma Brute Force

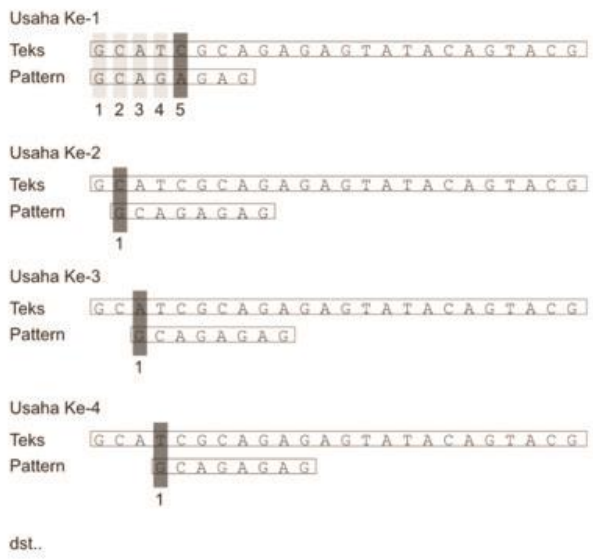
Seperti yang sudah kita ketahui, algoritma brute force merupakan algoritma yang ditulis tanpa memikirkan kinerja atau performa dari sebuah sistem. Begitu pula dalam hal pencocokan string ini, performa tidak dipermasalahkan oleh brute force, algoritma ini akan terus mencari sampai string tersebut ditemukan, atau stringnya sudah habis.

Cara kerja algoritma brute force saat pencocokan string adalah sebagai berikut:

1. Algoritma brute force mulai mencocokkan pattern

pada awal teks.

2. Dari kiri ke kanan, algoritma brute force akan mencocokkan karakter di pattern dengan karakter pada teks yang bersesuaian, sampai salah satu kondisi dibawah ini terpenuhi:
   1. Karakter di pattern dan di teks yang dibandingkan tidak cocok
   2. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan posisi ini.
3. Setelah salah satu kondisi diatas terpenuhi, pattern akan digeser ke kanan dan mengulangi langkah 2, dan berhenti saat mencapai ujung teks.



Gambar 2: Ilustrasi Pencocokan String Menggunakan Algoritma Brute Force

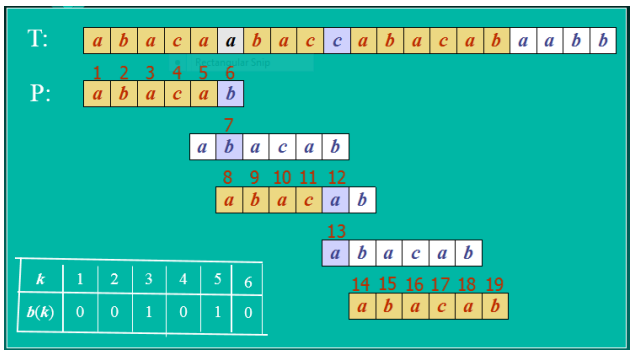
Berikut adalah pseudocode dari algoritma Brute Force

```
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

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
```

2.2 Algoritma Knuth-Morris-Pratt

Kita sudah mengetahui bahwa pencarian brute force dilakukan dari kiri ke kanan sampai teks habis, tanpa memikirkan performansi. Sekarang kita akan membahas algoritma yang lebih baik yaitu Knuth-Morris-Pratt atau biasa disingkat KMP.

Teknik algoritma Knuth-Morris-Pratt mirip dengan teknik brute force yaitu pencarian dilakukan dari kiri ke kanan. Akan tetapi, pergeseran yang dilakukan oleh algoritma Knuth-Morris-Pratt lebih efektif. Pergeseran dilakukan dengan mencari sebuah sub string yang sama di awal dan di akhir pattern. Saat pattern dan teks tidak cocok, maka akan dilakukan pergeseran dengan memperhitungkan border function, sehingga pengecekan yang percuma seperti pada algoritma brute force tidak akan dilakukan. Border function berguna untuk menghitung letak suatu urutan karakter dimana perbandingan harus dilakukan. Border function dihitung dengan cara menghitung panjang prefix yang ada di sebuah pattern, yang sama dengan suffixnya.



Gambar 3: Ilustrasi Pencocokan String Menggunakan Algoritma KMP

```
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
```

Diatas adalah pseudocode algoritma KMP pada fase pra-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

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
```

Diatas adalah pseudocode dari algoritma KMP.

2.3 Algoritma Boyer-Moore

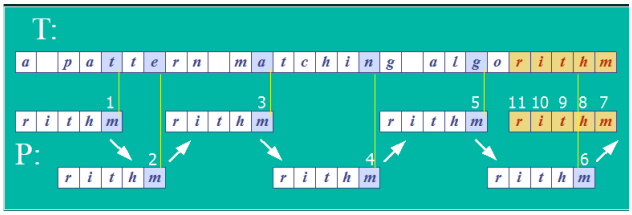
Algoritma Boyer-Moore adalah salah satu algoritma yang menurut banyak orang paling efisien pada aplikasi umum. Tidak seperti algoritma pencarian string yang lain, algoritma Boyer-Moore mulai mencocokkan karakter dari sebelah kanan karakter. Dengan mencocokkan karakter dari kanan, akan lebih banyak informasi yang didapat.

Algoritma Boyer-Moore adalah algoritma yang dilakukan dengan menerapkan 2 teknik, yaitu:

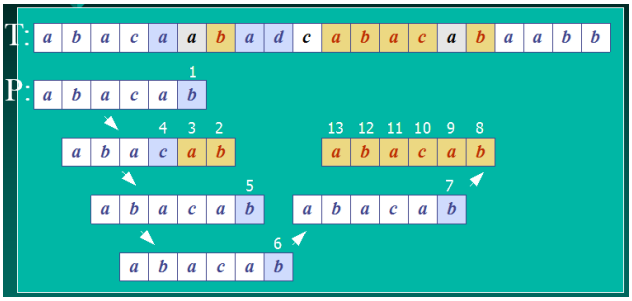
1. Teknik looking-glass
2. Teknik Character-jump

Teknik Looking-glass adalah teknik yang mencari suatu P di dalam kata T dengan mengecek secara mundur melalui P, mulai dari karakter yang terakhir. Sedangkan teknik character jump adalah teknik pelompatan ke target pengecekan berikutnya. Ada 3 kasus yang mungkin:

1. Jika pattern P mengandung x, maka pindahkan P ke kanan sehingga sebaris dengan kemunculan terakhir dari x di P dengan T[i]
2. Jika P mengandung x di suatu tempat, tetapi pergeseran tidak dimungkinkan maka geser P ke kanan sebanyak 1 karakter ke T[i+1]
3. Jika kasus 1 dan 2 tidak dilakukan, maka geser P agar membuat p[1] sebaris dengan T[i+1]



Gambar 4: Ilustrasi Pencocokan String Matching Menggunakan Algoritma Boyer-Moore



Gambar 5: Ilustrasi Pencocokan String Matching Menggunakan Algoritma Boyer-Moore 2

Berikut adalah contoh pseudocode algoritma boyer-moore 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

Algoritma:
  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

Algoritma:
  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
```

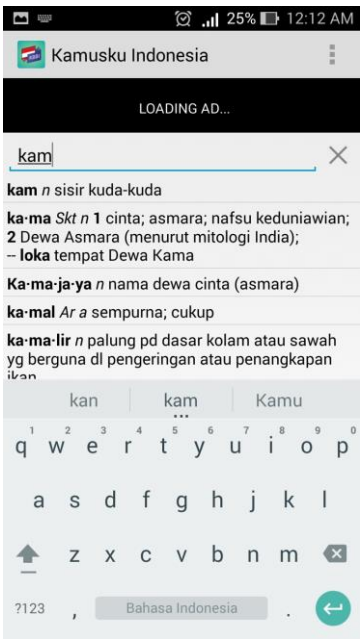
```
procedure preBmGs(
    input P : array[0..n-1] of char,
    input n : integer,
    input/output bmBc : array[0..n-1] of
integer
)
Deklarasi:
  i, j: integer
  suff: array [0..RuangAlpabet] of integer

  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
```

Berikut adalah pseudocode dari algoritma boyer-moore

```
  public static int bmMatch(String text,
   String pattern)
   {
   int     last[]     =     buildLast(pattern);
//mengembalikan index
  //dari last occurence tiap ASCII char di
pattern

   int n = text.length();
   int m = pattern.length();
   int i = m-1;
   if (i > n-1)
   return -1; // no match if pattern is
   // longer than text

   int j = m-1;
   do {
   if (pattern.charAt(j) == text.charAt(i))
   if (j == 0)
   return i; // match
   else { // looking-glass technique
   i--;
   j--;
   }
   else { // character jump technique
   int lo = last[text.charAt(i)]; //last occ
   i = i + m - Math.min(j, 1+lo);
   j = m - 1;
   }
   } while (i <= n-1);

   return -1; // no match
 } // end of bmMatch()
```

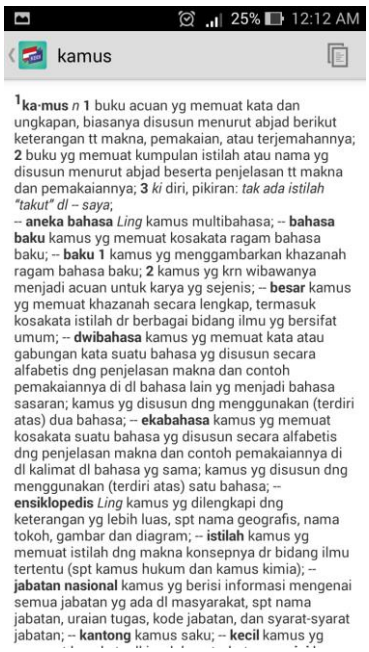
III. ANALISIS

Aplikasi kamusku menerima *input* dari pengguna dan mencari *input* tersebut secara *real-time*



Gambar 2. Pencarian Kata pada Aplikasi Kamusku Indonesia

Untuk mencari kata ‘kamus’ maka pengguna mengetikkan ‘kam’ pada kolom pencarian dan aplikasi akan menampilkan seluruh kata yang berawalan ‘kam’



Gambar 3. Hasil pencarian setelah dipilih

Setelah kata yang dimaksud ditemukan, maka pengguna bisa memilih kata tersebut untuk melihat definisi sesuai dengan yang tertera di Kamus Besar Bahasa Indonesia

IV. KESIMPULAN

Aplikasi Kamusku Indonesia adalah contoh sederhana sebuah perangkat lunak yang menerapkan algoritma String Matching. Dengan basis data yang besar, maka dibutuhkan pula algoritma pencarian yang sangat efektif. Karena pencarian pada kamus dilakukan secara *real-time* dan tulisan dimulai dari sebelah kiri, maka pencarian akan lebih cepat jika menggunakan algoritma KMP karena pencarian dilakukan dari awal *string*.

REFERENSI

[1] Munir, Rinaldi, Diktat Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika Institut Teknologi Bandung, 2009.
[2] Lecroq, Thierry Charras, Christian. 2001. Handbook of Exact String Matching Algorithm.

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, 4 Mei 2015

Reno Rasyad (13511045)