Pattern Matching dalam Aplikasi Pencarian Jodoh

Dini Lestari Tresnani - 13508096
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18096@students.if.itb.ac.id

***Abstract*—Pencarian jodoh bisa merupakan hal yang memusingkan bagi beberapa orang. Namun bukan rahasia umum jika sebagian besar pasangan di dunia ini memiliki biasanya adalah pasangan dengan memiliki banyak kesamaan satu sama lain. Mulai dari hobi, film kesukaan, makanan kesukaan, bahkan kisah hidup dan masa lalu. Sementara pencarian jodoh merupakan masalah bagi banyak orang, teknologi semakin berkembang menjadi pemecahan berbagai masalah baik yang besar maupun yang kecil. Sehingga banyak bermunculan aplikasi pencarian jodoh. Aplikasi pencarian jodoh ini menggunakan algoritma pattern matching sebagai dasar dari aplikasi tersebut.**

***Index Terms*—pencarian jodoh, pattern matching.**

I. PENDAHULUAN

Jodoh merupakan hal yang sangat diidam-idamkan oleh setiap orang di dunia. Dan hidup tanpa menemukan jodoh adalah mimpi buruk setiap orang di dunia. Namun pencarian jodoh ini merupakan sebuah masalah yang merepotkan bagi beberapa orang karena tidak setiap orang dapat menemukan jodohnya dalam sekali coba.

Tidak butuh teori khusus untuk membuktikan bahwa hampir setiap pasangan di dunia menjadi sebuah pasangan karena memiliki sebuah kesamaan. Baik itu kesamaan dalam hal hobi, makanan favorit, sejarah hidup, tempat favorit, film favorit, dll.

Mencari jodoh dapat membuat seseorang gila, depresi, putus asa, dan reaksi mental negatif lainnya. Sudah banyak cerita yang membuktikan hal ini.

Karena itu beberapa orang mencoba untuk membantu orang-orang yang belum menemukan jodoh dengan membuat aplikasi pencarian jodoh. Aplikasi ini menggunakan konsep teknologi yang sekarang ini sedang berkembang dengan pesatnya. Konsep dasar cara kerja aplikasi ini sebenarnya simpel. Aplikasi akan mencari lawan jenis (atau mungkin sesama jenis bagi beberapa orang) yang memiliki kesamaan. Semakin banyak kesamaannya, semakin besar kemungkinan berjodohnya.

Namun tentu harus diingat bahwa bukan berarti apabila ada 2 orang yang memiliki 100% kesamaan mereka memang benar berjodoh. Pasangan di dunia ini bervariasi. Ada yang memiliki sebagian besar kesamaan, ada juga yang justru berjodoh karena tidak memiliki kesamaan sama sekali. Bagaimanapun jodoh tetap ada di tangan Tuhan Yang Maha Esa, dan aplikasi pencarian jodoh ini hanya salah alat perantara. Karena kita dapat menemukan jodoh dimanapun dan dengan cara apapun.

II. PATTERN MATCHING

Pattern Matching jika dalam Bahasa Indonesia adalah pencocokan pola. Dari namanya saja sudah dapat diketahui bahwa pattern matching adalah istilah untuk mencari suatu pola dalam suatu hal (bisa berupa teks, gambar, dll). Pada kehidupan nyata, pattern matching dapat dikembangkan menjadi berbagai macam pencocokan. Contoh pengimplementasian pattern matching adalah pencocokan string, pencocokan dna, pencocokan sidik jari, dan lain-lain. Pada makalah ini, pattern matching yang akan menjadi fokus pembicaraan adalah pencocokan string.

Dengan pencocokan string, dapat diketahui apakah suatu pola yang diinginkan merupakan substring dari string tersebut. Secara notasi algoritmik, pencocokan string dapat diperlihatkan seperti berikut.

```
boolean PencocokanString(string Text, string
Pola)
{
      if (isIn(Text,Pola) == true) then
            return true;
      else
            return false;
}
```

Dapat dilihat pada notasi algoritmik di atas maksud dari pencocokan string. Untuk menemukan apakah Pola adalah substring dari Text, banyak algoritma yang dapat digunakan. Namun 3 algoritma yang terkenal dalam pencocokan string adalah Algoritma Brute-Force, Algoritma Knuth Morris Pratt, dan Algoritma Boyer Moore.

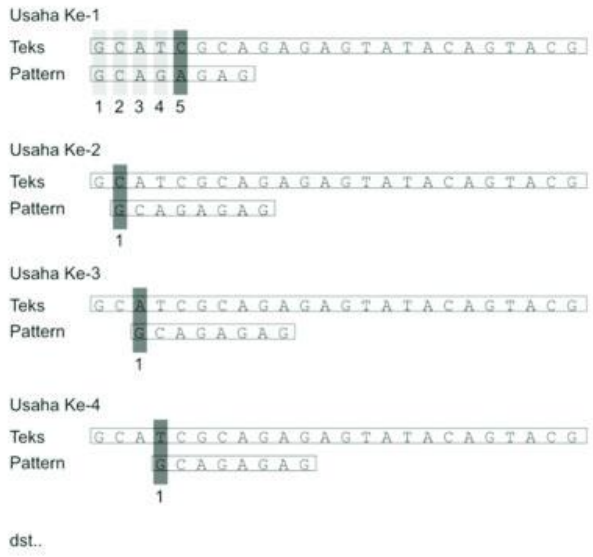
A. Algoritma Brute-Force

Algoritma Brute Force adalah algoritma yang lempang atau langsung ke pokok permasalahan. Pada pencocokan string, algoritma ini mencocokkan satu per satu karakter yang terdapat pada teks dan pola.

Langkah-langkah pencocokan string dengan menggunakan algoritma brute force adalah:

1. Algoritma brute force mencocokkan pola pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter demi karakter pola dengan karakter demi karakter pada teks yang bersesuaian hingga dipenuhi salah satu kondisi:
   - Karakter di pola dan di teks tidak sama.
   - Seluruh karakter pada pola ditemukan lalu kemudian algoritma mengembalikan nilai ketemu.
3. Algoritma kemudian menggeser karakter pada pola sejumlah 1 ke kanan dan mengulangi langkah 2 sampai teks berakhir.

Berikut adalah simulasi algoritma brute force pada pencocokan string.



Gambar 1. Simulasi Algoritma Brute Force

Berikut adalah pseudeucode 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
```

B. Algoritma Knuth Morris Pratt

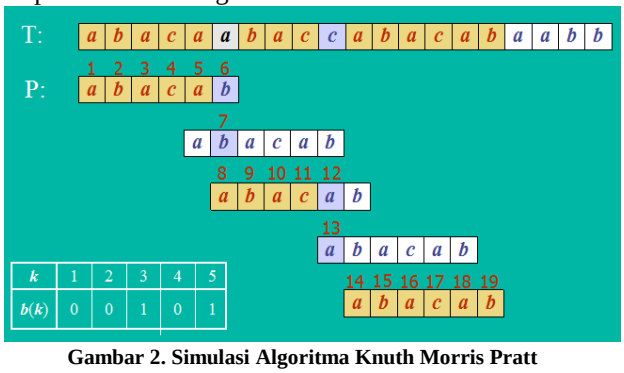
Algoritma Knuth Morris Pratt adalah salah satu algoritma pencocokan string yang lain. Algoritma ini sulit jika dijelaskan dalam kata-kata. Perhitungan penggeseran pada algoritma ini adalah sebagai berikut, bila terjadi ketidakcocokkan pada saat pattern sejajar dengan *teks*[*i*..*i* + *n* − 1], kita bisa menganggap ketidakcocokan pertama terjadi diantara *teks*[*i* + *j*] dan *pattern*[*j*], dengan 0 < *j* < *n*. Berarti, *teks*[*i*..*i* + *j* − 1] = *pattern*[0..*j* − 1] dan *a* = *teks*[*i* + *j*] tidak sama dengan *b* = *pattern*[*j*]. Ketika kita menggeser, sangat beralasan bila ada sebuah awalan *v* dari pattern akan sama dengan sebagian akhiran *u* dari sebagian teks. Sehingga kita bisa menggeser pattern agar awalan *v* tersebut sejajar dengan akhiran dari *u*.

Dengan kata lain, pencocokan string akan berjalan secara efisien bila kita mempunyai tabel yang menentukan berapa panjang kita seharusnya menggeser seandainya terdeteksi ketidakcocokkan di karakter ke-*j* dari pattern. Tabel itu harus memuat *next*[*j*] yang merupakan posisi karakter *pattern*[*j*] setelah digeser, sehingga kita bisa menggeser pattern sebesar *j* − *next*[*j*] relatif terhadap teks. [3]

Berikut langkah pencocokan string dengan menggunakan algoritma knuth morris pratt ini adalah:

1. Algoritma Knuth Morris Pratt memulai mencocokkan pola pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter demi karakter dari pola dengan karakter di teks yang sesuai, sampai salah satu kondisi berikut dipenuhi:
   - Karakter di pola dan di teks yang dibandingkan tidak sesuai.
   - Seluruh karakter pada pola ditemukan lalu kemudian algoritma mengembalikan nilai ketemu.
3. Algoritma kemudian menggeser pola berdasarkan tabel next, lalu kemudian mengulangi langkah 2 hingga pola berada di ujung teks.

Berikut adalah simulasi algoritma knuth morris pratt pada pencocokan string.



Gambar 2. Simulasi Algoritma Knuth Morris Pratt

Berikut adalah psedeucode algoritma knuth morris pratt

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

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
```

Berikut adalah pseudeucode algoritma knuth morris pratt 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

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
```

C. Algoritma Boyer Moore

Algoritma Boyer Moore adalah salah satu algoritma pencocokan string yang lain. Algoritma ini sulit jika dijelaskan dengan kata-kata. Misalnya ada sebuah pencocokan yang terjadi pada teks[i...(i+n-1)], dan misalkan ketidakcocokan pertama terjadi diantara teks[i+j] dan pola[j], dengan 0<j<n. Dengan begini, teks[(i+j+1)....(i+n-1)] = pola[(j+1)...(n-1)] dan a = teks[i+j] tidak sama dengan b = pola[j].

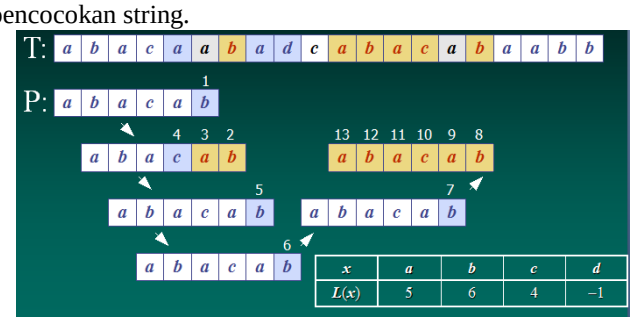
Jika u adalah akhiran pola sebelum b dan v adalah sebuah awalan pola, maka pergeseran-pergeseran yang mungkin adalah:

1. Penggeseran good-suffix yang terdiri dari mensejajarkan potongan *teks*[*i* + *j* + 1..*i* + *n* − 1] = *pattern*[*j* + 1..*n* − 1] dengan kemunculannya paling kanan di pattern yang didahului oleh karakter yang berbeda dengan pattern[j]. Jika tidak ada potongan seperti itu, maka algoritma akan mensejajarkan akhiran v dari teks[i + j + 1..i + n − 1] dengan awalan dari pattern yang sama.
2. Penggeseran bad-character yang terdiri dari mensejajarkan teks[i + j] dengan kemunculan paling kanan karakter tersebut di pattern. Bila karakter tersebut tidak ada di pattern, maka pattern akan disejajarkan dengan teks[i + n + 1]. [2]

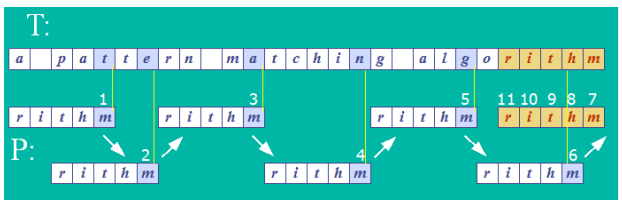
Berikut adalah langkah pencocokan string dengan menggunakan algoritma boyer moore.

1. Algoritma Boyer-Moore mulai mencocokkan pattern pada awal teks.
2. Dari kanan ke kiri, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
   - Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
   - Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma kemudian menggeser pattern dengan memaksimalkan nilai penggeseran good-suffix dan penggeseran bad-character, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

Berikut adalah contoh algoritma boyer moore pada pencocokan string.



Gambar 3. Contoh Algoritma Boyer Moore



Gambar 4. Contoh Lain Algoritma Boyer Moore

Berikut adalah psedeucode 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

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 pseudeucode pencarian string dengan menggunakan 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

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

III. PATTERN MATCHING PADA APLIKASI PENCARIAN JODOH

Banyak yang mempertanyakan bagaimana mengimplementasikan pattern matching pada aplikasi

pencarian jodoh. Namun jawabannya sangat sederhana.

Jika seseorang (sebut saja A) hendak mencari jodoh, yang perlu dilakukan hanyalah masuk ke aplikasi pencarian jodoh (biasanya aplikasi ini online), lalu mendaftarkan diri. Pada aplikasi tersebut A tinggal menuliskan deskripsi dirinya secara rinci namun singkat, lalu menuliskan deskripsi jodoh yang diinginkan olehnya secara rinci namun singkat. Kemudian aplikasi akan mencari kecocokan setiap kata pada deskripsi jodoh dengan seluruh data pada aplikasi tersebut. Pencarian ini dilakukan pada deskripsi diri setiap anggota lain aplikasi pencarian jodoh ini. Semakin banyak kecocokan yang ditemukan pada anggota lain, semakin besar prosentase kecocokan antara A dan anggota lain ini.

Contoh kasus, A mendaftar pada aplikasi pencarian jodoh X dengan data sebagai berikut.

```
Nama: A B C
Jenis Kelamin: Pria
Deskripsi Diri: Pintar, kaya, menarik, suka kucing, menyukai acara “American Next Top Model”.
Mencari: Wanita
Deskripsi Pencarian: Jelek, gendut, suka kucing.
```

Setelah itu aplikasi akan mencarikan anggota lain yang memiliki deskripsi diri dengan kata kunci jelek, gendut, dan suka kucing. Semakin banyak deskripsi diri yang cocok dengan kata kunci pencarian, semakin besar persentase berjodoh mereka.

Pattern matching diimplementasikan pada bagian pencarian ini. Pada saat aplikasi mencarikan jodoh yang sesuai, pattern matching yang melakukan pencarian dengan cara menjadikan kata kunci pencarian sebagai pola dan deskripsi diri setiap anggota lain sebagai teks lalu melakukan pencocokan string pada pola dan teks tersebut (entah itu menggunakan algoritma brute force, algoritma knuth morris pratt, ataupun algoritma boyer moore).

Berikut pseudeucode dari aplikasi pencarian jodoh ini.

```
struct
{
   int kodemember;
   string nama;
   int jeniskelamin; // 1 untuk pria, 2 untuk wanita
   int pencarian; // jenis kelamin yang dicari
   string[] deskripsi;
   string[] deskripsijodoh;
} member;

member[] aplikasi;
int[] ketemu;

procedure SearchJodoh(member a)
{
   foreach (member m in aplikasi)
   {
     if (a.kodemember != m.kodemember)
     {
           if (a.pencarian == m.jeniskelamin)
           {
            foreach (string s in a.deskripsijodoh)
            {
                  if (PencocokanString(s,m.deskripsi))
                  {
                        ketemu = m.kodemember;
                  }
            }
           }
     }
    }
}
```

Pseudeucode di atas belum menampilkan prosentase kecocokan dan pseudeucode di atas memang belum dicoba untuk diimplementasikan secara nyata, namun sudah banyak situs pencarian jodoh yang memiliki fitur pencarian seperti ini. Contoh situs pencarian jodoh yang memiliki fitur pencarian seperti ini adalah http://www.indonesiancupid.com.

IV. KESIMPULAN

Pattern matching dapat diimplementasikan dalam berbagai macam hal, salah satunya adalah pencarian string. Pencarian string dapat menggunakan banyak algoritma. Contoh algoritma pencocokan string yang terkenal adalah Algoritma Brute Force, Algoritma Knuth Morris Pratt, dan Algoritma Boyer Moore.

Dengan perkembangan teknologi saat ini, pencarian string dapat diimplementasikan pada berbagai macam aplikasi. Salah satu contoh implementasi dari pencarian string adalah aplikasi pencarian jodoh. Dengan menggunakan pencarian string, dapat dicari orang yang memiliki banyak kecocokan dengan kriteria yang diinginkan seseorang.

REFERENSI

[1] http://id.wikipedia.org/wiki/Algoritma_pencarian_string
[2] http://id.wikipedia.org/wiki/Algoritma_Boyer-Moore
[3] http://id.wikipedia.org/wiki/Algoritma_Knuth-Morris-Pratt
[4] Slide Pattern Matching Kuliah IF3051 Strategi Algoritma Semester 1 Tahun Pelajaran 2010/2011

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, 8 Desember 2010

Dini Lestari Tresnani 13508096