

String Matching Dalam Permainan “The Hunt for Gollum”

Ligar Mugi Syahid (10111053)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganessa 10 Bandung 40132, Indonesia
ligar@s.itb.ac.id

Abstrak - Pecocokan *String* merupakan algoritma yang penting dengan tujuan mencari lokasi *String* tertentu (*Pattern*) pada *String* (*Text*) yang lain. Contoh algoritma pencocokan *String*, Naïve String search (Brute Force), Knuth-Morris-Pratt (KMP) dan Booyer-Moore. Pencocokan *String* sangat banyak di kehidupan sehari-hari, contohnya mencari suatu subbab dalam buku, mencari buku dari nomer ISBN, dll.

Kata kunci: String Matching, KMP, Booyer-Moore

bababa
ababab
ababab
bababa
ababab

Input Program :

Text file berisi :

1. Ukuran matriks dari peta, dengan urutan “baris kolom”
2. Matriks peta karakter

Contoh :

3 3
aba
bab
aba
7 6
aababa
ababab
bababa
ababab
ababab
bababa
ababab

output Program :

setiap baris output merupakan index dari Peta Gollum yang terdapat pada Peta Aragorn. Jika tidak ada, *print* “Tidak di temukan”

contoh :
(1, 2)
(1, 4)
(2, 1)
(2, 3)
(5, 1)
(5, 3)

I. Pendahuluan

I.1 Peraturan Permainan

Permainan ini diambil dari <http://www.spoj.com/problems/ARDA1/> dengan deskripsi sebagai berikut,

Aragorn telah memulai perburuan Gollum, setelah beberapa hari, dia telah tiba di Rawa Kematian dan percaya Gollum berada di sana. Rawa Kematian tersebut sangat gelap, basah dan mengerikan. Namun Aragorn telah memiliki peta seluruh Rawa. Dengan kemampuan dan pengalamannya selama ini, Aragorn tahu karakteristik Rawa yang di sukai Gollum.

Peta yang dimiliki Aragorn adalah matriks dari karakter berukuran $M_1 \times M_2$. Dan karakteristik rawa yang di sukai Gollum juga matriks dengan ukuran $N_1 \times N_2$.

Sebagai Contoh :

Peta Gollum :

aba
bab
aba

Peta seluruh rawa :

aababa
ababab

II. Metode

II.1 String Matching

Misalkan n , jumlah karakter pada *pattern* dan m , jumlah karakter pada *text*.

II.1.1 Algoritma Brute Force

Secara sederhana *Brute Force* membandingkan tiap karakter pada *Pattern* ke *Text* dimulai dari huruf pertama pada *Pattern* dan *Text*.

Jika terjadi kesalahan geser 1 huruf, begitu seterusnya hingga *Pattern* ditemukan dalam *Text* atau tidak di temukan dan sudah sampai ke ujung *Text*.

Untuk kasus terburuk :

Tiap perbandingan $n-1$ karakter *Pattern* pada *Text* menghasilkan *true*, namun perbandingan yang ke n , *false*. Dan *Pattern* berada di ujung *Text*.

Maka,

- akan ada $(m-n)$ pergeseran
- tiap pergeseran ada n perbandingan
- total perbandingan $(m-n)*n = mn-n^2$.

jadi kompleksitasnya : $O(mn)$

II.1.2 Algoritma Knuth-Morris-Pratt (KMP)

Perbaikan dari *Brute Force*. Tidak seperti *Brute Force*, Algoritma ini menggeser *Pattern* pada *Text* berdasarkan *Border Function*, *array of integer* yang menyimpan ukuran terbesar dari *Prefix Pattern* yang juga *Suffix Pattern*.

Contoh :

Pattern : 0 0 1 1 0 0 0 1
Border Function : 0 1 0 0 1 2 2 -

Misalkan *Border Function* : $BF[i]$, dengan i adalah *index*.

Maka $BF[5]$, adalah *prefix* "0 0 1 1 0" yang mengandung *suffix* terbesar, yaitu "0 1 1 0" maka $BF[5] = 1$.

Maka kompleksitas dari KMP ini, $O(m+n)$.

Untuk kasus rata-rata, $O(m+n)$.

II.1.3 Algoritma Boyer-Moore (BM)

Algoritma ini didasarkan pada 2 teknik.

- Teknik *looking-glass*
Yaitu mencari *Pattern* pada *Text* dimulai dari ujung akhir *Pattern*
- Teknik *character-jump*
Misalkan $T[i]$ adalah karakter *Text* ke i .
 $P[i]$ adalah karakter *Pattern* ke i .
Jika terjadi perbandingan karakter yang salah, maka ada 3 kasus.
 - Jika ada kejadian terakhir karakter $T[i]$ dalam *Pattern*, misal $P[k]$, geser hingga membandingkan $P[k] = T[i]$.
 - Jika tidak dapat di geser, geser *Pattern* ke $T[i+1]$.
 - Jika tidak dapat melakukan point a dan c, geser hingga membandingkan $P[1] = T[i+1]$

Untuk itu kita membutuhkan *array*, misal *LastOccurent*, yang menyimpan kejadian terakhir suatu karakter.

Contoh :

Pattern : 0 0 1 1 0 0 0 1

Karakter	0	1	2
<i>LastOccurent</i>	6	7	-1

Tabel 1. LastOccurent

Kompleksitasnya untuk kasus rata-ratanya : $O(m+n)$.

Kasus Terburuk : $O(mn+A)$. dengan A jumlah alphabet berbeda pada *Pattern*.

III. Implementasi

III.1 Strategi

Pada permainan ini, Kita diminta mencari lokasi dari submatriks karakter (Peta Gollum) pada matriks karakter (Peta Aragorn).

Strategi pemecahan masalah,

1. Buat *array of string* dari Peta Gollum sebagai *pattern* dan Peta Aragorn sebagai *text*. Menggunakan contoh pada subbab I.1
 - a. Peta Gollum
P[0]="aba"
P[1]="bab"
Dst...
 - b. Peta Aragorn
M[0] = "aababa"
M[1] = "ababab"
Dst...

Batasan string input 100 untuk masing-masing peta.

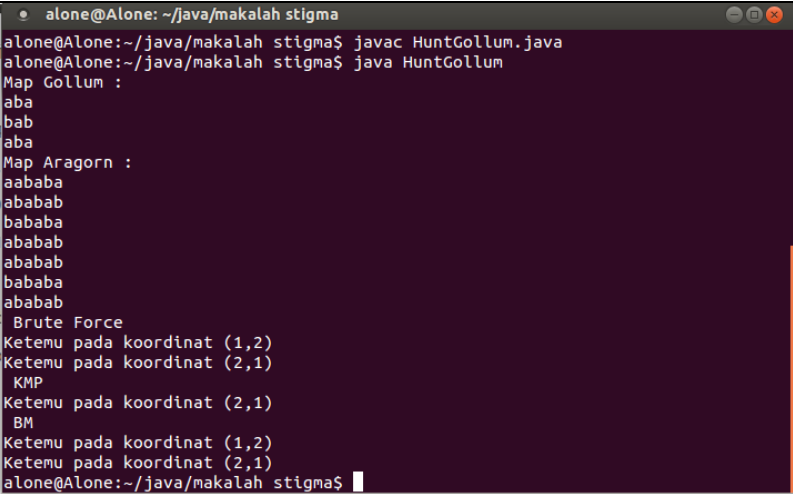
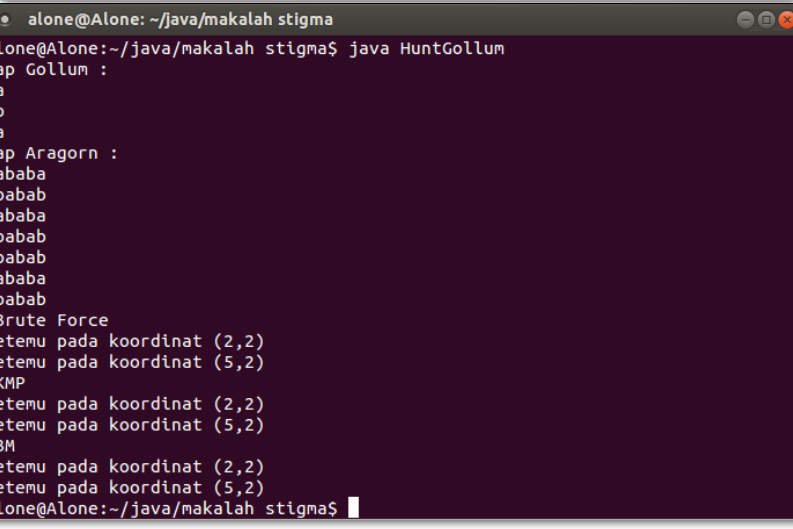
2. Selanjutnya cari *pattern* pada *text* dengan index yang bersesuaian.
 - a. Untuk Peta Gollum ukuran $N_1 \times N_2$ (baris x kolom), misal P dan Peta Aragorn ukuran $M_1 \times M_2$, misal M.
 - b. Mula-mula *counter* $i = 0$.
 - c. Selanjutnya akan di cari Posisi (x,y) untuk *pattern* pertama(P[0]) pada M(i). misal pada baris ke i dan karakter ke y ditemukan P(0). Maka tinggal mencocokkan M(i+1) dengan P(1), dan seterusnya hingga P(N_1) .
 - d. Jika di temukan. memberikan solusi (i, y).
 - e. Jika tidak *increase* $i + 1$. Kembali ke c.
 - f. Jika $i > (M_1 - N_1)$ program berhenti.

Skema Program

```
procedure Solve(input P, M : array of string, m :
integer) {
Mencari seluruh index kiri atas dari P pada M
dengan algoritma / mode tertentu.
Input : P, M : array of string problem. m : mode 1
Brute, mode 2 KMP, mode 3 BM
Output : koordinat P pada M, (x,y)
}
Kamus :
i, prec : integer
rowP : integer{ukuran baris P}
Algoritma :
Untuk mode m
While( i<(baris M – baris P +1))
  If(found P[0] in M[i])
    Prec = AlgoStringMatchmode(m);
    {koordinat y}
    {cek untuk P, M selanjutnya}
    While (found P[k] in M[m])
      {k=1, 2 ..rowP, m=i+k}
      Found = true;

  If found
    Print (i,prec);
```

IV. Hasil dan Analisis

Input	Output
<pre>3 3 aba bab aba aba 7 6 aababa ababab bababa ababab ababab bababa ababab</pre>	 <pre> alone@Alone: ~/java/makalah stigma alone@Alone:~/java/makalah stigma\$ javac HuntGollum.java alone@Alone:~/java/makalah stigma\$ java HuntGollum Map Gollum : aba bab aba Map Aragorn : aababa ababab bababa ababab ababab bababa ababab Brute Force Ketemu pada koordinat (1,2) Ketemu pada koordinat (2,1) KMP Ketemu pada koordinat (2,1) BM Ketemu pada koordinat (1,2) Ketemu pada koordinat (2,1) alone@Alone:~/java/makalah stigma\$ </pre>
<pre>3 2 ba ab ba 7 6 aababa ababab bababa ababab ababab bababa ababab</pre>	 <pre> alone@Alone: ~/java/makalah stigma alone@Alone:~/java/makalah stigma\$ java HuntGollum Map Gollum : ba ab ba Map Aragorn : aababa ababab bababa ababab ababab bababa ababab Brute Force Ketemu pada koordinat (2,2) Ketemu pada koordinat (5,2) KMP Ketemu pada koordinat (2,2) Ketemu pada koordinat (5,2) BM Ketemu pada koordinat (2,2) Ketemu pada koordinat (5,2) alone@Alone:~/java/makalah stigma\$ </pre>

Hasil belum cukup memuaskan, semua solusi belum berhasil di temukan.

Sedangkan untuk perbandingan algoritma, telah di hitung sebelumnya KMP dan BM memiliki kompleksitas yang sama. Namun pada praktiknya jumlah perbandingan karakter *pattern* dan *text* pada BM lebih sedikit dibanding KMP.

V. Kesimpulan

Masalah pencocokan *string* sering ditemukan dimana-mana. Pada perkuliahan Strategi Algoritma telah diajarkan pencocokan string dengan Brute Force, KMP, dan BM. Namun masih banyak lagi algoritma string pencocokan *string* yang lain seperti Rabin-Karp. Rata-rata, algoritma pencocokan string memiliki kompleksitas yang sama yaitu, $O(m+n)$.

Permainan “The Hunt for Gollum” adalah contoh kecil apabila kita ingin mencari daerah dengan kriteria tertentu pada suatu wilayah. Misalkan, daerah dengan tanah yang paling subur terluas pada suatu propinsi tertentu atau mencari lokasi dengan kandungan mineral logam tertentu.

Untuk kedepannya diharapkan penulis dapat menyelesaikan permainan ini.

VI .REFERENSI

- [1] Slide perkuliahan Strategi Algoritma 2013/2014 [http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014/Pencocokan%20String%20\(2013\).ppt](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014/Pencocokan%20String%20(2013).ppt)
- [2] <http://www.spoj.com/problems/ARDA1/>

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, 20 Desember 2013

A photograph of a handwritten signature in blue ink on a white background. The signature is stylized and appears to be 'Ligar Mugi Syahid'.

Ligar Mugi Syahid 10111053