Analisis Karakteristik Pemain *fighting game* dengan Algoritma *pattern matching*

Farizan Ramadhan - 13511081[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13511081@std.stei.itb.ac.id

Abstract*— fighting game sangat berkembang dari waktu ke waktu. Awalnya fighting game hanya berdasarkan single instruction. Tapi kini berkembang dari sisi perintah yang diberikan oleh pemain, hingga kombinasi gerakan fighting game semakin bervariasi. Makalah ini akan membahas cara menganalisis karakteristik pemain dari cara pemain tersebut memasukan instruksi ke dalam permainan. Cara menganalisisnya adalah dengan memeriksa pattern yang dimasukkan oleh pemain dengan algoritma string matching yang sudah diperlajari di kuliah Strategi Algoritma.

Index Terms*— fighting game, string matching, analisis karakteristik.

I. PENDAHULUAN

Fighting game semakin berkembang dari waktu ke waktu, sekitar 20 tahun yang lalu permainan kategori ini hanya berdasarkan *single instruction* dari pemain. Namun, seiring berjalannya waktu, permainan ini berkembang dengan *multi instruction* dan memperbanyak kombinasi gerakan dari avatar game yang dimainkan.

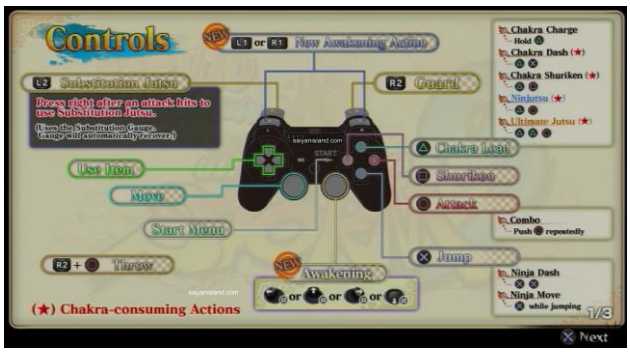
Fighting game pada masa awal, kita tidak bisa melihat karakteristik pemain yang kita lawan. Muncul kesulitan dalam menganalisa gerakan dan karakteristik pemain dalam memenangkan pertandingan, karena tidak muncul pola yang begitu jelas dari ronde ke ronde.

Pada dasarnya *fighting game* akan berlangsung pada waktu yang terbatas. Misal satu ronde dilaksanakan dalam 90 atau 60 detik. Sangat banyak varian permainan *fighting game* yang beredar di dunia.

Pada makalah ini, permainan dengan kategori *fighting game* yang menjadi bahan penelitian adalah "Naruto : Ultimate Ninja Storm 3 full burst". Permainan ini dipilih karena permainan ini mendukung *multi instruction* dalam pergerakan avatarnya.

Gambar 1 – *Gameplay*
Sumber : www.vg247.com

Sebagaimana yang telah disebutkan, permainan ini mendukung *multi instruction.* Permainan ini memilki pola instruksi yang dapat dipakai di semua avatar, perbedaannya adalah bentuk pergerakannya. Pada dasasrnya permainan ini dibagi menjadi gerakan : menyerang, bertahan, melarikan diri, mengisi tenaga 'chakra'.



Gambar 2 – Controller dan Instruksi
Sumber : www.saiyanisland.com

Permainan ini berjalan di multi konsol, diantaranya : PS3, XBOX 360, dan PC. Secara umum pemakaian dan instruksi tidak jauh berbeda antara satu konsol dan konsol yang lainnya.

II. DASAR TEORI

A. Pattern Matching

Pattern matching merupakan algoritma yang digunakan

untuk mencari suatu pola tulisan dalam suatu teks maupun memvalidasi suatu teks dengan pola tertentu.

Dalam ilmu komputer, *pattern matching* adalah sebuah aksi memeriksa urutan token-token dalam sebuah pola.

B. Konsep String

Asumsikkan 'S' adalah string dengan ukuran 'm'.

$S = x_1x_2x_3 \ldots x_m$

Prefix dari S adalah substring dari S[1 … k-1]

Suffix dari S adalah substring dari S[k-1 … m]

Dengan catatan :

- 'k' adalah indeks antara 1 dan m
- S[0] adalah character kosong / 'null'

C. Algoritma *Brute Force*

Brute Force akan membandingkan *pattern* ke sebuah teks, karakter per karakter setiap waktu, sampai menemukan karakter yang tidak sesuai dengan substring dari *pattern*. Algoritma ini, ketika menemukan karakter yang berbeda, akan maju satu karakter dan memeriksa dari awal *pattern* lagi ke teks. Algoritma ini akan terus berjalan hingga teks berakhir.

Kompleksitas algoritma *brute force* dapat kita lihat dari dua pendekatan, yaitu :

- Kasus terbaik (notasi Big-Oh) = O(M)
- Kasus terburuk (notasi Big-Oh) = O(MN)

Dengan M adalah jumlah perbandingan, dan N adalah panjang teks yang diperiksa.

D. Algoritma Rabin-Karp

Algoritma pencarian string/pattern Rabin-Karp akan mengkalkulasi nilai *hash (hash value)* untuk sebuah *pattern*, dan untuk setiap M-karakter yang berurutan untuk diperiksa.

Jika nilai *hash* tidak sama, maka algotima ini akan menghitung nilai *hash* untuk M-karakter selanjutnya dalam urutan.

Namun jika nilai *hash* sama, algoritma ini akan melakukan perbandingan dengan algoritma *brute force* antara *pattern* dan M-karakter yang berurutan.

Secara prinsip, algoritma ini akan meminimasi penggunaan *brute force*.

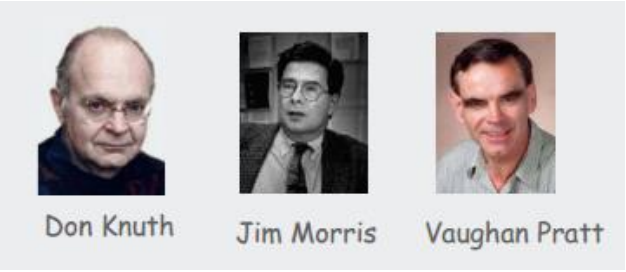
Kompleksitas Algoritma Rabin-Karp dapat kita lihat dari dua pendekatan, yaitu :

- Kasus terbaik (notasi Big-Oh) = O(N)
- Kasus terburuk (notasi Big-Oh) = O(MN)

Dengan N adalah panjang karakter dari sebuah teks.

E. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma KMP adalah algoritma pencarian string yang mirip dengan *brute force*, perbedaannya adalah cara bergeser atau metode bergeser *pattern* yang akan diperiksa dengan teks lebih baik.



Gambar 3 – Foto sosok dibalik algoritma KMP

Algoritma KMP akan memelihara informasi yang digunakan untuk melakukan jumlah pergeseran. Hal ini dimaksudkan agar pergeseran lebih jauh (tidak satu per satu karakter). Dengan algoritma KMP, jumlah pergeseran akan berkurang dengan sangat signifikan.

Komponen algoritma KMP adalah :

- Fungsi pinggiran (border function)

  Nama lain dari fungsi pinggiran diantaranya : fungsi *overlap*, fungsi *failure*, fungsi awalan, dsb.

  Fungsi ini akan mengindikasikan pergeseran 's' yang terbesar yang mungkin dengan menggunakan perbandingan yang dibentuk sebelum pencarian string. Dengan fungsi pinggiran ini, dapat dicegah pergeseran-pergeseran yang tidak berguna seperti pada algoritma *brute force*.

  Fungsi pinghgiran hanya bergantung pada karakter-karakter di dalam pattern, dan bukan pada karakter-karakter di dalam teks. Oleh karena itu, kita dapat melakukan perhitungan fungsi awalan sebelum pencarian *string* dilakukan.

  Fungsi pinggiran b(j) didefinisikan sebagai ukuran awalan terpanjang dari P (pattern) yang merupakan akhiran dari P[1…j].

  Kompleksitas Algoritma KMP dilihat dari perhitungan waktu untuk mendapatkan fungsi pinggiran dan jumlah waktu pencarian string, berikut detilnya :

  - Fungsi pinggiran = O(m)
  - Pencarian string = O(n)

  Total dari kompleksitas adalah O(m+n).

  [Disadur dari Diktat Kuliah IF2211, Rinaldi M.]

F. Algoritma Boyer-Moore

Algoritma Boyer-Moore merupakan variasi lain dari pencarian string dengan melompat maju sejauh mungkin. Tetapi, algoritma Boyer-Moore memiliki perbedaan dengan algoritma KMP, algoritma KMP akan melakukan perbandingan pattern dari kanan ke kiri, sedangkan Boyer-Moore sebaliknya (kiri ke kanan).

G. *Fighting Game*

Sebuah *Fighting Game* merupakan salah satu permainan yang karakter pemain (avatar) digerakan oleh pemain dengan suatu kontroler. Permainan ini didesain untuk perkelahian jarak dekat (close combat). Permainan ini minimal satu lawan satu, baik itu melawan AI (Artificial Intelegence) ataupun melawan player lain (multiplayer).

Awal permainan, karakter yang dimainkan akan

memiliki kadar kekuatan yang seimbang. Dalam satu pertandingan biasanya dibagi menjadi beberapa ronde dengan durasi yang bervariasi.

Pemain harus menguasai teknik-teknik dasar seperti menyerang (memukul dan menendang), *counter attack*, bertahan, berlari, menghindar, serangan spesial, dan lain-lain.

Permainan berkembang sehingga dikenal juga isitlah *combo*, yaitu serangan yang terstruktur yang dilakukan secara berurutan (*sequences of attack*), istilah ini mulai digunakan dan dikembangkan pada awal 1990an.

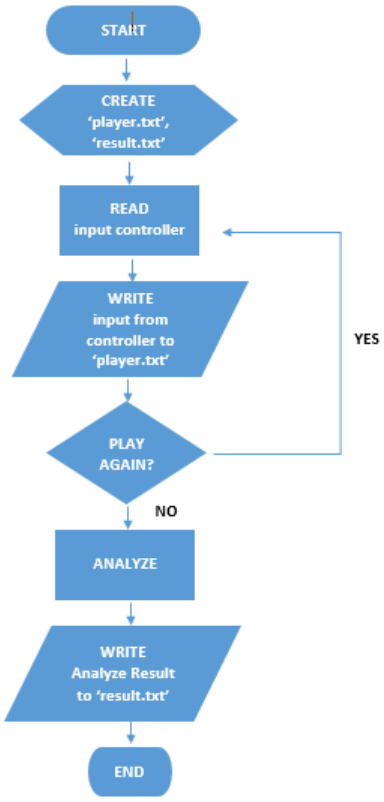
Serangan spesial (*special attack*) adalah serangan tingkat lanjut yang dapat dilakukan dengan menekan tombol-tombol spesifik yang berurutan atau bersamaan.

Permainan atau pertandingan akan berakhir jika waktu dalam suatu ronde telah habis atau salah satu karakter dalam permainan kehabisan tenaganya untuk bertanding karena dikalahkan oleh pemain (karakter) lainnya.

III. Analisis Penyelesaian Maslah

Ide dasar dari menganalisa karakteristik pemain adalah membaca dan mempelajari input yang pemain kirim melalui *joystick* atau *controller* yang menggerakkan avatar di permainan.

Untuk mendapatkan input dari *controller*, *controller* tersebut harus dijamin terhubung ke perangkat yang dapat menangkap sinyal tersebut, dalam hal ini *controller* mengirim sinyal melalui koneksi *bluetooth* atau *wifi*, pada dasarnya semua konsol sudah mengembangkan dan mengimplementasikan *wireless controller* dengan mekanisme seperti ini.



***Flowchart* Aplikasi**

Penjelasan flowchart :

1. START
   Memulai permainan di konsol dan aplikasi siap menerima input
2. CREATE
   Aplikasi selanjutnya akan membuat 2 file teks yaitu:
   a. 'player.txt', file ini nantinya akan diisi oleh hasil transalsi sinyal dari kontroller dan menjadi bahan analisa karakterisitk
   b. 'result.txt', file ini akan diisi oleh hasil analisa karakteristik.
3. READ
   Setelah dibuat file teks pendukung, perangkat penerima sinyal dalam hal ini adalah *personal computer* yang telah dipasang aplikasi ini akan menerima semua masukan dari pemain. Pada prinsipnya, sinyal akan diterima oleh dua perangkat yang telah 'pair' pada konfigurasi masing-masing perangkat penerima sinyal, maka satu sama lain perangkat penerima sinyal tidak akan salling mengganggu.

Ilustrasi penerimaan sinyal :

Ilustrasi Cara Kerja Penerimaan Sinyal

4. WRITE
   Setelah membaca input dari *controller*, selanjutnya input tersebut akan ditampung di memori sementara, kemudian input tersebut akan ditranslasikan sesuai aturan yang nantinya akan disimpan di 'player.txt' sebagai bahan untuk analisis.
5. PLAY AGAIN?
   Bagian ini, akan ditentukan pemain akan terus bermain atau selesai, jika akan bermain lagi, maka input akan kembali dibaca dan kemudian akan ditulis ke 'player.txt' (setelah dilakukan translasi).
6. ANALYZE
   Pada bagian ini file 'player.txt' akan dianalisis dengan algoritma pattern matching yang diimplementasikan.

7. WRITE
   Setelah hasil analisis muncul, maka hasil analisis tersebut akan disimpan pada file ‘result.txt’.
8. END
   Akhir dari proses analisis karakteristik pemain.

Selain itu, diperlukan komponen-komponen pendukung penyelesaian masalah, yaitu data-data yang menjadi bahan acuan dalam analisis karakteristik, diantaranya :

A. Daftar Instruksi Avatar (tabel A)

Isi dari tabel ini adalah daftar semua instruksi yang dapat pemain gunakan untuk menggerakan avatar pada permainan.

No	NAMA	PATTERN
1	Punch	C
2	Shuriken	S
3	Jump	X
4	Double Jump	XX
5	Chakra Charge	T
6	Chackra Shuriken	TS
7	Chakra Boost	TX
8	special move - Jutsu 1	TC
9	special move - Jutsu 2	TTC
10	Guard	4
11	Fake Guard	2
12	Combo Attack	CCCCC
13	Jump then Combo Attack	XCCCCC
14	Call Partner 1	1
15	Call Partner 2	3

Tabel 1 – Instruksi dan Pattern

B. Tabel Dampak Instruksi (tabel B)

Isi dari tabel ini adalah hubungan antara gerakan avatar dan dampak atau akibat dari gerakan tersebut pada avatar lawan atau pada diri avatar itu sendiri.

No	PATTERN	DAMAGE		
		LOW	MID	HIGH
1	C	v		
2	S	v		
3	X	-		
4	XX	-		
5	T	-		
6	TS		v	
7	TX	-		
8	TC		v	
9	TTC			v
10	4	-		
11	2	-		
12	CCCCC			v
13	XCCCCC			v
14	1		v	
15	3		v	

Tabel 2 – Pattern dan Impact Damage

C. Aturan Translasi Sinyal Controller

Pada dasarnya sinyal yang akan dikirim oleh controller adalah transmisi gelombang mikro dengan panjang gelombang yang pendek. Untuk itu, perlu ada translasi dari sinyal tersebut menjadi sebuah *string* yang nantinya akan diolah untuk analisis karakteristik.

Untuk aturan translasinya adalah sebagai berikut :

Gambar 1 – Tampak atas

Gambar 2 – Tampak depan

Aturan translasi ini berlaku untuk konsol Playstation 3.

Tombol	Huruf
Action Right	
X	X
Circle	C
Triangle	T
Square	S
Analog	
UP	U
DOWN	D
LEFT	L
RIGHT	R
Front	
L1	1

L2	2
R1	3
R2	4
Action Left	
UP'	5
DOWN'	6
LEFT'	7
RIGHT'	8

Tabel 3 – Translasi sinyal ke karakter

IV. IMPLEMENTASI

Berdasarkan flowchart yang telah disusun pada bagian III. Berikut hasil implementasinya :

A. Implementasi Bagian 'ANALYZE'

Seperti yang disebutkan sebelumnya, pada bagian ini, akan diterapkan beberapa algoritma *pattern matching* dalam menganalisa.

Algoritma BRUTE FORCE

```
do
   if (text letter == pattern letter) then
    compare next letter of pattern to next letter of text
  else
    move pattern down text by one letter
while (entire pattern found or end of text)
```

Algoritma Rabin Karp

```
hash_p = hash value of pattern
hash_t = hash value of first M letters in body of text

do
  if (hash+p == hash_t)
   bruteforce comparison of pattern
   and selected section of text
  hash_t = hash value of next section of text, one character over
while(end of text or bruteforce comparison == TRUE)
```

Algoritma KMP (search)

```
S = text (string)
W = pattern (string)
m ← 0 (index of text)
i ← 0 (index of pattern)
T = array of integer

while (m+1) < length(S) do
  if W[i] = S[m+1] then
     if i = length(W) – 1 then
        return m
     i ← i + 1
  else
     m ← m + 1 – T[i]
     if T[i] > -1 then
        i ← T[i]
     else
       i ← 0
```

Algoritma KMP (table)

```
W = pattern (string)
T = array of integer (the table to be filled)
pos ← 2
candidate ← 0

T[0] ← -1
T[1] ← 0

While pos < length(W) do
 /* first case */
  If W[pos-1] = W[candidate] then
     candidate ← candidate + 1
    T[pos] ← candidate
    pos ← pos + 1

   /* second case */
  Else if candidate > 0 then
        candidate ← T[candidate]

  /* third case */
  Else
     T[pos] ← 0
     pos ← pos + 1
```

Algoritma Boyer-Moore (search)

```
S = text (string)
W = patterm (string)

Last[] =buildLast(W)
n = length(S)
m = length(W)
i = m-1

if (I < n-1)
  return FALSE
/* not valid if pattern is longer than text */

j = m-1
do
  if (W.charAt(j) == S.charAt(i))
     if (j == 0)
        return TRUE /* match, found */
     else
       i ← i – 1
       j ← j – 1
```

```
  else
      lo = last[S.charAt(i)]
      i ← i + m – min(j, 1+lo)
      j ← m – 1
while ( i <= n -1)

 return FALSE
```

Algoritma Boyer-Moore (buildLast)

```
last = array of integer

for i=0  to i < 128 do
  last[i] = -1

for i=0  to i < length(pattern) do
  last[pattern.charAt(i)] = i

return last
```

B. Metode Analisis Karakteristik

Dari file teks yang berisi hasil translasi dari sinyal *controller*. Aplikasi akan diatur untuk memeriksa jumlah kemunculan *pattern* yang telah didefinisikan. Setelah diketahui jumlah kemunculannya, maka akan mudah menganalisis karakteristik pemain tersebut, prinsip dasarnya adalah menangkap kecenderungan seorang pemain dalam mengalahkan musuhnya.

Kemungkinan yang dapat dihasilkan dari hasil analisis karakteristik ini adalah :

1. Pemain bertipe penyerang *High Damage*
2. Pemain bertipe penyerang *Mid Damage*
3. Pemain bertipe penyerang *Low Damage*
4. Pemain bertipe bertahan, mengandalkan serangan balik

Cara perhitungannya adalah sebagai berikut :

Jumlah kemunculan masing-masing *pattern* akan dibagi dengan seluruh jumlah *pattern* yang terdeteksi. Sebagai ilustrasi, misal ada sebuah pattern 'TTX' muncul sebanyak 25 kali, sedangkan jumlah kemunculan seluruh jenis *pattern* dalam waktu permainan sebanyak 150 kali. Maka 25/150. Nilai tersebut akan dibandingkan, dan dicari nilai yang terbesar, artinya *pattern* yang memiliki nilai terbesar itu paling sering digunakan oleh pemain.

C. Contoh Test-case Aplikasi

Diberikan sebuah hasil translasi dari instruksi-instruksi pemain ke karakter permainan (avatar) sebagai berikut :

player.txt

```
CCCCTXXCXXCTTCTTCTCXXTXXXSSTSTCCCCTSXCTCT
TCSTXTXXTCT1243TX1STCXS134334TCTTCTCXXTXXXSS
TSTCCCCTSXCTCTTCSTXTXXTCT12TSTCCCCTSXCTCTT
CSCCTXXCXXCTCCTXXCXXCTCCCTSXCTCTTCSTCSTXT
XXTCT1243TX1STCXS134334TCTTCTCXXTXXXS
```

Analisis Pattern dari 'player.txt'

```
CCCCTXXCXXCTTCTTCTCXXTXXXSSTSTCCCCTSXCTCT
TCSTXTXXTCT1243TX1STCXS134334TCTTCTCXXTXXXSS
TSTCCCCTSXCTCTTCSTXTXXTCT12TSTCCCCTSXCTCTT
CSCCTXXCXXCTCCTXXCXXCTCCCTSXCTCTTCSTCSTXT
XXTCT1243TX1STCXS134334TCTTCTCXXTXXXS
```

result.txt

--- Player Characteristic ---

Type : Attacker, with HIGH Damage

Movement :

1. **Chackra Boost (10)**
2. **Combo Attack (3)**
3. **Special move – jutsu 1 (6)**
4. **Special move – jutsu 2 (3)**
5. **….**
6. **….**

Total Pattern Found in 'player.txt' = 26

V. KESIMPULAN

Algoritma *pattern matching* dapat digunakan dalam analisis karakterisitik pemain dalam game kategori *fighting*, tidak berhenti pada kategori ini saja, konsep ini dapat dipakai dalam menganalisis permainan yang lain, seperti sepak bola dan bola basket. Hanya saja *pattern* yang menjadi kamus (dictionary) akan berbeda.

Konsep analisis karakteristik ini akan menghasilkan tingkat akurasi yang tinggi jika pattern semakin panjang, karena pattern yang panjang sangat jelas dalam mengklasifikasikan gaya bermain seseorang.

Berdasarkan hasil implementasi, algoritma *pattern matching* yang paling baik untuk menganalisis adalah Algoritma KMP, karena algoritma ini akan memeriksa teks dari kiri ke kanan, hal ini sangat relevan karena analisis ini harus berurutan dari kiri ke kanan, ketika algoritma memeriksa dari kanan ke kiri, membuka peluang untuk berkurangnya pattern yang terbaca.

VI. UCAPAN TERIMA KASIH

Puji dan syukur kehadirat Allah SWT, karena atas rahmat dan karunianya, makalah ini bisa selesai dengan baik. Ucapan terima kasih juga saya sampaikan kepada kedua orang tua yang mendukung pengerjaan makalah ini. Tidak lupa kepada Dosen pengajar IF2211 Dr.Ir. Rinaldi Munir, M.T. dan Dr. Masayu Leyla Khodra, S.T., M.T. yang telah memberikan materi tentang Strategi Algoritma yang sangat bermanfaat untuk implementasi penyelesaian masalah di masa depan. Dan juga teman-teman seangkatan khususnya yang tergabung di K-01 IF2211 atas saran dan kritik selama pengerjaan.

REFERENSI

[1] Gimpel, J. F. 1973. A Theory of Discrete Patterns and Their Implementation in SNOBOL4. Commun. ACM 16, 2.

[2] Munir, Rinaldi. Diktat Kuliah IF2211 – Strategi Algoritma. 2009.

[3] http://www.dcs.gla.ac.uk/~pat/52233/slides/Strings1x1.pdf

[4] B. Smith, "An approach to graphs of linear forms (Unpublished work style)," unpublished.

[5] E. H. Miller, "A note on reflector arrays (Periodical style—Accepted for publication)," *IEEE Trans. Antennas Propagat.*, to be published.

[6] J. Wang, "Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication)," *IEEE J. Quantum Electron.*, submitted for publication.

[7] C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.

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, 19 Desember 2013

Farizan Ramadhan
13511081