Penerapan Algoritma Pattern Matching Metode Brute-Force pada Aplikasi Gacha Games Clicker Bot

M. Reyhanullah Budiaman / 13519045
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : muhammadreyhan2000@gmail.com

***Abstract*— Video game yang menerapkan sistem gacha memiliki aktivitas yang cukup melelahkan yang harus dilakukan oleh para pemain untuk dapat melanjutkan permainan atau membuka fitur-fitur baru selain menggunakan uang asli. Bot dan *macro* untuk mengautomasikan aktivitas pemain di video game tidak jarang ditemukan saat ini dengan kebutuhan *grinding* dalam gacha game. Namun, karena sebagian besar video game tidak mengimplementasikan sebuah API karena alasan keamanan, maka alternatif solusi untuk mengatasinya adalah dengan membandingkan tampilan pada layar sebagai *state* yang nantinya diproses oleh bot untuk menentukan aksi selanjutnya. Tujuan dari makalah ini adalah menerapkan algoritma brute-force dalam pattern matching sebagai salah satu alternatif algoritma yang dapat diimplementasikan pada clicker bot untuk memperoleh *state* permainan.**

Keywords—brute force; patern matching; clicker bot; state; macro

I. PENDAHULUAN

Permainan "Gacha" merupakan permainan dengan elemen kemungkinan dalam model bisnis freemium. Jenis permainan ini mengandalkan mata uang 'langka' pada permainan yang dapat diperoleh menggunakan uang asli atau dengan dengan memainkannya. Secara umum, mata uang 'langka' dapat diperoleh dengan *farming* atau *grinding* (istilah yang biasa digunakan pada komunitas-komunitas *gamer*, yaitu melakukan aktivitas yang sama berulang-ulang) yang bisa membosankan. Tidak hanya itu, beberapa permainan gacha juga membutuhkan proses *farming* atau *grinding* yang cukup banyak agar dapat melanjutkan permainan. Aktivitas ini bisa sangat membosankan dan membuang terlalu banyak waktu. Walaupun demikian, para pemain permainan gacha tetap melakukannya dengan senang karena kecanduan Kecanduan tersebut akan membuat pemain susah lepas dari permainan tersebut, walaupun mereka tidak punya waktu untuk memainkannya. Demi *progress*, beberapa pemain bahkan membayar orang lain untuk memainkan game mereka saat mereka tidak bisa main. Beberapa pemain yang berdedikasi juga menggunakan perangkat lunak seperti *macro* atau bot. *Macro* adalah perangkat lunak yang menyimpan urutan masukan pengguna yang dapat digunakan berulang kali. Oleh karena itu, penggunaan *macro* dapat menghemat waktu pemain untuk melakukan aktivitas berulang yang membosankan. Hanya saja beberapa *macro* yang digunakan oleh pemain memiliki beberapa masalah eksternal, seperti *latency*, yang dapat membuat *timing* dari *macro* sedikit meleset. Pada makalah ini dibahas salah satu solusinya yaitu dengan memanfaatkan *pattern recognition* atau *pattern matching* pada bot/*macro*.

Keuntungan dari penggunaan *pattern matching* adalah pemrosesan urutan masukan penggunan tidak terpengaruhi oleh *latency* dan *timing*, karena bot mencari pattern tertentu (dapat berupa tombol, text, gambar, dan lain-lain) yang didefinisikan oleh pengguna dan meminimumkan pengawasan pengguna karena peluang kesalahannya lebih rendah.

Gambar 1.1 Contoh Tampilan UI Game Gacha

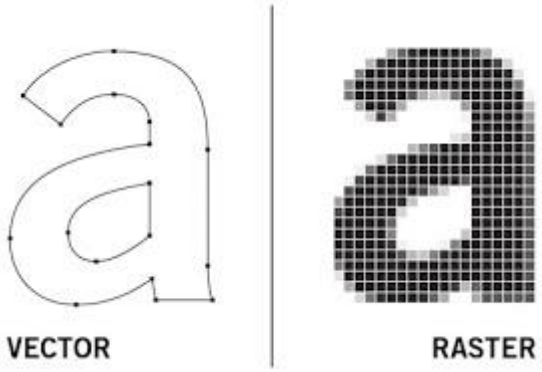
II. DASAR TEORI

A. *Gambar Digital*

Gambar digital adalah representasi sebuah citra dua dimensi dalam bentuk data. Sebuah gambar digital bisa termasuk kedalam dua jenis. Yang pertama adalah bitmap atau *raster* yang umum digunakan. Format bitmap merepresentasikan sebuah citra dua dimensi dalam sebuah larik pixel dua dimensi. Setiap pixel diberi nilai dalam byte. Pada gambar hitam-putih. Pixel hanya bisa berwarna hitam atau putih yang diwakili dengan kode biner. Pada bitmap terdapat dua istilah penting, yaitu resolusi yang dinyatakan dalam satuan dpi (dot per inch) dan intensitas atau kedalaman warna.

Jenis yang kedua adalah vector. Vektor adalah serangkaian instruksi matematis yang dijabarkan dalam bentuk, garis, dan bagian-bagian lain yang saling berhubungan dalam sebuah gambar. Gambar digital yang berbasis vektor biasanya tidak

tergantung pada resolusi sehingga dapat di ubah ukurannya dan di cetak pada resolusi berapapun tanpa merusak kualitas detailnya.



Gambar 2.1 Perbandingan Vector dan Raster
Sumber : seekacreative.co.nz

B. Pattern Matching

Pattern Matching adalah sebuah teknik pencarian sebuah pola (*pattern*) pada sebuah token (*text*). Walaupun pattern matching ini biasa digunakan pada string atau sebuah larik 1 dimensi, beberapa algoritma pattern matching dapat diimplementasikan pada gambar digital yang representasi datanya berupa larik 2 dimensi. Penggunaan pattern matching memberikan keluaran berupa lokasi kemunculan pola pada token.

Terdapat beberapa algoritma pada pattern matching seperti brute force, Knuth-Morris-Pratt (KMP), Boyer-Moore, Rabin-Karp, dan Finite Automata. Pada makalah ini, akan berfokus pada implementasi algoritma brute force.

C. Algoritma Brute Force

Pada algoritma ini, setiap karakter pada *pattern* dibandingkan dengan karakter pada *text* secara *straightforward*. Algoritma ini sederhana dan mudah diimplementasikan namun memiliki kompleksitas waktu yang tidak efisien. Pada masalah pencarian dalam struktur data larik, Algoritma Brute Force menelusuri setiap elemen pada larik.

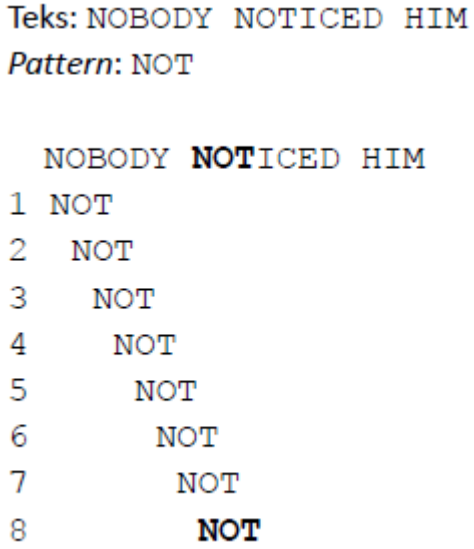
Karena merupakan sebuah algoritma yang memecahkan masalah secara jelas, dan melalui banyak opini atau pilihan, maka algoritma brute force merupakan sebuah metode pemecahan masalah logis yang memiliki kemampuan untuk memperoleh pemecahan masalah dengan baik. Dengan mempertimbangan banyak opsi, metode algoritma brute force mampu untuk menyaring satu dari sekian banyak solusi atau opsi yang ditawarkan,sehingga proses pemecahan masalah yang dilakukan akan menjadi lebih baik dan juga lebih optimal. Hampir semua masalah yang dipecahkan dengan menggunakan metode algoritma brute force ini berjalan dengan baik.

Namun demikian, meskipun memiliki kelebihan berupa pemecahan masalah yang mampu berjalan dengan baik dan juga sempurna, algoritma brute force sangat sulit untuk digunakan pada kebutuhan pemecahan masalah yagn cepat. Hal ini disebabkan karena algoritma brute force membutuhkan kumpulan banyak opsi terlebih dahulu sebulu dieksekusi. Hal ini membuat pertimbangan dalam memilih opsi akan menjadi lebih lambat.

Implementasi algoritma brute force dalam pattern matching adalah sebagai berikut. Diberikan sebuah text T[1..n] dan pattern P[1..m] adalah larik dari karakter dengan panjang masing-masing n dan m karakter. Langkah-langkah penyelesaiannya adalah sebagai berikut.

1. Sejajarkan *pattern* dan *text* pada awal karakter (P[*i*] dan T[*j*]) untuk dibandingkan, dimana $i = j = 1$
2. Bergerak dari kiri ke kanan, bandingkan setiap karakter pada *pattern* dengan karakter pada *text* sampai memenuhi kondisi berikut.
    a. Semua karakter pada *pattern* cocok, pencarian diberhentikan
    b. Terjadi mismatch (ketidakcocokan), maka ulangi langkah ini dengan $j = j + 1$ (melakukan pergeseran ke kanan sebanyak 1 karakter)
3. Jika karakter terakhir pada *pattern* dicocokan dengan karakter terakhir pada *text* dan terjadi mismatch, maka *pattern* tidak terdapat pada *text*.

Berikut adalah ilustrasi algoritma Brute Force:



Gambar 2.2 Ilustrasi algoritma Brute Force
Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Pada ilustrasi diatas, terjadi pergeseran sebanyak 8 kali dan perbandingan sebanyak 12 kali.

Jika menghitung jumlah operasi perbandingan yang dilakukan, kompleksitas waktu terbaik algoritma Brute Force adalah $O(n)$ dan kompleksitas waktu terburuknya adalah

$O(mn)$. Untuk kasus rata-ratanya, algoritma ini memiliki kompleksitas waktu $O(n)$. .

D. *Matriks*

Matriks adalah larik dua dimensi yang secara umum elemennya merupakan angka yang diatur dalam baris dan kolom.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ m \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & n \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \end{array}$$

Gambar 2.3 Sebuah matriks berukuran m x n
Sumber : https://en.wikipedia.org/wiki/File:Matris.png

III. Analisis Pemecahan Masalah

Permasalahan pencocokan gambar dapat diselesaikan dengan mula-mula memodelkan hal yang merupakan masalah utama terlebih dahulu, yaitu gambar tersebut. Seperti yang telahdisebutkan pada dasar teori, gambar adalah representasi citra dalam bentuk pixel yang tersusun dalam larik 2 dimensi yang dapat dimodelkan dengan sebuah matriks.

Matriks model akan memiliki elemen berupa byte tiap piksel dari gambar. Matriks tersebut memiliki ukuran lebar x tinggi, sesuai ukuran gambar, dan seterusnya sampai (m,n). Matriks ini dapat dikatakan sebagai sebuah area pada sebuah gambar. Setiap area adalah sebuah submatriks yang berukuran sama dengan ukuran matriks dari *pattern*. Proses perbandingan akan dilakukan sama seperti membandingkan string seperti yang telah diuraikan pada dasar teori. Proses iterasi perbandingan area dari *text* (yang dalam kasus ini adalah *screenshot* layar utama) dengan *pattern* dapat dilakukan perbaris ataupun perkolom. Iterasi perbandingan akan berhenti ketika terjadi mismatch. Jika terjadi mismatch maka pergeseran akan dilakukan dengan $j = j + 1$ yang dalam kasus ini j adalah kolom dari elemen a_{11} pada *text*. Ketika terdapat *exact match* dari *pattern*, maka iterasi pencarian diberhentikan dan program mengembalikan posisi pertama kalit terjadinya *exact match* tersebut.

Pada implementasi ini, terdapat beberapa batasan. Yang pertama adalah resolusi dari game gacha tidak pernah berubah dan resolusi gambar dari *pattern* harus sama dengan game gacha yang sedang berjalan.

Asumsi yang digunakan pada implementasi ini ialah pencarian *pattern* akan dilakukan pada *state* yang telah didefinisikan oleh pengguna sebelumnya, sehingga sudah pasti *pattern* terdapat pada UI game gacha pada *state*-nya yang terkait.

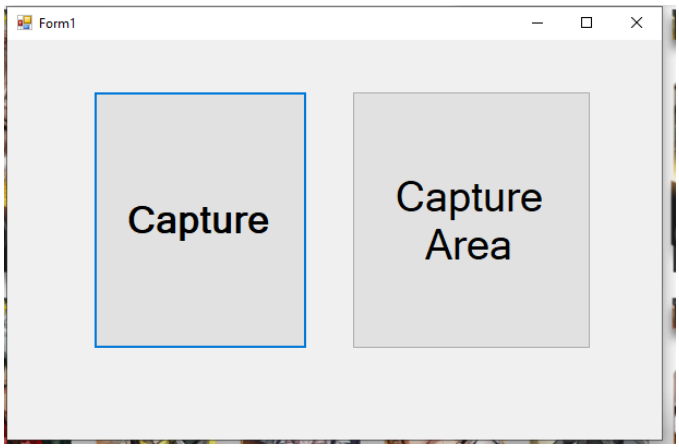
- “approximately” or “effectively.”

IV. Implementasi dan Analisis Pengujian

Pada tahap implementasi, penulis membuat sebuah program yang dijadikan percobaan untuk menyelesaikan permasalahan pencarian letak potongan citra ini. Program dibuat dengan bahasa C# dan dengan pustaka WinForm sebagai antarmuka.

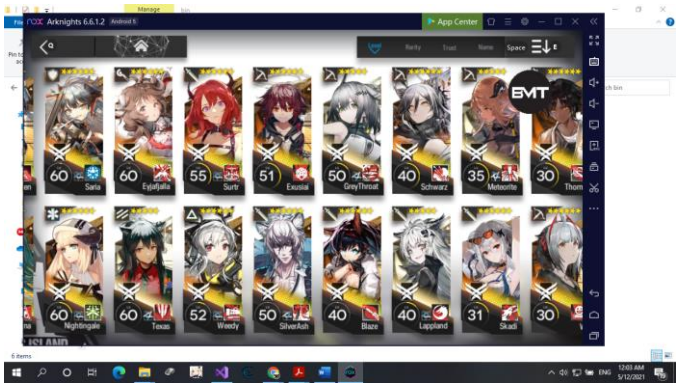
Program ini memiliki fitur mengambil potongan gambar pada layar yang akan disimpan sebagai template (*pattern*). Dan menangkap seluruh layar sebagai eksekutor program utama. Pada program ini, penulis juga menambahkan fitur untuk memindahkan posisi kursor ke tempat *pattern* pertama ditemukan.

Berikut adalah tampilan program.

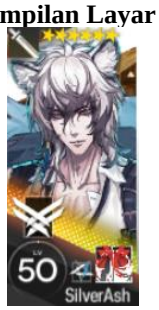


Berikut merupakan hasil eksekusi program pada permainan gacha game Arknights pada emulator Nox Player.

A. *Percobaan 1*

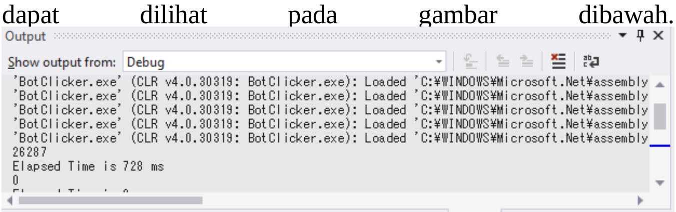


Gambar 4.1 Tampilan Layar List Karakter



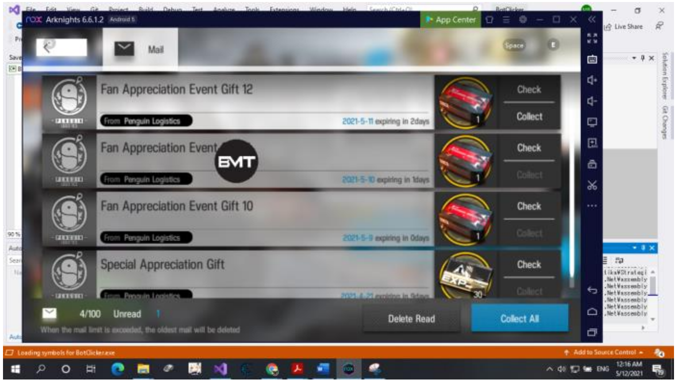
Gambar 4.2 Pattern karakter yang dicari

Pada percobaan pertama ini, program menemukan *pattern* pada gambar 4.1. Jumlah perbandingan yang dilakukan adalah 26287 kali dan waktu eksekusinya adalah 728ms. Hasilnya dapat dilihat pada gambar dibawah.

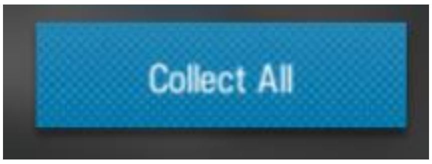


Gambar 4.3 Hasil Percobaan 1

B. Percobaan 2

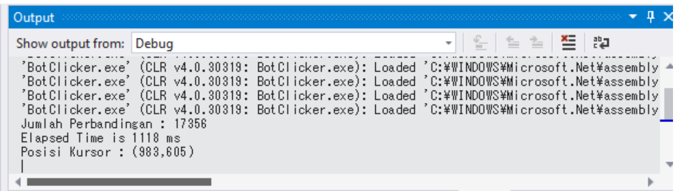


Gambar 4.4 Tampilan Layar Mailbox



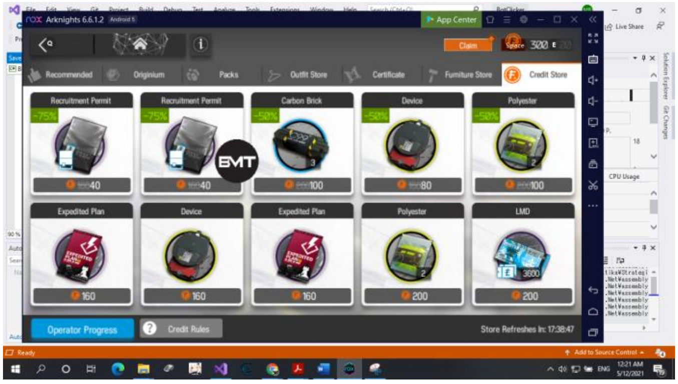
Gambar 4.5 Pattern Collect All

Pada percobaan kedua ini, program menemukan *pattern* pada gambar 4.4. Jumlah perbandingan yang dilakukan adalah 17356 kali dan waktu eksekusinya adalah 1118ms. Hasilnya dapat dilihat pada gambar dibawah.



Gambar 4.6 Hasil Percobaan 2

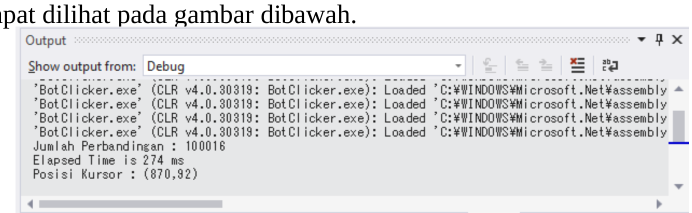
C. Percobaan 3



Gambar 4.7 Tampilan Layar Credit Store

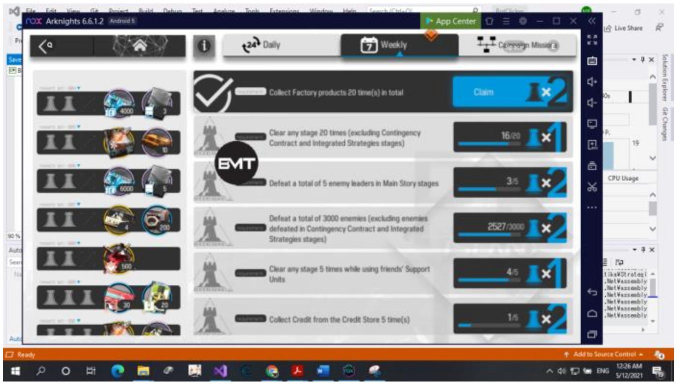
Gambar 4.8 Pattern Claim

Pada percobaan ketiga ini, program menemukan *pattern* pada gambar 4.7. Jumlah perbandingan yang dilakukan adalah 100016 kali dan waktu eksekusinya adalah 274ms. Hasilnya dapat dilihat pada gambar dibawah.



Gambar 4.9 Hasil Percobaan 3

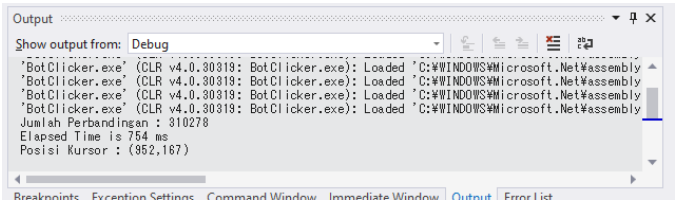
D. Percobaan 4



Gambar 4.10 Tampilan Layar Weekly Missions

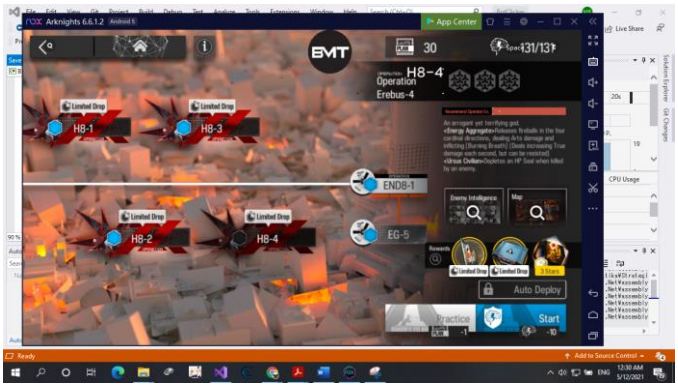
Gambar 4.11 Pattern Claim Weekly Reward

Pada percobaan keempat ini, program menemukan *pattern* pada gambar 4.10. Jumlah perbandingan yang dilakukan adalah 310278 kali dan waktu eksekusinya adalah 754ms. Hasilnya dapat dilihat pada gambar dibawah.

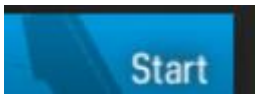


Gambar 4.12 Hasil Percobaan 4

E. Percobaan 5

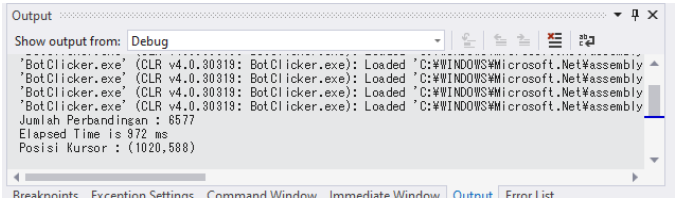


Gambar 4.13 Tampilan Layar Stage



Gambar 4.14 Pattern Start Button

Pada percobaan kelima ini, program menemukan *pattern* pada gambar 4.13. Jumlah perbandingan yang dilakukan adalah 6577 kali dan waktu eksekusinya adalah 972ms. Hasilnya dapat dilihat pada gambar dibawah.



Gambar 4.15 Hasil Percobaan 5

Dari seluruh hasil percobaan diatas, dapat dilihat bahwa jumlah perbandingan pada layar dengan variasi warna yang beragam memiliki jumlah perbandingan sedikit. Ambillah percobaan 5 dan 2, ekspektasi jumlah perbandingan yang dihasilkan adalah tidak memilii perbedaan yang sangat jauh dari percobaan 3 dan 4 karena posisi pattern lebih jauh dari titik awal pencarian pada percobaan 5 dan 2. Jumlah perbandingan yang banyak muncul pada tampilan layar yang variasi warnanya tidak begitu luas.

VIDEO LINK AT YOUTUBE

Tidak ada.

SIMPULAN DAN SARAN

Pattern matching adalah algoritma pemecahan masalah yang banyak diaplikasikan dalam sebuah program komputer. Salah satu metode dalam algoritma pattern matching adalah brute force yang dapat digunakan untuk mencari pola suatu gambar dari gambar lain. Walaupun algoritma brute force ini merupakan algoritma yang tidak terlalu efisien untuk pencarian pola gambar, waktu eksekusi untuk algoritma ini cukup cepat (< 3s).

Beberapa algoritma pattern matching (seperti KMP dan Boyer-Moore) yang lebih efisien tidak diimplementasikan pada program ini karean penulis kesulitan dalam menentukan border function(KMP) dan last occurrence (Boyer-Moore) untuk larik 2 dimensi.

Pencarian pola dalam gambar sebenarnya telah berkembang dan algoritma – algoritmanya dapat dikatakan efisien jika dibandingkan dengan brute force.

Untuk kedepannya, pencocokan pola gambar ini tidak dilakukan secara *exact matching* mengingat tampilan UI akan selalu berubah untuk game gacha. Salah satunya adalah dengan menggunakan approximate string matching ataupun dengan algoritma-algoritma template matching.

Penulis meminta maaf apabila ada kesalahan dalam penulisan atau penyampaian materi tulisan ini. Semoga pembaca bisa mendapatkan manfaat sebanyak-banyaknya dan, jika berkenan, memberi saran dan kritik kepada penulis.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas selesainya pembuatan makalah ini. Tidak lupa, penulis berterima kasih kepada dosen mata kuliah IF2251 Strategi Algoritma, yakni Bapak Rila Mandala yang telah memberikan materi dan ilmu, yang bermanfaat dalam pembuatan makalah ini.

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf.
Diakses pada tanggal 10 Mei 2021.

[2] https://docs.microsoft.com/en-us/dotnet/csharp/pattern-matching.
Diakses pada tanggal 10 Mei 2021.

[3] https://www.apockdesign.com/2020/04/pengertian-rasterbitmap-dalam-dunia.html.
Diakses pada tanggal 10 Mei 2021.

[4] https://www.cloudflare.com/learning/bots/what-is-a-bot/
Diakses pada tanggal 10 Mei 2021.

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.

Makassar, 11 Mei 2021

M. Reyhanullah Budiaman 13519045