Penyelesaian Permainan Domino LINE Let's Get Rich Menggunakan Algoritma Runut-Balik

Studi Kasus: Pengembangan Permainan Domino Interaktif dengan Kecerdasan Buatan

Abrar Abhirama Widyadhana - 13523038
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: abrar.widyadhana.b@gmail.com , 13523038@std.stei.itb.ac.id

***Abstrak*—LINE Let's Get Rich adalah salah satu permainan populer yang didalamnya memiliki salah satu mode permainan Domino. Kunci untuk memenangkan mode permainan ini adalah dengan menyusun langkah yang lebih baik dari lawan komputer. Makalah ini membahas perancangan dan implementasi sebuah AI yang kompetitif untuk permainan Domino dengan menerapkan algoritma runut-balik (*backtracking*). Aplikasi simulasi permainan dibangun menggunakan bahasa Java dengan library Swing untuk antarmuka grafis yang interaktif. Algoritma runut-balik diimplementasikan sebagai otak utama AI, yang memungkinkannya untuk menganalisis status permainan saat ini dan mencari jalur langkah optimal yang mengarah pada kemenangan. Hasilnya adalah sebuah prototipe fungsional di mana AI mampu bermain secara strategis, tidak hanya memainkan langkah acak, sehingga memberikan tantangan yang signifikan bagi pemain manusia.**

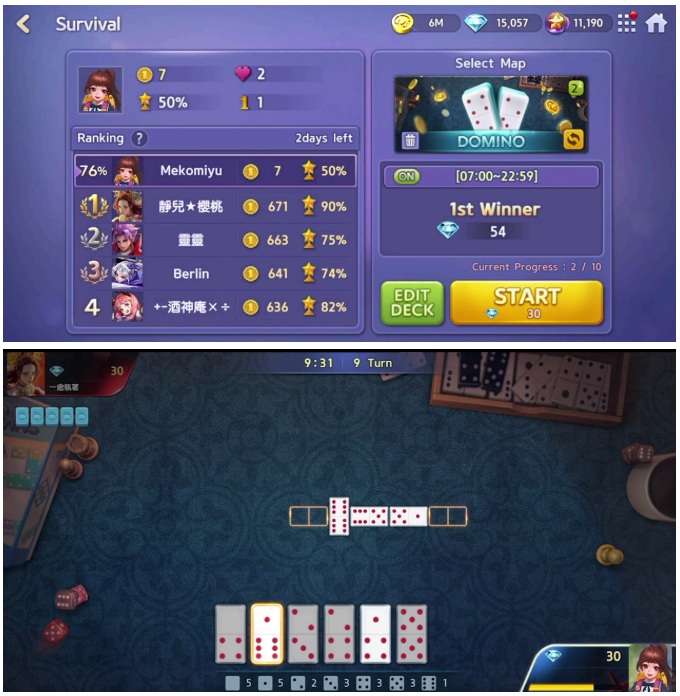
***Keyword*—*backtracking*, runut-balik, permainan domino, kecerdasan buatan, Java Swing**

I. Pendahuluan

A. *Latar Belakang*

LINE Let's Get Rich merupakan salah satu permainan digital yang populer di Indonesia. Salah satu mode permainan yang ditawarkan dan banyak diminati karena sistem hadiahnya adalah mode permainan Domino. Dalam mode ini, seorang pemain akan berhadapan dengan lawan komputer (Artificial Intelligence atau AI). Kualitas pengalaman bermain sangat bergantung pada tingkat kecerdasan AI tersebut. Lawan komputer yang hanya bergerak secara acak atau dengan pola sederhana tidak akan memberikan tantangan yang berarti.

Hal ini memunculkan sebuah permasalahan rekayasa perangkat lunak yang menarik: bagaimana merancang sebuah AI yang mampu bermain secara strategis. Makalah ini akan membahas penerapan salah satu strategi algoritma fundamental, yaitu algoritma runut-balik (backtracking), untuk membangun otak dari AI yang kompetitif dalam permainan Domino, sebagai studi kasus yang terinspirasi dari game LINE Let's Get Rich.



Gambar 1.1 Tampilan Mode Permainan Domino di LINE Let's Get Rich

B. *Rumusan Masalah*

Berdasarkan latar belakang tersebut, permasalahan yang akan dibahas dalam makalah ini dapat dirumuskan sebagai berikut:

1. Bagaimana algoritma runut-balik (backtracking) dapat diterapkan untuk menciptakan AI yang kompetitif dalam sebuah permainan Domino?
2. Bagaimana tahapan perancangan dan implementasi sebuah aplikasi yang mensimulasikan permainan Domino secara utuh, mulai dari antarmuka pengguna hingga logika pengambilan keputusan AI?

C. *Tujuan*

Penulisan makalah dan pengembangan algoritma ini bertujuan untuk:

1. Mengimplementasikan algoritma runut-balik sebagai decision-making engine (mesin pengambil keputusan) untuk AI pada permainan Domino, sehingga AI dapat menentukan langkah terbaik yang mengarah pada kemenangan.
2. Membangun sebuah prototipe aplikasi permainan Domino yang fungsional dan interaktif menggunakan Java Swing, yang mensimulasikan mekanisme inti dari sebuah permainan domino melawan AI.

D. *Ruang Lingkup*

Agar Pembahasan tetap fokus dan mendalam. Akan ditekankan bahwa aplikasi yang di kembangkan merupakan sebuah simulasi mandiri yang terinspirasi dari mekanisme permainan Domino pada LINE Let's Get Rich, dan bukan merupakan modifikasi atau intervensi terhadap game aslinya.

Fokus utama dari makalah ini adalah pada perancangan dan implementasi kecerdasan buatan dengan menggunakan algoritma runut-balik. Oleh karena itu, fitur-fitur lain yang ada di dalam game LINE Let's Get Rich, seperti sistem ekonomi, karakter, atau item, tidak termasuk dalam cakupan pembahasan.

II. Dasar Teori

A. *Permainan Domino*

Permainan Domino adalah permainan strategi berbasis ubin di mana pemain secara bergiliran meletakkan kartu domino ke sebuah rantai di atas meja. Setiap kartu domino memiliki dua sisi dengan nilai angka tertentu. Sebuah kartu hanya dapat diletakkan jika salah satu sisinya memiliki nilai yang sama dengan salah satu dari dua nilai di ujung rantai yang terbuka. Pemain pertama yang berhasil menghabiskan semua kartunya dinyatakan sebagai pemenang. Jika terjadi kondisi di mana tidak ada pemain yang bisa bergerak lagi (permainan macet atau stuck), pemenang ditentukan berdasarkan pemain dengan total nilai sisa kartu di tangan yang paling kecil. Permainan ini menuntut pemain untuk tidak hanya memikirkan langkah saat ini, tetapi juga merencanakan langkah-langkah berikutnya dan memprediksi kemungkinan kartu lawan.

B. *Algoritma Runut-Balik*

Algoritma runut-balik adalah sebuah metode yang memecahkan suatu permasalahan dengan efisien dan terstruktur, bisa digunakan dalam permasalahan optimasi maupun non-optimasi. Algoritma ini adalah perbaikan dari algoritma *exhaustive search*. Pada algoritma *exhaustive search*, semua kemungkinan solusi akan dieksplorasi dan dievaluasi satu persatu. Sedangkan algoritma *backtracking* bekerja dengan cara mengeksplorasi pilihan-pilihan yang berpotensi mengarah ke solusi dan akan "memangkas" cabang pencaharian yang sudah pasti tidak akan menghasilkan solusi.

1. *Organisasi Ruang Solusi*

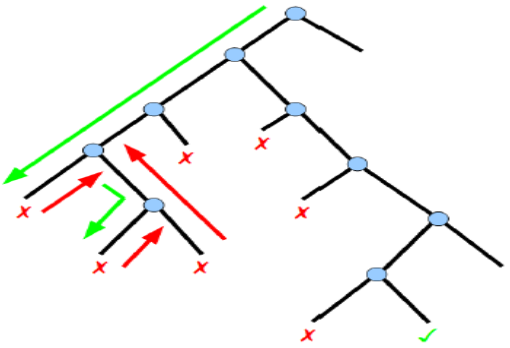
Semua Kemungkinan solusi dari suatu persoalan disebut ruang solusi. Dalam pendekatan backtracking, ruang solusi ini dimasukan ke dalam sebuah struktur pohon yang disebut pohon ruang status.

- Setiap simpul di dalam pohon ini merepresentasikan sebuah status dari suatu permasalahan
- Setiap lintasan dari akar menuju daun merepresentasikan satu kemungkinan solusi
- Pencarian solusi dilakukan dengan membangkitkan simpul-simpul status dari akar dengan aturan DFS, simpul yang dibangkitkan desbut simpul hidup dan simpul yang diperluas disebut simpul-E

2. *Prinsip Kerja dan Fungsi Pembatas*

Saat sebuah simpul-E diperluas, lintasan solusi akan terbangun dan bertambah panjang. Jika langkah tersebut tidak valid, maka simpul-E tersebut akan "dimatikan", proses "mematikan" ini dipengaruhi oleh fungsi pembatas. Fungsi pembatas $B(x_1, x_2, ..., x_□)$ adalah sebuah predikat yang akan bernilai true jika rangkaian pilihan $(x_1, x_2, ..., x_□)$ masih berpotensi mengarah ke solusi, dan bernilai false jika sebaliknya. Ketika sebuah simpul dinyatakan sebagai simpul mati, algoritma secara implisit telah memangkas semua kemungkinan solusi di bawah simpul tersebut, sehingga jauh lebih efisien daripada exhaustive search.

Jika pencarian berakhir pada simpul mati, maka proses akan melakukan runut-balik ke simpul di level atasnya untuk mencoba membangkitkan simpul anak yang lain. Proses maju-mundur ini terus berlanjut hingga solusi ditemukan atau semua kemungkinan telah dieksplorasi.



Gambar 2.2.1 Ilustrasi Proses Backtracking
Sumber:
http://www.w3.org/2011/Talks/01-14-steven-phenotype/

III. Perancangan Dan Implementasi

Pada bab ini akan dijelaskan mengenai tahapan perancangan arsitektur perangkat lunak, implementasi logika permainan, implementasi kecerdasan buatan untuk lawan, dan implementasi antarmuka pengguna untuk aplikasi permainan domino.

A. *Arsitektur Perangkat Lunak*

Aplikasi ini dibangun dengan pendekatan berorientasi objek menggunakan bahasa pemrograman Java. Strukturnya terdiri dari empat kelas utama yang masing-masing memiliki tanggung jawab yang spesifik untuk menjaga agar kode modular.

- DominoSolverGUI.java
  Kelas ini berfungsi sebagai kerangka atau jendela utama aplikasi. Tugasnya sangat sederhana, yaitu membuat sebuah JFrame yang menjadi wadah untuk semua komponen visual permainan.
- GamePanel.java
  Ini adalah kelas inti yang menjadi jantung sekaligus otak dari keseluruhan aplikasi. GamePanel bertanggung jawab untuk mengelola seluruh status permainan menangani logika alur permainan, memproses semua interaksi pengguna melalui tetikus, dan yang terpenting, menjalankan logika AI musuh. Kelas ini juga menangani semua proses penggambaran ke layar.
- Domino.java
  Sebuah kelas data yang berfungsi sebagai model untuk merepresentasikan satu buah kartu domino. Kelas ini dirancang untuk menyimpan dua nilai pada setiap sisinya. Di dalamnya, metode equals() dan hashCode() telah di-override secara khusus agar dapat menangani perbandingan kartu secara logis (misalnya, kartu [2|5] dianggap sama dengan [5|2]), yang sangat esensial untuk operasi pada koleksi data.
- Move.java
  Sebuah kelas data sederhana yang digunakan oleh algoritma AI untuk membungkus informasi sebuah langkah. Ini mencakup kartu domino yang akan dimainkan dan di mana penempatannya, apakah di sisi kiri atau kanan rantai.

B. Implementasi Logika Permainan

Keseluruhan alur dan logika permainan diatur oleh sebuah mesin status sederhana yang diimplementasikan di dalam kelas GamePanel. Proses permainan diawali dengan inisialisasi melalui metode startNewGame, yang dipicu saat pengguna menekan tombol "Game Baru". Metode ini bertanggung jawab untuk membuat satu set kartu domino lengkap, mengocoknya secara acak, membagikannya kepada pemain dan musuh, serta mengatur ulang semua variabel status ke kondisi awal.

Saat giliran pemain tiba, interaksi utama terjadi melalui MouseListener yang ditanamkan pada panel. Logika handleMouseClick akan mendeteksi dua jenis klik: klik pertama untuk memilih kartu dari tangan, yang akan disimpan sementara dalam variabel selectedDomino, dan klik kedua untuk mencoba meletakkan kartu tersebut ke ujung papan yang valid. Setelah pemain menentukan langkah, validitasnya diperiksa oleh metode tryPlayDomino. Metode ini akan memverifikasi apakah nilai pada kartu yang dipilih cocok dengan nilai di ujung rantai yang terbuka (currentLeftEnd atau currentRightEnd) sebelum mengizinkan langkah tersebut.

Jika pemain tidak memiliki langkah yang valid, ia dapat memilih untuk melewati gilirannya dengan menekan tombol "Pass", yang akan mengubah status playerPassed menjadi benar dan langsung menyerahkan giliran ke musuh. Alur permainan ini akan terus berlanjut hingga salah satu kondisi akhir terpenuhi, yaitu salah satu pemain kehabisan kartu, atau ketika kedua pemain sama-sama melakukan "Pass" secara berturut-turut yang menandakan permainan macet. Pada kondisi macet, pemenang akan ditentukan dengan memanggil metode calculateHandScore untuk menghitung dan membandingkan total nilai sisa kartu di tangan masing-masing pemain.

C. Implementasi AI Musuh

Kecerdasan buatan untuk musuh diimplementasikan dengan strategi hibrida agar dapat bermain secara efektif dan realistis. Logika ini terdapat dalam metode executeEnemyTurn. AI tidak hanya bergerak acak, tetapi mengikuti hirarki prioritas sebagai berikut

- Prioritas 1, Mencari Jalur Kemenangan dengan Runut-Balik
  Saat giliran musuh, AI pertama kali akan memanggil metode solve() yang berisi algoritma runut-balik. Sesuai dengan teori, metode ini akan menjelajahi pohon ruang status secara rekursif untuk mencari sebuah "jalur kemenangan", yaitu sebuah sekuens lengkap yang bisa menghabiskan semua kartu di tangannya. Jika sebuah jalur kemenangan ditemukan, AI akan memainkan langkah pertama dari jalur optimal tersebut.
- Prioritas 2, Mode Aman
  Di tengah permainan, seringkali jalur kemenangan yang pasti belum tentu ada. Jika metode solve() tidak menemukan solusi lengkap dan mengembalikan null, AI tidak langsung menyerah. Sebaliknya, ia akan beralih ke "mode aman", ia akan mengiterasi seluruh kartunya dan memainkan langkah valid pertama yang bisa ia temukan. Ini memastikan AI tetap bermain secara logis meskipun tidak ada jaminan kemenangan.
- Prioritas 3, Pass
  AI akan melakukan pass, jika dan hanya jika kedua prioritas gagal. Hal ini akan terjadi saat tidak ada satu pun kartu yang bisa dimainkan di papan

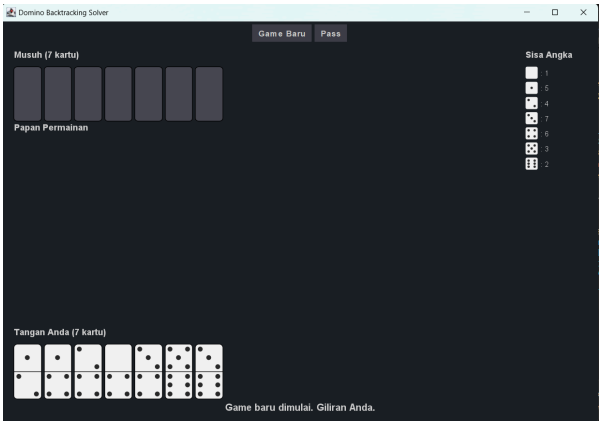
Tabel 3.3 Implementasi Algoritma Runut Balik

```
private List<Move> solve(List<Domino> hand,
Deque<Domino> board, int leftEnd, int rightEnd) {
   if (hand.isEmpty()) {
      return new ArrayList<>();
   }

   for (int i = 0; i < hand.size(); i++) {
      Domino domino = hand.get(i);

      List<Domino> nextHand = new ArrayList<>(hand);
      nextHand.remove(i);

      if (domino.getVal1() == rightEnd || domino.getVal2()
== rightEnd) {
         board.addLast(domino);

         int nextRightEnd = (domino.getVal1() == rightEnd)
? domino.getVal2() : domino.getVal1();

         List<Move> solution = solve(nextHand, board,
leftEnd, nextRightEnd);
```

```
            board.removeLast();

            if (solution != null) {
                solution.add(0, new Move(domino, "KANAN"));
                return solution;
            }
        }

        if ((leftEnd != rightEnd) && (domino.getVal1() ==
leftEnd || domino.getVal2() == leftEnd)) {
            board.addFirst(domino);

            int nextLeftEnd = (domino.getVal1() == leftEnd) ?
domino.getVal2() : domino.getVal1();

            List<Move> solution = solve(nextHand, board,
nextLeftEnd, rightEnd);

            board.removeFirst();

            if (solution != null) {
                solution.add(0, new Move(domino, "KIRI"));
                return solution;
            }
        }
    }

    return null;
}
```

D. Implementasi Antarmuka Grafis

Keseluruhan antarmuka pengguna grafis atau GUI untuk aplikasi ini dibangun menggunakan library Java Swing, dengan GamePanel sebagai kanvas utama tempat semua komponen visual digambar. Proses penggambaran yang ditangani oleh metode paintComponent memanfaatkan objek Graphics2D untuk menghasilkan visual yang lebih halus dan kontrol yang lebih baik atas elemen-elemen yang ada. Salah satu implementasi visual yang penting adalah papan permainan dinamis, di mana posisi rantai domino dihitung secara real-time agar selalu tampil terpusat di layar.



Gambar 3.4 Ilustrasi Proses Backtracking

Logika ini juga secara otomatis mendeteksi kartu angka kembar untuk digambar secara vertikal, sementara kartu lainnya tetap horizontal, sesuai dengan standar permainan. Untuk memberikan umpan balik yang jelas kepada pengguna, kartu yang sedang dipilih akan diberi sorotan berwarna kuning transparan. Selain itu, sebagai fitur strategis, di bagian pojok kanan atas layar ditampilkan panel informasi jumlah sisa dari setiap nilai angka, yang datanya diperbarui setiap giliran untuk membantu pemain dalam mengambil keputusan.

IV. Pengujian Dan Analisis

Pada bab ini, dilakukan serangkaian pengujian untuk memverifikasi fungsionalitas dan menganalisis kinerja dari aplikasi permainan Domino yang telah diimplementasikan. Pengujian difokuskan pada dua aspek utama, yaitu kebenaran logika permainan interaktif dan efektivitas dari AI lawan.

A. Skenario Pengujian

Pengujian difokuskan pada tiga skenario kunci yang merepresentasikan kondisi kritis dalam alur permainan:

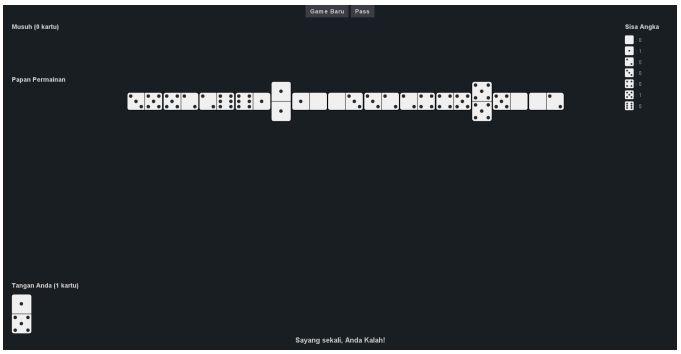
1. Skenario Langkah Optimal AI
   Sebuah kondisi permainan disiapkan di mana AI memiliki setidaknya satu jalur kemenangan yang dapat ditemukan oleh algoritma runut-balik. Skenario ini bertujuan untuk menguji apakah AI mampu mengidentifikasi dan mengeksekusi langkah pertama dari jalur kemenangan tersebut.
2. Skenario Musuh Tidak Memiliki Langkah (Pass)
   Skenario ini menguji kemampuan AI untuk mendeteksi bahwa tidak ada satu pun kartu di tangannya yang bisa dimainkan, sehingga AI harus melakukan "pass" dan melewatkan gilirannya.
3. Skenario Permainan Macet (Stuck)
   Sebuah kondisi disiapkan di mana kedua pemain pada akhirnya tidak dapat bergerak. Tujuannya adalah untuk menguji apakah program dapat mendeteksi kondisi macet dengan benar (kedua pemain pass berturut-turut) dan menentukan pemenang berdasarkan perhitungan skor sisa kartu terendah.

B. Hasil Pengujian

Berikut adalah hasil eksekusi setiap skenario pengujian yang telah dirancang

1. Pengujian Skenario Langkah Optimal AI

Pengujian ini bertujuan untuk memverifikasi bahwa AI mampu menggunakan algoritma runut-balik untuk mengidentifikasi sebuah jalur kemenangan.

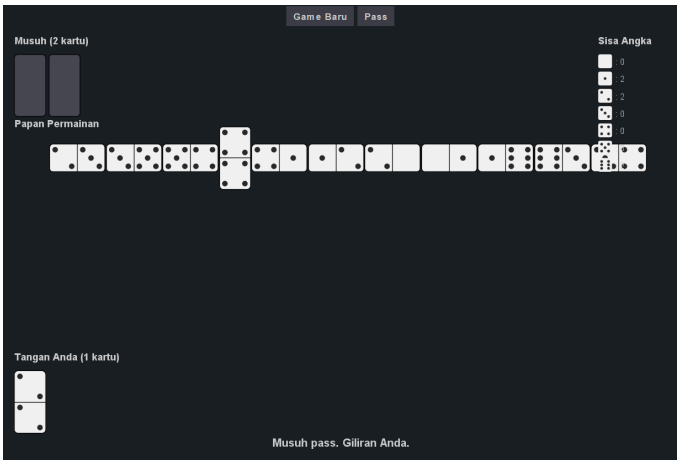


Gambar 4.2.1 AI berhasil memenangkan permainan

Hasil ini menunjukkan bahwa implementasi algoritma runut-balik pada AI telah berhasil. AI tidak hanya memilih langkah acak yang valid, tetapi mampu melakukan pencarian di dalam pohon ruang status untuk menemukan sebuah sekuens langkah yang optimal dari sudut pandangnya. Ini membuktikan bahwa komponen inti dari kecerdasan buatan telah berfungsi sesuai perancangan.

2. Pengujian Skenario Musuh Pass

Pengujian ini bertujuan untuk memverifikasi kemampuan AI untuk mendeteksi bahwa tidak ada kartu valid yang bisa dimainkan dan kemudian melakukan aksi "pass" dengan benar.



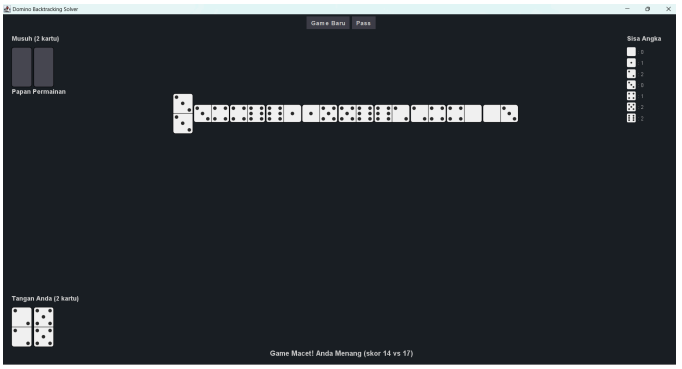

```
Musuh mencari langkah...
Musuh tidak punya langkah dan pass.
```

Gambar 4.2.2 AI melakukan “pass” karena tidak ada kartu yang memungkinkan

Pengujian ini berhasil memvalidasi logika AI dalam menangani kasus di mana tidak ada langkah yang bisa diambil. AI mampu memeriksa seluruh tangannya terhadap kedua ujung papan yang valid dan membuat keputusan yang benar untuk melewatkan gilirannya, alih-alih menyebabkan *error* atau berhenti. Ini menunjukkan bahwa logika dasar permainan telah diimplementasikan.

3. Pengujian Skenario Permainan Macet

Pengujian ini bertujuan untuk memverifikasi kemampuan program dalam mendeteksi kondisi macet dan menentukan pemenang berdasarkan skor sisa kartu.




```
Musuh mencari langkah...
Musuh tidak punya langkah dan pass.
Game Selesai: Game Macet! Anda Menang (skor 14 vs 17)
```

Gambar 4.2.3 AI melakukan “pass” karena tidak ada kartu yang memungkinkan

Skenario ini membuktikan bahwa mekanisme deteksi permainan macet dan perhitungan skor telah diimplementasikan dengan benar. Program mampu melacak status 'pass' dari kedua pemain dan memicu fungsi calculateHandScore untuk menentukan pemenang berdasarkan aturan skor terendah. Ini menunjukkan bahwa semua kondisi akhir permainan telah ditangani.

4. Tabel Hasil Pengujian

Hasil seluruh skenario pengujian dirangkum dalam tabel di bawah ini

Tabel 4.2.4 Rangkuman Hasil Pengujian Fungsionalitas

No.	Skenario Pengujian	Hasil Aktual	Kesesuaian
1	Skenario Langkah Optimal AI	Musuh mencari langkah... AI memainkan: [0|1] di kiri. Musuh mencari langkah... AI memainkan: [2|3] di kanan. Musuh mencari langkah... Musuh tidak punya langkah dan pass. Musuh mencari langkah... AI memainkan: [2|6] di kiri. Musuh mencari	Ya

		langkah... AI memainkan: [5|5] di kanan. Musuh mencari langkah... AI memainkan: [2|5] di kiri. Musuh mencari langkah... AI memainkan: [0|2] di kanan. Musuh mencari langkah... AI memainkan: [3|5] di kiri. Game Selesai: Sayang sekali, Anda Kalah!	
2	Skenario Musuh Pass	Musuh tidak punya langkah dan pass. Musuh mencari langkah...	Ya
3	Skenario Permainan Macet	Musuh mencari langkah... Musuh tidak punya langkah dan pass. Game Selesai: Game Macet! Anda Menang (skor 14 vs 17)	Ya

C. *Analisis Hasil*

Berdasarkan hasil pengujian yang disajikan pada Tabel 4.2.4, dapat disimpulkan bahwa fungsionalitas inti dari aplikasi permainan Domino telah berjalan sesuai perancangan. Algoritma runut-balik terbukti berhasil diimplementasikan sebagai otak dari AI, yang ditunjukkan pada keberhasilan Skenario 1 di mana AI mampu bermain secara optimal. Selain itu, program juga mampu menangani kondisi-kondisi khusus dalam permainan seperti saat AI harus melewatkan giliran (Skenario 2) dan saat permainan berakhir macet (Skenario 3), di mana penentuan pemenang berdasarkan skor berhasil dilakukan dengan benar. Secara keseluruhan, implementasi AI dan logika permainan sudah solid dan fungsional.

V. Kesimpulan dan Saran

Bab ini berisi kesimpulan yang ditarik dari keseluruhan proses perancangan, implementasi, dan pengujian aplikasi permainan Domino, serta saran untuk pengembangan di masa yang akan datang.

A. *Kesimpulan*

Berdasarkan perancangan dan pengujian yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Algoritma runut-balik telah berhasil diimplementasikan sebagai inti dari kecerdasan buatan untuk permainan Domino. Algoritma ini terbukti mampu menemukan sebuah jalur kemenangan yang optimal dari sebuah kondisi permainan jika jalur tersebut memang ada, yang ditunjukkan pada keberhasilan skenario pengujian langkah optimal AI.
2. Pendekatan AI hibrida, yang menggabungkan pencarian solusi optimal dengan mode aman, menciptakan sebuah lawan komputer yang tangguh namun tetap realistis. AI tidak akan "mogok" atau *pass* jika masih memiliki langkah yang bisa dimainkan, meskipun langkah tersebut tidak menjamin kemenangan.
3. Aplikasi permainan Domino interaktif yang fungsional telah berhasil dibangun menggunakan bahasa pemrograman Java dan *library* Swing. Aplikasi ini mampu menangani seluruh alur permainan, mulai dari pembagian kartu, giliran pemain dan musuh, hingga penentuan pemenang berdasarkan kondisi kartu habis atau skor terendah saat permainan macet.

B. *Saran*

Meskipun aplikasi yang dibangun sudah fungsional, terdapat beberapa area yang dapat dikembangkan lebih lanjut. Saran utama adalah meningkatkan kecerdasan AI dengan menambahkan logika heuristik untuk memprediksi kartu lawan dan melakukan blokade yang lebih efektif. Selain itu, fitur permainan dapat diperkaya dengan mengimplementasikan boneyard di mana pemain dapat mengambil kartu jika tidak bisa bergerak. Untuk pengembangan jangka panjang, aplikasi ini berpotensi untuk dikembangkan menjadi mode multiplayer melalui jaringan dan menambahkan opsi pengaturan tingkat kesulitan AI untuk memberikan pengalaman bermain yang lebih bervariasi bagi pengguna.

Kode Program

Link GitHub :

https://github.com/Abrar-Abhirama/DominoWithBackTracking

Ucapan dan Terima Kasih

Penulis mengucapkan rasa syukur yang sebesar-besarnya kepada Tuhan Yang Maha Esa atas segala nikmat dan petunjuk-Nya, sehingga penulis dapat menyelesaikan penelitian ini dengan lancar. Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada dosen pengampu mata kuliah IF2211, Dr. Nur Ulfa Maulidevi, S.T, M.Sc. dan Dr. Ir. Rinaldi Munir, M.T., atas segala pengabdian dan usahanya yang luar biasa dalam mengajar dan membimbing mahasiswa. Bimbingan, ilmu, dan perhatiannya sangat membantu penulis dalam memahami pokok bahasan dan menyelesaikan tugas ini. Semoga segala kebaikan dan ilmu yang telah dibagikannya dapat memberikan manfaat yang abadi.

Referensi

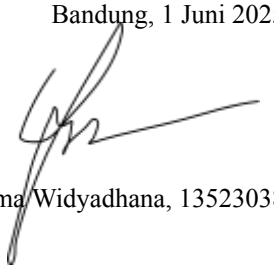
[1] R. Munir, "Algoritma Runut-balik (Backtracking) (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, Institut Teknologi Bandung, 2025.[Online].Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/15-Algoritma-backtracking-(2025)-Bagian1.pdf

[2] R. Munir, "Algoritma Runut-balik (Backtracking) (Bagian 2)," Bahan Kuliah IF2211 Strategi Algoritma, Institut Teknologi Bandung, 2025.[Online].Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/16-Algoritma-backtracking-(2025)-Bagian2.pdf

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, 1 Juni 2025



Abrar Abhirama Widyadhana, 13523038