JUSINFO
(Jurnal Sains dan Informatika)
Journal homepage: https://ejournal.pelitabina.com/index.php/Jusinfo
Pp: 18-28
Published By: Pelita Bina Neysha

Volume. 1 Nomor 1
Januari 2025

Analisis Efisiensi dan Kompleksitas Algoritma Backtracking dalam Permainan Math Maze

Virky Akassatya[1], Aqila Putri[2]

[1] Teknik Informatika, STMIK Pelita Nusantara, Medan (email: akassatyav@gmail.com)
[2] Teknik Informatika, STMIK Pelita Nusantara, Medan (email: putriaqilaaa25@gmail.com)

[Diserahkan: 23 Desember 2024, Direvisi: 30 Desember 2024, Diterima: 10 Januari 2025]
Corresponding Author: Virky Akassatya (email: akassatyav@gmail.com)

Kata Kunci: : Algoritma Backtracking, Permainan Math Maze, *Depth-First Search*, Kompleksitas, Pencarian Solusi

Abstrak : Kemajuan teknologi informasi saat ini mendorong kebutuhan akan metode pencarian solusi yang efisien dan optimal dalam berbagai aplikasi, termasuk permainan edukatif. Penelitian ini bertujuan untuk mengevaluasi efisiensi dan kompleksitas algoritma *backtracking* dalam permainan *Math Maze*. Permainan ini mengusung konsep labirin berbasis angka, di mana pemain harus menemukan jalur yang benar berdasarkan angka panduan di sisi grid. Metode yang digunakan adalah implementasi algoritma *backtracking* dengan pendekatan *Depth-First Search* (DFS) untuk membangun dan menyelesaikan labirin.Hasil penelitian menunjukkan bahwa algoritma *backtracking* mampu membangkitkan maze dengan solusi yang valid dan unik. Waktu rata-rata yang dibutuhkan untuk membentuk maze berukuran 10x10 adalah sekitar 0,5 detik, dengan peningkatan waktu seiring bertambahnya ukuran grid. Keunggulan utama dari algoritma ini mencakup keberhasilan 100% dalam menemukan jalur solusi, pencegahan loop atau ruang kosong yang tidak terpakai, serta kemampuan meminimalkan eksplorasi jalur yang tidak relevan. Penelitian ini menyimpulkan bahwa algoritma *backtracking* efektif dan efisien dalam menyelesaikan masalah eksplorasi jalur pada permainan *Math Maze*. Dengan solusi yang optimal serta struktur maze yang terarah, permainan ini menawarkan tantangan logis yang edukatif dan menarik bagi pemain.

PENDAHULUAN

Dalam era perkembangan teknologi informasi yang sangt pesat, kebutuhan akan pencarian data secara cepat, akurat, dan efisien menjadi hal yang esensial, terutama dalam pengelolaan perpustakaan dan penyelesaian masalah berbasis logika. Salah satu pendekatan algoritmik yang efektif untuk memecahkan masalah pencarian adalah algoritma *backtracking*. Algoritma ini memungkinkan eksplorasi solusi secara sistematis dengan melakukan penelusuran mendalam dan kemampuan untuk mundur (*backtrack*) ketika menemui jalan buntu.

Penelitian mengenai implementasi algoritma *backtracking* telah diterapkan dalam berbagai bidang, seperti pencarian judul buku di perpustakaan, pemecahan teka-teki labirin, permainan edukatif seperti Math Maze, dan permainan catur. Misalnya, dalam sistem informasi perpustakaan Universitas Muhammadiyah Bengkulu. algoritma *backtracking* membantu mempercepat pencarian judul buku, sehingga meningkatkan

efisiensi pelayanan perpustakaan. Sementara itu, dalam konteks permainan labirin, algoritma ini mampu menemukan rute keluar yang optimal dengan memangkas langkah-langkah yang tidak diperlukan.

Dengan fleksibilitas dan efisiensinya, algoritma *backtracking* menjadi solusi yang relevan untuk berbagai tantangan pencarian yang melibatkan eksplorasi banyak kemungkinan solusi. Penelitian ini bertujuan untuk membahas implementasi algoritma *backtracking* dalam konteks yang berbeda, serta mengidentifikasi efektivitasnya dalam menyelesaikan berbagai permasalahan yang memerlukan pencarian solusi optimal.

Backtracking adalah metode pencarian yang digunakan untuk menyelesaikan masalah dengan cara mencoba semua kemungkinan solusi secara rekursif. Jika suatu solusi yang sedang dicoba tidak mengarah ke hasil yang diinginkan, algoritma ini akan mundur (*backtrack*) ke langkah sebelumnya dan mencoba kemungkinan lainnya. Teknik ini berbasis pada algoritma *Depth-First Search* (DFS) dan dirancang untuk mencari solusi dalam ruang pencarian secara efisien. Algoritma *backtracking* adalah pengembangan dari algoritma *brute-force* yang bekerja dengan cara lebih sistematis dalam menjelajahi semua kemungkinan solusi untuk menemukan penyelesaian suatu permasalahan.

Algoritma *Backtracking* digunakan untuk memecahkan berbagai masalah yang melibatkan eksplorasi banyak kemungkinan solusi. Dalam pencarian judul buku di perpustakaan, algoritma ini membantu mengatasi masalah pencarian buku dalam *database* yang besar, seperti yang diterapkan di perpustakaan Universitas Muhammadiyah Bengkulu. Dalam pemecahan labirin, backtracking efektif menemukan rute keluar dengan mempertimbangkan semua kemungkinan jalur dan menghindari jalan buntu. Pada permainan edukatif seperti Math Maze, algoritma ini digunakan untuk menentukan jalur solusi yang melibatkan kombinasi angka dan logika. Selain itu, backtracking juga berguna untuk menghasilkan semua kombinasi atau permutasi elemen dalam himpunan, seperti yang diterapkan pada teka-teki sudoku dan berbagai jenis puzzle lainnya.

Algoritna *Backtrcking* menyediakan solusi dengan pendekatan yang sistematis dan efisien. Algoritma backtracking bekerja dengan mengeksplorasi setiap kemungkinan solusi secara mendalam. Jika suatu jalur tidak mengarah ke solusi yang diinginkan, algoritma akan mundur dan mencoba cara lainnya, sehingga proses ini memungkinkan penelusuran semua opsi yang tersedia. Dengan kemampuan untuk mundur, *backtracking* juga menghindari eksplorasi ulang terhadap jalur yang sudah terbukti tidak efektif. Kelebihan utama dari *backtracking* adalah kemampuannya untuk menemukan solusi optimal dengan memangkas langkah-langkah yang tidak diperlukan, menjadikannya lebih efisien dibandingkan metode *brute-force* murni yang memeriksa semua kemungkinan tanpa strategi eliminasi(Adiguna et al., 2020a).

Masalah *backtracking* mengacu pada penggunaan algoritma *backtracking* untuk menyelesaikan berbagai jenis persoalan yang memerlukan pencarian solusi secara terorganisir dan terstruktur. Masalah-masalah ini biasanya melibatkan eksplorasi ruang solusi yang luas dengan tujuan menemukan hasil yang valid atau optimal. Berdasarkan hal ini, beberapa contoh masalah yang sering diselesaikan menggunakan algoritma ini:

1. Menemukan jalan keluar dari labirin dengan mengeksplorasi setiap jalur yang memungkinkan dan mundur jika menemui jalan buntu. Menentukan rute pergerakan kuda catur untuk mengunjungi semua petak di papan catur tanpa mengunjungi petak yang sama lebih dari satu kali.
2. Menempatkan N buah ratu pada papan catur berukuran N x N sehingga tidak ada dua ratu yang saling menyerang, baik secara horizontal, vertikal, maupun diagonal.
3. Mengisi grid Sudoku sesuai dengan aturan tertentu, yaitu tidak ada angka yang berulang pada setiap baris, kolom, atau subgrid. Menghasilkan semua kemungkinan susunan elemen, seperti variasi urutan huruf untuk anagram atau kombinasi angka tertentu.
4. Membagi sekumpulan angka ke dalam dua subset dengan jumlah total nilai yang sama pada masing-masing subset. meliputi pencarian jalur Hamiltonian (jalur yang melewati setiap simpul di graf tepat satu kali), pewarnaan graf (memberikan warna pada simpul graf tanpa ada warna yang sama pada simpul bertetangga), atau mencari jalur tertentu sesuai dengan kriteria dalam graf.

Algoritma *Backtracking* menyelesaikan masalah-masalah tersebut dengan cara menelusuri solusi secara bertahap. Jika jalur yang dipilih tidak memenuhi kriteria, algoritma akan kembali ke langkah sebelumnya dan mencoba jalur lain, sehingga lebih efisien dibandingkan dengan metode *brute force*

TINJAUAN LITERATUR

Penelitian (Teneng et al., 2011a) dengan judul penelitian "Penerapan Algorima Backtracking Pada Permainan Math Maze". Algoritma backtracking (runut balik) adalah salah satu metode penyelesaian masalah yang menggunakan pendekatan berbasis pencarian dalam ruang status. Algoritma ini bekerja secara rekursif dengan menelusuri semua kemungkinan solusi secara sistematis. Karena algoritma ini didasarkan pada pendekatan *Depth-First Search* (DFS), pencarian solusi dilakukan dengan menelusuri struktur berupa pohon berakar menggunakan urutan *preorder*.

Penelitian (Serdano et al., 2019) dengan judul penelitian "Implementasi Algoritma Backtracking Pada Knight's Tour Problem". Algoritma Backtracking adalah algoritma yang berkaitan dengan *Depth-First Search* (DFS) untuk menemukan solusi secara lebih efisien dibandingkan dengan algoritma brute force. Secara umum, algoritma ini bekerja dengan memangkas (*pruning*) bagian-bagian yang tidak mengarah pada solusi dari seluruh kemungkinan yang ada. Penelitian ini bertujuan untuk mengimplementasikan algoritma Backtracking dalam menyelesaikan semua kemungkinan solusi pada permasalahan *Knight's Tour* di papan catur berukuran m x n. Dalam penelitian ini, papan catur yang digunakan berukuran 8 x 8.

Penelitian (Widjaja & Sudirman, 2013) dengan judul penelitian "Implementasi Algoritma Backtracking Dengan optimasi Menggunakan Teknik Hidden Single Pada Penyelesaian Permainan Sudoku". Algoritma ini bekerja dengan membangun solusi parsial dari suatu kandidat solusi dan mengevaluasi setiap solusi parsial tersebut secara bertahap. Jika solusi parsial yang dibentuk tidak memenuhi kriteria, maka proses pengembangan kandidat tersebut dihentikan, dan algoritma akan kembali (*backtrack*) ke kandidat lain yang masih memenuhi syarat. Proses backtracking ini dilakukan secara berulang hingga ditemukan solusi yang memenuhi semua kriteria yang ditentukan.

Penelitian (Rahmawati yunia, 2020) dengan judul penelitian "Penerapan Algoritma Backtracking Pada N-Queen Problem Permainan Catur". Penelitian ini menggunakan pendekatan metode kepustakaan (*library research*), yaitu dengan mengumpulkan data dari berbagai referensi, seperti jurnal, skripsi, dan dokumen lain yang relevan dengan topik penelitian. Selain itu, peneliti memanfaatkan strategi *Depth-First Search* (DFS) pada algoritma backtracking untuk proses pencarian solusi, serta menerapkan konsep pohon solusi (*State Space Tree*) sebagai representasi langkah-langkah yang dapat diambil dalam upaya menemukan solusi.

Penelitian (Putra et al., 2010) dengan judul penelitian "Penerapan dan Implementasi Algoritma Backtracking". Algoritma backtracking memiliki prinsip dasar yang mirip dengan brute-force, yaitu mencoba semua kemungkinan solusi. Namun, perbedaannya terletak pada pendekatan utamanya, di mana backtracking membentuk semua kemungkinan solusi dalam struktur seperti pohon solusi (yang bersifat abstrak). Algoritma ini kemudian menelusuri pohon tersebut menggunakan metode pencarian *depth-first search* (DFS) hingga menemukan solusi yang memenuhi kriteria.

Penelitian (|hasani & Dan Mustafidah, 2016) dengan judul penelitian "Algoritma Backtracking untuk Penyelesaian Puzzle Gambar Bendera". Penelitian ini membahas tentang pengembangan atau *Research and Development* (R&D), yang berfokus pada pembuatan perangkat lunak permainan dengan menerapkan algoritma backtracking untuk menyelesaikan puzzle gambar bendera. Penelitian ini melibatkan dua variabel utama, yaitu sembilan bagian benua dan gambar bendera negara. Kedua variabel tersebut akan diproses menggunakan algoritma backtracking. Dengan penerapan algoritma ini, diharapkan proses pencarian solusi pada puzzle gambar bendera dapat dilakukan secara efektif dan efisien.

Penelitian (Adiguna et al., 2020) dengan judul penelitian "Implementasi Algoritma Backtracking untuk Mencari Jalan Keluar Labirin". Labirin merupakan teka-teki yang sering digunakan dalam permainan, dengan banyak cabang dan jalan buntu yang membuat pencarian jalan keluar menjadi sulit. Algoritma pencarian dapat membantu dalam menemukan jalan keluar dari labirin, tetapi tidak semua algoritma pencarian cocok diterapkan pada labirin. Algoritma Backtracking adalah salah satu algoritma pencarian yang efisien, karena algoritma ini melakukan langkah mundur dari simpul tujuan untuk memeriksa apakah solusi yang sedang dicari mengarah pada simpul tujuan yang diinginkan.

METODE PENELITIAN

Algoritma *Backtracking* pertama kali diperkenalkan oleh D.H Lehmer pada tahun 1950. Algoritma ini cukup baik untuk digunakan dalam beberapa pemecahan masalah dan juga untuk memberikan kecerdasan buatan dalam permainan. Beberapa permainan populer seperti Sudoku, Labirin, Catur juga dapat diimplementasikan dengan menggunakan algoritma *Backtracking*. Algoritma *Backtracking* berbsis pada DFS (*Depth First Search*) sehingga aturan pencariannya akan mengikuti aturan pencarian DFS (*Depth First Search*) yaitu dengan mencari solusi dari akar ke daun (dalam pohon ruang solusi) dengan pencarian mendalam. Titik-titik yang sudah dibangkitkan (diperiksa) disebut simpul hidup (Live node). Simpul hidup yang sedang diperluas disebut simpul-E atau *Expand Node*(Yulyanto et al., 2023).

*Expand nod*e adalah langkah dalam algoritma pencarian di mana simpul yang sedang diproses dipecah menjadi simpul-simpul anaknya. Proses ini memungkinkan algoritma untuk menjelajahi lebih banyak kemungkinan solusi dengan mempertimbangkan semua opsi yang tersedia dari simpul tersebut. Dalam konteks algoritma backtracking dan pencarian lainnya, expand node merupakan bagian penting dari proses eksplorasi ruang solusi.

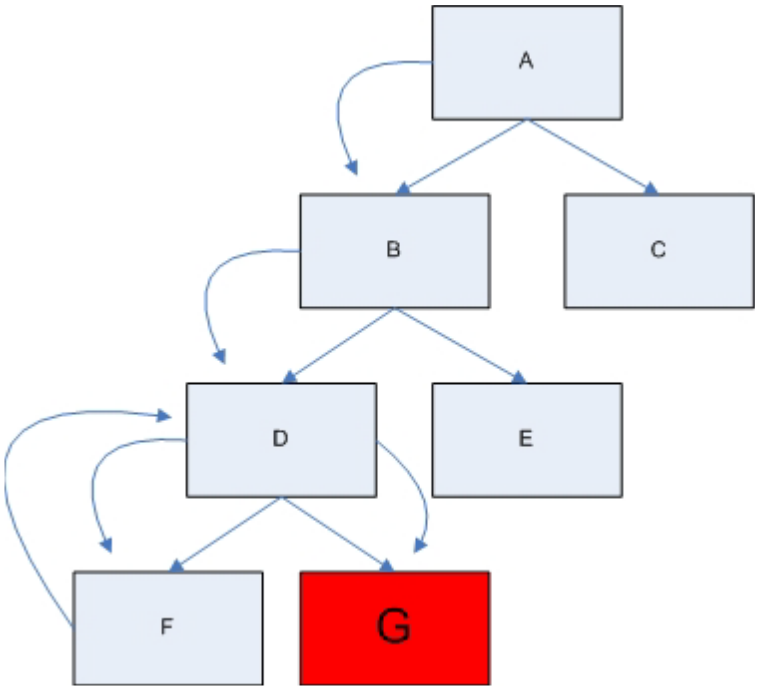
Algoritma *Backtracking* adalah metode pencarian yang digunakan untuk menyelesaikan masalah dengan cara mencoba semua kemungkinan solusi secara rekursif. Jika suatu solusi yang sedang dicoba tidak mengarah ke hasil yang diinginkan, algoritma ini akan mundur *(backtrack)* ke langkah sebelumnya dan mencoba kemungkinan lainnya. Teknik ini berbasis pada algoritma *Depth-First Search* (DFS) dan dirancang untuk mencari solusi dalam ruang pencarian secara efisien. Algoritma *backtracking* adalah pengembangan dari algoritma *brute-force* yang bekerja dengan cara lebih sistematis dalam menjelajahi semua kemungkinan solusi untuk menemukan penyelesaian suatu permasalahan.

Langkah-langkah pencarian solusi dengan backtracking adalah sebagai berikut:
Jika lintasan yang dihasilkan dari perluasan simpul-E tidak mengarah ke solusi, simpul tersebut menjadi simpul mati dan tidak akan diperluas lebih lanjut.Solusi dicari dengan membentuk lintasan dari akar ke daun. Simpul yang sudah dilahirkan dinamakan simpul hidup dan simpul hidup yang diperluas dinamakan simpul-E *(Expand-node*).

Jika lintasan yang dihasilkan dari perluasan simpul-E tidak mengarah ke solusi, simpul tersebut menjadi simpul mati dan tidak akan diperluas lebih lanjut. Ketika proses mencapai simpul mati, pencarian dilanjutkan dengan membangkitkan simpul anak lainnya. Jika tidak ada lagi simpul anak yang bisa dibangkitkan, dilakukan backtracking ke simpul induk (*parent*).

Proses pencarian akan berhenti ketika solusi ditemukan atau tidak ada lagi simpul hidup yang dapat diperluas.

Berikut dibawah ini gambar 1 merupakan flowchart dari algoritma Backtraking.



Sumber : (Rifqo & Apridiansyah, 2017)

Gambar 1. Contoh Algoritma Backtraking

- **Depth-First Search (DFS)**

Depth first search (DFS) pertama kali diperkenalkan oleh Tarjan dan Hopcroft pada abad ke-19. Mereka menunjukkan cara DFS dapat digunakan untuk membuat banyak algoritma grafik yang efektif. Metode pencarian mendalam adalah metode Depth First Search (DFS). Metodenya adalah dengan menelusuri salah satu kemungkinan yang tersedia hingga akar-akar maksimal, dan kemudian menelusuri kemungkinan lain hingga akar-akar maksimal juga.

Perlu diperhatikan bahwa kebutuhan waktu proses pencarian dengan DFS sejalan dengan kedalaman pohon pencarian yang paling tinggi. Algoritma mungkin tidak akan berhenti jika kedalaman pohon tidak terbatas. Ini dapat terjadi dalam kasus ruang pencarian tidak terbatas atau jika ruang pencarian mengandung siklus keadaan. Oleh karena itu, algoritma DFS hanya menunjukkan beberapa ruang pencarian dan tidak menunjukkan sifat komplitnya.

Pada penelitian ini penulis mengambil sampel dari jurnal/artikel dari (Teneng et al., 2011), berikut penjelasannya.

1. Identifikasi Masalah

Permainan Math Maze bertujuan untuk membantu pemain menemukan jalur yang tepat dalam sebuah labirin yang menggunakan angka sebagai panduan. Masalah yang diidentifikasi adalah bagaimana merancang dan mengimplementasikan papan permainan Math Maze yang efektif, serta bagaimana algoritma backtracking dapat digunakan untuk menghasilkan maze yang valid dan solusi yang efisien.

Penelitian ini menganalisis pengembangan permainan *Math Maze* yang memanfaatkan algoritma *backtracking*. Permainan ini dirancang untuk membantu pemain menemukan jalur yang tepat dengan mengikuti panduan angka yang terletak di sisi kiri dan atas papan permainan. Berbeda dengan permainan labirin tradisional yang menggunakan tembok sebagai penghalang, *Math Maze* mengharuskan pemain mengikuti angka-angka tersebut sebagai pedoman.

Salah satu tantangan dalam penelitian ini adalah memastikan bahwa setiap papan permainan yang dihasilkan memiliki solusi yang valid dan unik. Selain itu, pembangkitan papan permainan harus dapat

dilakukan secara dinamis untuk menyesuaikan kebutuhan pemain. Permasalahan lainnya adalah merancang algoritma yang efisien untuk diterapkan pada grid berukuran besar hingga 20x20. Tantangan ini mencakup fleksibilitas dalam penempatan titik awal (start) dan titik akhir (finish) yang dapat ditentukan oleh pemain. Selain itu, diperlukan metode untuk memastikan bahwa jalur yang dihasilkan tidak membentuk loop atau meninggalkan ruang yang tidak terpakai.

2. Pengumpulan Data

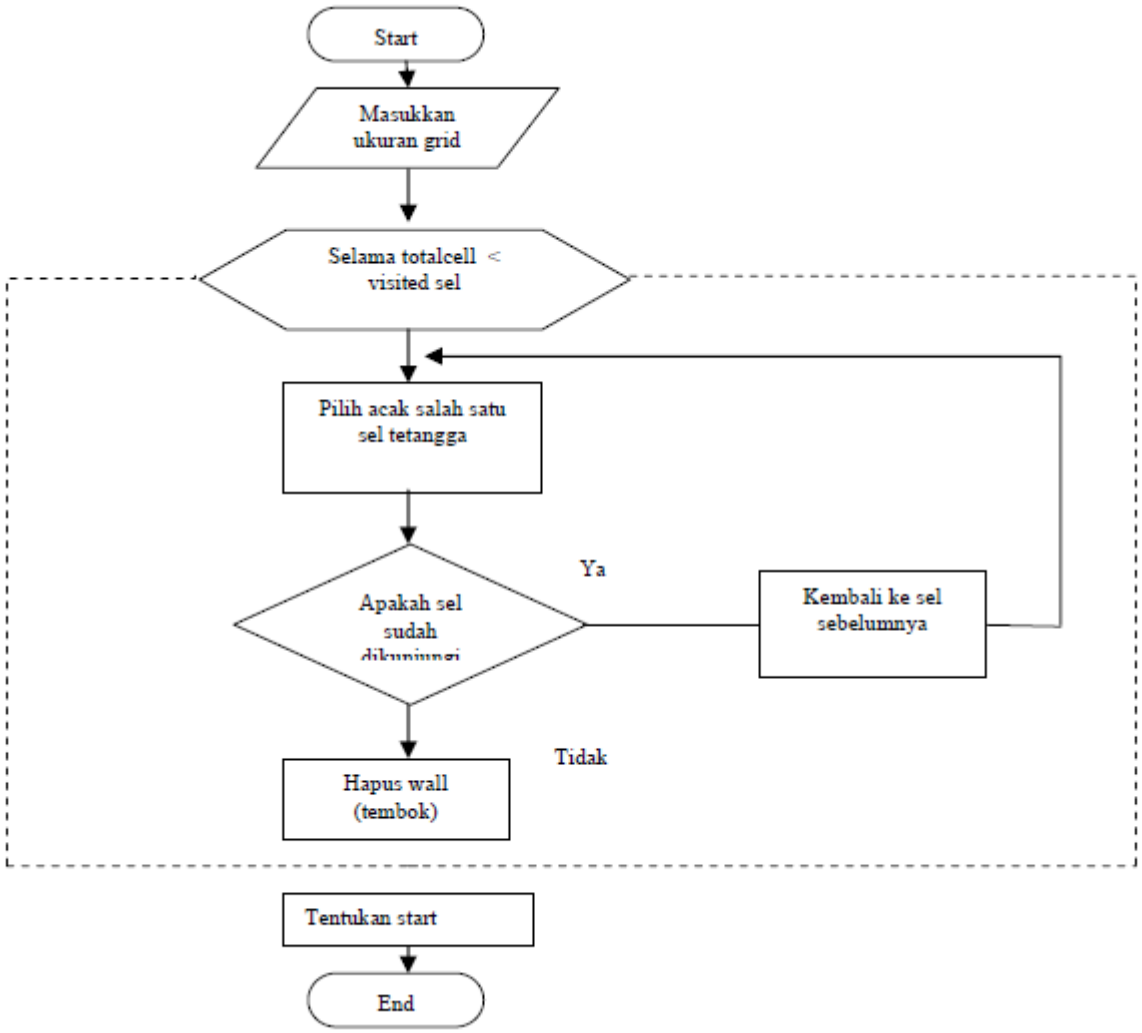
Penelitian ini mengacu pada berbagai teori dan konsep yang berkaitan dengan algoritma *backtracking*, metode pencarian *Depth First Search* (DFS), dan prinsip dasar permainan labirin. Beberapa sumber utama mencakup buku teks tentang kecerdasan buatan, jurnal penelitian terkait, serta artikel daring yang membahas implementasi algoritma backtracking dalam permainan dan aplikasi lainnya.

Kecerdasan buatan (AI) digunakan sebagai dasar dalam penelitian ini untuk menyelesaikan masalah yang membutuhkan eksplorasi ruang pencarian yang kompleks. Algoritma backtracking dipilih karena kemampuannya untuk mencari solusi secara sistematis dengan mengeksplorasi semua kemungkinan jalur, serta kemampuannya untuk kembali ke kondisi sebelumnya jika menemui jalan buntu. Selain itu, teori DFS mendukung pencarian solusi secara mendalam yang sesuai dengan kebutuhan permainan *Math Maze*.

3. Penerapan Algoritma Backtraking

Langkah pertama dalam Proses pembangkitan ini dilakukan dengan Algoritma *Backtracking* untuk membentuk sebuah *Maze*. Mulai

a. Masukkan ukuran *grid* (m x n)
b. Selama Total sel pada *grid* < sel yang telah di kunjungi lakukan langkah d, e, dan f.
c. Pilih secara acak sel tetangga (kanan, kiri , atas, bawah)
d. Cek apakah sel sudah pernah dikunjungi, jika ya kembali ke langkah d, jika tidak lakukan langkah f.
e. Hapus *wall* atau pembatas pada sel
f. Tentukan pintu masuk (start) dan goal (pintu keluar)
g. Selesai.



Gambar 2. Flowchart kebangkitan maze

Setelah maze terbentuk dan titik awal (pintu masuk) serta titik akhir (pintu keluar) ditentukan, proses pencarian solusi dilakukan menggunakan algoritma *Backtracking*. Algoritma ini bekerja dengan menelusuri setiap jalur yang memungkinkan untuk mencapai tujuan. Selama proses penelusuran, komputer akan mengikuti jalur yang tersedia secara acak. Jika menemui jalan buntu, komputer akan melakukan *backtrack* dengan kembali ke jalur sebelumnya untuk mencari jalur alternatif yang belum dijelajahi.

Dalam proses pembangkitan maze dan pencarian solusinya, arah gerakan yang diperbolehkan adalah ke atas, ke bawah, ke kiri, dan ke kanan. Gerakan lain, seperti diagonal atau melompati sel, tidak diperbolehkan. Setelah solusi ditemukan, tahap selanjutnya adalah menghitung jumlah sel yang dilewati pada jalur solusi di sepanjang baris dan kolom. Hasil perhitungan tersebut digunakan untuk membentuk angka-angka panduan di sisi kiri dan atas papan permainan yang membantu pemain menemukan jalur yang benar.

Langkah terakhir adalah menyembunyikan tampilan maze beserta solusinya, sehingga pemain hanya melihat grid dengan sel awal (start), sel tujuan (finish), dan angka-angka panduan di sisi kiri dan atas papan permainan. Algoritma dari program tersebut adalah sebagai berikut :

a. Mulai
b. Buat Stack generate untuk sturktur data
c. Set totalcell = jumlah sel pada grid
d. Pilih satu sel secara acak sebagai current sel
e. Set visited sel=1
f. Selama Visited sel < totalcell lakukan langkah g sampai i

g. Cek sel tetangga dari current sel, jika lebih dari 1 lakukan langkah h sampai k, jika =1 lakukan langkah l
h. pilih salah satu secara acak.
i. Simpan sebagai visited sel
j. Masukan (push) data visited sel ke stack generate.
k. Jadikan sel tetangga tersebut menjadi current sel yang baru
l. Selesai

Pada algoritma ini proses *backtracking* ditunjukan dengan dengan mengeluarkan data dari *stack generated* di mana data tersebut mengarah ke sel yang telah dikunjungi kembali ke data di mana sel masih memiliki kemungkinan pengecekan sel tetangga lainnya.

3.1 Algoritma Pencarian Solusi pada Maze yang Dibangkitkan

Algoritma yang digunakan untuk mencari solusi pada maze yang dihasilkan dapat dijelaskan sebagai berikut:

1. Inisialisasi Sel Awal: Pilih sel pertama secara acak dari grid dan tetapkan sebagai sel saat ini (ptCurrent).
2. Pengecekan Sel Tetangga: Periksa sel-sel di sekitar sel saat ini untuk mengetahui apakah sudah pernah dikunjungi. Karena ini adalah langkah awal, belum ada sel yang dikunjungi. Sel saat ini kemudian disimpan dalam *Stack generate*.
3. Arah Pengecekan: Pengecekan dilakukan dengan mengikuti empat arah yang ditentukan, yaitu:
    - Atas: (x, y-1)
    - Kanan: (x+1, y)
    - Bawah: (x, y+1)
    - Kiri: (x-1, y)
4. Proses ini berlanjut sampai jumlah total sel yang dikunjungi sama dengan jumlah total sel dalam grid, dan sel saat ini (ptCurrent) adalah sel tujuan (ptEnd). Jika ditemukan sel yang sudah dikunjungi, proses *backtracking* dilakukan dengan mengeluarkan data dari *stack generate*. Backtracking ini mengembalikan proses ke sel sebelumnya yang masih memiliki tetangga yang belum diperiksa.

Melalui proses ini, algoritma akan menemukan solusi untuk maze yang telah dihasilkan.

Hasil Percobaan:

Start (Baris, Kolom)	Finish (Baris, Kolom)	Apakah Solusi Di Tenukan?	Jumlah Solusi
0,0	**0,5**	**ya**	**1**
0,1	**0,4**	**ya**	**1**
0,2	**0,3**	**ya**	**1**
0,3	**0,2**	**ya**	**1**
0,4	**0,1**	**ya**	**1**
0,5	**0,0**	**ya**	**1**

Tabel 1. Pencarian solusi dengan Start dan Finish ditentukan Pemakai

HASIL PENELITIAN

Hasil dari penelitian mengenai penerapan algoritma *backtracking* dalam pengembangan permainan Math Maze menunjukkan tingkat keberhasilan yang tinggi dalam menghasilkan solusi yang valid dan efisien. Berikut adalah ringkasan hasil yang merangkum kinerja algoritma:

1. Tingkat Keberhasilan Algoritma
   Algoritma backtracking mampu menghasilkan solusi tunggal untuk setiap maze yang dibuat. Hal ini menunjukkan bahwa algoritma dapat menentukan jalur yang sesuai dengan panduan nilai yang diberikan.
2. Waktu Pemrosesan

Rata-rata waktu pemrosesan untuk membentuk maze berukuran 10x10 adalah sekitar 0,5 detik. Namun, durasi ini meningkat seiring bertambahnya ukuran maze, menandakan adanya hubungan langsung antara ukuran grid dan kompleksitas algoritma.

3. Keberhasilan Pencarian Solusi
   Algoritma backtracking menunjukkan keberhasilan 100% dalam menemukan jalur dari awal hingga pintu keluar pada setiap pengujian, membuktikan efektivitasnya dalam memecahkan masalah pencarian jalur pada permainan Math Maze.
4. Loop dan Ruang Terbuang
   Salah satu keunggulan utama algoritma ini adalah kemampuannya untuk membentuk maze tanpa loop atau ruang yang tidak terpakai. Hal ini memastikan bahwa setiap langkah permainan memiliki tujuan yang jelas tanpa redundansi.

Penelitian ini mengungkapkan bahwa algoritma backtracking adalah metode yang sangat handal untuk membangun permainan Math Maze. Dengan kemampuan menghasilkan solusi unik, efisiensi dalam pemrosesan waktu, dan keberhasilan tinggi dalam pencarian solusi, algoritma ini sangat cocok untuk menciptakan pengalaman bermain yang menarik dan menantang. Selain itu, maze yang dihasilkan memiliki struktur optimal, yang meningkatkan kualitas permainan secara keseluruhan dan memastikan jalur permainan lebih terarah.

No	Parameter yang Dinilai	Hasil	Keterangan
1	Jumlah Solusi yang Dihasilkan	1 solusi per problem	Setiap maze yang dibangkitkan menghasilkan satu solusi unik.
2	Waktu Eksekusi	Rata-rata 0.5 detik untuk ukuran 10x10	Waktu eksekusi meningkat seiring dengan ukuran maze.
3	Keberhasilan Pencarian Solusi	100% berhasil menemukan jalur	Semua percobaan berhasil menemukan jalur menuju pintu keluar.
4	Jumlah Loop dan Ruang Terbuang	Tidak ada loop atau ruang terbuang	Maze yang dihasilkan adalah maze sempurna tanpa siklus.

Tabel 2 Tabel Hasil Penelitian Algoritma Backtracking dalam Permainan Math Maze

PEMBAHASAN

Pembahasan penelitian ini Evaluasi terkait efisiensi dan kompleksitas algoritma backtracking dilakukan dalam konteks pengembangan sebuah permainan edukatif bernama Math Maze. Permainan ini dirancang untuk membantu pemain menemukan jalur yang tepat di dalam labirin angka. Algoritma backtracking diterapkan untuk membangun struktur labirin sekaligus mencari solusi dari jalur yang benar.

1. Efisiensi dan Kompleksitas Algoritma

Waktu pemrosesan untuk maze berukuran 10x10, rata-rata waktu pemrosesan adalah sekitar 0,5 detik. Namun, waktu eksekusi meningkat seiring dengan bertambahnya ukuran grid. Hal ini menunjukkan bahwa kompleksitas algoritma terkait langsung dengan ukuran maze. Algoritma backtracking meminimalkan eksplorasi jalur yang tidak relevan, sehingga lebih efisien dibandingkan metode brute-force murni.

2. Validitas Maze dan Solusi

Algoritma backtracking memastikan bahwa setiap maze yang dihasilkan memiliki solusi yang valid dan unik. Hal ini menghindari terbentuknya loop atau ruang yang tidak terpakai. Maze yang dihasilkan adalah maze sempurna (perfect maze), di mana tidak ada redundansi dalam jalur.

3. Keberhasilan Pencarian Solusi

Algoritma menunjukkan keberhasilan 100% dalam menemukan solusi, bahkan untuk maze dengan ukuran besar hingga 20x20. Pada setiap percobaan, algoritma mampu menemukan jalur dari titik awal ke titik akhir tanpa kesalahan.

4. Penerapan Angka Panduan

Angka panduan pada sisi grid membantu pemain dalam menemukan jalur solusi, membuat permainan lebih edukatif dan logis. Panduan ini didasarkan pada jumlah sel yang dilewati pada jalur solusi, sehingga mendukung gameplay yang menantang namun terstruktur.

5. Tantangan dan Fleksibilitas

Salah satu tantangan utama adalah memastikan bahwa algoritma tetap efisien untuk grid yang lebih besar. Penyesuaian titik awal dan akhir secara dinamis memberikan fleksibilitas tambahan pada permainan.

Algoritma backtracking terbukti mampu memberikan solusi optimal dan efisien dalam permainan Math Maze. Struktur maze yang dihasilkan memberikan pengalaman bermain yang menarik, logis, dan menantang, dengan memanfaatkan kemampuan algoritma untuk menghindari jalur tidak relevan dan menghasilkan jalur solusi yang unik.

KESIMPULAN

Secara keseluruhan, penelitian ini membuktikan bahwa algoritma backtracking adalah metode yang sangat efisien dalam pengembangan permainan Math Maze. Algoritma ini mampu menghasilkan solusi tunggal, memiliki waktu eksekusi yang optimal, serta tingkat keberhasilan yang sangat tinggi dalam pencarian jalur. Keunggulan ini menjadikannya andal untuk menciptakan pengalaman bermain yang seru dan menantang bagi para pemain. Selain itu, penerapan backtracking memastikan bahwa maze yang dihasilkan memiliki struktur yang optimal, sehingga meningkatkan kualitas keseluruhan permainan.

REFERENCES

|hasani, A. K., & Dan Mustafidah, H. (2016). Algoritma Backtracking untuk Penyelesaian Puzzle Gambar Bendera (Backtracking Algorithm for Completing Puzzle Flag). *Juita*, *2*(November), 87–95.

Adiguna, Y., Swanjaya, D., Informatika, T., Teknik, F., Nusantara, U., & Kediri, P. (2020a). Implementasi Algoritma Backtracking untuk Mencari Jalan Keluar Labirin. *Prosiding SEMNAS INOTEK ...*, 131–136. https://proceeding.unpkediri.ac.id/index.php/inotek/article/view/75

Adiguna, Y., Swanjaya, D., Informatika, T., Teknik, F., Nusantara, U., & Kediri, P. (2020b). Implementasi Algoritma Backtracking untuk Mencari Jalan Keluar Labirin. *Prosiding SEMNAS INOTEK ...*, 131–136. https://proceeding.unpkediri.ac.id/index.php/inotek/article/view/75

Putra, D. N., Sardjito, O. O., & Lawrence, C. (2010). Penerapan dan Implementasi Algoritma

Backtracking. *Departemen Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Bandung (ITB) Kampus ITB Jl. Ganesha No. 10 Bandung*, *10*, 5–7.

Rahmawati yunia. (2020). *Penerapan Algoritma Backtracking Pada N-Queen Problem Permainan Catur*. *3*(July), 1–23.

Serdano, A., Zarlis, M., & Hartama, D. (2019). Implementasi Algoritma Backtracking Pada Knight's Tour Problem. *Seminar Nasional Teknologi Informasi Dan Komunikasi STI&K (SeNTIK)*, *3*(1).

Teneng, T., Purwadi, J., & Kurniawan, E. (2011a). Penerapan Algoritma Backtracking Pada Permainan Math Maze. *Jurnal Informatika*, *6*(2). https://doi.org/10.21460/inf.2010.62.89

Teneng, T., Purwadi, J., & Kurniawan, E. (2011b). Penerapan Algoritma Backtracking Pada Permainan Math Maze. *Jurnal Informatika*, *6*(2). https://doi.org/10.21460/inf.2010.62.89

Widjaja, V. S., & Sudirman, D. Z. (2013). Implementasi Algoritma Backtracking Dengan Optimasi Menggunakan Teknik Hidden Single Pada Penyelesaian Permainan Sudoku. *Seminar Nasional Aplikasi Teknologi Informaso (SNATI)* , 15–2013.

Yulyanto, A. B., Maudia Arsyad, F., & Sugiharto, T. (2023). Pengembangan Game Puzzle Find Grass Menggunakan. *Bulletin of Information Technology (BIT)*, *4*(2), 275–280.