Laporan Akhir

Implementasi Algoritma Backtracking pada Game Labirin Escape untuk Pemecahan Maze Dinamis

Dibuat Oleh :

NIM	Nama
11422002	Benyamin Sibarani
11422018	Andien Laura T. Panjaitan
11422032	Handika Sukri Husni Harahap
11422050	Listra Imelda Sidabutar

Untuk :

Institut Teknologi Del

Sitoluama

Kabupaten Toba, Sumatera Utara, Indonesia

DAFTAR ISI

1 Pendahuluan

Bab Pendahuluan memberikan gambaran umum proyek, mencakup Deskripsi Proyek tentang konsep game Labirin Escape dengan implementasi algoritma backtracking untuk maze dinamis, Rumusan Masalah yang mengidentifikasi isu utama, Tujuan yang merumuskan hasil proyek, dan Scope yang menetapkan fitur, teknologi, serta batasan teknis untuk menjaga fokus pengembangan.

1.1 Latar Belakang

Labirin Escape adalah sebuah game interaktif yang dirancang untuk menantang pemain dalam menemukan jalan keluar dari labirin yang kompleks. Terinspirasi dari konsep labirin klasik, permainan ini menguji kemampuan orientasi, navigasi, serta pengambilan keputusan di lingkungan yang penuh liku. Meskipun konsepnya sederhana, pengembangan permainan ini melibatkan tantangan algoritmik yang signifikan, khususnya dalam merancang mekanisme petunjuk jalan yang efisien dan pengacakan labirin untuk memastikan setiap sesi permainan terasa unik.

Proyek ini memanfaatkan algoritma backtracking sebagai inti dari fitur petunjuk arah. Algoritma ini memungkinkan permainan memberikan fitur solve, yang dapat digunakan ketika pemain merasa buntu. Fitur ini menyediakan jalur yang benar menuju jalan keluar, sehingga membantu pemain melanjutkan permainan tanpa mengurangi esensi tantangan utama. Selain itu, fitur random maze generator diterapkan untuk menghasilkan labirin yang selalu berbeda setiap kali permainan dimulai atau diulang, mencegah pemain mengandalkan hafalan dan menambah elemen kejutan dalam setiap permainan.

Dikembangkan menggunakan bahasa pemrograman seperti Python. Dengan menggabungkan elemen permainan klasik dan teknologi modern, Labirin Escape menawarkan pengalaman bermain yang segar dan menantang.

Selain sebagai media hiburan, game ini juga memiliki nilai edukatif. Pemain dan pengembang dapat mempelajari penerapan algoritma komputasi dalam konteks nyata, seperti pemecahan masalah dinamis dan pengambilan keputusan berbasis logika. Proyek ini bertujuan tidak hanya untuk menghadirkan permainan yang menghibur, tetapi juga untuk menjadi sarana pembelajaran algoritma yang menarik dan aplikatif bagi pengguna.

1.2 Deskripsi Project

Labirin Escape adalah permainan interaktif yang dirancang untuk melatih keterampilan logika, strategi, dan navigasi pemain dalam menemukan jalan keluar dari labirin dengan tingkat kesulitan yang dinamis. Pemain dihadapkan pada tantangan untuk menghindari

jalan buntu dan mencapai tujuan akhir di tengah jalur labirin yang terus berubah, menciptakan pengalaman bermain yang seru dan menantang bagi berbagai kalangan.

Permainan ini dilengkapi dengan fitur solve, yang dirancang untuk membantu pemain ketika menghadapi kebuntuan. Fitur ini memberikan solusi berupa jalur yang benar secara langsung, memastikan pemain dapat melanjutkan permainan tanpa kehilangan semangat, sekaligus tetap menjaga aspek strategis dan menantang dalam permainan.

Proyek ini menggunakan algoritma backtracking sebagai inti logika permainan, yang memungkinkan solusi optimal ditemukan secara efisien untuk memenuhi kebutuhan gameplay utama dan pemberian solusi. Dengan penggabungan elemen permainan klasik dan teknologi modern, Labirin Escape tidak hanya menjadi sumber hiburan, tetapi juga menyediakan pengalaman pembelajaran yang mendalam tentang penyelesaian masalah secara sistematis.

1.3 Rumusan Masalah

Permasalahan utama yang ingin dipecahkan dalam proyek ini adalah:

1. Bagaimana cara menemukan solusi yang efisien untuk menyelesaikan labirin dinamis yang terus berubah?
2. Bagaimana mengimplementasikan algoritma backtracking secara efektif untuk menemukan jalur keluar dalam permainan labirin?
3. Bagaimana memastikan algoritma dapat beradaptasi dengan kompleksitas labirin yang di-generate secara acak?

1.4 Tujuan

Proyek ini bertujuan untuk:

1. Mengembangkan game Labirin Escape yang interaktif dan dinamis, di mana pemain ditantang untuk menemukan jalan keluar dari labirin yang dapat berubah-ubah.
2. Mengimplementasikan algoritma backtracking sebagai solusi utama untuk menyelesaikan labirin, termasuk fitur solve, yang memungkinkan pemain mendapatkan solusi lengkap saat merasa buntu.
3. Menguji performa algoritma backtracking dalam menyelesaikan berbagai skenario maze dinamis, termasuk tingkat keberhasilan, efisiensi waktu, dan stabilitas.
4. Memberikan solusi edukatif yang dapat memperkenalkan pemain pada prinsip-prinsip dasar algoritma dan penyelesaian masalah berbasis logika.
5. Menganalisis pengaruh kompleksitas labirin terhadap performa algoritma backtracking.

1.5 Lingkup (Scope)

Ruang lingkup proyek ini meliputi;

1. Pengembangan Game:
    - Labirin dinamis yang dapat berubah berdasarkan kondisi tertentu
    - Sistem kontrol pemain untuk navigasi dalam labirin.
    - Fitur solve, yang memberikan solusi lengkap ketika pemain merasa buntu.
2. Implementasi Algoritma:
- Penerapan algoritma backtracking untuk pencarian solusi jalur keluar.
- Integrasi algoritma dengan fitur solve untuk menampilkan jalur yang benar kepada pemain.
3. Lingkup Teknologi:
- Bahasa pemrograman yang digunakan: Python.
- Teknologi visualisasi: Representasi grafis labirin dan jalur solusi.
4. Batasan Proyek:
- Labirin yang dihasilkan bersifat dua dimensi.
- Dinamika labirin terbatas pada pola perubahan yang telah dirancang sebelumnya.
- Pengujian dilakukan pada perangkat desktop dengan performa standar.
5. Fitur yang Tidak Dikembangkan:
- Tidak ada implementasi multiplayer atau mode permainan online.
- Tidak mencakup algoritma lain selain backtracking untuk penyelesaian maze.

2 Metodologi

Bab Metodologi menjelaskan langkah-langkah dan prosedur yang digunakan dalam pengembangan proyek, mencakup desain, pengumpulan data, serta tahapan implementasi untuk mencapai hasil yang diinginkan.

2.1 Pendekatan Pengembangan

Pendekatan yang digunakan dalam proyek ini adalah pendekatan berbasis algoritma, dengan fokus pada penerapan algoritma backtracking untuk menyelesaikan maze dinamis. Pendekatan ini dipilih karena algoritma backtracking dapat menangani kompleksitas labirin dengan baik melalui proses eksplorasi dan pengembalian (backtracking) yang sistematis. Selain itu, pendekatan incremental digunakan dalam pengembangan game untuk memungkinkan validasi di setiap tahap penyelesaian fitur utama.

2.2 Tahapan Pelaksanaan

1. Analisis Masalah

   Tahap ini melibatkan identifikasi kebutuhan permainan dan permasalahan yang akan diselesaikan. Langkah-langkahnya meliputi:

   - Menentukan format labirin, seperti ukuran grid dan posisi awal serta tujuan.
   - Menganalisis kemungkinan kendala seperti jalan buntu, labirin bercabang, dan labirin tanpa Solusi.
   - Mengidentifikasi kebutuhan fitur tambahan, seperti fitur solve untuk memberikan solusi parsial kepada pemain.

1. Desain Solusi

   Tahap perancangan sistem mencakup langkah-langkah utama dalam merancang struktur data dan logika algoritma untuk pemecahan masalah. Labirin direpresentasikan sebagai matriks 2D, di mana setiap elemen menunjukkan status sel, seperti jalan, tembok, atau pintu keluar, sehingga mempermudah identifikasi kondisi setiap posisi.

   Logika algoritma yang digunakan adalah backtracking berbasis Depth First Search (DFS), yang efektif untuk menelusuri jalur dari titik awal ke tujuan. Implementasinya dilakukan secara rekursif, dengan memeriksa setiap langkah yang memungkinkan pada tiap arah. Jika menemui jalan buntu, algoritma akan kembali (backtrack) ke langkah sebelumnya untuk mencoba alternatif jalur lain, hingga solusi ditemukan atau semua kemungkinan habis dievaluasi.

2. Implementasi

   Tahap implementasi mencakup penerapan algoritma dan pengembangan fitur utama dalam kode program untuk menciptakan labirin yang interaktif. Proses ini meliputi:

- Pengkodean Fitur Utama: Mengembangkan mekanisme untuk menggerakkan pemain menggunakan input keyboard atau mouse, serta menerapkan algoritma backtracking untuk solusi otomatis dari titik awal ke tujuan.
- Integrasi Fitur Solusi: Menyediakan solusi parsial bagi pemain yang kesulitan, dengan menampilkan jalur secara visual langsung di labirin.
- Pengembangan Antarmuka Pengguna (UI): Mendesain labirin sebagai grid visual interaktif yang menampilkan jalur, tembok, dan pintu keluar, serta menambahkan tombol "Solve" untuk mengaktifkan fitur solusi.

3. Pengujian dan Validasi
   Tahap ini bertujuan untuk memastikan bahwa sistem berfungsi sesuai dengan spesifikasi yang telah ditetapkan melalui serangkaian langkah pengujian yang menyeluruh.
   - Pengujian Fungsional: Langkah ini difokuskan pada validasi setiap fitur utama dalam sistem. Pengujian memastikan pemain dapat bergerak melalui labirin sesuai dengan input yang diberikan, algoritma backtracking mampu menghasilkan solusi yang benar, dan fitur "Solve" memberikan petunjuk yang valid serta dapat diandalkan.
   - Pengujian Performa: Pengujian ini bertujuan mengevaluasi efisiensi sistem, khususnya dalam waktu eksekusi algoritma. Performa diuji untuk berbagai ukuran labirin guna memastikan algoritma tetap responsif dan dapat menangani skenario dengan kompleksitas yang lebih tinggi.

2.3 Tools dan Teknologi

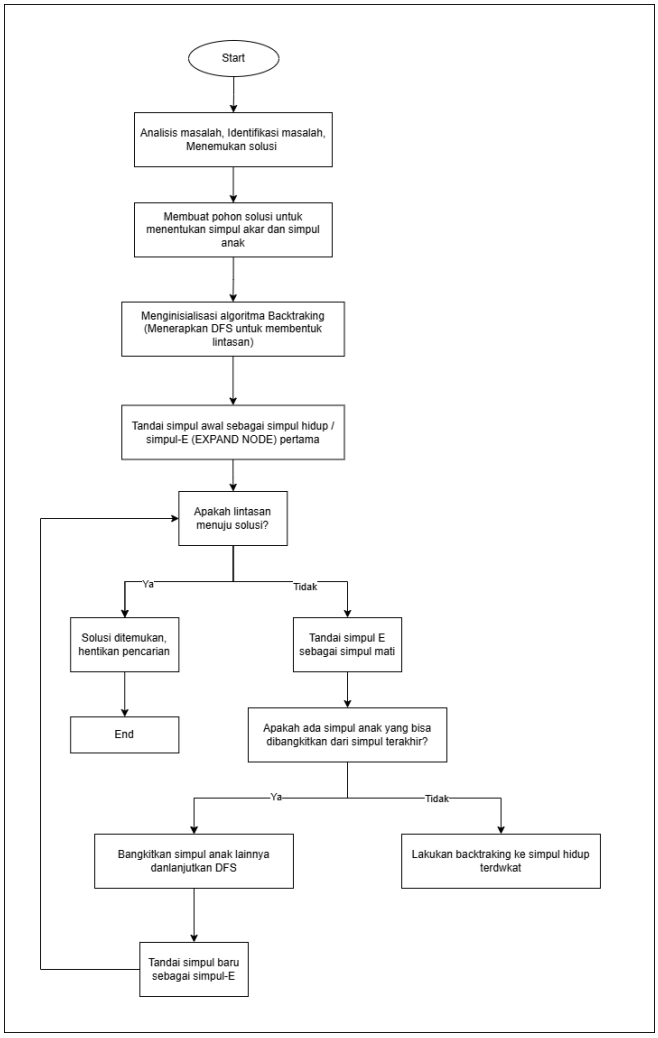
Untuk mendukung pengembangan proyek, berbagai tools dan teknologi berikut diterapkan:

- Bahasa Pemrograman: Python dipilih karena kemudahan sintaksisnya yang sederhana, serta kemampuan untuk mendukung implementasi algoritma backtracking secara efisien. Python juga menawarkan banyak pustaka (library) yang kaya, memudahkan pengembangan berbagai fitur.
- Library:
  - turtle: Digunakan untuk membuat visualisasi grafis interaktif, membantu menggambarkan perjalanan solusi dalam game Labirin Escape secara visual dan menarik.
  - time: Memfasilitasi pengaturan waktu dalam program, seperti memberikan jeda waktu pada tampilan atau interaksi pengguna menggunakan fungsi seperti time.sleep().

    - random: Digunakan untuk menghasilkan angka acak, yang berguna dalam elemen-elemen acak dalam game, seperti penempatan rintangan atau posisi awal karakter.
- IDE: Visual Studio Code dipilih sebagai lingkungan pengembangan karena kemudahan penggunaannya, dukungan ekstensi yang luas, serta fitur-fitur produktivitas yang meningkatkan efisiensi pengkodean.
- Version Control: GitHub digunakan untuk mengelola versi proyek, memungkinkan kolaborasi yang lebih baik, pelacakan perubahan kode, serta kemudahan dalam mengelola pengembangan proyek secara terstruktur.

2.4 Diagram Alur Metode Proses Penerapan Backtracking

Berikut adalah diagram alur yang menggambarkan tahapan proses penerapan algoritma backtracking dalam menyelesaikan maze dinamis pada game Labirin Escape:



Gambar 1. Diagram Alur Metode Algoritma Backtracking

Penjelasan Diagram Alur Metode Proses Penerapan Backtracking

- Pengumpulan Data
  Langkah pertama adalah mengumpulkan data terkait maze yang akan dipecahkan, termasuk ukuran labirin, posisi awal, dan tujuan akhir. Data ini akan digunakan untuk membangun representasi maze yang akan diproses dalam algoritma backtracking.
- Menganalisis Masalah, Identifikasi Tujuan, dan Menentukan Solusi

Setelah data terkumpul, langkah berikutnya adalah menganalisis maze untuk memahami bagaimana cara bergerak di dalamnya. Tujuan dari algoritma backtracking adalah menemukan jalur yang menghubungkan titik awal dengan titik tujuan. Berdasarkan analisis ini, solusi berupa pencarian jalur akan ditentukan.

- Perancangan Aplikasi
  Selanjutnya, aplikasi atau program untuk game Labirin Escape dirancang. Pada tahap ini, struktur data seperti matriks maze akan disiapkan, serta pemrograman logika dasar untuk menggerakkan karakter dalam labirin.
- Implementasi Coding
  Di tahap ini, kode program ditulis untuk mengimplementasikan seluruh logika yang dibutuhkan. Salah satu bagian terpenting adalah implementasi algoritma backtracking, yang akan digunakan untuk mencari jalur menuju tujuan.
- Implementasi Algoritma Backtracking
  Pada tahap ini, algoritma backtracking diterapkan pada maze. Berikut adalah alur dari proses backtracking yang akan dilakukan:
    - Perluasan Lintasan: Algoritma akan mencoba memperluas lintasan atau jalur dengan menggerakkan karakter ke simpul-simpul baru dalam maze.
    - Pengecekan Simpul: Setiap simpul yang diperluas akan diperiksa untuk memastikan apakah lintasan tersebut mengarah ke solusi (yaitu, tujuan akhir maze). Jika lintasan tidak mengarah ke solusi, maka simpul tersebut dianggap "mati" (dead node).
    - Penghapusan Simpul Mati: Simpul mati adalah simpul yang tidak lagi bisa diperluas ke tujuan. Simpul mati tidak akan digunakan lagi dalam pencarian.
    - Pencarian Solusi Lebih Lanjut: Jika lintasan berakhir dengan simpul mati, algoritma akan mencari simpul anak lainnya untuk diteruskan. Jika tidak ada simpul anak yang tersisa, algoritma melakukan backtracking untuk mundur ke simpul yang sebelumnya masih dapat dilanjutkan. Proses ini akan berlanjut sampai solusi ditemukan.
    - Runut Balik (Backtracking): Jika semua simpul yang diperluas sudah tidak mengarah ke solusi, algoritma akan kembali ke simpul hidup sebelumnya, mencoba jalur lain. Proses ini terus berlanjut, mundur ke simpul-simpul sebelumnya jika perlu, hingga solusi ditemukan atau tidak ada lagi jalur yang bisa dicoba.
- Pencarian Dihentikan
  Proses pencarian dihentikan dalam dua kondisi:
    - Solusi Ditemukan: Jika jalur ditemukan yang menghubungkan titik awal dengan tujuan, pencarian dihentikan, dan solusi dicatat sebagai jalur yang menghubungkan titik tersebut.

- Tidak Ada Simpul Hidup: Jika semua simpul sudah diperiksa dan tidak ada jalur yang mengarah ke tujuan, berarti tidak ada solusi yang ditemukan, dan pencarian dihentikan.

3 Landasan Teori

Bab ini membahas teori-teori dasar yang menjadi landasan dalam pengembangan proyek Labirin Escape. Pembahasan mencakup algoritma backtracking, dan kelebihan dan kekurangannya sebagai metode penyelesaian masalah.

3.1 Algoritma Backtracking

Algoritma Backtracking adalah teknik pemrograman berbasis pencarian (searching) yang digunakan untuk menyelesaikan masalah dengan mencoba semua kemungkinan solusi secara sistematis. Pendekatan ini bekerja secara rekursif dengan mengeksplorasi satu jalur solusi, mundur (backtrack) jika jalur tersebut tidak valid, dan mencoba jalur lain hingga solusi ditemukan atau semua kemungkinan dievaluasi.

Prinsip Dasar Algoritma Backtracking

1. Eksplorasi: Mengevaluasi semua kemungkinan keputusan pada setiap tahap.
2. Validasi: Memastikan langkah yang diambil memenuhi kriteria solusi.
3. Backtrack: Mundur ke langkah sebelumnya jika jalur saat ini tidak mengarah ke solusi.

3.2 Kelebihan dan Kekurangan Algoritma Backtracking

Kelebihan

1. Sederhana dan Fleksibel: Mudah diimplementasikan untuk berbagai jenis masalah.
2. Menjamin Solusi (Complete): Mampu menemukan semua solusi jika ruang pencarian lengkap dievaluasi.
3. Pendekatan Sistematis: Mengurangi redundansi dalam pencarian solusi dengan menggunakan validasi pada setiap langkah.

Kekurangan

1. Kompleksitas Waktu Tinggi: Pada masalah dengan banyak kemungkinan solusi, algoritma dapat menjadi tidak efisien.
2. Ketergantungan pada Validasi: Keberhasilan algoritma sangat bergantung pada fungsi validasi yang baik.
3. Tidak Optimal: Backtracking sering kali mencari solusi yang "cukup baik" tetapi bukan yang paling optimal.

4 Pembahasan

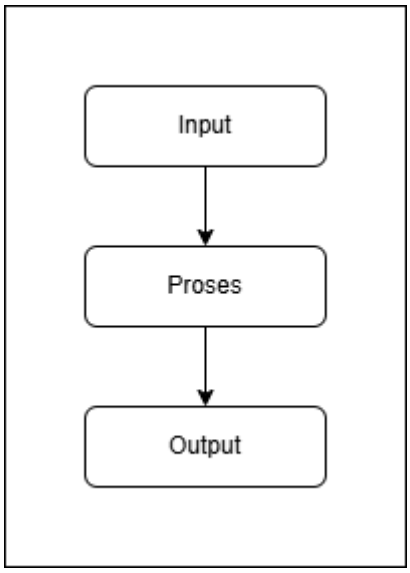
Bab ini menjelaskan proses teknis pengembangan proyek, mulai dari arsitektur algoritma, skema kerja algoritma, hingga implementasi algoritma backtracking dan desain antarmuka pengguna.

4.1 Arsitektur Algoritma

Pada sub bab ini akan dijelaskan arsitektur algoritma yang digunakan dalam pengembangan game Labirin Escape. Arsitektur ini menjelaskan tahapan dari input hingga proses dan output yang dihasilkan. Proses ini mencakup bagaimana algoritma backtracking digunakan untuk menyelesaikan permasalahan maze dinamis.

Diagram Arsitektur Algoritma

Gambaran arsitektur dari implementasi algoritma backtracking pada game Labirin Escape adalah sebagai berikut:



Gambar 2. Arsitektur Algoritma Backtracking

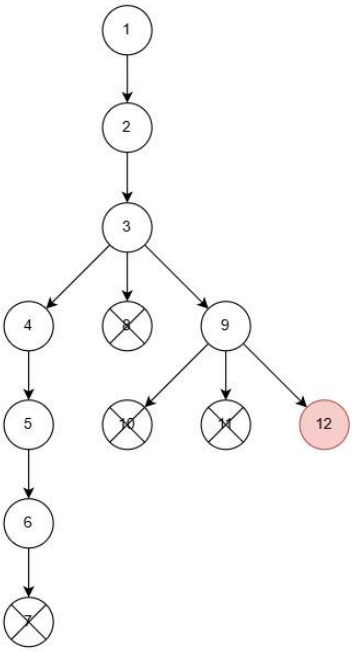
1. Input:
    - Maze dinamis berupa grid (matriks) dengan simpul awal (start) dan simpul tujuan (exit).
    - Kondisi maze yang dapat berubah ketika pemain berinteraksi dengan permainan.
2. Proses:
    - Implementasi algoritma backtracking untuk menelusuri semua kemungkinan jalur dari simpul awal ke simpul tujuan.

- Langkah-langkah proses melibatkan rekursi dan mekanisme backtrack jika jalur yang dicoba tidak valid atau menemui jalan buntu.

3. Output:
   - Jalur yang berhasil ditemukan dari simpul awal menuju simpul tujuan.
   - Informasi tambahan seperti visualisasi proses penyelesaian atau solusi jika pemain menggunakan fitur “solve”.

4.2 Skema Kerja Backtracking

Proses pencarian solusi dilakukan dengan membentuk lintasan dari akar hingga daun pada sebuah pohon pencarian. Metode yang digunakan adalah Depth First Search (DFS), di mana simpul-simpul baru yang dihasilkan selama pencarian disebut sebagai simpul hidup. Simpul hidup yang sedang diperluas dinamakan simpul-E. Setiap simpul diberi nomor secara berurutan dari atas ke bawah sesuai urutan kelahirannya.



Gambar 3. Pohon Solusi Permasalahan

Pohon pada Gambar 3 menggambarkan jalur solusi yang dibentuk dari akar menuju daun. Simpul hidup pada contoh ini adalah simpul bernomor 7, 8, 10, 11, dan 12. Simpul hidup dapat diperluas, tetapi jika tidak memungkinkan atau tidak mengarah ke solusi, maka simpul tersebut menjadi simpul mati (ditandai dengan silang pada gambar, seperti simpul 7, 8, 10, dan 11).

Ketika sebuah simpul mati ditemukan, pencarian dilanjutkan dengan membangkitkan simpul anak lainnya. Proses ini terus berjalan hingga solusi ditemukan, yaitu simpul 12 (ditandai dengan warna pada gambar).

Untuk menyelesaikan permasalahan ini, digunakan algoritma backtracking, yang bekerja dengan memeriksa setiap jalur dari akar ke daun:

1. Pencarian dimulai dari simpul 7, dengan jalur (1 → 2 → 3 → 4 → 5 → 6 → 7). Jika simpul ini bukan solusi, pencarian dilanjutkan.
2. Berikutnya, simpul 8 diperiksa melalui jalur (1 → 2 → 3 → 8). Jika bukan solusi yang optimal, pencarian diteruskan.
3. Algoritma kemudian mencoba simpul 10, melalui jalur (1 → 2 → 3 → 9 → 10). Jika simpul ini juga bukan solusi, pencarian dilanjutkan ke simpul 11.

Simpul 10 dan 11 memiliki jalur awal yang sama, yaitu (1 → 2 → 3). Untuk menghindari pemeriksaan ulang, hasil lintasan yang sudah dihitung disimpan, sehingga algoritma langsung memeriksa simpul 11.

Pencarian ini diulang hingga algoritma menemukan solusi yang benar-benar optimal, yaitu simpul 12. Algoritma backtracking memastikan bahwa setiap kemungkinan diperiksa secara sistematis, namun tetap menghindari pengulangan jalur yang sama untuk meningkatkan efisiensi.

5 Implementasi

Bab ini menjelaskan penerapan algoritma backtracking pada pengembangan game.

5.1 Penerapan Algoritma Backtracking

1) Algoritma Pembangkit Labirin

Algoritma pembangkit labirin menggunakan pendekatan Depth-First Search (DFS) untuk menghasilkan jalur secara acak di dalam labirin, dengan memanfaatkan teknik backtracking. Proses dimulai dari sebuah sel awal yang ditandai sebagai telah dikunjungi. Algoritma secara acak memilih tetangga yang belum dikunjungi, menghapus dinding antara kedua sel, dan berpindah ke tetangga tersebut, sambil menyimpan posisi saat ini ke dalam stack. Jika tidak ada tetangga yang tersedia dari sel saat ini, algoritma akan kembali (backtrack) ke sel sebelumnya menggunakan stack. Proses ini berlanjut hingga semua sel telah dikunjungi. Penggunaan stack sebagai penyimpan jejak perjalanan memastikan algoritma dapat menangani situasi jalan buntu, sementara penandaan sel menghindari kunjungan berulang, sehingga menghasilkan labirin dengan jalur unik tanpa siklus.

```python
# Fungsi untuk menghasilkan labirin besar dan lebih kompleks
def generate_maze(rows, cols):
    class MazeCell:
        def __init__(self, x, y):
            self.x, self.y = x, y  # Koordinat sel labirin
            self.visited = False  # Status apakah sel sudah dikunjungi
            self.walls = [True, True, True, True]  # Dinding: Atas, Kanan, Bawah, Kiri
```

Gambar 4. Algoritma Pembangkit Labirin

2) Algoritma Pencari Solusi

Fungsi ini dirancang untuk mencari solusi dari titik awal ke titik akhir menggunakan pendekatan pencarian berbasis frontier. Frontier berfungsi sebagai penyimpan posisi yang akan dieksplorasi selanjutnya, memastikan setiap langkah pencarian terorganisasi. Jika tidak ada jalur yang valid dari posisi saat ini, fungsi akan menandai posisi tersebut sebagai sudah dievaluasi, kemudian melanjutkan eksplorasi ke posisi berikutnya di dalam frontier. Proses ini berulang hingga solusi ditemukan atau semua kemungkinan jalur telah dievaluasi.

```python
# Fungsi untuk mencari solusi labirin menggunakan metode pencarian jalur
Tabnine | Edit | Test | Explain | Document | Ask
def search(x, y):
    global solver_in_progress

    if solver_in_progress:  # Jika solver sudah berjalan, hentikan eksekusi baru
        return

    solver_in_progress = True
    frontier.append((x, y))  # Mulai dari posisi awal
    solution[x, y] = x, y  # Menyimpan jalur dari posisi awal

    while len(frontier) > 0:  # Selama masih ada titik dalam frontier
        time.sleep(0.05)  # Menambahkan jeda untuk menampilkan proses pencarian
        current = frontier.pop()  # Ambil titik terakhir dari frontier
        current_x, current_y = current

        # Jika titik saat ini adalah titik akhir
        if (current_x, current_y) == (end_position_x, end_position_y):
            solution_marker.goto(current_x, current_y)
            solution_marker.stamp()
            print("Labirin selesai!")
            break
```

```python
        # Tandai jalur yang sedang dijelajahi
        backtrack_marker.goto(current_x, current_y)
        backtrack_marker.stamp()

        # Mengeksplorasi semua arah yang mungkin
        directions = [
            (current_x - 24, current_y),  # Kiri
            (current_x, current_y - 24),  # Bawah
            (current_x + 24, current_y),  # Kanan
            (current_x, current_y + 24)   # Atas
        ]

        explored = False  # Flag untuk melacak apakah ada langkah yang valid

        for next_x, next_y in directions:
            if (next_x, next_y) in path and (next_x, next_y) not in visited:
                explored = True
                frontier.append((next_x, next_y))  # Tambahkan ke frontier
                solution[next_x, next_y] = current  # Simpan jalur
                visited.append((next_x, next_y))

        # Jika tidak ada jalur yang ditemukan (jalan buntu), tandai dengan warna biru
        if not explored:
            wall.goto(current_x, current_y)
            wall.stamp()
            visited.append((current_x, current_y))

        # Tandai langkah yang dikunjungi
        solution_marker.goto(current_x, current_y)
        solution_marker.stamp()

    print("Pencarian selesai!")
```

Gambar 5. Algoritma Pencari Solusi

3) Algoritma Melacak Kembali Jalur dan Pencarian Solusi

Kode ini menggunakan dua fungsi utama, yaitu backtrack dan start_solver, untuk melacak kembali jalur solusi dan memulai pencarian solusi.

a. **Fungsi backtrack:**

Fungsi ini bertugas melacak kembali jalur solusi dari titik akhir ke titik awal menggunakan struktur data berupa dictionary bernama solution. Dictionary ini menyimpan hubungan antara titik saat ini dengan titik sebelumnya yang dilalui. Proses dimulai dari titik akhir, dan fungsi akan terus mengunjungi setiap titik yang dicatat dalam solution hingga mencapai titik awal. Setelah jalur ditemukan, fungsi ini menampilkan rute solusi secara berurutan.

b. Fungsi start_solver:

Fungsi ini bertanggung jawab memulai proses pencarian solusi. Fungsi ini pertama-tama memanggil fungsi search untuk mengeksplorasi jalur dari titik awal ke titik akhir menggunakan algoritma pencarian tertentu. Setelah pencarian selesai, fungsi ini melanjutkan dengan memanggil backtrack untuk menelusuri kembali jalur solusi yang ditemukan.

Secara keseluruhan, algoritma ini bekerja secara terorganisasi dengan memisahkan proses eksplorasi dan pelacakan solusi, memastikan bahwa jalur yang dihasilkan optimal dan mudah ditelusuri kembali.

```
# Fungsi untuk melacak kembali jalur solusi dari titik akhir ke titik awal
def backtrack(x, y):
    path_marker.goto(x, y)  # Mulai dari titik akhir
    path_marker.stamp()
    while (x, y) != (start_position_x, start_position_y):  # Sampai mencapai titik awal
        path_marker.goto(solution[x, y])
        path_marker.stamp()
        x, y = solution[x, y]  # Pindah ke titik sebelumnya

# Fungsi untuk memulai pencarian solusi
def start_solver():
    global solver_in_progress

    if not solver_in_progress:
        start_position_x, start_position_y = player.xcor(), player.ycor()
        search(start_position_x, start_position_y)
        backtrack(end_position_x, end_position_y)
```

Gambar 6. Algoritma Melacak Kembali Jalur dan Pencarian Solusi

A. Fitur-fitur Utama

Fitur-fitur utama dalam permainan labirin berbasis algoritma backtracking ini mencakup berbagai elemen yang dirancang untuk meningkatkan pengalaman pengguna. Berikut adalah penjelasan mengenai fitur-fitur utamanya:

- Labirin Dinamis

  Permainan ini menggunakan algoritma untuk menghasilkan labirin secara acak setiap kali pemain memulai permainan. Hal ini memastikan bahwa pemain menghadapi tantangan baru di setiap sesi, meningkatkan daya tarik dan replayability.

- Algoritma Backtracking

  Algoritma backtracking digunakan untuk menyelesaikan labirin secara otomatis. Fitur ini membantu menunjukkan solusi terbaik untuk menemukan jalur dari titik awal ke titik akhir, sehingga pengguna dapat mempelajari logika penyelesaian labirin.

- Fungsi Restart

  Fitur ini memungkinkan pemain untuk memulai ulang permainan kapan saja. Fungsi ini mengatur ulang labirin, posisi pemain, dan status permainan, memberikan pemain kesempatan untuk mencoba ulang dengan labirin baru atau strategi berbeda tanpa harus keluar dari aplikasi.

- Indikator Solusi

  Permainan menyediakan visualisasi jalur yang benar untuk menyelesaikan labirin. Fitur ini sangat berguna sebagai referensi untuk pemain yang

mungkin mengalami kesulitan atau ingin belajar bagaimana labirin dapat diselesaikan.

- Interaktivitas dan Kontrol

  Pemain dapat mengontrol pergerakan karakter melalui antarmuka sederhana, seperti keyboard atau tombol pada layar. Ini memberikan pengalaman bermain yang intuitif dan responsif.

- Tampilan Responsif dan User-Friendly

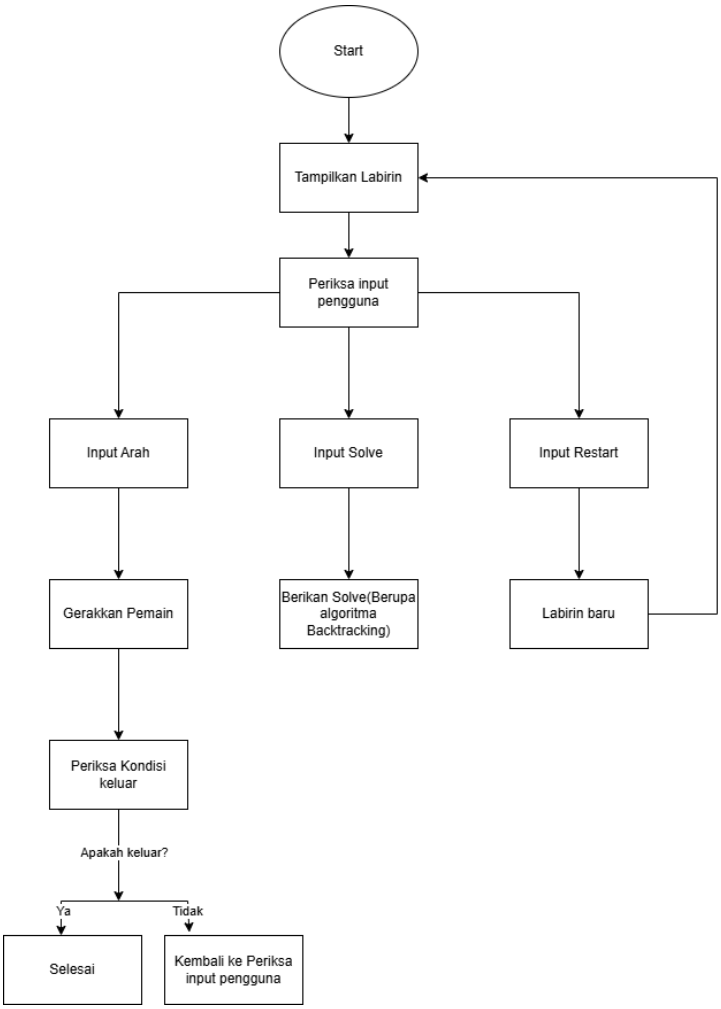
  Antarmuka permainan dirancang dengan tampilan yang menarik dan responsif, memastikan permainan dapat berjalan dengan baik di berbagai perangkat, termasuk desktop dan ponsel.

- Penilaian Performa Pemain

  Beberapa versi permainan juga dilengkapi dengan fitur untuk melacak waktu yang dibutuhkan pemain dalam menyelesaikan labirin, memberikan elemen kompetitif yang memotivasi pemain untuk meningkatkan performa.

B. Proses Output Game

Berikut adalah proses dalam menghasilkan output game, mulai dari awal permainan hingga pemain menyelesaikan labirin.



Gambar 7. Proses Output Game

1. Mulai Permainan
   Pada langkah ini, permainan dimulai dengan inisialisasi. Sistem secara otomatis mengacak struktur labirin dan menentukan posisi awal pemain. Posisi awal biasanya berada di titik masuk tertentu, sedangkan titik keluar diacak untuk memberikan variasi tantangan kepada pemain.
2. Tampilkan Labirin
   Setelah inisialisasi, labirin ditampilkan pada layar. Tampilan ini mencakup representasi grafis dari dinding labirin, jalur yang dapat dilalui, serta titik masuk dan titik keluar. Elemen ini memberikan visualisasi yang jelas bagi pemain untuk memulai eksplorasi.
3. Periksa Input Pengguna

Sistem menunggu masukan dari pemain, yang dapat berupa pergerakan ke arah tertentu (misalnya, atas, bawah, kiri, atau kanan), permintaan petunjuk jalur, atau pengaturan ulang permainan melalui tombol restart.

4. Input Arah - Gerakkan Pemain
   Ketika pemain memberikan input arah (misalnya, menekan tombol panah pada keyboard), sistem memeriksa apakah jalur di arah tersebut tersedia. Jika jalur tersedia, pemain akan dipindahkan ke posisi baru. Jika tidak, pemain tetap berada di posisi sebelumnya, menandakan bahwa dinding labirin menghalangi jalur tersebut.
5. Input Solve - Berikan Solve
   Jika pemain meminta petunjuk dengan menekan tombol solve, algoritma backtracking aktif. Sistem akan menampilkan sebagian jalur yang benar, memberikan petunjuk tanpa memperlihatkan solusi lengkap. Hal ini dirancang agar pemain tetap tertantang untuk menemukan sisa jalur sendiri.
6. Input Restart - Labirin Baru
   Ketika tombol restart ditekan, labirin diacak ulang, menghasilkan konfigurasi baru. Pemain kemudian dikembalikan ke posisi awal. Fitur ini memberikan kesempatan untuk mencoba ulang dengan tantangan yang berbeda.
7. Periksa Kondisi Keluar
   Program secara terus-menerus memeriksa apakah posisi pemain sudah berada di titik keluar. Jika pemain mencapai titik tersebut, sistem mengenali bahwa permainan telah selesai.
8. Selesai
   Permainan berakhir ketika pemain berhasil mencapai titik keluar. Pada tahap ini, sistem dapat memberikan notifikasi kemenangan atau ringkasan waktu yang dibutuhkan untuk menyelesaikan labirin.

6. Hasil

Bab ini akan membahas hasil yang diperoleh dari penerapan algoritma yang telah dijelaskan sebelumnya, serta analisis mengenai efektivitas dan kinerja algoritma dalam menyelesaikan masalah yang dihadapi.

a) Pembuatan Labirin

Proses pembuatan labirin dimulai dengan menggunakan algoritma DFS (Depth First Search) untuk membangkitkan labirin acak. Algoritma ini memastikan bahwa setiap sel dalam labirin dapat diakses, dan dinding-dinding labirin akan dihapus secara acak sesuai dengan aturan untuk menciptakan jalur yang dapat dilalui. Sebuah maze grid dua dimensi yang terdiri dari karakter + (dinding) dan (jalur kosong) dihasilkan, di mana posisi mulai ('s') dan keluar ('e') ditandai secara spesifik.

Labirin yang dihasilkan memiliki struktur yang cukup kompleks, dengan banyak kemungkinan jalur, serta adanya penambahan jalur tambahan dengan probabilitas tertentu untuk memberikan variasi. Penggunaan algoritma DFS dengan backtracking memungkinkan labirin untuk dibuat dengan jalur yang panjang dan memerlukan pencarian yang lebih rumit untuk menemukan solusi.

b) Implementasi Algoritma Backtracking

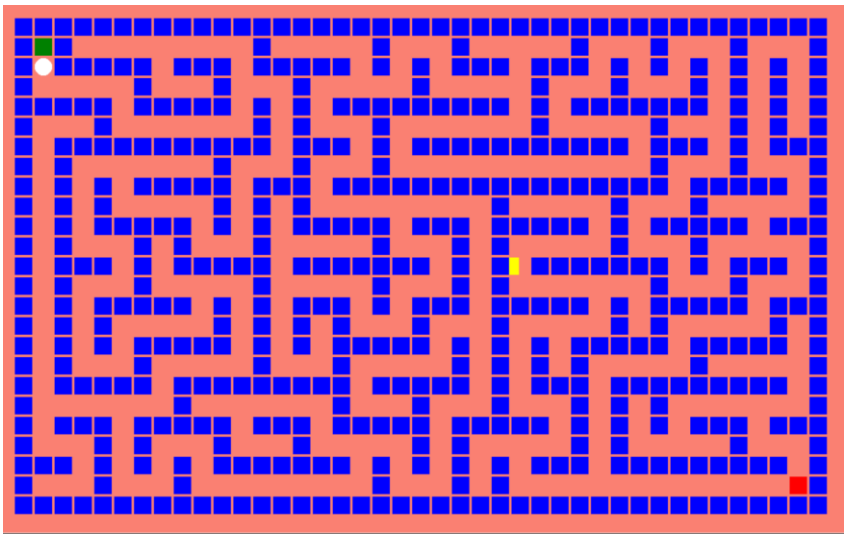
Setelah labirin terbentuk, pemain dapat memulai pencarian untuk menyelesaikan labirin dengan menggunakan algoritma backtracking. Algoritma pencarian mencari jalan dari posisi awal (titik 's') menuju posisi akhir (titik 'e') dengan memeriksa tiap sel yang mungkin dijangkau. Setiap sel yang dijelajahi oleh algoritma akan ditandai, dan jika tidak ada jalur yang dapat ditempuh lebih lanjut,

algoritma akan melakukan backtracking, yakni kembali ke titik sebelumnya yang memiliki jalur yang belum dieksplorasi.

Proses pencarian ditampilkan secara visual dengan penanda yang berbeda:

- Penanda jalur berwarna hijau menunjukkan jalur yang sedang dijelajahi.
- Penanda backtracking berwarna merah menunjukkan sel yang ditinggalkan karena tidak dapat diteruskan.
- Penanda solusi berwarna kuning menunjukkan jalur yang membentuk solusi dari labirin.

Pencarian berhenti ketika titik akhir ditemukan, dan jalur solusi ditampilkan di layar.

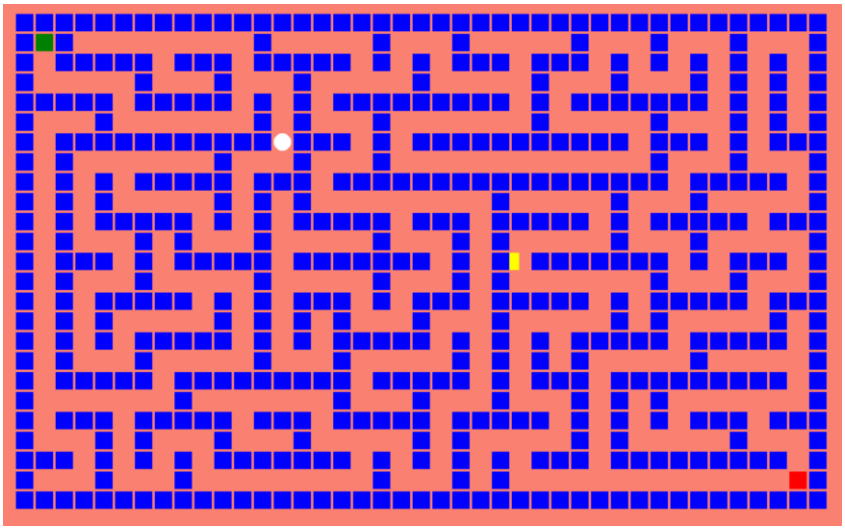


c) Interaksi Pemain dengan Permainan

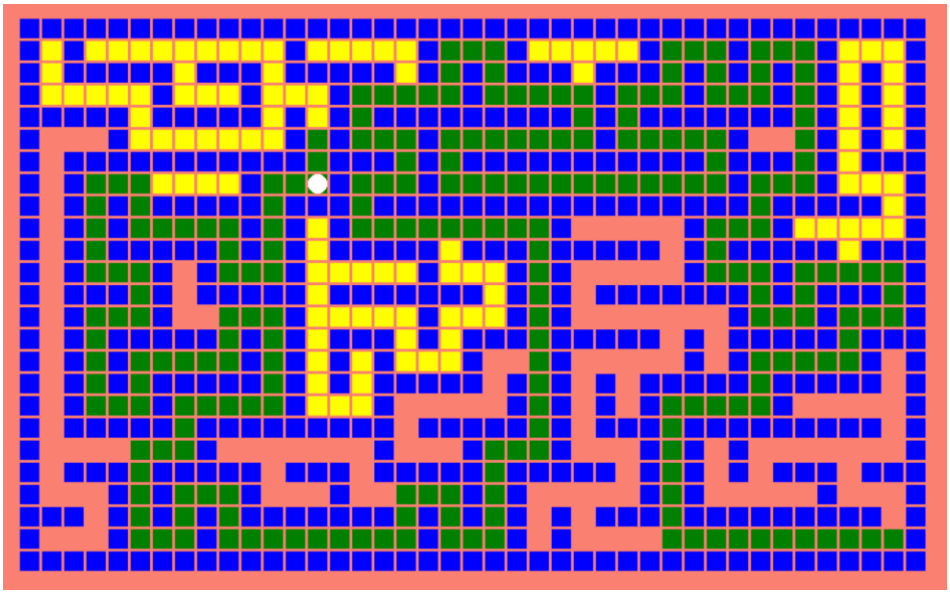
Pemain dapat menggerakkan karakter mereka di dalam labirin menggunakan tombol arah pada keyboard. Pemain dapat bergerak ke atas, bawah, kiri, atau kanan, dengan batasan bahwa mereka hanya bisa bergerak ke sel yang merupakan jalur terbuka dan tidak bertabrakan dengan dinding. Setiap pergerakan pemain diperiksa menggunakan fungsi can_move, yang memastikan bahwa gerakan yang diinginkan tidak menabrak dinding.

Proses pengecekan ini memungkinkan pemain untuk berinteraksi dengan labirin dan mencoba menemukan jalur menuju titik keluar secara manual. Jika pemain mencapai titik keluar, permainan akan

berakhir, dan pemain dapat memulai kembali dengan labirin baru yang dihasilkan secara acak.



d) **Penggunaan Fungsi Pencarian dan Backtracking**
Untuk mempercepat penyelesaian labirin, terdapat opsi untuk menggunakan fungsi otomatis yang akan mencari solusi menggunakan algoritma pencarian jalur dan backtracking. Fungsi search memulai pencarian dari titik awal, dan setiap langkah yang diambil oleh algoritma akan ditampilkan secara visual. Setelah menemukan solusi, algoritma akan melakukan backtracking untuk menunjukkan jalur yang telah diambil dari titik akhir ke titik awal, memberikan gambaran yang jelas tentang jalur solusi.



7. Kesimpulan

Proyek Labirin Escape berhasil menghadirkan permainan interaktif yang menggabungkan hiburan, tantangan, dan edukasi. Dengan memanfaatkan algoritma seperti backtracking untuk fitur petunjuk arah dan random maze generator untuk labirin acak, game ini menawarkan pengalaman yang selalu segar dan menantang. Pengembangan menggunakan bahasa pemrograman Python menunjukkan bagaimana konsep algoritma dapat diterapkan dalam konteks nyata, menjadikannya tidak hanya sebagai media permainan tetapi juga alat pembelajaran algoritmik.

Melalui fitur-fitur seperti solve, kontrol pergerakan intuitif, dan desain ulang labirin yang dinamis, Labirin Escape memberikan keseimbangan antara membantu pemain dan menjaga tingkat kesulitan yang menarik. Secara keseluruhan, proyek ini menunjukkan potensi besar dalam memadukan teknologi komputasi dan desain game untuk menciptakan produk yang inovatif dan bermanfaat.