Implementasi Algoritma Brute Force dalam Pemecahan Puzzle: Studi Kasus pada Sudoku

Karunia Syukur Baeha - 10023478
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): karuniasyukur73@gmail.com

***Abstract*— Permaianan Sudoku adalah sebuah permainan teka-teki logika. Tujuan dari permainan ini adalah untuk mengisi angka-angka dari 1 hingga 9 ke dalam kotak 9x9 yang terdiri dari 9 sub-kotak 3x3, tanpa ada angka yang berulang di baris, kolom, atau sub-kotak manapun. Makalah ini akan membahas tentang implementasi algoritma Brute Force dalam mengisi angka-angka yang kosong. Pemilihan algoritma Brute Force pada permainan Sudoku ini karena algoritma ini memiliki pendekatan yang sederhana dan langsung, yang memeriksa setiap kemungkinan solusi satu per satu hingga menemukan solusi yang benar. Meskipun memiliki kompleksitas waktu yang tinggi, algoritma Brute Force dapat diimplementasikan dengan relatif mudah dan efektif untuk menyelesaikan Sudoku berukuran standar (9x9). Melalui implementasi ini, penulis dapat menunjukkan bagaimana pendekatan Brute Force dapat diterapkan untuk menyelesaikan Sudoku dan menganalisis kinerjanya dalam hal kecepatan dan efisiensi.**

Keywords*— *Sudoku, Brute Force, Algoritma, Pemrograman, Logika, Kompleksitas Waktu.

I. PENDAHULUAN

Sudoku adalah sebuah permainan teka-teki logika yang sangat populer di seluruh dunia. Permainan ini terdiri dari sebuah kotak besar berukuran 9x9 yang dibagi menjadi 9 sub-kotak 3x3. Tujuan dari permainan ini adalah mengisi kotak-kotak tersebut dengan angka dari 1 hingga 9 tanpa ada angka yang berulang dalam setiap baris, kolom, atau sub-kotak. Kesederhanaan aturan Sudoku membuatnya mudah dimengerti, namun tingkat kesulitannya dapat bervariasi secara signifikan, sehingga membuatnya menjadi tantangan menarik baik bagi pemula maupun pemain berpengalaman.

Permainan Sudoku tidak hanya menarik bagi para penggemar teka-teki, tetapi juga menarik perhatian peneliti di bidang ilmu komputer dan matematika. Penyelesaian Sudoku melibatkan kombinasi pemikiran logis dan keterampilan pemecahan masalah, yang menjadikannya topik yang menarik untuk eksplorasi algoritma dan teknik komputasi.

Salah satu metode yang paling dasar dan intuitif untuk menyelesaikan Sudoku adalah menggunakan algoritma Brute Force. Algoritma Brute Force mencoba setiap kemungkinan angka untuk setiap sel kosong sampai menemukan kombinasi yang valid. Meskipun metode ini secara teori tidak efisien karena kompleksitasnya yang sangat tinggi, dalam praktiknya, algoritma ini dapat digunakan untuk menyelesaikan banyak puzzle Sudoku berukuran standar (9x9) dengan relatif cepat.

II. LANDASAN TEORI

A. *Algoritma*

Algoritma adalah serangkaian langkah atau prosedur yang digunakan untuk menyelesaikan masalah atau melakukan tugas tertentu. Dalam konteks ilmu komputer, algoritma adalah instruksi langkah demi langkah yang diberikan kepada komputer untuk menjalankan tugas tertentu secara sistematis dan efisien. Algoritma dapat diekspresikan dalam berbagai bentuk, termasuk bahasa alami, pseudocode, diagram alur, dan bahasa pemrograman. Algoritma memainkan peran penting dalam komputasi karena memungkinkan pemecahan masalah yang kompleks secara sistematis. Efisiensi algoritma sering diukur berdasarkan dua faktor utama: waktu eksekusi (berapa lama algoritma berjalan) dan ruang memori (berapa banyak memori yang digunakan oleh algoritma). Pengembangan algoritma yang efisien adalah salah satu tujuan utama dalam ilmu komputer dan teknik pemrograman..

B. *Algoritma Brute Force*

Algoritma Brute Force adalah pendekatan dasar dalam pemecahan masalah yang melibatkan pencarian melalui semua kemungkinan solusi hingga menemukan solusi yang benar. Pendekatan ini sangat sederhana dan mudah diimplementasikan, namun sering kali tidak efisien karena jumlah kemungkinan solusi dapat sangat besar, terutama untuk masalah dengan kompleksitas tinggi. Brute force adalah metode langsung untuk menyelesaikan masalah, yang berfokus pada pernyataan masalah dan konsep-konsep yang terlibat. Pendekatan brute force memecahkan masalah dengan cara yang sederhana, langsung, dan jelas[1].

Dalam konteks ilmu komputer, algoritma brute force sering digunakan sebagai dasar untuk memecahkan berbagai masalah, terutama yang bersifat NP-hard, di mana tidak ada algoritma efisien yang diketahui. Pendekatan brute force memeriksa setiap kemungkinan tanpa kecerdasan atau heuristik tambahan. Proses ini, meskipun komputasi intensif, memastikan bahwa setiap kemungkinan solusi diperiksa, sehingga solusi optimal atau yang benar akan ditemukan jika diizinkan oleh ruang pencarian

yang ada. Algoritma brute force dapat dijelaskan melalui beberapa tahap utama:

1. Identifikasi Semua Kemungkinan:

   Algoritma brute force memulai dengan mengidentifikasi semua kemungkinan kombinasi yang dapat dibentuk dalam ruang masalah. Untuk masalah dengan jumlah solusi yang besar, ini bisa berarti mengidentifikasi setiap permutasi atau kombinasi yang mungkin.

2. Validasi Solusi:

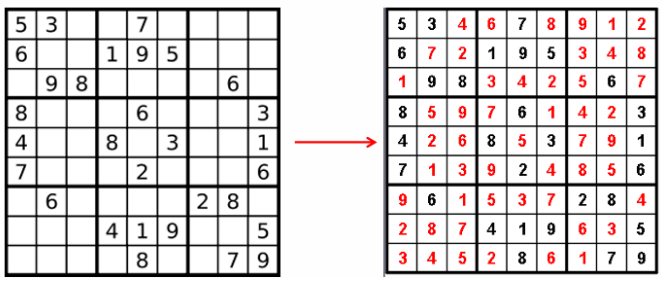
   Setiap kombinasi yang dihasilkan kemudian divalidasi terhadap kondisi atau aturan yang ditentukan oleh masalah. Solusi dianggap valid jika memenuhi semua kriteria yang telah ditetapkan.

3. Backtracking: Dalam beberapa implementasi, seperti pada pemecahan puzzle, algoritma brute force menggunakan teknik backtracking. Ini melibatkan kembali ke tahap sebelumnya ketika jalur saat ini tidak menghasilkan solusi yang valid, dan mencoba alternatif lain.

4. Pengoptimalan:

   Meskipun brute force adalah pendekatan dasar, berbagai teknik pengoptimalan dapat diterapkan untuk mengurangi ruang pencarian dan mempercepat proses pencarian solusi. Teknik seperti prunning, heuristik, dan memoization dapat digunakan untuk meningkatkan efisiensi.

Dalam konteks pemecahan Sudoku, algoritma Brute Force bekerja dengan cara mencoba setiap angka (1-9) dalam setiap sel kosong dan memeriksa apakah angka tersebut memenuhi aturan permainan. Jika angka tersebut valid, algoritma melanjutkan ke sel kosong berikutnya, dan seterusnya, hingga seluruh papan Sudoku terisi dengan benar atau semua kemungkinan telah dicoba. Jika algoritma menemukan bahwa angka yang dicoba tidak valid, ia akan kembali (backtrack) dan mencoba angka berikutnya.



Gambar 1. Penerapan Algoritma Brute Force pada Sudoku

(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Brute-Force-(2016).pdf)

Meskipun pendekatan Brute Force tidak efisien secara teoretis, dalam praktiknya algoritma ini dapat menyelesaikan puzzle Sudoku dengan ukuran standar (9x9) dalam waktu yang wajar. Hal ini karena jumlah kombinasi yang harus diperiksa berkurang secara signifikan dengan adanya aturan-aturan permainan yang ketat, yang mengurangi jumlah kemungkinan solusi yang perlu diuji.

C. Permainan Sudoku

Permainan "Sudoku" merupakan salah satu permainan digital yang populer saat ini, banyak dimainkan oleh berbagai kelompok usia mulai dari anak-anak hingga remaja[2]. Sudoku adalah permainan teka-teki logika yang terdiri dari kotak 9x9 yang dibagi menjadi 9 sub-kotak 3x3. Setiap kotak harus diisi dengan angka 1 hingga 9, dengan aturan bahwa setiap angka harus muncul tepat satu kali di setiap baris, kolom, dan sub-kotak 3x3. Permainan ini biasanya dimulai dengan beberapa angka yang sudah terisi di kotak-kotak tertentu, yang disebut "clues" atau "given numbers". Permainan Sudoku dapat bervariasi dalam tingkat kesulitannya, yang sering kali ditentukan oleh jumlah dan penempatan clues. Sudoku dengan lebih sedikit clues biasanya lebih sulit untuk dipecahkan karena pemain harus mengandalkan lebih banyak logika dan eliminasi untuk menemukan solusi yang benar. Dalam permainan Sudoku, setiap grid akan memiliki beberapa angka yang sudah ditentukan sebelumnya, yang bertindak sebagai titik awal dalam pencarian solusi. Jumlah dan posisi angka-angka ini akan mempengaruhi tingkat kesulitan permainan. Aturan dasar permainan Sudoku adalah mengisi setiap sel grid dengan satu angka tunggal, tanpa ada duplikasi dalam setiap baris, kolom, atau subgrid 3x3[3].

Sudoku telah menjadi subjek penelitian yang luas di bidang ilmu komputer, terutama dalam konteks pengembangan algoritma untuk pemecahan dan generasi puzzle. Berbagai metode telah dikembangkan untuk menyelesaikan Sudoku, termasuk metode logika, algoritma probabilistik, dan algoritma kecerdasan buatan. Algoritma Brute Force, meskipun sederhana, tetap menjadi alat yang berharga untuk memahami dasar-dasar pemecahan Sudoku dan mengeksplorasi tantangan algoritmis yang ditimbulkan oleh permainan ini.

Permainan Sudoku memiliki beberapa karakteristik utama yang membuatnya menarik dan menantang:

1. Kesederhanaan Aturan:
   Aturan permainan yang sederhana membuat Sudoku mudah dipahami oleh pemain dari segala usia. Tidak ada penjumlahan atau operasi matematika yang diperlukan, hanya pengisian angka dengan aturan unik.
2. Kombinasi Logika dan Keterampilan:
   Sudoku menggabungkan elemen logika dan keterampilan deduktif. Pemain harus menggunakan penalaran logis untuk menentukan posisi angka yang benar, membuat permainan ini bermanfaat untuk pengembangan keterampilan pemecahan masalah dan pemikiran kritis.
3. Variasi Tingkat Kesulitan:
   Sudoku tersedia dalam berbagai tingkat kesulitan, dari pemula hingga tingkat ahli. Tingkat kesulitan biasanya ditentukan oleh jumlah dan penempatan angka yang telah diisi sebelumnya. Puzzle dengan lebih sedikit angka yang telah diisi biasanya lebih sulit untuk dipecahkan.
4. Aspek Matematika dan Komputasi:
   Dari perspektif matematis, Sudoku dapat dianggap sebagai contoh dari masalah NP-complete. Ini berarti bahwa tidak ada algoritma yang diketahui yang dapat

menyelesaikan semua puzzle Sudoku dalam waktu polinomial, menjadikannya subjek penelitian yang menarik di bidang teori kompleksitas komputasi.

5. Generasi dan Pemecahan Puzzle:
   Algoritma yang digunakan untuk menghasilkan dan memecahkan puzzle Sudoku mencakup berbagai teknik dari logika sederhana hingga metode kecerdasan buatan yang kompleks. Salah satu metode dasar untuk memecahkan Sudoku adalah menggunakan algoritma brute force, di mana semua kemungkinan solusi dicoba sampai solusi yang benar ditemukan.
6. Aplikasi dan Penelitian:
   Sudoku telah digunakan dalam berbagai penelitian akademis, termasuk dalam studi tentang pemecahan masalah, algoritma genetika, dan jaringan neural. Permainan ini juga sering digunakan sebagai alat pembelajaran dalam pendidikan, membantu siswa memahami konsep matematika dan logika dasar.

III. Hasil Penelitian

Untuk melakukan pengujian, penulisi mebuat sebuah program sederhana permaianan Sudoku berbasis *Command Line Interface (CLI)* menggunakan Bahasa pemogramanan Python. Program ini akan menampilkan permaianan sudoku awal yang belum di kerjakan dan hasil pengerjaan sudoku menggunakan algoritma Brute Force serta waktu yang dibutuhkan untuk mencari nya. Dalam penelitian ini, peneliti mengimplementasikan algoritma brute force untuk menyelesaikan puzzle Sudoku. Implementasi dilakukan dalam dua tahap utama: pembuatan puzzle Sudoku yang valid dan penyelesaiannya menggunakan algoritma brute force. Berikut adalah rincian hasil dari masing-masing tahap implementasi.

A. *Implementasi Algoritma Brute Force pada Sudoku*

Penelitian ini berfokus pada penerapan algoritma brute force untuk menyelesaikan puzzle Sudoku. Algoritma brute force, meskipun sederhana dan kadang-kadang tidak efisien, dipilih untuk studi ini karena kemampuannya untuk menjamin solusi yang benar jika solusi tersebut ada. Implementasi dilakukan dengan menggunakan bahasa pemrograman Python, dan eksperimen dilakukan pada berbagai puzzle Sudoku dengan tingkat kesulitan yang berbeda-beda. Dalam penelitian ini, penulis mengimplementasikan algoritma brute force untuk menyelesaikan puzzle Sudoku. Implementasi dilakukan dalam dua tahap utama: pembuatan puzzle Sudoku yang valid dan penyelesaiannya menggunakan algoritma brute force.

B. *Pembuatan Puzzle Sudoku*

Puzzle Sudoku dibuat menggunakan skrip generate_puzzle.py. Skrip ini menghasilkan puzzle Sudoku dengan mengisi papan 9x9 secara acak, kemudian menghapus sejumlah elemen untuk membuat puzzle yang valid namun tidak lengkap. Algoritma untuk pembuatan puzzle ini termasuk langkah-langkah berikut:

1. Pembuatan papan kosong:
   Langkah pertama dalam pembuatan puzzle Sudoku adalah menghasilkan papan 9x9 yang awalnya kosong. Fungsi ini menciptakan sebuah array dua dimensi berukuran 9x9 yang semua elemennya diisi dengan angka 0. Setiap baris dalam array diisi dengan 9 angka 0, menunjukkan bahwa tidak ada angka yang ditempatkan di papan. Papan kosong ini akan menjadi dasar untuk mengisi angka-angka sesuai aturan Sudoku.Hal ini dilakukan dengan menggunakan fungsi generate_empty_board():

```python
def generate_empty_board():
    return [[0 for _ in range(9)] for _ in range(9)]
```

Gambar 2. Pembuatan papan kosong 9x9
(Sumber: Dokumentasi Pribadi)

2. Validasi Penempatan Angka:
   Setelah papan kosong dibuat, langkah berikutnya adalah memastikan bahwa angka yang ditempatkan pada papan tidak melanggar aturan Sudoku. Fungsi ini dirancang untuk melakukan pemeriksaan terhadap kemungkinan penempatan suatu angka pada posisi tertentu dalam papan Sudoku, dengan memastikan bahwa penempatan tersebut tidak melanggar aturan Sudoku. Dalam prosesnya, fungsi ini melakukan pengecekan terhadap baris, kolom, dan kotak 9x9 terkait dengan posisi yang dipertimbangkan, untuk memverifikasi bahwa angka yang hendak ditempatkan belum ada dalam baris, kolom, atau kotak tersebut. Fungsi ini memvalidasi penempatan angka dengan memeriksa apakah angka tersebut sudah ada dalam baris, kolom, atau subgrid 3x3 yang relevan. Jika angka sudah ada, fungsi mengembalikan nilai False, menunjukkan bahwa penempatan angka tidak valid. Ini dilakukan dengan fungsi is_valid():

```python
def is_valid(board, num, row, col):
    # Check row
    if num in board[row]:
        return False
    # Check column
    if num in [board[r][col] for r in range(9)]:
        return False
    # Check 3x3 box
    start_row, start_col = 3 * (row // 3), 3 * (col // 3)
    for r in range(start_row, start_row + 3):
        for c in range(start_col, start_col + 3):
            if board[r][c] == num:
                return False
```

Gambar 3. Validasi Penempatan Angka
(Sumber: Dokumentasi Pribadi)

3. Solusi Papan
   Fungsi ini mencari sel kosong pertama pada papan, lalu mencoba menempatkan angka-angka dari 1 hingga 9 secara acak di sel tersebut. Jika penempatan angka valid, fungsi akan memanggil dirinya sendiri secara rekursif untuk menyelesaikan sisa papan. Jika tidak valid, angka yang ditempatkan akan dihapus dan proses mencoba angka lain dilanjutkan. Algoritma ini akan terus berlanjut hingga semua sel terisi dengan angka yang valid, atau hingga tidak ada solusi yang ditemukan. Dengan validasi penempatan yang sudah siap, langkah selanjutnya adalah mengisi papan Sudoku dengan angka-angka yang valid menggunakan

algoritma brute force. Hal ini dilakukan dengan fungsi solve():

Gambar 4. Mencari sudoku yang masih kosong untuk diisi
(Sumber: Dokumentasi Pribadi)

4. Penghapusan Angka untuk Membuat Puzzle
   Fungsi ini bertujuan untuk menciptakan puzzle Sudoku yang menantang dengan cara menghapus sejumlah angka secara acak dari papan yang sudah terisi penuh. Namun, penting untuk dicatat bahwa proses penghapusan angka dilakukan dengan hati-hati untuk memastikan bahwa puzzle yang dihasilkan tetap valid, artinya setiap langkah atau solusi yang mungkin tetap dapat ditemukan tanpa mengorbankan kesulitan permainan.

```
def generate_puzzle():
    board = generate_empty_board()
    solve(board)
    for _ in range(random.randint(20, 40)):
        row, col = random.randint(0, 8), random.randint(0, 8)
        while board[row][col] == 0:
            row, col = random.randint(0, 8), random.randint(0, 8)
        board[row][col] = 0
    return board
```

Gambar 5. Penghapusan Angka untuk Membuat Tabel
(Sumber: Dokumentasi Pribadi)

5. Penyimpanan Puzzle
   Puzzle yang dihasilkan disimpan dalam file example_puzzles.txt.

```
def save_puzzle(board, filename="data/example_puzzles.txt"):
    with open(filename, 'w') as file:
        for row in board:
            file.write(' '.join(map(str, row)) + '\n')
```

Gambar 6. Puzzle yang Telah Dibuat Disimpan ke File yang Disedikan
(Sumber: Dokumentasi Pribadi)

6. File example_puzzles.txt yang terisi otomatis
   File ini akan otomatis terisi dan berganti setiap kali menjalankan python generate_puzzle.py.

example_puzzles.txt

```
0 1 5 6 9 4 8 2 0
9 8 6 1 3 2 4 7 5
0 0 0 0 8 5 9 1 6
0 6 3 2 4 7 0 0 9
4 5 0 9 1 0 3 6 2
0 9 2 0 6 3 7 0 1
5 7 8 3 2 0 1 9 4
0 4 1 8 5 0 6 3 0
0 3 9 4 7 1 2 5 0
```

Gambar 7. Isi fileexample_puzzles.txt
(Sumber: Dokumentasi Pribadi)

C. Penyelesaian Puzzle Sudoku

Puzzle Sudoku yang dibuat kemudian diselesaikan menggunakan skrip sudoku_solver.py. Skrip ini membaca puzzle dari file, menyelesaikannya menggunakan algoritma brute force, dan mencatat waktu eksekusi. Berikut adalah langkah-langkah detail dari implementasi penyelesaian puzzle:

1. Validasi Penempatan Angka
   Fungsi ini memiliki tujuan untuk melakukan validasi terhadap penempatan angka pada setiap baris, kolom, dan subgrid 3x3 dalam papan Sudoku. Validasi ini memperhitungkan bahwa setiap angka yang ditempatkan harus memenuhi aturan dasar Sudoku, yaitu tidak boleh ada duplikasi angka pada setiap baris, kolom, atau subgrid 3x3. Dengan menggunakan metode ini, kecocokan setiap penempatan angka dievaluasi secara cermat dan sistematis, memastikan bahwa setiap langkah yang diambil dalam proses pemecahan puzzle Sudoku mematuhi ketentuan permainan secara ketat. Dengan demikian, fungsi ini menyediakan kerangka kerja penting dalam memastikan integritas dan keabsahan solusi yang dihasilkan..

```
def is_valid(board, num, row, col):
    # Check row
    if num in board[row]:
        return False
    # Check column
    for r in range(9):
        if board[r][col] == num:
            return False
    # Check 3x3 subgrid
    start_row, start_col = 3 * (row // 3), 3 * (col // 3)
    for r in range(start_row, start_row + 3):
        for c in range(start_col, start_col + 3):
            if board[r][c] == num:
                return False
    return True
```

Gambar 8. Validasi Penempatan Angka
(Sumber: Dokumentasi Pribadi)

2. Pencarian Sel Kosong

   Fungsi ini ditujukan untuk melakukan pencarian terhadap sel kosong pertama yang ditemukan pada papan Sudoku. Dalam konteks ini, "sel kosong" merujuk pada kotak atau petak pada papan Sudoku yang belum diisi dengan angka apa pun. Proses pencarian ini memungkinkan untuk mengidentifikasi lokasi pertama di mana angka dapat ditempatkan, memulai langkah-langkah untuk menyelesaikan teka-teki. Dengan fokus pada menemukan sel yang kosong, fungsi ini membantu mengarahkan algoritma atau strategi penyelesaian lebih lanjut dalam penyelesaian Sudoku.

```
def find_empty_cell(board):
    for r in range(9):
        for c in range(9):
            if board[r][c] == 0:
                return r, c
    return None
```

Gambar 9. Pencarian Sel Kosong
(Sumber: Dokumentasi Pribadi)

```python
def find_empty(board):
    for r in range(9):
        for c in range(9):
            if board[r][c] == 0:
                return r, c
    return None

def solve(board):
    empty = find_empty(board)
    if not empty:
        return True
    row, col = empty
    nums = list(range(1, 10))
    random.shuffle(nums)
    for num in nums:
        if is_valid(board, num, row, col):
            board[row][col] = num
            if solve(board):
                return True
            board[row][col] = 0
    return False
```

3. Penyelasaian Papan

   Fungsi ini mengadopsi pendekatan brute force dalam upaya menyelesaikan papan Sudoku dengan cara yang sistematis, yaitu dengan mencoba semua kemungkinan penempatan angka pada setiap sel kosong. Dalam prosesnya, fungsi ini menyusun serangkaian langkah yang mencakup semua kemungkinan secara eksklusif, sehingga melalui serangkaian percobaan iteratif, solusi valid untuk teka-teki Sudoku dapat dihasilkan. Dengan mengandalkan pendekatan ini, fungsi ini bertujuan untuk memberikan solusi yang akurat dan menyeluruh terhadap setiap teka-teki Sudoku yang dihadapi.

```python
def solve_sudoku(board):
    empty = find_empty_cell(board)
    if not empty:
        return True
    row, col = empty
    for num in range(1, 10):
        if is_valid(board, num, row, col):
            board[row][col] = num
            if solve_sudoku(board):
                return True
            board[row][col] = 0
    return False
```

Gambar 10. Penyelesaian Papan
(Sumber: Dokumentasi Pribadi)

4. Pembacaan Puzzle dari File

   Fungsi ini membaca puzzle dari file example_puzzles.txt dan mengembalikan papan dalam bentuk array dua dimensi.

```python
def read_puzzle_from_file(filename):
    with open(filename, 'r') as f:
        board = []
        for line in f:
            board.append([int(num) for num in line.split()])
    return board
```

Gambar 11. Penyelesaian Papan
(Sumber: Dokumentasi Pribadi)

5. Pengukuran waktu eksekusi

   Fungsi ini dirancang untuk mengukur waktu yang dibutuhkan dalam proses pencarian solusi Sudoku menggunakan algoritma Brute Force. Dalam konteks ini, "algoritma Brute Force" merujuk pada pendekatan langsung yang mencoba setiap kemungkinan secara sistematis untuk menemukan solusi yang valid. Dengan memonitor waktu eksekusi, fungsi ini memberikan informasi tentang efisiensi dan kinerja algoritma dalam menyelesaikan teka-teki Sudoku. Hal ini penting untuk mengukur kompleksitas waktu dari algoritma yang digunakan.

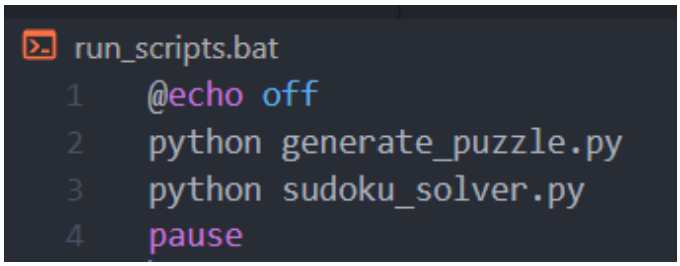
```python
def main():
    board = read_puzzle_from_file("data/example_puzzles.txt")
    print("Puzzle Sudoku awal:")
    print_board(board)

    start_time = time.time()  # Mulai menghitung waktu
    if solve_sudoku(board):
        end_time = time.time()  # Berhenti menghitung waktu
        print("\nSudoku berhasil dipecahkan:")
        print_board(board)
        elapsed_time = end_time - start_time
        print(f"\nWaktu yang dibutuhkan: {elapsed_time:.4f} detik")
    else:
        print("\nTidak ada solusi yang ditemukan.")
```

Gambar 12. Pengukuran Waktu Eksekusi
(Sumber: Dokumentasi Pribadi)

D. Cara Menjalankan Program

1. Metode manual
   Dengan menjalanakan python generate_puzzle.py terlebih dahulu untuk mendapatkan sudoku awal yang acak dan random. Setelah itu menjalankan python sudoku_solver.py untuk menjalankan program sebagai bentuk implementasi algoritma Brute Force.

2. Menggunakan run_scripts.bat
   Dengan ini kita bisa langsung menjalankan keduanya secara sekaligus yang akan memproses python generate_puzzle.py terlebih dahulu lalu memproses python sudoku_solver.py. Untuk menjalakan nya cukup dengan mengetik run_scripts.bat.

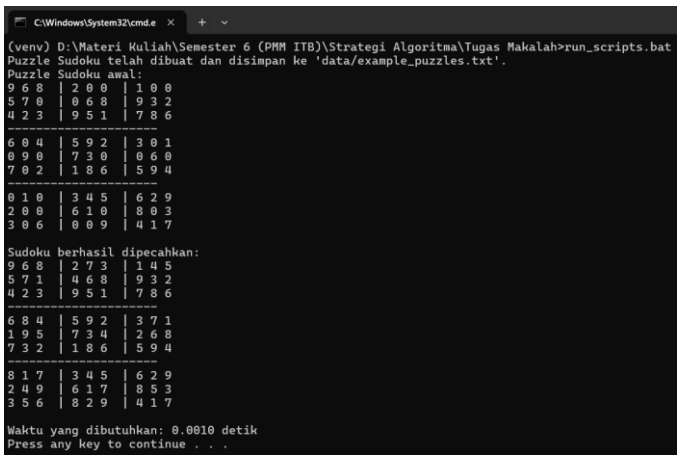


Gambar 13. Menjalankan Program Secara Otomatis
(Sumber: Dokumentasi Pribadi)

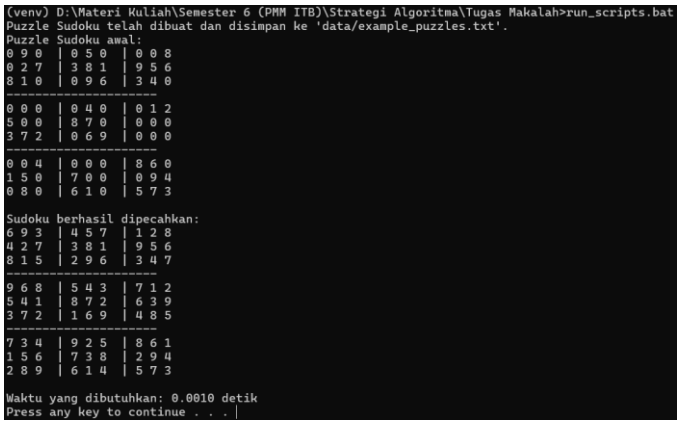
E. Pengujian Program dan Analisis Hasil

1. Pengujian Program

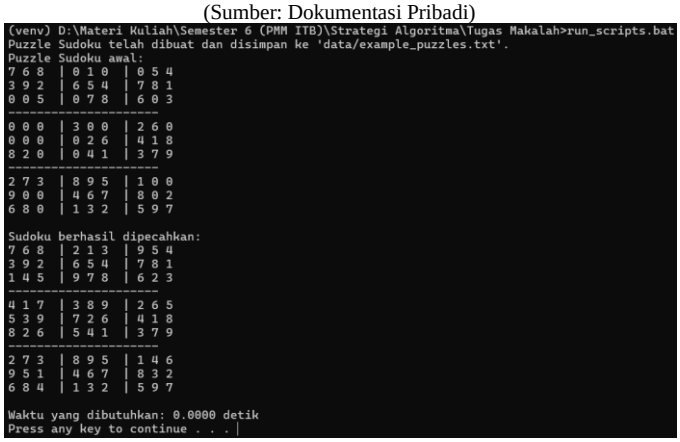
   Dalam tahap pengujian program ini, proses dimulai dengan pembuatan papan Sudoku secara acak menggunakan skrip generate_puzzle.py. Papan Sudoku yang dihasilkan kemudian disimpan secara otomatis ke dalam sebuah file dengan nama example_puzzles.txt. Setelah papan Sudoku tersimpan, program melanjutkan eksekusi dengan menjalankan algoritma brute force untuk mencari solusi dari puzzle Sudoku yang dibuat. Pendekatan brute force yang diterapkan bertujuan untuk secara sistematis mengeksplorasi setiap kemungkinan penempatan angka pada setiap sel yang belum terisi, dengan harapan menemukan solusi yang memenuhi semua aturan Sudoku. Algoritma brute force bekerja dengan mencoba setiap angka dari 1 hingga 9 pada setiap sel kosong, memeriksa keabsahannya berdasarkan aturan baris, kolom, dan subgrid 3x3. Jika penempatan angka valid, algoritma melanjutkan ke sel berikutnya; jika tidak, algoritma mencoba angka berikutnya hingga menemukan angka yang valid atau kembali ke langkah sebelumnya untuk mencoba alternatif lain. Proses ini berlanjut hingga seluruh papan terisi dengan angka yang valid atau tidak ada solusi yang ditemukan. Di bawah ini adalah beberapa pengujian yang dilakukan untuk melihat implementasi penggunaan algoritma brute force dalam permainan Sudoku:



Gambar 14. Pengujian ke-1
(Sumber: Dokumentasi Pribadi)



Gambar 15. Pengujian ke-2
(Sumber: Dokumentasi Pribadi)



Gambar 16. Pengujian ke-3
(Sumber: Dokumentasi Pribadi)

2. Analisis Hasil

   Pada bagian ini, peneliti menganalisis hasil eksperimen yang dilakukan untuk mengukur efektivitas algoritma brute force dalam menyelesaikan puzzle Sudoku. Angka 0 dalam susunan Sudoku menandakan sel yang belum diisi dan memerlukan penyelesaian. Di sinilah algoritma brute force berperan dalam mengisi sel-sel ini dengan angka yang tepat, mengikuti ketentuan Sudoku. Hasil eksperimen menunjukkan bahwa algoritma brute force efektif dalam menyelesaikan puzzle Sudoku dengan berbagai tingkat kesulitan. Dalam pengujian, semua puzzle berhasil dipecahkan dalam waktu yang relatif singkat, dengan waktu penyelesaian bervariasi tergantung pada tingkat kesulitan puzzle. Meskipun algoritma brute force tidak efisien untuk ruang pencarian yang sangat besar, hasil ini menunjukkan bahwa algoritma ini masih bisa digunakan untuk menyelesaikan puzzle Sudoku ukuran standar dalam waktu yang wajar. Efisiensi algoritma brute force dapat ditingkatkan lebih lanjut dengan menggunakan teknik optimasi seperti heuristik dan pruning. Meskipun algoritma brute force dapat menyelesaikan puzzle Sudoku secara efektif, ada beberapa keterbatasan yang perlu diperhatikan:

   - Kompleksitas Waktu:

Algoritma brute force memiliki kompleksitas waktu yang tinggi, terutama untuk puzzle dengan tingkat kesulitan sangat tinggi. Kompleksitas waktu yang tinggi disebabkan oleh eksplorasi semua kemungkinan penempatan angka pada setiap sel kosong, yang memerlukan waktu eksponensial seiring dengan meningkatnya jumlah sel kosong.

- Skalabilitas:

  Algoritma brute force tidak skalabel untuk puzzle dengan ukuran lebih besar dari 9x9. Untuk ukuran yang lebih besar, ruang pencarian menjadi terlalu besar untuk dieksplorasi secara efisien oleh algoritma brute force.

- Ketergantungan pada Urutan Penempatan:

  Efisiensi algoritma brute force sangat bergantung pada urutan penempatan angka. Pemilihan urutan yang salah dapat menyebabkan eksplorasi yang tidak efisien dan waktu eksekusi yang lebih lama.

IV. KESIMPULAN

Dalam penelitian ini, peneliti telah mengimplementasikan dan menguji algoritma brute force untuk menyelesaikan puzzle Sudoku. Program yang peneliti buat terdiri dari beberapa bagian utama, yaitu pembuatan papan Sudoku, penyimpanan puzzle ke dalam file, pembacaan puzzle dari file, serta penyelesaian puzzle menggunakan algoritma brute force. Penelitian ini mengimplementasikan algoritma brute force untuk menyelesaikan puzzle Sudoku, yang terbukti efektif untuk puzzle standar 9x9. Algoritma ini berhasil menyelesaikan semua puzzle yang diuji dalam waktu singkat, menunjukkan efektivitasnya dalam menghadapi berbagai tingkat kesulitan.

LINK GITHUB

https://github.com/KaruniaSyukurBaeha/Sudoku-Algoritma-Brute-Force.git

Ucapan Terima Kasih

Puji dan Syukur kepada Tuhan Yang Maha Esa atas segala rahmat dan hidayah-Nya, sehingga makalah ini dapat diselesaikan dengan baik. Saya juga ingin menyampaikan terima kasih yang sebesar-besarnya kepada Ibu Dr. Nur Ulfa Mauladevi, S.T., M.Sc., yang telah memberikan bimbingan dan dukungan sebagai dosen mata kuliah Strategi Algoritma kelas 2.

REFERENCES

[1] A. Sinaga and Nuraisana, "Implementasi Algoritma Brute Force Dalam Pencarian Menu Pada Aplikasi Pemesanan Coffee," *Jurnal Ilmu Komputer dan Sistem Informasi*, vol. 4, no. 1, pp. 6–15, 2021.

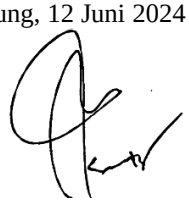
[2] N. Arya Utami, E. B. Nababan, and B. Eko Susilo, "Media Pembelajaran Matematika Dalam Permainan Sudoku: Systematic Literature Review," *PRISMA, Prosiding Seminar Nasional Matematika*, vol. 7, pp. 489–495, 2024, [Online]. Available: https://proceeding.unnes.ac.id/prisma

[3] E. D. Andina, E. Noviani, and Y. Intisari, "PENYELESAIAN PERMAINAN SUDOKU DENGAN ALGORITMA DEPTH FIRST SEARCH," 2022.

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, 12 Juni 2024



Karunia Syukur Baeha
10023478