Penerapan Algoritma Backtracking dalam Menyelesaikan Permainan Number Mazes

Muhammad Dava Fathurrahman - 13522114
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522114@std.stei.itb.ac.id

***Abstrak*—Number Mazes adalah permainan teka-teki di mana pemain harus melakukan navigasi dari titik awal hingga titik akhir, memastikan bahwa total angka yang dilewatinya sesuai dengan aturan tertentu. Dalam tingkat kesulitan yang berbeda, pemain harus menyesuaikan total angka yang dilewatinya untuk mencapai jumlah yang ditentukan. Makalah ini memberikan gambaran tentang aturan permainan, konsep bruteforce, konsep backtracking, dan bagaimana konsep ini diimplementasikan dalam menyelesaikan permainan Number Maze.**

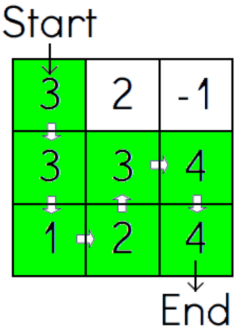
Kata Kunci—Backtracking, Number Mazes, Brute Force, Exhaustive Search.

I. Latar Belakang

Pada perkembangannya, konsep backtracking telah diaplikasikan dalam berbagai permasalahan yang berkaitan dengan pemecahan masalah, termasuk dalam bidang permainan komputer. Salah satu permainan yang menggunakan konsep backtracking adalah labirin angka atau number mazes. Number mazes merupakan permainan teka-teki yang mengharuskan pemain untuk menemukan jalur yang tepat dari titik awal hingga titik akhir tanpa melanggar aturan permainan.

Kita dapat menemukan banyak sekali variasi permainan beserta aturannya. Permainan number mazes yang penulis maksud adalah permainan yang menantang pemain untuk memilih jalur yang tepat dari sejumlah angka yang tersusun dalam bentuk matriks. Angka-angka ini memiliki nilai positif, nol, atau negatif, dan pemain harus menjumlahkan angka-angka yang dilewatinya agar mencapai jumlah yang ditentukan sesuai dengan level kesulitan permainan. Hal ini membuat permainan number mazes menjadi menarik sebagai objek studi dalam penerapan konsep backtracking.

Aturan permainan dalam Number Mazes adalah sebagai berikut: Pemain harus menemukan jalur dari titik awal hingga titik akhir pada papan permainan (maze), sambil memastikan bahwa total jumlah angka yang dilewati sama dengan jumlah yang ditentukan. Pemain dapat memilih tingkat kesulitan yang diinginkan, yaitu Easy (Mudah) dengan papan permainan berukuran 3x3, di mana total jumlah angka yang dilewati harus sama dengan 20; Moderate (Sedang) dengan papan permainan berukuran 4x3, di mana total jumlah angka yang dilewati harus sama dengan 50; Hard (Sulit) dengan papan permainan berukuran 4x4, di mana total jumlah angka yang dilewati harus sama dengan 100, dan Very Hard (Sangat Sulit) dengan papan permainan berukuran 5x3, di mana total jumlah angka yang dilewati harus sama dengan 100. Pemain dapat bergerak dari satu sel ke sel berikutnya secara horizontal atau vertikal, tetapi tidak boleh bergerak secara diagonal. Selain itu, pemain hanya boleh melewati setiap sel (kotak) dalam papan permainan satu kali saja. Dengan aturan ini, pemain diharapkan dapat menggunakan keterampilan pemecahan masalah dan strategi untuk menavigasi maze, menjumlahkan angka-angka yang dilewati dengan benar, dan mencapai tujuan akhir dengan total yang sesuai.



Gambar 1. Ilustrasi Permainan Number Mazes Level Easy

(Sumber: https://www.dr-mikes-math-games-for-kids.com/number-mazes.html)

Ilustrasi di atas menunjukkan pemecahan untuk permainan number mazes pada tingkat kesulitan mudah. Setiap langkah dalam jalur tersebut hanya bergerak secara vertikal atau horizontal, dengan total jumlah angka yang dilewati sama dengan 20, dan jalur dimulai dan berakhir pada posisi yang benar.

II. Landasan Teori

A. Algoritma Bruteforce

Algoritma Brute Force didefinisikan sebagai pendekatan sederhana dan langsung dalam penyelesaian masalah yang mencoba semua kemungkinan solusi secara berurutan, tanpa melakukan pemangkasan atau optimasi yang signifikan. Pendekatan ini seringkali digambarkan sebagai metode yang

"lempang" (straightforward) karena kejelasannya dalam proses pemecahan masalah. Algoritma brute force memecahkan persoalan dengan cara yang sangat sederhana, jelas, dan langsung, tanpa adanya strategi atau heuristik yang rumit.

Meskipun dapat digunakan untuk memecahkan hampir semua jenis masalah, algoritma brute force umumnya tidak dianggap sebagai metode yang cerdas atau optimal karena membutuhkan volume komputasi yang besar dan waktu yang lama dalam penyelesaiannya. Namun, kelebihan dari algoritma brute force adalah kemampuannya untuk menyelesaikan hampir semua jenis masalah, sehingga memiliki aplikabilitas yang luas.

Kekuatan utama dari algoritma brute force meliputi:

1. Kemampuannya untuk diterapkan pada berbagai jenis masalah (wide applicability).
2. Kedekatannya dengan konsep sederhana dan mudah dimengerti.
3. Penerapannya dalam menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan string, dan perkalian matriks.
4. Kontribusinya dalam menghasilkan algoritma baku (standard) untuk tugas-tugas komputasi seperti penjumlahan/perkalian n buah bilangan dan menentukan elemen minimum atau maksimum dalam suatu himpunan data.

Namun, algoritma brute force juga memiliki beberapa kelemahan, antara lain:

1. Jarang menghasilkan algoritma yang optimal atau efisien.
2. Lambat dalam menangani masukan berukuran besar, sehingga tidak dapat diterima dalam situasi di mana kinerja waktu sangat penting.
3. Kurangnya kreativitas dalam strategi pemecahan masalah dibandingkan dengan pendekatan lain yang lebih canggih.

B. Algoritma Backtracking

Algoritma backtracking merupakan metode pemecahan masalah yang efisien, terstruktur, dan sistematis, baik untuk persoalan optimasi maupun non-optimasi. Backtracking pertama kali diperkenalkan oleh D. H. Lehmer pada tahun 1950, kemudian dikembangkan lebih lanjut oleh R. J. Walker, Golomb, dan Baumert yang memberikan uraian umum tentang algoritma ini. Algoritma ini dianggap sebagai perbaikan dari metode exhaustive search. Pada exhaustive search, semua kemungkinan solusi dieksplorasi dan dievaluasi satu per satu, sedangkan pada backtracking, hanya pilihan yang mengarah ke solusi yang dieksplorasi, sementara pilihan yang tidak mengarah ke solusi dipangkas (pruning).

Terdapat tiga buah properti umum dari algoritma runut balik, yaitu:

1. Solusi Persoalan

   Solusi dinyatakan sebagai vektor dengan n-tuple:

   X = (x1 , x2 , …, xn), di mana xi ∈ Si. Umumnya, S1 = S2 = … = Sn.

2. Fungsi pembangkit nilai xk

   Fungsi ini dinyatakan dengan predikat T(). Predikat ini, T(x[1], x[2], …, x[k-1]) digunakan untuk membangkitkan nilai xk, yang merupakan komponen vektor solusi.

3. Fungsi pembatas (bounding function)

   Fungsi ini dinyatakan sebagai predikat B(x1, x2, …, xn). Predikat B akan bernilai true jika argumen-argumennya tidak melanggar batasan atau mengarah ke solusi.

   Jika bernilai true, pembangkitan nilai untuk xk+1 dilanjutkan, tetapi jika false, vektor (x1 , x2 , …, xk) diabaikan.

Semua kemungkinan solusi dari suatu masalah disebut ruang solusi (solution space). Ruang solusi diorganisasikan dalam bentuk struktur pohon berakar, di mana setiap simpul pohon mewakili status (state) dari masalah, dan cabang-cabangnya diberi label dengan nilai-nilai xi. Lintasan dari akar ke daun menggambarkan solusi yang mungkin, dan keseluruhan lintasan dari akar ke daun membentuk ruang solusi. Pengorganisasian ruang solusi dalam bentuk pohon disebut sebagai pohon ruang status (state space tree).

Backtracking dapat dipandang sebagai pencarian dalam pohon dari akar menuju daun (simpul solusi). Solusi dicari dengan membangkitkan simpul-simpul status sehingga menghasilkan lintasan dari akar ke daun. Aturan pembangkitan simpul yang digunakan adalah depth-first search (DFS). Simpul-simpul yang telah dibangkitkan disebut simpul hidup (live node), dan simpul hidup yang sedang diperluas disebut simpul-E (expand-node). Setiap kali simpul-E diperluas, lintasan yang dibentuk olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, simpul-E tersebut "dimatikan" dan menjadi simpul mati (dead node). Fungsi pembatas (bounding function) digunakan untuk mematikan simpul-E ini. Ketika sebuah simpul mati, anak-anak simpul tersebut secara implisit telah dipangkas (pruning).

Jika pembentukan lintasan berakhir pada simpul mati, proses pencarian akan mundur (backtrack) ke simpul pada level sebelumnya. Proses ini dilanjutkan dengan membangkitkan simpul anak lainnya, yang kemudian menjadi simpul-E yang baru. Pencarian dihentikan ketika simpul tujuan (goal node) telah tercapai atau tidak ada lagi simpul yang dapat diekspan.

Dengan mengimplementasikan algoritma backtracking, proses pencarian solusi menjadi lebih efisien dibandingkan dengan metode exhaustive search, karena cabang-cabang yang tidak mungkin menghasilkan solusi dipangkas lebih awal, sehingga mengurangi jumlah simpul yang harus dieksplorasi.

C. Teknik Heuristik

Teknik heuristik adalah metode yang digunakan untuk mempercepat proses pencarian solusi dalam algoritma exhaustive search dengan mengeliminasi beberapa kemungkinan solusi tanpa harus mengeksplorasi seluruh kemungkinan solusi secara penuh. Dalam konteks pemecahan

masalah, heuristik berfungsi sebagai pendekatan berbasis pengalaman, proses pembelajaran, dan penemuan solusi, meskipun tidak selalu menjamin hasil yang optimal. Heuristik memanfaatkan terkaan, intuisi, dan common sense yang, meskipun secara matematis tidak dapat dibuktikan, sangat berguna dalam praktik.

Teknik heuristik bekerja dengan mengarahkan pencarian solusi ke jalur-jalur yang lebih menjanjikan berdasarkan informasi yang tersedia, sehingga dapat mempercepat waktu pencarian dibandingkan dengan pencarian solusi secara exhaustive search. Heuristik menggunakan berbagai strategi untuk mengurangi ruang pencarian, seperti mengabaikan jalur-jalur yang dianggap tidak akan mengarah ke solusi optimal atau memprioritaskan jalur-jalur yang memiliki probabilitas lebih tinggi untuk mencapai solusi.

Keunggulan utama dari teknik heuristik adalah kemampuannya untuk memberikan solusi yang cukup baik dalam waktu yang lebih singkat, terutama untuk masalah-masalah yang kompleks dan memiliki ruang solusi yang sangat besar. Meskipun tidak selalu memberikan hasil yang optimal, teknik ini sering kali dapat menyelesaikan masalah dengan cukup memuaskan untuk kebanyakan kasus praktis. Oleh karena itu, heuristik menjadi alat yang sangat berguna dalam berbagai bidang seperti optimasi, kecerdasan buatan, dan pemrograman, di mana waktu dan sumber daya komputasi menjadi faktor yang penting.

Namun, penting untuk diingat bahwa heuristik tidak menjamin selalu dapat memecahkan persoalan. Efektivitas heuristik sangat bergantung pada sifat masalah yang dihadapi dan kualitas heuristik yang diterapkan. Dalam beberapa kasus, heuristik mungkin tidak menemukan solusi atau hanya menemukan solusi suboptimal. Meskipun demikian, dalam banyak situasi, teknik heuristik seringkali memberikan hasil yang cukup baik dan lebih cepat dibandingkan dengan metode exhaustive search yang mengeksplorasi semua kemungkinan solusi secara penuh

D. Permainan Number Mazes

Number Mazes adalah permainan teka-teki yang menantang pemain untuk menavigasi grid angka menuju nilai tertentu melalui jalur yang valid dari titik awal hingga titik akhir. Dalam permainan ini, pemain harus menggunakan logika, perhitungan matematis, dan strategi perencanaan jalur untuk menyelesaikan teka-teki yang diberikan. Struktur permainan terdiri dari grid persegi atau persegi panjang yang terdiri dari sel-sel dengan angka positif, nol, atau negatif. Pemain dimulai dari titik awal dan harus mencari jalur menuju titik akhir. Mereka dapat bergerak secara horizontal atau vertikal antara sel-sel, tetapi tidak diperbolehkan bergerak secara diagonal. Penting untuk mencatat bahwa setiap sel hanya boleh dilewati sekali selama perjalanan dari titik awal ke titik akhir.

Aturan permainan menetapkan target jumlah yang harus dicapai pemain sepanjang jalur, yang bervariasi tergantung pada tingkat kesulitan permainan. Mulai dari tingkat mudah dengan grid 3x3 dan target jumlah 20, hingga tingkat sangat sulit dengan grid 5x3 dan target jumlah 100. Dengan mengikuti aturan ini, pemain harus menemukan jalur yang tepat untuk menyelesaikan teka-teki dan mencapai tujuan akhir.

III. PEMBAHASAN

Dalam makalah ini, akan dijelaskan penggunaan algoritma brute force dan backtracking untuk menentukan jalur penyelesaian pada permainan number maze. Sumber persoalan ini diambil dari worksheet yang tersedia pada situs Dr. Mike's Math Games for Kids. Setiap persoalan hanya memiliki tepat satu solusi.

A. Deskripsi Persoalan

Permainan Number Mazes menantang pemain untuk menemukan jalur dari titik awal di kiri atas hingga titik akhir di kanan bawah dalam sebuah grid angka. Dalam permainan ini, pemain harus memastikan bahwa total jumlah angka yang dilewati oleh jalur mereka sesuai dengan jumlah yang ditentukan. Terdapat empat tingkat kesulitan: Mudah, Sedang, Sulit, dan Sangat Sulit, masing-masing dengan ukuran grid yang berbeda dan target jumlah angka yang berbeda. Pemain hanya dapat bergerak secara horizontal atau vertikal dari satu sel ke sel berikutnya, namun tidak diperbolehkan bergerak secara diagonal. Setiap sel dalam grid hanya boleh dilewati sekali.

B. Pemodelan Persoalan

Tiga properti umum dari algoritma backtracking dalam persoalan number mazes adalah sebagai berikut:

1. Solusi Persoalan:

   Solusi dalam persoalan ini adalah sebuah jalur yang menghubungkan sel-sel pada grid dari titik awal hingga titik akhir, di mana jumlah angka yang dilewati sesuai dengan aturan yang ditentukan. Dalam grid yang diberikan, solusi merupakan urutan langkah-langkah yang membentuk jalur dari angka pertama hingga angka terakhir, memastikan bahwa total jumlah angka yang dilewati sesuai dengan persyaratan permainan. Panjang jalur penyelesaian dapat berbeda-beda pada setiap persoalan.

   Berikut adalah pemodelan yang mungkin,

   $X = (x1, x2, \ldots, xn)$, dengan $x \in Z$.

2. Fungsi Pembangkit

   Fungsi pembangkit nilai (T) bertanggung jawab untuk menghasilkan kemungkinan langkah selanjutnya dari posisi saat ini dalam grid. Dalam implementasi, fungsi ini akan menghasilkan tetangga-tetangga yang valid dari posisi saat ini dan belum pernah dilalui, sehingga pemain dapat memilih langkah selanjutnya berdasarkan tetangga yang tersedia.

3. Fungsi pembatas

   Fungsi pembatas (B) digunakan untuk membatasi pencarian hanya pada jalur-jalur yang memiliki potensi untuk mencapai solusi yang valid. Fungsi ini akan memeriksa apakah posisi saat ini adalah tujuan akhir (goal), namun jumlah angka yang dilewati tidak

sesuai dengan jumlah yang ditentukan (goal value). Ini berarti jika pemain telah mencapai sel yang merupakan tujuan akhir, namun total jumlah angka yang dilewatinya tidak sesuai dengan nilai tujuan yang ditetapkan untuk permainan tersebut, maka jalur tersebut tidak memenuhi syarat dan harus dieliminasi dari pencarian.

C. Penyelesaian Permainan Number Mazes dengan Algoritma Brute Force

Algoritma brute force yang digunakan berupa teknik exhaustive search dengan langkah-langkah sebagai berikut:

1. Inisialisasi current_sum dengan nilai awal dari sel pada posisi awal.
2. Lakukan pencarian dengan metode rekursif untuk menjelajahi semua kemungkinan langkah dari posisi saat ini.
3. Saat menjelajahi setiap langkah yang mungkin, tambahkan nilai dari sel tersebut ke current_sum.
4. Periksa apakah posisi saat ini sama dengan posisi tujuan dan nilai current_sum sama dengan nilai tujuan yang diinginkan. Jika ya, maka solusi telah ditemukan.
5. Jika tidak ada langkah yang tersisa untuk dieksplorasi, kembali ke langkah sebelumnya (backtrack) untuk mencoba langkah alternatif.
6. Ulangi langkah-langkah 2-5 sampai solusi ditemukan atau semua kemungkinan telah dieksplorasi dan tidak ada solusi yang ditemukan.

```
def solve_maze(self):
    start_x, start_y = 0, 0
    initial_sum = self.maze[start_y][start_x]
    initial_path = [(start_x, start_y, 'Start', initial_sum)]

    if self.find_path(start_x, start_y, initial_sum, self.goal_value, initial_path):
        return initial_path
    else:
        return None
```

```
def find_path(self, x, y, current_sum, goal_value, path):
    if self.is_goal_reached(x, y, current_sum, goal_value):
        return True

    self.visited.add((x, y))
    self.explored_nodes += 1

    for nx, ny, move in self.neighbors(x, y):
        if (nx, ny) not in self.visited:
            next_value = self.maze[ny][nx]
            path.append((nx, ny, move, next_value))
            if self.find_path(nx, ny, current_sum + next_value, goal_value, path):
                return True
            path.pop()

    self.visited.remove((x, y))

    return False
```

D. Penyelesaian Permainan Number Mazes dengan Algoritma Backtracking

Algoritma backtracking yang digunakan dengan langkah-langkah sebagai berikut:

1. Inisialisasi current_sum dengan nilai awal dari sel pada posisi awal.
2. Lakukan pencarian dengan metode rekursif untuk menjelajahi semua kemungkinan langkah dari posisi saat ini.
3. Saat menjelajahi setiap langkah yang mungkin, tambahkan nilai dari sel tersebut ke current_sum.
4. Periksa apakah posisi saat ini sama dengan posisi tujuan.

   Jika ya, periksa apakah nilai current_sum sama dengan nilai tujuan yang diinginkan. Jika ya, maka solusi telah ditemukan - Jika tidak, kembali ke langkah sebelumnya (backtrack) untuk mencoba langkah alternatif.
5. Jika tidak ada langkah yang tersisa untuk dieksplorasi, kembali ke langkah sebelumnya (backtrack) untuk mencoba langkah alternatif.
6. Ulangi langkah-langkah 2-5 sampai solusi ditemukan atau semua kemungkinan telah dieksplorasi dan tidak ada solusi yang ditemukan.

```
def solve_maze(self):
    start_x, start_y = 0, 0
    initial_sum = self.maze[start_y][start_x]
    initial_path = [(start_x, start_y, 'Start')]

    if self.find_path(start_x, start_y, initial_sum, self.goal_value, initial_path):
        return initial_path
    else:
        return None
```

```
def is_goal_unreachable(self, x, y, current_sum, goal_value):
```

```
        return x == self.goal_x and y == self.goal_y and current_sum != goal_value
```

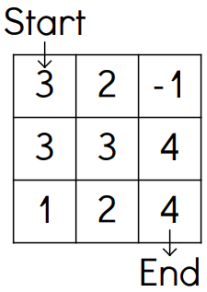
```
def find_path(self, x, y, current_sum, goal_value, path):
    if self.is_goal_reached(x, y, current_sum, goal_value):
        return True

    if self.is_goal_unreachable(x, y, current_sum, goal_value):
        return False

    self.visited.add((x, y))
    self.explored_nodes += 1

    for nx, ny, direction in self.neighbors(x, y):
        if (nx, ny) not in self.visited:
            next_value = self.maze[ny][nx]
            path.append((nx, ny, direction))
    if self.find_path(nx, ny, current_sum + next_value, goal_value, path):
            return True

    self.visited.remove((x, y))

    return False
```

Algoritma di atas menggunakan fungsi pembatas seperti yang dijelaskan pada bagian pemodelan persoalan. Algoritma backtracking dapat mengidentifikasi dan mengeliminasi jalur yang tidak mungkin mengarah pada solusi lebih awal dalam proses pencarian (pruning).

E. Uji Coba

Sumber persoalan ini diambil dari worksheet yang tersedia pada situs Dr. Mike's Math Games for Kids. Setiap persoalan hanya memiliki tepat satu solusi.

1. Level Easy



Gambar 2. Soal Uji Level Easy

(Sumber: Dokumen Pribadi)

```
$ python NumberMazeBF.py
Welcome to Number Maze Solver using Brute Force Algorithm!
Choose a difficulty level:
1. Easy
2. Moderate
3. Hard
4. Very Hard
Enter the number of your choice: 1
Enter 3 numbers for row 1: 3 2 -1
Enter 3 numbers for row 2: 3 3 4
Enter 3 numbers for row 3: 1 2 4
3       2       -1
3       3       4
1       2       4

Goal value: 20
Rows: 3, Columns: 3, Goal Value: 20
Solver found a path:
Start(3) -> Down(6) -> Down(7) -> Right(9) -> Up(12) -> Right(16) -> Down(20) -> End
Number of explored nodes: 70
Time taken: 0.000145800004247576 seconds
```

Gambar 3. Simulasi Penyelesaian Soal Uji Level Easy dengan Algoritma Brute Force

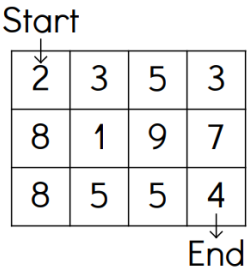
(Sumber: Dokumen Pribadi)

```
$ python NumberMazeBT.py
Welcome to Number Maze Solver using Backtracking Algorithm!
Choose a difficulty level:
1. Easy
2. Moderate
3. Hard
4. Very Hard
Enter the number of your choice: 1
Enter 3 numbers for row 1: 3 2 -1
Enter 3 numbers for row 2: 3 3 4
Enter 3 numbers for row 3: 1 2 4
3       2       -1
3       3       4
1       2       4

Goal value: 20
Rows: 3, Columns: 3, Goal Value: 20
Solver found a path:
Start(3) -> Down(6) -> Down(7) -> Right(9) -> Up(12) -> Right(16) -> Down(20) -> End
Number of explored nodes: 40
Time taken: 0.00011510000331327319 seconds
```

Gambar 4. Simulasi Penyelesaian Soal Uji Level Easy dengan Algoritma Backtracking

(Sumber: Dokumen Pribadi)

2. Level Moderate



Gambar 5. Soal Uji Level Moderate

(Sumber: Dokumen Pribadi)

```
$ python NumberMazeBF.py
Welcome to Number Maze Solver using Brute Force Algorithm!
1. Easy
2. Moderate
3. Hard
4. Very Hard
Enter the number of your choice: 2
Enter 4 numbers for row 1: 2 3 5 3
Enter 4 numbers for row 2: 8 1 9 7
Enter 4 numbers for row 3: 8 5 5 4
2       3       5       3
8       1       9       7
8       5       5       4

Goal value: 50
Rows: 3, Columns: 4, Goal Value: 50
Solver found a path:
Start(2) -> Right(5) -> Right(10) -> Down(19) -> Left(20) -> Left(28)
-> Down(36) -> Right(41) -> Right(46) -> Right(50) -> End
Number of explored nodes: 51
Time taken: 0.00011339999036863446 seconds
```

Gambar 6. Simulasi Penyelesaian Soal Uji Level Moderate dengan Algoritma Brute Force

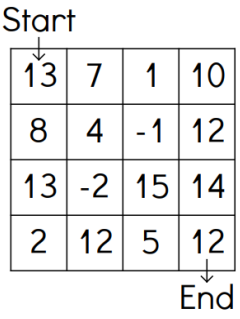
(Sumber: Dokumen Pribadi)

```
$ python NumberMazeBT.py
Welcome to Number Maze Solver using Backtracking Algorithm!
Choose a difficulty level:
1. Easy
2. Moderate
3. Hard
4. Very Hard
Enter the number of your choice: 2
Enter 4 numbers for row 1: 2 3 5 3
Enter 4 numbers for row 2: 8 1 9 7
Enter 4 numbers for row 3: 8 5 5 4
2       3       5       3
8       1       9       7
8       5       5       4

Goal value: 50
Rows: 3, Columns: 4, Goal Value: 50
Solver found a path:
Start(2) -> Right(5) -> Right(10) -> Down(19) -> Left(20) -> Left(28)
-> Down(36) -> Right(41) -> Right(46) -> Right(50) -> End
Number of explored nodes: 30
Time taken: 9.14999982342124e-05 seconds
```

Gambar 7. Simulasi Penyelesaian Soal Uji Level Moderate dengan Algoritma Backtracking

(Sumber: Dokumen Pribadi)

3. Level Hard



Gambar 8. Soal Uji Level Hard

(Sumber: Dokumen Pribadi)

```
$ python NumberMazeBF.py
Welcome to Number Maze Solver using Brute Force Algorithm!
Choose a difficulty level:
1. Easy
2. Moderate
3. Hard
4. Very Hard
Enter the number of your choice: 3
Enter 4 numbers for row 1: 13 7 1 10
Enter 4 numbers for row 2: 8 4 -1 12
Enter 4 numbers for row 3: 13 -2 15 14
Enter 4 numbers for row 4: 2 12 5 12
13      7       1       10
8       4       -1      12
13      -2      15      14
2       12      5       12

Goal value: 100
Rows: 4, Columns: 4, Goal Value: 100
Solver found a path:
Start(13) -> Down(21) -> Right(25) -> Up(32) -> Right(33) -> Down(32) ->
 Right(44) -> Down(58) -> Left(73) -> Left(71) -> Down(83) -> Right(88)
-> Right(100) -> End
Number of explored nodes: 1157
Time taken: 0.0016306000179611146 seconds
```

Gambar 9. Simulasi Penyelesaian Soal Uji Level Hard dengan Algoritma Brute Force

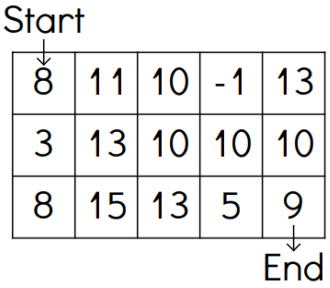
(Sumber: Dokumen Pribadi)

```
$ python NumberMazeBT.py
Welcome to Number Maze Solver using Backtracking Algorithm!
Choose a difficulty level:
1. Easy
2. Moderate
3. Hard
4. Very Hard
Enter the number of your choice: 3
Enter 4 numbers for row 1: 13 7 1 10
Enter 4 numbers for row 2: 8 4 -1 12
Enter 4 numbers for row 3: 13 -2 15 14
Enter 4 numbers for row 4: 2 12 5 12
13      7       1       10
8       4       -1      12
13      -2      15      14
2       12      5       12

Goal value: 100
Rows: 4, Columns: 4, Goal Value: 100
Solver found a path:
Start(13) -> Down(21) -> Right(25) -> Up(32) -> Right(33) -> Down(32) ->
 Right(44) -> Down(58) -> Left(73) -> Left(71) -> Down(83) -> Right(88)
-> Right(100) -> End
Number of explored nodes: 605
Time taken: 0.0012003000010736287 seconds
```

Gambar 10. Simulasi Penyelesaian Soal Uji Level Hard dengan Algoritma Backtracking

(Sumber: Dokumen Pribadi)

4. Level Very Hard



Gambar 11. Soal Uji Level Very Hard

(Sumber: Dokumen Pribadi)

```
$ python NumberMazeBF.py
Welcome to Number Maze Solver using Brute Force Algorithm!
Choose a difficulty level:
1. Easy
2. Moderate
3. Hard
4. Very Hard
Enter the number of your choice: 4
Enter 5 numbers for row 1: 8 11 10 -1 13
Enter 5 numbers for row 2: 3 13 10 10 10
Enter 5 numbers for row 3: 8 15 13 5 9
8       11      10      -1      13
3       13      10      10      10
8       15      13      5       9

Goal value: 100
Rows: 3, Columns: 5, Goal Value: 100
Solver found a path:
Start(8) -> Down(11) -> Right(24) -> Down(39) -> Right(52) -> Up(62)
-> Up(72) -> Right(71) -> Down(81) -> Right(91) -> Down(100) -> End
Number of explored nodes: 753
Time taken: 0.0012530999956652522 seconds
```

Gambar 12. Simulasi Penyelesaian Soal Uji Level Very Hard dengan Algoritma Brute Force

(Sumber: Dokumen Pribadi)

```
$ python NumberMazeBT.py
Welcome to Number Maze Solver using Backtracking Algorithm!
Choose a difficulty level:
1. Easy
2. Moderate
3. Hard
4. Very Hard
Enter the number of your choice: 4
Enter 5 numbers for row 1: 8 11 10 -1 13
Enter 5 numbers for row 2: 3 13 10 10 10
Enter 5 numbers for row 3: 8 15 13 5 9
8       11      10      -1      13
3       13      10      10      10
8       15      13      5       9

Goal value: 100
Rows: 3, Columns: 5, Goal Value: 100
Solver found a path:
Start(8) -> Down(11) -> Right(24) -> Down(39) -> Right(52) -> Up(62)
-> Up(72) -> Right(71) -> Down(81) -> Right(91) -> Down(100) -> End
Number of explored nodes: 397
Time taken: 0.0006729000015184283 seconds
```

Gambar 13. Simulasi Penyelesaian Soal Uji Level Very Hard dengan Algoritma Backtracking

(Sumber: Dokumen Pribadi)

Berdasarkan analisis terhadap keempat kasus uji, algoritma backtracking menonjol dalam efisiensi penggunaan memori dan waktu. Dalam hal penggunaan memori, backtracking berhasil mengeksplorasi simpul-simpul dengan efisiensi yang signifikan lebih tinggi daripada brute force. Dalam keempat kasus uji, backtracking hanya membutuhkan memori untuk mengeksplorasi kurang dari 50% dari jumlah simpul yang dieksplorasi oleh brute force. Artinya, backtracking mampu menemukan solusi dengan menggunakan jumlah memori yang lebih sedikit, menjadikannya pilihan yang lebih efisien dalam pengelolaan sumber daya sistem.

Selain itu, dalam hal waktu, backtracking berhasil menyelesaikan tugas dengan cepat, memerlukan waktu kurang dari 75% dari waktu yang diperlukan oleh brute force untuk mengeksplorasi solusi. Ini menunjukkan bahwa meskipun brute force secara eksplisit menjelajahi semua kemungkinan solusi, backtracking secara cerdas mengurangi jumlah simpul yang harus dieksplorasi, sehingga menghasilkan kinerja yang lebih efisien dalam menyelesaikan permasalahan. Dengan demikian, baik dari segi penggunaan memori maupun waktu, algoritma backtracking menonjol sebagai pendekatan yang lebih efisien dan efektif dibandingkan dengan brute force pada permainan Number Mazes.

IV. Kesimpulan

Dalam permainan Number Mazes, algoritma backtracking menunjukkan keunggulan yang signifikan dalam efisiensi penggunaan memori dan waktu dibandingkan dengan metode brute force. Backtracking berhasil mengeksplorasi simpul-simpul dengan memori yang lebih sedikit, hanya menggunakan kurang dari 50% dari jumlah memori yang diperlukan oleh brute force. Selain itu, dalam hal waktu, backtracking mampu menyelesaikan tugas dengan lebih cepat, hanya memerlukan waktu kurang dari 75% dari waktu yang diperlukan oleh brute force. Dengan demikian, backtracking terbukti menjadi pilihan yang lebih efisien dalam menyelesaikan permasalahan pada permainan Number Mazes, dengan mengoptimalkan penggunaan sumber daya sistem secara efektif.

Ucapan Terima Kasih

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya, penulis dapat menyelesaikan makalah ini. Penulis juga ingin menyampaikan terima kasih kepada dosen pengampu mata kuliah Strategi Algoritma IF2211 2023/2024, yaitu Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc dan Bapak Dr. Rinaldi Munir, yang telah memberikan ilmunya sehingga penulis dapat menyusun makalah ini. Terakhir, penulis juga ingin menyampaikan terima kasih kepada semua pihak yang telah membantu langsung maupun tidak langsung dalam penyelesaian makalah ini.

Video Link at Youtube

https://youtu.be/-tevxChpmmg

Referensi

[1] R. Munir, 'IF2211 Strategi Algoritma · Semester II Tahun 2023/2024', https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf Diakses pada 11 Juni 2024

[2] R. Munir, 'IF2211 Strategi Algoritma · Semester II Tahun 2023/2024', https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf Diakses pada 11 Juni 2024

[3] R. Munir, 'IF2211 Strategi Algoritma · Semester II Tahun 2023/2024', https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf Diakses pada 11 Juni 2024

[4] Dr Mike's Math Games for Kids, 'Printable Number Mazes', https://www.dr-mikes-math-games-for-kids.com/number-mazes.html Diakses pada 11 Juni 2024

[5] Repository Github penulis, 'Number Mazes Solver', https://github.com/mdavaf17/Number-Mazes-Solver

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

Muhammad Dava Fathurrahman, 13522114