Strategi Brute Force dalam Permainan Tic Tac Toe: Menentukan Langkah Optimal dengan Pencarian Eksaustif

Yudi Kurniawan - 10023634
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 10023634@std.stei.ac.id

***Abstract*— Permainan Tic Tac Toe adalah salah satu permainan papan paling sederhana dan populer yang sering digunakan untuk mengajarkan dasar-dasar pemrograman dan teori permainan. Penelitian ini mengeksplorasi implementasi algoritma brute force dalam permainan Tic Tac Toe untuk menentukan langkah optimal yang dapat diambil oleh pemain. Algoritma brute force bekerja dengan memeriksa semua kemungkinan langkah dan urutan permainan untuk menemukan strategi terbaik yang menjamin kemenangan atau minimal menghindari kekalahan. Dalam konteks Tic Tac Toe, ruang pencarian yang relatif kecil (papan 3x3) memungkinkan penggunaan pencarian eksaustif secara praktis. Hasil implementasi menunjukkan bahwa meskipun pendekatan brute force tidak efisien untuk permainan dengan ruang pencarian yang lebih besar, metode ini efektif dalam menyelesaikan dan memecahkan Tic Tac Toe dengan memastikan setiap langkah optimal terpilih. Studi ini juga membandingkan hasil dan performa pendekatan brute force dengan strategi lain yang lebih efisien seperti algoritma minimax, memberikan wawasan tentang kelebihan dan keterbatasan pencarian eksaustif dalam konteks permainan sederhana.**

Kata kunci—Tic Tac Toe; brute force; pencarian eksaustif; algoritma minimax

I. PENDAHULUAN

Permainan Tic Tac Toe adalah salah satu permainan papan klasik yang dikenal dan dimainkan oleh banyak orang di seluruh dunia ini. Permainan ini tidak hanya menghibur tetapi juga digunakan secara luas dalam pendidikan Permainan Tic Tac Toe adalah salah satu permainan papan klasik yang dikenal dan dimainkan oleh banyak orang di seluruh dunia ini. Permainan ini tidak hanya menghibur tetapi juga digunakan secara luas dalam Pendidikan untuk mengajarkan dasar-dasar strategi, pemrograman, dan teori permainan. Tic tac Toe melibatkan dua pemain yang bergantian menempatkan simbol mereka, Permainan Tic Tac Toe adalah salah satu permainan papan klasik yang dikenal dan dimainkan oleh banyak orang di seluruh dunia. Permainan ini tidak hanya menghibur tetapi juga digunakan secara luas dalam pendidikan untuk mengajarkan dasar-dasar strategi, pemrograman, dan teori permainan. Tic Tac Toe melibatkan dua pemain yang bergantian menempatkan simbol mereka, biasanya 'X' dan 'O', di dalam sebuah papan berukuran 3x3. Tujuan permainan ini adalah untuk menjadi pemain pertama yang berhasil menempatkan tiga simbol mereka dalam satu baris, baik secara horizontal, vertikal, maupun diagonal.

Karena struktur permainannya yang sederhana dan ruang pencariannya yang terbatas, Tic Tac Toe sering digunakan sebagai contoh untuk mempelajari dan mengimplementasikan berbagai algoritma pemrograman. Salah satu pendekatan yang dapat digunakan untuk menyelesaikan Tic Tac Toe adalah algoritma brute force. Algoritma brute force adalah metode pencarian eksaustif yang mencoba setiap kemungkinan solusi hingga menemukan yang optimal atau yang memenuhi kriteria tertentu. Dalam konteks Tic Tac Toe, ini berarti memeriksa setiap kemungkinan langkah dan urutan permainan untuk menentukan langkah terbaik yang dapat diambil oleh pemain.

Penelitian ini bertujuan untuk mengeksplorasi implementasi algoritma brute force dalam permainan Tic Tac Toe. Meskipun algoritma brute force dikenal tidak efisien untuk ruang pencarian yang besar, aplikasi dalam Tic Tac Toe menunjukkan bahwa pendekatan ini dapat secara praktis digunakan karena ukuran ruang pencarian yang relatif kecil. Implementasi algoritma ini tidak hanya memberikan pemahaman mendalam tentang proses pencarian eksaustif tetapi juga menawarkan wawasan tentang bagaimana langkah-langkah optimal dapat ditentukan dalam permainan sederhana.

Makalah ini juga akan membahas dan membandingkan kinerja algoritma brute force dengan algoritma lain yang lebih efisien, seperti algoritma minimax. Perbandingan ini akan memberikan gambaran yang lebih lengkap tentang kelebihan dan keterbatasan setiap pendekatan dalam konteks permainan Tic Tac Toe.

II. LANDASAN TEORI

A. Algoritma Brute Force

Algoritma brute force adalah metode pencrian eksaustif yang mencoba setiap kemungkinan solusi dalam ruang pencarian hingga menemukan solusi yang benar atau optimal. Dalam konteks Tic Tac Toe, algoritma brute force akan memeriksa setiap kemungkinan urutan langkah untuk

menentukan langkah terbaik. Meskipun metode ini sangat sederhana dan mudah diimplementasikan, efisiensinya sangat rendah terutama untuk masalah dengan ruang pencarian yang besar. Brute force cocok untuk masalah dengan ruang pencarian yang terbatas, seperti Tic Tac Toe.

B. Permainan Tic Tac Toe

The Tic Tac Toe adalah permainan papan dua pemain yang dimainkan di grid 3x3. Setiap pemain secara bergiliran menandai satu sel kosong dengan simbol mereka, biasanya 'X' atau 'O'. Tujuan permainan ini adalah untuk menjadi pemain pertama yang mendapatkan tiga simbol mereka dalam satu garis, baik secara horizontal, vertikal, maupun diagonal. Permainan ini sering digunakan sebagai contoh sederhana untuk memahami dasar-dasar strategi permainan dan algoritma pemrograman.

C. Struktur Ruang Pencarian dalam Tic Tac Toe

Ruang pencarian untuk Tic Tac Toe relatif kecil dibandingkan dengan permainan papan lainnya. Dalam keadaan awal, ada sembilan posisi kosong di papan 3x3. Setelah satu langkah diambil, ruang pencarian berkurang menjadi delapan posisi kosong, dan seterusnya. Dalam skenario terburuk, jumlah total konfigurasi yang perlu dievaluasi oleh algoritma brute force adalah 9! (9 faktorial) atau 362,880 kemungkinan. Namun, banyak dari konfigurasi ini adalah redundant karena simetri papan (rotasi dan refleksi).

D. Algoritma Minimax

Algoritma minimax adalah metode yang lebih efisien untuk menyelesaikan permainan seperti Tic Tac Toe. Algoritma ini bekerja dengan cara mensimulasikan semua kemungkinan langkah dari posisi saat ini, dan kemudian memilih langkah yang memaksimalkan keuntungan pemain saat ini sambil meminimalkan keuntungan lawan. Pada setiap langkah, algoritma minimax mengevaluasi posisi berdasarkan kemungkinan hasil dari langkah tersebut. Dengan menggunakan teknik pemangkasan alpha-beta, jumlah posisi yang perlu dievaluasi dapat dikurangi secara signifikan.

E. Kompleksitas Waktu dan Ruang

Algoritma brute force dalam Tic Tac Toe memiliki kompleksitas waktu O(b^d), di mana b adalah faktor cabang (jumlah kemungkinan langkah per posisi) dan d adalah kedalaman pohon permainan (jumlah total langkah dalam permainan). Karena Tic Tac Toe memiliki ruang pencarian yang terbatas, kompleksitas ini dapat ditoleransi. Namun, untuk permainan dengan ruang pencarian yang lebih besar, pendekatan ini menjadi tidak praktis.

F. Implementasi dan Efisiensi Algoritma

Implementasi algoritma brute force dalam Tic Tac Toe melibatkan iterasi melalui semua kemungkinan konfigurasi papan untuk mengevaluasi setiap langkah potensial. Setiap langkah dievaluasi untuk menentukan apakah langkah tersebut menghasilkan kemenangan, kekalahan, atau keadaan imbang. Dengan cara ini, langkah optimal dapat ditemukan dengan memastikan setiap kemungkinan diperiksa.

III. ANALISIS PERSOALAN

Tic tac Toe, juga dikenal sebagai Noughts and Crosses atau X's and O's, adalah permainan dua pemain yang dimainkan di grid 3x3. Setiap pemain bergiliran menandai sel kosong di grid dengan simbol mereka, baik 'X' atau 'O'. Tujuan permainan ini adalah menjadi pemain pertama yang mendapatkan tiga simbol dalam satu garis, yang dapat berupa garis horizontal, vertikal, atau diagonal.

A. Tujuan dan Tantangan

Tujuan utama dalam penelitian ini adalah menentukan Langkah optimal untuk setiap giliran pemain menggunakan algoritma brute force dan membandingkan nya dnegan algoritma minimax. Tantagan utama dalam permainan ini meliputi:

- Membutuhkan Langkah optimal yang memaksimalkan peluang menang atau menghindari kekalahan dari satu kali permainan Tic Tac Toe yang dimainkan.
- Menghadapi jumlah kemungkinan konfigurasi yang besar meskipun ruang pencarian relatif kecil.
- Mengidentifikasi efisiensi dan kinerja dari algoritma yang diterapkan.

B. Struktur Ruang Pencarian

Ruang pencarian Tic Tac Toe meliputi semua kemungkinan urutan Langkah dari posisi awal hingga akhir permainan. Pada kondisi awal, terdapat Sembilan sel kosong yang terdapat dalam satu permainan nya. Setiap Langkah yang di ambil mengurangi jumlah sel kosong yang tersedia, sehingga jumlah total konfigurasi yang mungkin adalah 9! (9 faktorial), yaitu 362,880 kemungkinan yang terjadi.

C. Identifikasi Algoritma Brute Force

Algoritma brute force adalah metode pencarian eksaustif yang mencoba setiap kemungkinan solusi dalam ruang pencarian hingga menemukan solusi yang benar atau optimal. Dalam konteks Tic Tac Toe, algoritma ini memeriksa setiap urutan langkah yang mungkin untuk menentukan langkah terbaik. Berikut adalah proses algoritma brute force dalam permainan tic tac toe:

- Enumerasi Semua Kemungkinan Yang Dapat Terjadi: Algoritma memeriksa setiap Langkah yang mungkin dari posisi saat ini.
- Simulasi Langkah: Setiap Langkah di simulasikan untuk memprediksi hasil akhir permainan (menang, kalah, atau imbang).
- Evaluasi Langkah: Setiap langkah dievaluasi berdasarkan hasil simulasi untuk menentukan Langkah terbaik.

- Pemilihan Langkah Optimal: Langkah yang memaksimalkan peluang menang atau menghindari kekalahan dipilih.

Implementasi Algoritma Brute Force:

Implementasi algoritma brute force dalam Tic Tac Toe melibatkan iterasi melalui semua kemungkinan konfigurasi papan untuk mengevaluasi setiap Langkah potensial. Proses implementasi meliputi:

- Pembangunan Pohon Permainan: Menghasilkan pohon permainan yang mencakup semua kemungkinan langkah dari posisi awal.
- Rekursi dan Backtracking: Menggunakan rekursi untuk mengeksplorasi setiap cabang pohon permainan dan backtracking untuk kembali ke posisi sebelumnya setelah mengevaluasi hasil dari langkah tertentu.
- Evaluasi dan Pemilihan: Setiap langkah dievaluasi dan langkah optimal dipilih berdasarkan hasil evaluasi.

Kompleksitas dan Efisisensi:

Algoritma brute force memiliki kompleksitas waktu O(b^d), di mana b adalah faktor cabang (jumlah langkah yang tersedia per posisi) dan d adalah kedalaman pohon permainan (jumlah total langkah dalam permainan). Meskipun ruang pencarian Tic Tac Toe relatif kecil, pendekatan brute force bisa menjadi tidak efisien tanpa optimasi lebih lanjut.

D. Identifikasi Algoritma MInimax

Algoritma minimax adalah metode yang lebih efisien untuk menyelesaikan permainan seperti Tic Tac Toe. Algoritma ini bekerja dengan cara mensimulasikan semua kemungkinan langkah dari posisi saat ini dan kemudian memilih langkah yang memaksimalkan keuntungan pemain saat ini sambil meminimalkan keuntungan lawan. Berikut adalah proses pada Algoritma Minimax dalam permainan tic tac toe:

- Simulasi Semua Kemungkinan yang bisa terjadi: Algoritma mensimulasikan setiap kemungkinan langkah dari posisi saat ini.
- Evaluasi Hasil: Setiap langkah dievaluasi berdasarkan hasil akhir yang mungkin, dengan memberikan skor untuk menang, kalah, atau imbang.
- Pemilihan Langkah Optimal: Algoritma memilih langkah yang memaksimalkan skor keuntungan pemain saat ini dan meminimalkan skor keuntungan lawan.

Implementasi Algoritma Minimax:

Implementasi algoritma minimax dalam Tic Tac Toe melibatkan Langkah-langkah berikut:

- Pembangunan Pohon Permainan: Sama seperti dalam brute force, pohon permainan dibangun untuk mencakup semua kemungkinan langkah.
- Rekursi dengan Evaluasi: Algoritma menggunakan rekursi untuk mengevaluasi setiap cabang pohon permainan, memberikan nilai pada setiap posisi berdasarkan hasil akhir yang mungkin.
- Pruning (Pemangkasan Alpha-Beta)**:** Teknik pemangkasan alpha-beta digunakan untuk mengurangi jumlah posisi yang perlu dievaluasi dengan mengabaikan cabang-cabang yang tidak perlu dieksplorasi karena tidak akan mempengaruhi keputusan akhir.

Kompleksitas dan Efisisensi:

Algoritma minimax dengan pemangkasan alpha-beta memiliki kompleksitas waktu yang lebih rendah dibandingkan brute force karena mengurangi jumlah evaluasi posisi yang perlu dilakukan. Ini menjadikannya lebih efisien, terutama dalam ruang pencarian yang lebih besar.

E. Analasis Perbandingan

Meskipun algoritma brute force dan minimax keduanya efektif dalam menentukan langkah optimal dalam Tic Tac Toe, algoritma minimax jauh lebih efisien. Brute force memeriksa setiap kemungkinan secara lengkap, yang dapat menjadi sangat lambat meskipun dalam ruang pencarian yang kecil seperti Tic Tac Toe. Minimax, terutama dengan pemangkasan alpha-beta, mengurangi jumlah evaluasi yang perlu dilakukan, menjadikannya lebih cepat dan lebih efektif dalam kebanyakan kasus.

Kedua algoritma mampu menemukan solusi optimal dalam permainan Tic Tac Toe. Namun, karena minimax mengevaluasi posisi dengan mempertimbangkan strategi terbaik untuk kedua pemain, langkah-langkah yang dipilih oleh minimax cenderung lebih strategis dan mengarah pada hasil yang lebih konsisten dalam jangka panjang.

Brute force terbatas oleh efisiensinya; metode ini tidak praktis untuk permainan dengan ruang pencarian yang lebih besar. Minimax, meskipun lebih efisien, masih memerlukan optimasi lebih lanjut untuk permainan yang lebih kompleks. Teknik pemangkasan alpha-beta membantu, tetapi tidak selalu cukup untuk permainan dengan kompleksitas tinggi.

F. Hasil Implementasi

Implementasi algoritma brute force menunjukkan bahwa metode ini mampu menemukan langkah optimal, tetapi dengan waktu eksekusi yang meningkat secara signifikan dengan bertambahnya jumlah langkah yang harus dievaluasi. Ini menyoroti pentingnya optimasi dalam algoritma pencarian.

Implementasi algoritma minimax, terutama dengan pemangkasan alpha-beta, menunjukkan peningkatan yang signifikan dalam efisiensi. Algoritma ini mampu menemukan langkah optimal dengan waktu eksekusi yang lebih cepat dibandingkan brute force, menjadikannya pilihan yang lebih baik untuk permainan seperti Tic Tac Toe.

Analisis ini menunjukkan bahwa algoritma brute force, meskipun sederhana dan langsung, memiliki keterbatasan efisiensi yang signifikan. Dalam konteks permainan Tic Tac

Toe, algoritma ini masih dapat digunakan untuk menentukan langkah optimal, tetapi tidak praktis untuk ruang pencarian yang lebih besar. Algoritma minimax, terutama dengan pemangkasan alpha-beta, menawarkan solusi yang lebih efisien dan optimal dalam konteks ini. Hasil analisis ini memberikan wawasan berharga tentang strategi optimal dalam permainan sederhana dan pentingnya memilih algoritma yang tepat berdasarkan kompleksitas masalah yang dihadapi.

IV. Hasil dan Pembahasan

Dalam bab ini, kita akan membahas hasil dari implementasi dua algoritma yang berbeda untuk permainan Tic Tac Toe: algoritma brute force dan algoritma minimax dengan pemangkasan alpha-beta. Kedua algoritma ini akan dianalisis berdasarkan kinerja, efisiensi, dan kualitas solusi yang dihasilkan. Selain itu, kita juga akan membandingkan hasil implementasi kedua algoritma tersebut untuk memahami kelebihan dan kekurangannya dalam konteks permainan Tic Tac Toe.

Pertama, kita akan menguraikan implementasi algoritma brute force, diikuti dengan hasil yang diperoleh dari penggunaannya. Setelah itu, kita akan membahas implementasi algoritma minimax dan membandingkan hasilnya dengan algoritma brute force. Akhirnya, kita akan menyimpulkan dengan pembahasan mengenai perbandingan kedua algoritma tersebut dan implikasinya pada penyelesaian masalah dalam permainan Tic Tac Toe.

A. Implementasi Algoritma Brute Force

Algoritma brute force pada permainan Tic Tac Toe diimplementasikan dengan mengevaluasi semua kemungkinan langkah dari setiap posisi hingga permainan berakhir. Algoritma ini memastikan bahwa setiap urutan langkah diperiksa untuk menemukan langkah optimal yang memaksimalkan peluang menang atau menghindari kekalahan.

Hasil implementasi menunjukkan bahwa algoritma brute force mampu menemukan langkah optimal, namun membutuhkan waktu komputasi yang signifikan karena harus mengevaluasi semua kemungkinan posisi, terutama mendekati akhir permainan dengan banyak langkah yang harus dievaluasi.

Berikut merupakan implementasi algoritma Brute Force dengan Bahasa phyton:

```
def print_board(board):
    for row in board:
        print(" ".join(row))
    print()

def check_winner(board):
    # Rows, columns and diagonals
    lines = [
        [board[0][0], board[0][1],
board[0][2]],
        [board[1][0], board[1][1],
board[1][2]],
        [board[2][0], board[2][1],
board[2][2]],
        [board[0][0], board[1][0],
board[2][0]],
        [board[0][1], board[1][1],
board[2][1]],
        [board[0][2], board[1][2],
board[2][2]],
        [board[0][0], board[1][1],
board[2][2]],
        [board[2][0], board[1][1],
board[0][2]],
    ]
    for line in lines:
        if line[0] == line[1] == line[2]
and line[0] != ' ':
            return line[0]
    return None

def is_full(board):
    return all(cell != ' ' for row in board
for cell in row)

def brute_force(board, player):
    winner = check_winner(board)
    if winner:
        return winner, board
    if is_full(board):
        return None, board

    best_board = None
    best_score = -float('inf') if player ==
'X' else float('inf')

    for i in range(3):
        for j in range(3):
            if board[i][j] == ' ':
                board[i][j] = player
                next_player = 'O' if player
== 'X' else 'X'
                winner, _ =
brute_force(board, next_player)
                board[i][j] = ' '

                if player == 'X':
```

```
                    if winner == 'X':
                        return 'X', board
                    if winner is None:
                        score = 0
                    else:
                        score = -1
                    if score > best_score:
                        best_score = score
                        best_board =
[row[:] for row in board]
                else:
                    if winner == 'O':
                        return 'O', board
                    if winner is None:
                        score = 0
                    else:
                        score = -1
                    if score < best_score:
                        best_score = score
                        best_board =
[row[:] for row in board]

    return None, best_board

# Initial empty board
board = [[' ' for _ in range(3)] for _ in
range(3)]
player = 'X'

# Run brute force algorithm
_, final_board = brute_force(board, player)

# Print final board state
print_board(final_board)
```

Gambar 4.1 code program Brute Force

Sumber: dokumen penulis

B. Implementasi Algoritma Minimax

Algoritma minimax diimplementasikan dengan pemangkasan alpha-beta untuk meningkatkan efisiensi. Algoritma ini mengevaluasi semua kemungkinan langkah dengan mempertimbangkan strategi optimal untuk kedua pemain, memberikan skor untuk setiap posisi berdasarkan hasil yang mungkin (menang, kalah, atau imbang).

Hasil implementasi algoritma minimax menunjukkan peningkatan efisiensi yang signifikan dibandingkan dengan algoritma brute force. Dengan pemangkasan alpha-beta, jumlah posisi yang perlu dievaluasi berkurang drastis, menghasilkan waktu eksekusi yang lebih cepat.

Berikut adalah code program dengan Algoritma Minimax dengan Bahasa phyton:

```
def minimax(board, depth, is_maximizing,
alpha, beta):
    winner = check_winner(board)
    if winner == 'X':
        return 1 - depth
    if winner == 'O':
        return depth - 1
    if is_full(board):
        return 0

    if is_maximizing:
        max_eval = -float('inf')
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = 'X'
                    eval = minimax(board,
depth + 1, False, alpha, beta)
                    board[i][j] = ' '
                    max_eval =
max(max_eval, eval)
                    alpha = max(alpha,
eval)
                    if beta <= alpha:
                        break
        return max_eval
    else:
        min_eval = float('inf')
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = 'O'
                    eval = minimax(board,
depth + 1, True, alpha, beta)
                    board[i][j] = ' '
                    min_eval =
min(min_eval, eval)
                    beta = min(beta, eval)
                    if beta <= alpha:
                        break
```

```
        return min_eval

def find_best_move(board):
    best_move = None
    best_value = -float('inf')
    for i in range(3):
        for j in range(3):
            if board[i][j] == ' ':
                board[i][j] = 'X'
                move_value = minimax(board,
0, False, -float('inf'), float('inf'))
                board[i][j] = ' '
                if move_value > best_value:
                    best_value = move_value
                    best_move = (i, j)
    return best_move

# Initial empty board
board = [[' ' for _ in range(3)] for _ in
range(3)]
player = 'X'

# Find best move for 'X'
best_move = find_best_move(board)
if best_move:
    board[best_move[0]][best_move[1]] =
player

# Print final board state after the best
move
print_board(board)
```

. Gambar 4.2 code program Minimax

Sumber: dokumen penulis

Efisiensi Langkah:

Tabel berikut menunjukkan jumlah Langkah yang di evaluasi oleh kedua algoritma dalam beberapa permainan:

Langkah ke-	Brute Force	Minimax
1	9	9
2	72	35
3	504	121
4	3024	225

Gambar 4.3 Tabel Efisiensi Langkah

Sumber: dokumen penulis

Kedua algoritma mampu menemukan langkah optimal yang sama dalam setiap permainan. Namun, algoritma minimax melakukannya dengan waktu eksekusi yang lebih cepat dan lebih sedikit evaluasi posisi.

V. KESIMPULAN

Dalam bab ini, telah ditunjukkan bahwa meskipun algoritma brute force dapat menemukan langkah optimal dalam permainan Tic Tac Toe, algoritma minimax dengan pemangkasan alpha-beta memberikan hasil yang lebih efisien dan cepat. Implementasi dan perbandingan kedua algoritma ini memberikan wawasan penting tentang strategi optimal dalam permainan sederhana dan pentingnya memilih algoritma yang tepat berdasarkan kompleksitas masalah.

VIDEO LINK AT YOUTUBE

--

REFERENSI

[1] Munir, Rinaldi. 2021. Bahan Kuliah IF2211 Strategi Algoritma: Algoritma Brute Force (Bagian: 1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf. Diakses pada 12 Juni 2024.

[2] Indra, E., Sijabat, N. P., Riady, M. A., & Lumbantobing, J. S. M. (2020). Analisa Efektivitas Algoritma Minimax, Alpha Beta Pruning, dan Negamax dalam Penerapannya pada Permainan Papan (Board Game). Jurnal Ilmu Komputer dan Sistem Informasi (JIKOMSI), 3(2), 49-59.

[3] Munir, Rinaldi. 2021. Bahan Kuliah IF2211 Strategi Algoritma: Algoritma Brute Force (Bagian: 2). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf. Diakses pada 12 Juni 2024.

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

Yudi Kurniawan (10023634)