LAPORAN TUGAS MATA KULIAH KECERDASAN BUATAN PRAKTIKUM

"IMPLEMENTASI ALGORITMA MINIMAX PADA PERMAINAN CONNECT 4"

Disusun Oleh :

Amelia Nur Indah Puspita	(434221052)
Lady Cessa Nadinda	(434221056)
Lintang Sekar Wangi	(434221053)
Salma Aida Yasmi	(434221037)

Program Studi D4 Teknik Informatika

Fakultas Vokasi

UNIVERSITAS AIRLANGGA

09 Mei 2024

DAFTAR ISI

BAB I - PENDAHULUAN

1.1 Latar Belakang

Permainan strategi merupakan permainan yang memerlukan kecerdasan dan ketelitian dalam memilih jurus yang optimal untuk mencapai kemenangan. Dalam permainan strategi seringkali terdapat banyak gerakan yang dapat dilakukan setiap giliran, sehingga diperlukan algoritma khusus untuk memprediksi gerakan yang paling optimal.

Salah satu algoritma yang paling umum digunakan dalam permainan strategi, permainan strategi adalah Algoritma *minimax*. Algoritma ini bekerja dengan cara mensimulasikan seluruh kemungkinan gerakan yang dapat dilakukan pada setiap putaran, kemudian memilih gerakan yang menghasilkan keuntungan terbesar atau kerugian terkecil, tergantung peran pemain dalam permainan tersebut.

Dalam permainan Connect 4, Algoritma *Minimax* dapat digunakan untuk memprediksi gerakan optimal di setiap putaran dengan mempertimbangkan kemungkinan gerakan lawan dan gerakan Berikutnya yang mungkin tersedia. Namun karena *game tree* pada Connect 4 bisa berukuran sangat besar, maka penggunaan algoritma *Minimax* perlu dioptimalkan dengan teknik pemangkasan dan heuristik untuk mempercepat proses evaluasi *node game tree*. Selain itu, teknik *Greedy* juga dapat digunakan untuk mempercepat proses evaluasi pada *node* pohon permainan dengan skor heuristik yang buruk. Dengan menggabungkan teknik *Greedy* dengan algoritma *Minimax* diharapkan dapat meningkatkan efisiensi dan efektifitas dalam memprediksi pergerakan optimal pada setiap putaran permainan Connect 4.

1.2 Tujuan

Laporan ini bertujuan untuk membahas implementasi algoritma *Minimax* dengan teknik *Greedy* pada game strategi Connect 4. Penulis akan menjelaskan secara detail implementasi algoritma *Minimax* dan teknik *Greedy* pada game Connect 4 serta hasil yang dicapai. Eksperimen dilakukan untuk menguji keefektifan algoritma ini.

Selain itu, laporan ini juga akan membahas heuristik yang dapat digunakan pada game Connect 4 untuk mengevaluasi posisi game dengan cepat dan akurat. Kami berharap laporan ini dapat berkontribusi pada pengembangan aplikasi strategi permainan yang lebih efektif dan efisien.

BAB II - DASAR TEORI

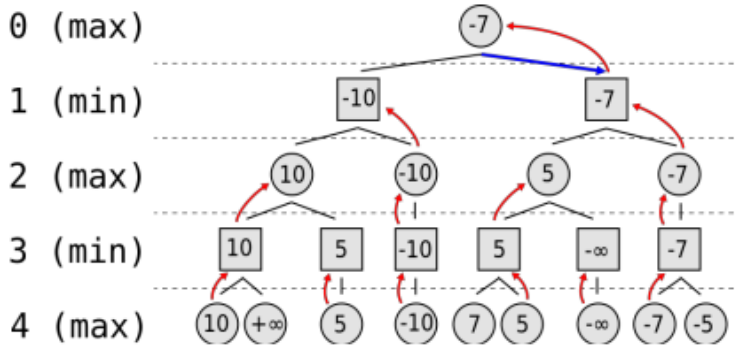
2.1 Kecerdasan Buatan

Kecerdasan buatan (AI) adalah cabang ilmu komputer yang bertujuan untuk mengembangkan sistem atau entitas yang mampu meniru, mewakili, dan bahkan meningkatkan kemampuan manusia dalam melakukan tugas-tugas menuntut kecerdasan. AI bertujuan untuk membantu mesin memahami lingkungannya, berpikir rasional, dan melakukan tugas-tugas seperti pemecahan masalah, pengambilan keputusan, pengenalan pola, dan bahasa alami, yang sebelumnya hanya dapat dilakukan oleh manusia.

Dalam konteks *game* , AI berperan dalam menciptakan agen cerdas yang mampu berperan aktif dalam lingkungan *game*. Agen cerdas ini diprogram untuk membuat keputusan optimal dalam situasi tertentu, beradaptasi dengan strategi lawannya, dan belajar dari pengalaman mereka. meningkatkan performa *game* mereka Melalui penggunaan teknik seperti algoritma pencarian, pembelajaran mesin, dan pengetahuan pemodelan, AI dapat menciptakan agen *game* yang mampu menantang permainan manusia atau bahkan bermain setara dengan manusia terbaik dalam *game* tertentu.

2.2 Algoritma Minimax

Algoritma Minimax adalah salah satu algoritma yang sering digunakan pada permainan strategi untuk memprediksi gerakan yang optimal pada setiap giliran, dengan mempertimbangkan kemungkinan gerakan lawan dan kemungkinan gerakan selanjutnya. Algoritma ini bekerja dengan cara membangun pohon permainan untuk merepresentasikan semua kemungkinan gerakan yang dapat diambil pada setiap giliran.



Gambar 2.1 Contoh Algoritma MiniMax

Pada setiap simpul pohon, algoritma akan mengevaluasi keuntungan atau kerugian yang mungkin terjadi pada pemain, dan kemudian memilih gerakan yang memberikan

keuntungan terbesar atau kerugian terkecil, tergantung pada peran pemain pada simpul tersebut. Algoritma kemudian akan melakukan pengecekan pada simpul selanjutnya, dan seterusnya hingga mencapai simpul daun pohon. Pada simpul daun, algoritma akan mengevaluasi keuntungan atau kerugian dari posisi permainan yang telah dihasilkan.
Setelah algoritma mengevaluasi semua simpul pada pohon permainan, algoritma akan memilih gerakan yang memberikan keuntungan terbesar atau kerugian terkecil pada simpul awal pohon. Gerakan tersebut kemudian diambil sebagai gerakan terbaik pada giliran tersebut.

Ada dua jenis pemain dalam algoritma Minimax, yaitu pemain maks dan pemain min. Pemain maks berusaha untuk memaksimalkan keuntungan dan meminimalkan kerugian, sementara pemain min berusaha untuk meminimalkan keuntungan dan memaksimalkan kerugian. Pada setiap giliran, algoritma akan bergantian sebagai pemain maks dan pemain min untuk memprediksi gerakan yang optimal.

Namun, karena pohon permainan pada permainan strategi bisa sangat besar, penggunaan algoritma Minimax harus dioptimalkan dengan teknik pruning dan heuristik. Pruning dapat membantu memotong cabang-cabang dari pohon permainan yang tidak perlu dievaluasi karena tidak memberikan kontribusi signifikan pada solusi optimal.

Sementara itu, heuristik dapat digunakan untuk mengevaluasi posisi permainan dengan cepat dan memberikan skor yang cukup akurat, sehingga algoritma Minimax dapat mengabaikan simpul-simpul yang memiliki skor yang buruk. Algoritma Minimax telah diterapkan pada berbagai jenis permainan strategi, seperti catur, dam, dan permainan video seperti tic-tac-toe dan permainan real-time strategy. Algoritma ini juga dapat dikombinasikan dengan teknik kecerdasan buatan lainnya, seperti algoritma Alpha-Beta Pruning, untuk meningkatkan efisiensi dan efektivitas dalam memprediksi gerakan yang optimal pada setiap giliran.

2.3 Permainan Connect 4

Connect Four adalah permainan strategi yang dimainkan oleh dua pemain menggunakan papan permainan berukuran 6x7 yang terdiri dari 42 slot. Tujuan dari permainan ini adalah untuk menjadi pemain pertama yang berhasil membentuk garis horizontal, vertikal, atau diagonal yang terdiri dari empat cakram pemain sendiri.

Setiap pemain memiliki cakram berbeda, biasanya dengan warna yang berbeda pula. Pemain bergantian menempatkan cakram mereka pada salah satu kolom di bagian atas

papan permainan. Cakram akan jatuh ke posisi terbawah pada kolom tersebut, mengisi slot kosong yang tersedia.

Pemain harus secara cerdas memilih kolom yang tepat untuk menempatkan cakram mereka. Mereka perlu mempertimbangkan langkah mereka sendiri untuk membentuk garis empat cakram secara horizontal, vertikal, atau diagonal, sambil juga menghalangi lawan dari mencapai tujuan yang sama.

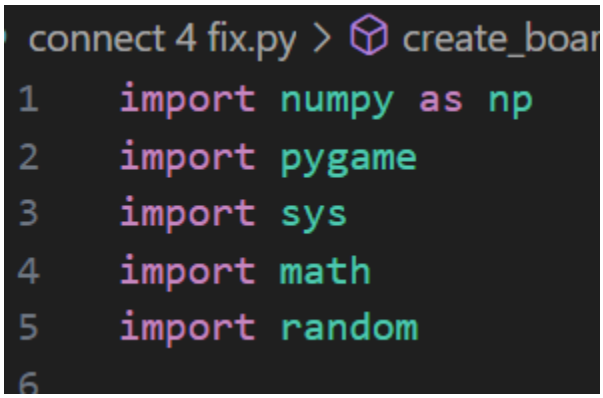
Permainan Connect Four menawarkan kombinasi strategi ofensif dan defensif. Pemain harus merencanakan langkah mereka dengan baik, mencari peluang untuk membentuk garis sendiri sambil mencegah lawan melakukan hal yang sama. Kecepatan dan kemampuan untuk melihat kemungkinan langkah berikutnya juga menjadi faktor penting dalam meraih kemenangan.

Connect Four adalah permainan yang populer di berbagai platform, termasuk versi fisik, permainan komputer, dan aplikasi seluler. Permainan ini dapat dimainkan oleh pemain dengan berbagai tingkat keahlian, dari pemula hingga tingkat kompetitif.

Dalam pengembangan algoritma atau kecerdasan buatan, Connect Four menjadi tantangan menarik untuk dipecahkan. Algoritma Minimax, yang diterapkan dalam contoh kode yang diberikan sebelumnya, adalah salah satu cara untuk mengembangkan bot atau program komputer yang cerdas dalam memainkan permainan ini.

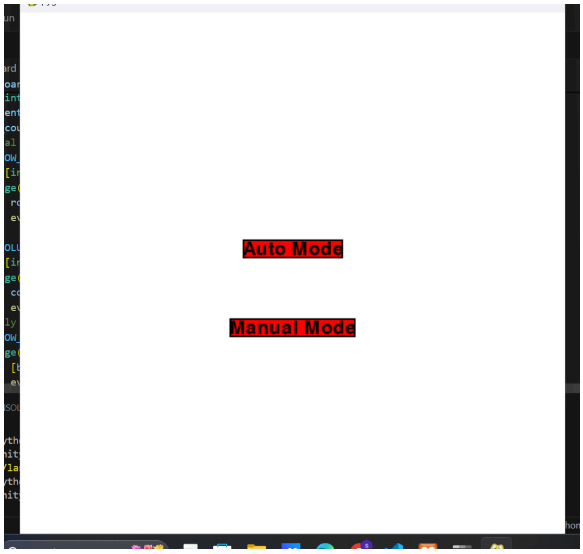
BAB III - IMPLEMENTASI

Permainan Connect 4 ini dirancang menggunakan bahasa pemrograman *python* dengan beberapa modul yang digunakan yaitu numpy, pygame, sys, math, dan random



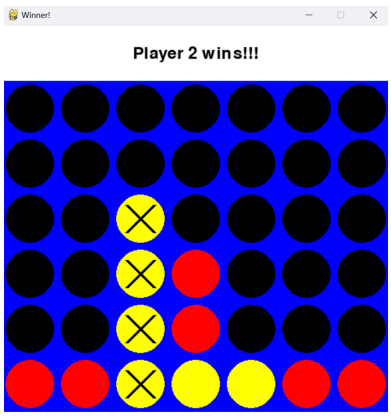
gambar 1. library python

Pada permainan ini kami membuat 2 mode permainan yaitu *Auto* mode dan *Manual* mode, dimana ini dapat memberikan kebebasan akses pengguna dalam memilih ingin bermain melawan komputer (Auto Mode) atau melawan pengguna atau user lain (Manual Mode).

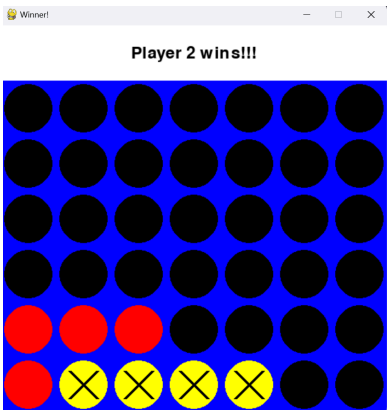


gambar 2. mode permainan

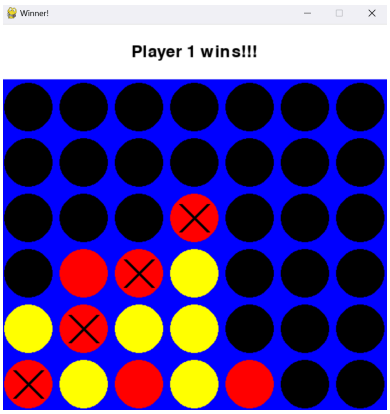
Kriteria kemenangan dari permainan ini yaitu terdapat pola barisan yang terdiri dari 4 kepingan baik secara vertikal, horizontal, maupun diagonal



gambar 3. posisi menang vertikal



gambar 4. posisi menang horizontal



gambar 5. posisi menang diagonal

Berikut adalah penjelasan dari source code yang kami terapkan untuk membuat permainan Connect 4 yang menerapkan algoritma Minimax :

3.1 Tampilan Permainan

```python
# Function to draw the board
def draw_board(board, winning_pos=None):
    for c in range(COLUMN_COUNT):
        for r in range(ROW_COUNT):
            pygame.draw.rect(screen, BLUE, (c * SQUARESIZE, r * SQUARESIZE + SQUARESIZE, SQUARESIZE, SQUARESIZE))
            pygame.draw.circle(screen, BLACK, (int(c * SQUARESIZE + SQUARESIZE / 2), int(r * SQUARESIZE + SQUARESIZE + SQUARESIZE / 2)), RADIUS)

    for c in range(COLUMN_COUNT):
        for r in range(ROW_COUNT):
            if board[r][c] == 1:
                pygame.draw.circle(screen, RED, (int(c * SQUARESIZE + SQUARESIZE / 2), height - int(r * SQUARESIZE + SQUARESIZE / 2)), RADIUS)
            elif board[r][c] == 2:
                pygame.draw.circle(screen, YELLOW, (int(c * SQUARESIZE + SQUARESIZE / 2), height - int(r * SQUARESIZE + SQUARESIZE / 2)), RADIUS)
```

gambar 6. code penggambaran board

Function `draw_board` digunakan untuk menggambarkan papan permainan connect 4 beserta bidak-bidak permainannya, terdapat beberapa looping dimana loop pertama menggambarkan kotak persegi untuk setiap sel di papan, loop kedua digunakan untuk menggambar lingkaran untuk setiap bidak yang telah ditempatkan. Pemain pertama dilambangkan dengan lingkaran atau bidak berwarna merah sedangkan lingkaran atau bidak berwarna kuning melambangkan pemain kedua

```python
# Function to display the menu screen
def menu_screen():
    screen.fill((255, 255, 255))
    font = pygame.font.Font(None, 36)
    text_auto = font.render("Auto Mode", True, (0, 0, 0))
    text_manual = font.render("Manual Mode", True, (0, 0, 0))
    text_auto_rect = text_auto.get_rect(center=(width // 2, height // 2 - 50))
    text_manual_rect = text_manual.get_rect(center=(width // 2, height // 2 + 50))

    # Draw buttons with red background and black border
    pygame.draw.rect(screen, RED, text_auto_rect, 0)  # Red button for "Auto Mode"
    pygame.draw.rect(screen, RED, text_manual_rect, 0)  # Red button for "Manual Mode"

    # Check if cursor is hovering over the button
    if text_auto_rect.collidepoint(pygame.mouse.get_pos()):
        pygame.draw.rect(screen, BLACK, text_auto_rect, 2)  # Add black border if hovering
        pygame.mouse.set_cursor(*pygame.cursors.tri_left)  # Change cursor to indicate clickable
    else:
        pygame.draw.rect(screen, BLACK, text_auto_rect, 2)  # Black border for "Auto Mode" button

    if text_manual_rect.collidepoint(pygame.mouse.get_pos()):
        pygame.draw.rect(screen, BLACK, text_manual_rect, 2)  # Add black border if hovering
        pygame.mouse.set_cursor(*pygame.cursors.tri_left)  # Change cursor to indicate clickable
    else:
        pygame.draw.rect(screen, BLACK, text_manual_rect, 2)  # Black border for "Manual Mode" button

    screen.blit(text_auto, text_auto_rect)
    screen.blit(text_manual, text_manual_rect)
    pygame.display.update()
    return text_auto_rect, text_manual_rect
```

gambar 7. code display screen permainan

Function `menu_screen` digunakan untuk menampilkan screen atau layar permainan. Pada tampilan ini terdapat dua mode yaitu Auto Mode dan Manual Mode yang terletak ditengah layar.

```python
# Function to display winner notification and return to menu
def winner_notification(winner, winning_pos):
    pygame.display.set_caption("Winner!")
    screen.fill((255, 255, 255))
    font = pygame.font.Font(None, 36)
    text_winner = font.render(f"{winner} wins!!!", True, BLACK)
    text_rect = text_winner.get_rect(center=(width // 2, SQUARESIZE // 2))
    screen.blit(text_winner, text_rect)

    # Draw winning positions
    draw_board(board, winning_pos)

    pygame.display.update()
    pygame.time.wait(3000)

    # Reset game variables
    return True
```

gambar 8. code display winner notification

Function `winner_notification` digunakan untuk menampilkan pesan kemenangan ketika pemain memenangkan permainan connect 4, pesan tersebut akan tampil di layar selama 3 detik sebelum game akan kembali ke tampilan awal

3.2 Perhitungan Skor

```python
# Function to evaluate the score for a particular move
def evaluate_window(window, piece):
    score = 0
    opp_piece = 1 if piece == 2 else 2
    if window.count(piece) == 4:
        score += 100
    elif window.count(piece) == 3 and window.count(0) == 1:
        score += 5
    elif window.count(piece) == 2 and window.count(0) == 2:
        score += 2
    if window.count(opp_piece) == 3 and window.count(0) == 1:
        score -= 4
    return score
```

gambar 9. code evaluasi potensi kemenangan

Function `winner_notification` digunakan untuk mengevaluasi nilai dari permainan connect 4 yang dilambangkan dengan window. Window yang dimaksud ialah sejumlah sel berturut-turut dalam baris, kolom, atau diagonal pada papan permainan

connect 4. Nilai ditentukan berdasarkan jumlah bidak yang sama berada pada papan connect 4, dengan nilai tertentu diberikan sesuai dengan pola berikut :

1. Jika ada 4 bidak pemain yang sama dalam papan, maka nilai += 100
2. Jika ada 3 bidak pemain yang sama dan satu sel kosong dalam papan, maka nilai += 5
3. Jika ada 2 bidak pemain yang sama dan dua sel kosong dalam papan, maka nilai += 2
4. Jika ada 3 bidak lawan yang salam dan satu sel kosong dalam papan, maka nilai -= 4

```python
# Function to calculate the score of a position
def score_position(board, piece):
    score = 0
    # Score center column
    center_array = [int(i) for i in list(board[:, COLUMN_COUNT // 2])]
    center_count = center_array.count(piece)
    score += center_count * 3
    # Score horizontal
    for r in range(ROW_COUNT):
        row_array = [int(i) for i in list(board[r, :])]
        for c in range(COLUMN_COUNT - 3):
            window = row_array[c:c + 4]
            score += evaluate_window(window, piece)
    # Score vertical
    for c in range(COLUMN_COUNT):
        col_array = [int(i) for i in list(board[:, c])]
        for r in range(ROW_COUNT - 3):
            window = col_array[r:r + 4]
            score += evaluate_window(window, piece)
    # Score positively sloped diagonal
    for r in range(ROW_COUNT - 3):
        for c in range(COLUMN_COUNT - 3):
            window = [board[r + i][c + i] for i in range(4)]
            score += evaluate_window(window, piece)
    # Score negatively sloped diagonal
    for r in range(ROW_COUNT - 3):
        for c in range(COLUMN_COUNT - 3):
            window = [board[r + 3 - i][c + i] for i in range(4)]
            score += evaluate_window(window, piece)
    return score
```

gambar 10. code perhitungan skor

Function `score_position` digunakan menilai posisi pada papan permainan connect 4 berdasarkan skor relatif terhadap pemain lain. Proses perhitungan ini dilakukan dengan menghitung skor pada berbagai arah, termasuk arah horizontal, vertikal, dan diagonal.

3.3 Implementasi Algoritma Minimax

```python
# Function to perform a minimax search
def minimax(board, depth, maximizingPlayer):
    valid_locations = get_valid_locations(board)
    if depth == 0 or winning_move(board, 1)[0] or winning_move(board, 2)[0] or is_board_full(board):
        if winning_move(board, 2)[0]:
            return None, 100000000000000
        elif winning_move(board, 1)[0]:
            return None, -10000000000000
        else:
            return None, 0
    if maximizingPlayer:
        value = -math.inf
        column = random.choice(valid_locations)
        for col in valid_locations:
            row = get_next_open_row(board, col)
            b_copy = board.copy()
            drop_piece(b_copy, row, col, 2)
            new_score = minimax(b_copy, depth - 1, False)[1]
            if new_score > value:
                value = new_score
                column = col
        return column, value
    else:
        value = math.inf
        column = random.choice(valid_locations)
        for col in valid_locations:
            row = get_next_open_row(board, col)
            b_copy = board.copy()
            drop_piece(b_copy, row, col, 1)
            new_score = minimax(b_copy, depth - 1, True)[1]
            if new_score < value:
                value = new_score
                column = col
        return column, value
```

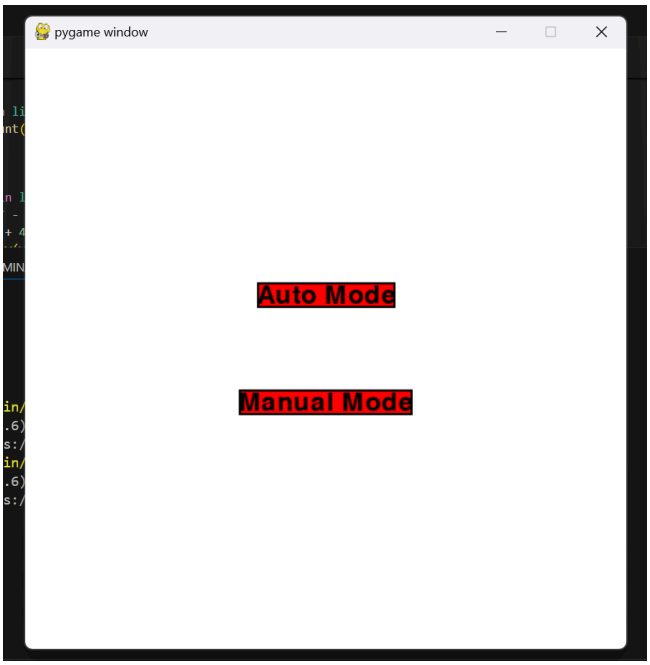
gambar 7. Penerapan algoritma minimax

Algoritma minimax digunakan untuk mencari langkah terbaik yang dapat diambil oleh pemain pada permainan yang memiliki konsep bergilir, seperti *board game* pada coding diatas terdapat function `minimax` yang memiliki tiga parameter input yaitu `board, depth, maximizingPlayer (merupakan sebuah boolean yang menunjukkan apakah pemain yang melakukan pencarian saat ini merupakan pemain yang sedang memaksimalkan skor atau meminimalkan skor)` selain itu terdapat kondisi terminasi dari rekursi. Jika kedalaman pencarian sudah mencapai batas yang ditentukan (`if depth == 0`) atau terdapat pemenang (`winning_move(board, 1)[0] or winning_move(board, 2)[0]`) atau papan permainan sudah dalam kondisi penuh atau tidak ada area kosong (`or is_board_full(board)`) maka function tersebut akan mengembalikan skor sesuai dengan kondisi tersebut. Jika pemain 2 menang maka skor positif akan dikembalikan, jika pemain 1 yang menang maka skor negatif yang akan

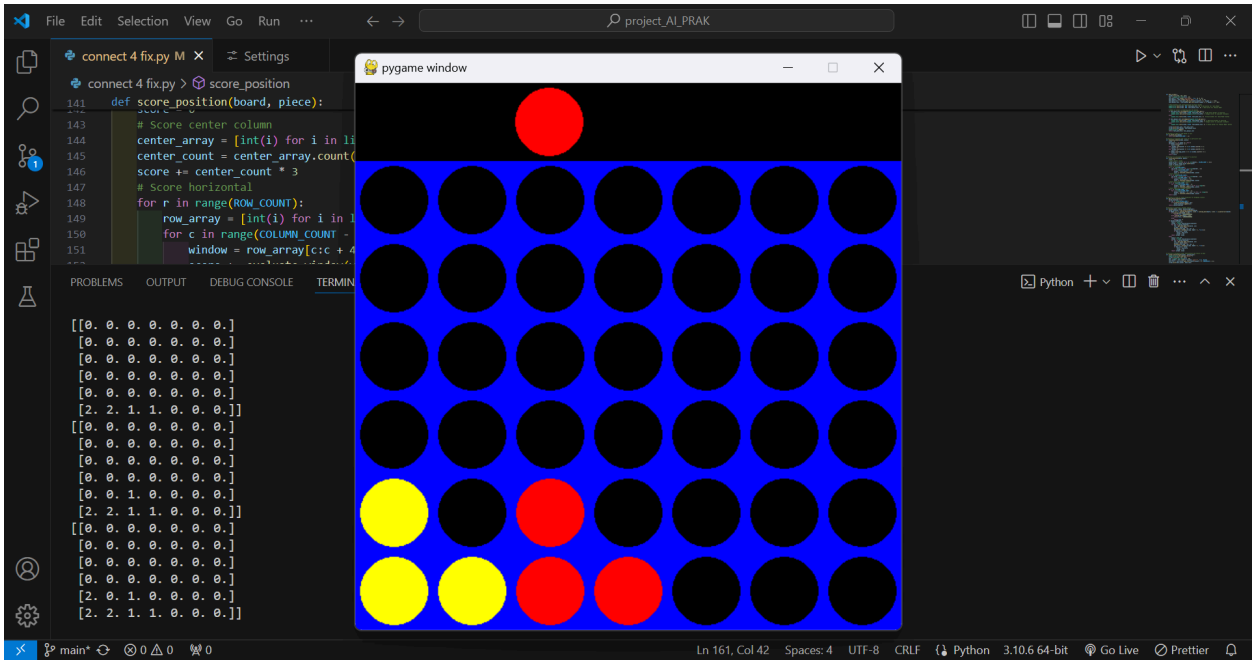
dikembalikan, jika tidak ada pemenang atau papan sudah penuh maka skor nol yang akan dikembalikan.

Jika kedalaman pencarian belum mencapai o dan permainan belum selesai, function `minimax` akan terus melakukan pencarian secara rekursif. Setelah mencoba semua langkah yang mungkin function `minimax` akan mengembalikan nilai kolom terbaik yang dipilih, serta nilai skor yang terkait dengan langkah tersebut.

BAB IV - DOKUMENTASI HASIL



gambar 8. tampilan awal permainan



gambar 9. tampilan board dan terminal

BAB V - KESIMPULAN

Algoritma minimax adalah salah satu algoritma yang digunakan untuk pengambilan keputusan dalam permainan bergilir seperti connect 4, tujuan dari algoritma ini yaitu menemukan langkah terbaik dengan memaksimalkan keuntungan pemain berdasarkan semua kemungkinan yang diuji dalam algoritma ini. Dalam permainan connect 4 meskipun memerlukan perhitungan intensif, terutama jika permainan ini dimainkan dalam papan ukuran besar namun, pendekatan ini memungkinkan pemain untuk mencari langkah terbaik yang dapat meningkatkan peluang kemenangan dan mengurangi kemungkinan kekalahan.

LINK GITHUB

https://github.com/salmaaida0504/AI-PRAKTIKUM_CONNECT-4.git

DAFTAR PUSTAKA

[1]
https://connect4.gamesolver.org/ diakses pada 21 Mei 2023
[2]
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Makalah/Makalah-Stima-2023-(118).pdf diakses pada 9 Mei 2024