Penerapan Algoritma *MiniMax* dengan Memanfaatkan *Greedy* dalam Aplikasi Strategi Game *Connect Four*

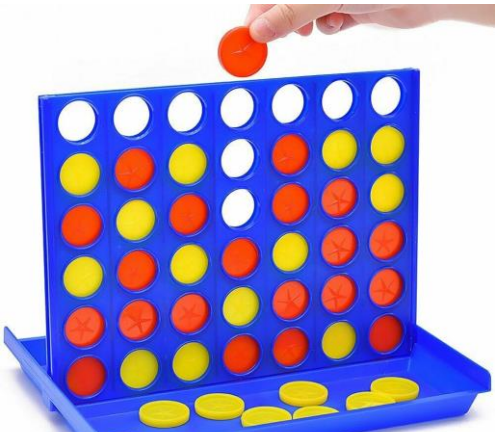
Vieri Fajar Firdaus - 13521099
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521099@std.stei.itb.ac.id

***Abstract*— Makalah ini membahas penerapan algoritma Minimax dengan memanfaatkan teknik Greedy pada permainan strategi Connect Four. Algoritma Minimax digunakan untuk memprediksi gerakan yang optimal pada setiap giliran dengan mempertimbangkan kemungkinan gerakan lawan dan kemungkinan gerakan selanjutnya. Pada permainan Connect Four, algoritma Minimax dapat dioptimalkan dengan teknik pruning dan heuristik untuk mempercepat proses evaluasi simpul-simpul pohon permainan yang besar. Selain itu, teknik Greedy juga digunakan untuk mempercepat proses evaluasi pada simpul-simpul pohon permainan yang memiliki skor heuristik yang buruk. Dalam makalah ini, penulis menjelaskan secara detail tentang implementasi algoritma Minimax dan teknik Greedy pada permainan Connect Four, serta hasil uji coba yang dilakukan untuk menguji keefektifan algoritma ini. Hasil uji coba menunjukkan bahwa penggunaan algoritma Minimax dengan memanfaatkan teknik Greedy dalam permainan Connect Four dapat menghasilkan solusi yang optimal dengan waktu yang lebih singkat. Makalah ini diharapkan dapat memberikan kontribusi dalam pengembangan aplikasi strategi game yang lebih efisien dan efektif.**

Keywords*— *Connect Four, algoritma MiniMax, Greedy

I. PENDAHULUAN

Permainan strategi merupakan salah satu jenis permainan yang memerlukan kecerdasan dan ketelitian dalam memilih gerakan yang optimal untuk mencapai kemenangan. Dalam permainan strategi, seringkali terdapat banyak kemungkinan gerakan yang dapat diambil pada setiap giliran, sehingga memerlukan algoritma khusus untuk memprediksi gerakan yang paling optimal.



Gambar 1 Connect Four
https://www.google.com/search?q=connect+four&source=lnms&tbm=isch&sa=X&ved=2ahUKEwju1Lz4mon_AhWX-DgGHeGHB40Q_AUoAXoECAEQAw&biw=1536&bih=746

Salah satu algoritma yang sering digunakan pada permainan strategi adalah algoritma Minimax. Algoritma ini bekerja dengan melakukan simulasi pada setiap kemungkinan gerakan yang dapat diambil pada setiap giliran, dan kemudian memilih gerakan yang memberikan keuntungan terbesar atau kerugian terkecil, tergantung pada peran pemain pada permainan tersebut.

Pada permainan Connect Four, algoritma Minimax dapat digunakan untuk memprediksi gerakan yang optimal pada setiap giliran dengan mempertimbangkan kemungkinan gerakan lawan dan kemungkinan gerakan selanjutnya. Namun, karena pohon permainan pada Connect Four dapat sangat besar, maka penggunaan algoritma Minimax harus dioptimalkan dengan teknik pruning dan heuristik untuk mempercepat proses evaluasi simpul-simpul pohon permainan.

Selain itu, teknik Greedy juga dapat digunakan untuk mempercepat proses evaluasi pada simpul-simpul pohon permainan yang memiliki skor heuristik yang buruk. Dengan memadukan teknik Greedy dengan algoritma Minimax, diharapkan dapat meningkatkan efisiensi dan efektivitas dalam memprediksi gerakan yang optimal pada setiap giliran dalam permainan Connect Four.

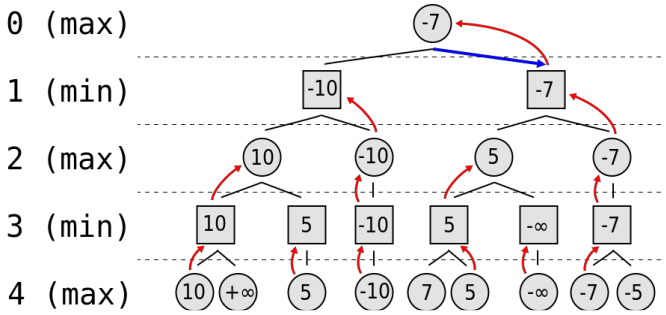
Makalah ini membahas tentang penerapan algoritma Minimax dengan memanfaatkan teknik Greedy pada permainan strategi Connect Four. Penulis akan menjelaskan secara detail tentang implementasi algoritma Minimax dan teknik Greedy pada permainan Connect Four, serta hasil uji coba yang dilakukan untuk menguji keefektifan algoritma ini.

Dalam makalah ini, penulis juga akan membahas tentang heuristik yang dapat digunakan dalam permainan Connect Four untuk mengevaluasi posisi permainan secara cepat dan akurat. Makalah ini diharapkan dapat memberikan kontribusi dalam pengembangan aplikasi strategi game yang lebih efisien dan efektif.

II. LANDASAN TEORI

A. *Algoritma MinMax*

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 contoh algoritma MinMax
https://www.google.com/search?q=algoritma+minimax&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiD4rDY64j_AhUKzqACHSz3AF0Q_AUoAXoECAIQAw&

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 diambilsebagai 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.

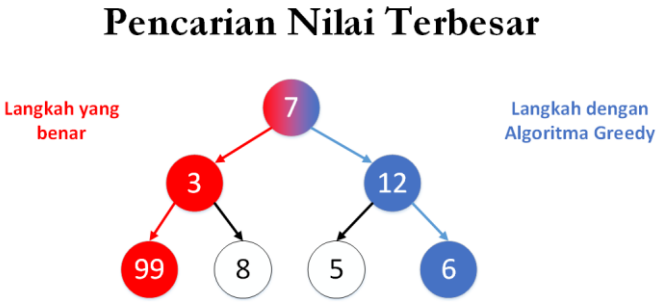
Namun, karena pohon permainan pada permainan strategi bisa sangat besar, penggunaan algoritma Minimax harus dioptimalkan dengan teknik pruning dan heuristik. Pruning dapat membantu memotong cabang-cabang dari pohon permainan yang tidak perlu dievaluasi karena tidak memberikan kontribusi signifikan pada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

B. *Algoritma Greedy*

Algoritma Greedy adalah algoritma yang digunakan dalam pemrograman untuk memecahkan masalah optimasi dengan cara memilih opsi yang terbaik pada setiap langkah. Algoritma ini bekerja dengan cara memilih opsi yang paling menguntungkan pada setiap langkah, tanpa mempertimbangkan konsekuensi jangka panjang atau alternatif yang lebih baik di masa depan.

Pada dasarnya, algoritma Greedy berusaha untuk mencapai solusi terbaik dengan mengambil keputusan yang terbaik pada setiap langkah, tanpa perlu mempertimbangkan semua kemungkinan solusi. Oleh karena itu, algoritma Greedy sering digunakan untuk memecahkan masalah yang kompleks dan memiliki banyak kemungkinan solusi, dengan cara mencari solusi yang cukup baik dalam waktu yang relatif singkat.



Gambar 2 contoh Penerapan Greedy
https://www.google.com/search?q=algoritma+greedy&tbm=isch&ved=2ahUKEwjBmazZ64j_AhVfg2MGHV9pDj4Q2-

Salah satu contoh penerapan algoritma Greedy adalah pada masalah Knapsack, yaitu masalah memilih sejumlah item dari kumpulan item yang tersedia dengan kapasitas tertentu, sedemikian rupa sehingga nilai item yang dipilih maksimum dan kapasitas yang digunakan tidak melebihi kapasitas yang tersedia. Pada masalah ini, algoritma Greedy akan memilih item dengan nilai tertinggi pada setiap langkah, tanpa mempertimbangkan kapasitas yang tersisa atau nilai item di masa depan. Algoritma ini dapat memberikan solusi yang cukup baik dalam waktu yang relatif singkat, namun tidak selalu dapat menghasilkan solusi yang optimal.

Pada aplikasi strategi game, algoritma Greedy dapat digunakan untuk memprediksi gerakan yang optimal pada setiap giliran, dengan memilih gerakan yang memberikan keuntungan terbesar pada saat itu. Namun, seperti pada masalah Knapsack, algoritma Greedy tidak selalu menghasilkan solusi yang optimal dalam permainan strategi yang kompleks dan memiliki banyak kemungkinan gerakan.

Oleh karena itu, algoritma Greedy sering digunakan bersama dengan algoritma lain, seperti algoritma Minimax, untuk meningkatkan efisiensi dan efektivitas dalam memprediksi gerakan yang optimal pada setiap giliran dalam permainan strategi yang kompleks. Dengan memadukan algoritma Greedy dengan algoritma Minimax, diharapkan dapat meningkatkan kecepatan dalam memprediksi gerakan yang optimal tanpa mengorbankan kualitas solusi secara signifikan.

Meskipun demikian, algoritma Greedy tetap memiliki kegunaannya sendiri dalam memecahkan masalah optimasi sederhana yang tidak terlalu kompleks dan memiliki struktur yang jelas. Algoritma ini juga dapat digunakan sebagai strategi heuristik untuk memecahkan masalah yang tidak memiliki solusi yang optimal atau dalam situasi di mana waktu yang tersedia sangat terbatas

C. *Connect Four*

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.

III. IMPLEMENTASI

Dari landasan teori yang sudah dijelaskan di atas, kita dapat membuat sebuah strategi berdasarkan algoritma minimax dengan memanfaatkan greedy untuk permainan connect four. Secara garis besar, strategi yang akan kita buat adalah strategi yang mengambil gerakan terbaik pada tiap giliran, dengan gerakan yang terbaik dinilai dari apakah gerakan ini membuat menang ataupun apabila tidak diambil akan membuat kalah. Apabila tidak ada gerakan tertentu yang akan membuat kalah atau menang, maka diambil gerakan yang akan membuat baris paling panjang.

A. *Penggunaan Pendekatan Greedy pada Algoritma Minimax*

Akan ditentukan pilihan dari pohon keputusan yang digunakan dalam pembuatan algoritma minimax :

a. Memilih kolom tengah score bernilai tiga

b. Memilih kolom sehingga terbentuk dua titik dalam baris score bernilai dua

c. Memilih kolom sehingga terbentuk tiga titik dalam baris score bernilai lima

d. Memilih kolom yang dapat memenangkan permainan akan mendapatkan poin tak hingga namun disini akan diberikan poin 100

   Memenangkan permainan dapat memberikan poin tak hingga

e. Musuh memillih kolom sehingga terbentuk dua titik dalam baris akan mengurangi score sebanyak dua poin

f. Musuh memilih kolom sehingga memenangkan permainan akan mengurangi score sebanyak tak hingga poin

```
def evaluateWindow(window, piece):
    score = 0
    opp_piece = PLAYER_PIECE
    if piece == PLAYER_PIECE:
        opp_piece = AI_PIECE

    if window.count(piece) == 4:
        score += 100
    elif window.count(piece) == 3 and
 window.count(EMPTY) == 1:
        score += 5
    elif window.count(piece) == 2 and
 window.count(EMPTY) == 2:
        score += 2

    if window.count(opp_piece) == 3
and window.count(EMPTY) == 1:
        score -= 4

    return score
```

Gambar 3 kode evaluateWindow

Fungsi diatas adalah untuk menentukan score dari tiap posisi dari tiap titik pada window, window dapat berupa baris

kolom maupun diagonal, dan dibawah ini adalah fungsi untuk mendapatkan total score.

```python
def scorePosition(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 * 4

    ## 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+WINDOW_LENGTH]
            score += evaluateWindow(window, piece)
            # print(evaluateWindow(window, piece),r,c)

    ## 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+WINDOW_LENGTH]
            score += evaluateWindow(window, piece)
            # print(evaluateWindow(window, piece),r,c)

    ## Score posiive 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(WINDOW_LENGTH)]
            score += evaluateWindow(window, piece)
            # print(evaluateWindow(window, piece),r,c)

    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(WINDOW_LENGTH)]
            score += evaluateWindow(window, piece)
            # print(evaluateWindow(window, piece),r,c)

    return score
```

Gambar 4 kode scorePosition

B. *Algoritma Minimax*

Setelah didapatkan total skor dapat ditentukan suatu value papan melalui algoritma minimax

```python
def minimax(board, depth, alpha, beta, maximizingPlayer):
    valid_locations = get_valid_locations(board)
    is_terminal = isTerminal(board)
    if depth == 0 or is_terminal:
        if is_terminal:
            if bestMove(board, AI_PIECE):
                return (None, 100000000000000)
            elif bestMove(board, PLAYER_PIECE):
                return (None, -100000000000000)
            else: # Game is over, no more valid moves
                return (None, 0)
        else: # Depth is zero
            return (None, scorePosition(board, AI_PIECE))
    if maximizingPlayer:
        value = -math.inf
        column = random.choice(valid_locations)
        for col in valid_locations:
            row = getNext(board, col)
            b_copy = board.copy()
            getBoard(b_copy, row, col, AI_PIECE)
            new_score = minimax(b_copy, depth-1, alpha, beta, False)[1]
            print(new_score,end=" ")
            if new_score > value:
                value = new_score
                column = col
            alpha = max(alpha, value)
            if alpha >= beta:
                break
        print()
        return column, value

    else: # Minimizing player
        value = math.inf
        column = random.choice(valid_locations)
        for col in valid_locations:
            row = getNext(board, col)
            b_copy = board.copy()
            getBoard(b_copy, row, col, PLAYER_PIECE)
            new_score = minimax(b_copy, depth-1, alpha, beta, True)[1]
            if new_score < value:
                value = new_score
                column = col
            beta = min(beta, value)
            if alpha >= beta:
                break
        return column, value
```

Gambar 5 kode algoritma minMax

Sebelum dilakukan pencarian kolom terbaik menurut algoritma minimax, akan dicek terlebih dahulu bahwa lokasi kolom yang akan dipilih valid atau tidak, lalu akan dicek juga is_terminal, is_terminal berupa fungsi untuk mengecek player/AI apakah sudah menang atau imbang. Jika belum terdapat pemenang, atau papan belum terisi semua, maka akan dilakukan pemeriksaan secara rekursif menggunakaan fungsi miniMax.

Pertama, fungsi ini memeriksa apakah kedalaman pencarian sudah mencapai 0 atau permainan telah berakhir (is_terminal). Jika ya, maka dilakukan evaluasi skor posisi pada papan berdasarkan pemain AI (AI_PIECE). Jika AI menang, skor yang sangat tinggi (100000000000000) dikembalikan. Jika pemain lawan (PLAYER_PIECE) menang, skor yang sangat rendah (-100000000000000) dikembalikan. Jika tidak ada lagi langkah yang valid atau permainan berakhir dengan skor imbang, skor 0 dikembalikan.

Jika belum mencapai kedalaman 0 dan permainan belum berakhir, fungsi akan memilih langkah terbaik berdasarkan algoritma Minimax. Jika pemain yang sedang dipertimbangkan adalah pemain yang memaksimalkan, maka fungsi akan mencoba semua langkah yang valid dan memanggil rekursif fungsi minimax pada setiap langkah tersebut. Fungsi akan memperbarui nilai maksimum (value) dan kolom terbaik (column) berdasarkan skor yang dihasilkan dari pemanggilan rekursif tersebut. Pruning Alpha-Beta digunakan dengan memperbarui nilai alpha menjadi maksimum antara alpha dan value setiap kali nilai value diperbarui, dan memeriksa apakah alpha sudah melebihi atau sama dengan beta. Jika iya, iterasi dihentikan karena langkah lain tidak akan menghasilkan solusi yang lebih baik. Pada

akhirnya, kolom terbaik dan nilai maksimum tersebut dikembalikan.

Jika pemain yang sedang dipertimbangkan adalah pemain yang meminimalkan, proses serupa dilakukan, namun nilai minimum (value) dan kolom terbaik (column) diperbarui berdasarkan skor yang lebih rendah. Pruning Alpha-Beta juga digunakan dengan memperbarui nilai beta menjadi minimum antara beta dan value, serta memeriksa apakah alpha sudah melebihi atau sama dengan beta.

Fungsi ini akan optimal jika kedalaman semakin besar, karena program akan menjelajahi sebanyak kedalaman tersebut. Namun semakin besar kedalaman pastinya semakin besar pula kompleksitasnya.

C. *Implementasi Kode Utama*

Setiap algoritma yang dibutuhkan sudah diterapkan dan dapat langsung diaplikasikan. Berikut adalah kode utama untuk bot *connect four*

```python
from miniMax import *
from turn import *
board = createBoard()
game_over = False
turn = random.randint(PLAYER1,PLAYER2)

print("Keterangan :")
print("1.0 PLAYER1\n2.0 PLAYER2\n")

while (True):

    if turn == PLAYER1:
        inp = ""
        while(True):
            inp = input("Masukkan kolom: ")
            if(inp=="AI" or inp=="ai"):
                break
            if(inp.isdigit()):
                if(int(inp)<0 or int(inp)>6):
                    continue
                else:
                    break


        if(inp=="AI" or inp=="ai"):
            col, minimax_score = minimax(board, 5, -math.inf, math.inf,
True)
        else :
            col=int(inp)
        print("Masukan kolom PLAYER1 : ", col)

        if isValidLoc(board, col):
            row = getNext(board, col)
            getBoard(board, row, col, PLAYER1_PIECE)

            if bestMove(board, PLAYER1_PIECE):
                game_over = True
                print("PLAYER1 menang")
                printBoard(board)
                break

            turn += 1
            turn = turn % 2
            print("Giliran PLAYER1")
            printBoard(board)
```

```python
    else:
        inp = ""
        while(True):
            inp = input("Masukkan kolom: ")
            if(inp=="AI" or inp=="ai"):
                break
            if(inp.isdigit()):
                if(int(inp)<0 or int(inp)>6):
                    continue
                else:
                    break


        if(inp=="AI" or inp=="ai"):
            col, minimax_score = minimax(board, 1, -math.inf, math.inf,
True)
        else :
            col=int(inp)
        print("Masukan kolom PLAYER2 : ", col)
        if isValidLoc(board, col):
            row = getNext(board, col)
            getBoard(board, row, col, PLAYER2_PIECE)

            if bestMove(board, PLAYER2_PIECE):
                game_over = True
                print("PLAYER2 menang")
                printBoard(board)
                break
            print("giliran PLAYER2")
            printBoard(board)

            turn += 1
            turn = turn % 2
    if(isFull(board)):
        print("Draw")
        break
```

Gambar 6 kode program utama

Pada program utama dipilih kedalaman lima karena kompleksitasnya masih terjangkau dan kemungkinan untuk menang masih besar. Kedalaman lima berarti AI menebak jalan terbaik setelah melakukan lima kali giliran / penelusuran. Setelah lima kali penelusuran akan didapatkan kolom terbaik.

D. *Pengujian*

```
Masukkan kolom: 3
Masukan kolom PLAYER2 :  3
giliran PLAYER2
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 2 0 0 0
0 0 0 2 1 0 0
0 0 0 2 1 0 0

Masukkan kolom: ai
Masukan kolom PLAYER1 :  3
Giliran PLAYER1
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 0 0 2 0 0 0
0 0 0 2 1 0 0
0 0 0 2 1 0 0
```

Gambar 7 Pengujian menghindar dari kekalahan

Pada pengujian diatas player 2 menggunakan AI dan telah dipilih langkah terbaik agar tidak kalah adalah meletakkan pada kolom empat, dengan menggunakan fungsi minimax dipilih kolom empat karena apabila AI memilih kolom selain empat maka AI akan kalah.

```
Masukkan kolom: 0
Masukan kolom PLAYER2 :  0
giliran PLAYER2
0 0 0 0 0 0 0
0 0 0 2 0 0 0
0 0 0 1 1 0 0
0 0 1 2 2 0 0
0 1 2 2 1 0 0
2 1 2 2 1 0 0

Masukkan kolom: ai
Masukan kolom PLAYER1 :  4
PLAYER1 menang
0 0 0 0 0 0 0
0 0 0 2 1 0 0
0 0 0 1 1 0 0
0 0 1 2 2 0 0
0 1 2 2 1 0 0
2 1 2 2 1 0 0
```

Gambar 8 Pengujian menuju kemenangan

Pada pengujian diatas player 2 memiliih kolom satu dan player 1 memilih menggunakan AI, player 1 dapat memenangkan jika memilih kolom 5, dan ternyata dari kombinasi kolom terbaik dengan algoritma minimax diperoleh kolom 5 sebagai kolom terbaik

IV. Kesimpulan dan Saran

Dalam makalah ini, telah dibahas penerapan algoritma Minimax dengan memanfaatkan strategi greedy dalam aplikasi strategi game Connect Four. Connect Four adalah permainan strategi yang melibatkan dua pemain yang bergantian menempatkan cakram mereka pada papan permainan dengan tujuan membentuk garis horizontal, vertikal, atau diagonal yang terdiri dari empat cakram.

Dalam makalah ini, algoritma Minimax digunakan untuk mencari langkah terbaik yang dapat diambil oleh pemain AI dalam permainan Connect Four. Algoritma Minimax bekerja dengan mempertimbangkan semua kemungkinan langkah yang mungkin terjadi dari posisi saat ini, baik oleh pemain AI maupun oleh lawan, dan memilih langkah yang mengoptimalkan skor akhir.

Bot akan memiliki persentase kemenangan lebih besar jika kedalaman dari algoritma Minimax semakin besar, hal ini dikarenakan semakin besar kedalaman semakin besar pula penjelajahan setiap kemungkinan yang terjadi, namun semakin besar pula kompleksitas dari algoritma ini.

Secara keseluruhan, penggunaan algoritma Minimax dengan memanfaatkan strategi greedy dalam aplikasi strategi game Connect Four memberikan pemahaman yang lebih baik tentang bagaimana algoritma tersebut dapat diterapkan dalam konteks permainan.

V. Ucapan Terima Kasih

Penulis mengucapkan terima kasih serta puji dan syukur kepada Tuhan Yang Maha Esa karena berkat rahmatnya penulis dapat menyelesaikan makalah ini dengan tepat waktu. Penulis juga mengucapkan terima kasih kepada seluruh tim dosen mata kuliah IF2211 Strategi Algoritma yang telah dengan sabar membimbing penulis selama satu semester. Tanpa bimbingan dan ajaran dari tim dosen IF2211 makalah ini tidak mungkin dapat terselesaikan dengan baik.

Penulis juga berterima kasih kepada seluruh tim asisten mata kuliah IF2211 yang telah dengan sabar memberi ilmu serta membimbing selama mata kuliah IF2211 semester ini baik secara langsung saat demo maupun tidak langsung. Penulis juga berterima kasih kepada seluruh teman penulis di Teknik Informatika ITB angkatan 2021 yang telah membantu penulis menemukan judul makalah serta memberi masukkan dan saran pada proses pengerjaan makalah ini sehingga makalah ini dapat menjadi lebih baik lagi.

Terakhir, penulis juga berterima kasih kepada orang tua penulis yang berkat doa dan dukungannya memungkinkan penulis untuk menyelesaikan mata kuliah IF2211 Strategi Algoritma dan menyelesaikan makalah ini dengan tepat waktu.

Video Link at Youtube

https://youtu.be/bvOQHM9reh8

References

[1] https://medium.com/analytics-vidhya/artificial-intelligence-at-play-connect-four-minimax-algorithm-explained-3b5fc32e4a4f diakses pada 22 Mei 2023

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf diakses pada 20 Mei 2023

[3] https://connect4.gamesolver.org/ diakses pada 21 Mei 2023

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, 22 Mei 2023

Vieri Fajar Firdaus 13521099