LAPORAN FINAL PROJECT

Konsep Kecerdasan Artifisial

"Permainan *Tic Tac Toe* dengan *Adversarial Search*"

Kelompok "Bingung"

1. Helsa Sriprameswari Putri (5025221154)
2. Yasmin Putri Sujono (5025221273)
3. Nadya Saraswati Putri (5025221246)

DEPARTEMEN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS

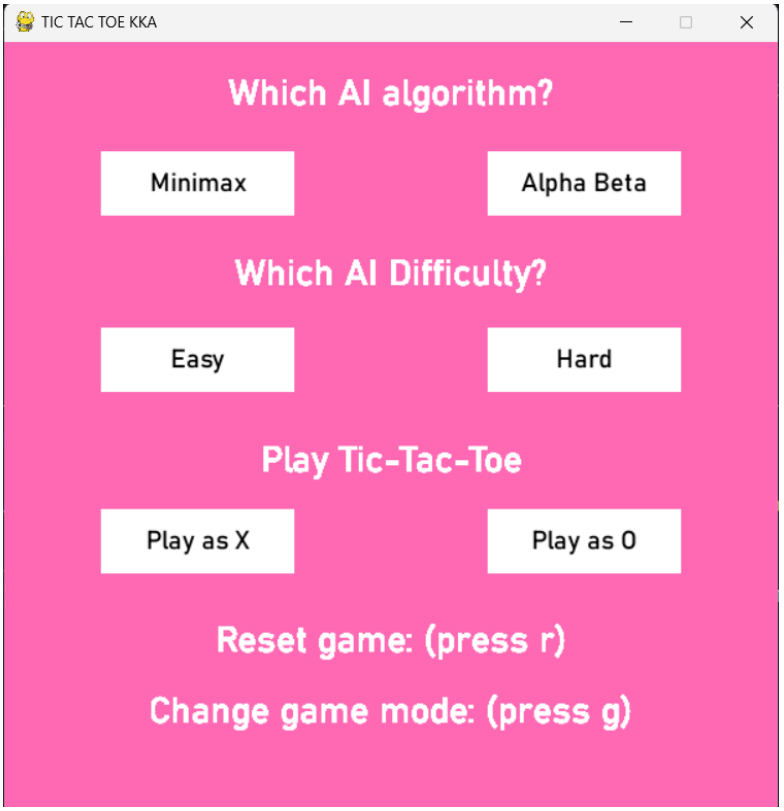
INSTITUT TEKNOLOGI SEPULUH NOPEMBER

2023

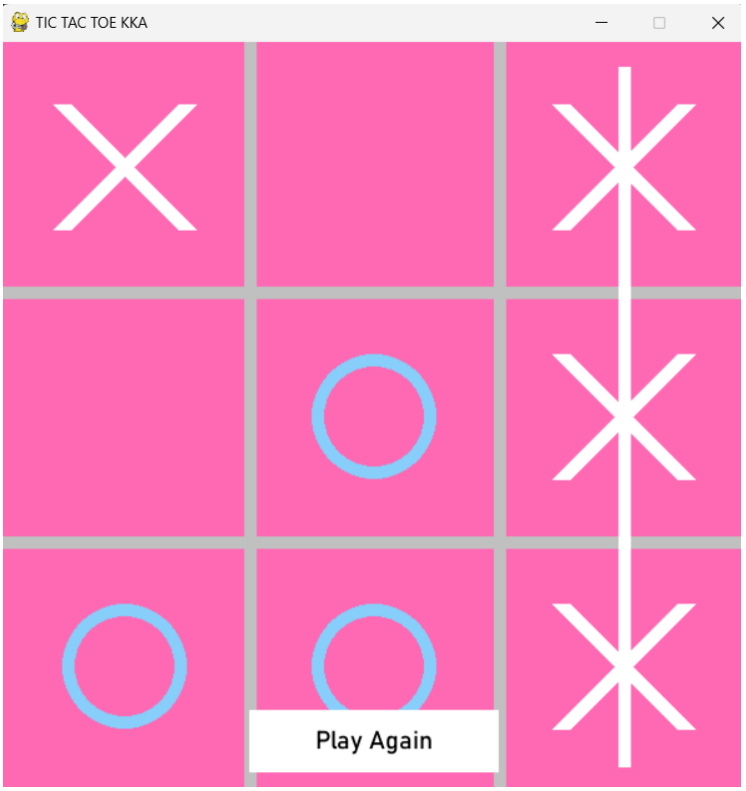
1. Deskripsi Permasalahan

Pada *final project* ini, kami membuat program game sederhana yang terinspirasi dari permainan *Tic Tac Toe*. *Tic Tac Toe* merupakan permainan papan yang berisi kotak-kotak berukuran 3x3. Dalam permainan terdapat 2 simbol yang dapat dimainkan oleh pemain, yaitu simbol **X** dan simbol **O**. Simbol **X** dan **O** ini digunakan untuk menunjukkan langkah permainan pada permainan *Tic Tac Toe* ini. Pada program ini, permainan hanya memiliki satu mode, yaitu mode melawan *AI* atau komputer dengan pilihan kesulitan, *easy* dan *hard*. Terdapat pilihan algoritma yang ingin digunakan, yaitu algoritma "*Minimax*" dan "*Alpha Beta Pruning*". Permainan dimulai dengan tampilan awal di mana pemain dapat memilih mode AI, tingkat kesulitan, dan simbol yang akan mereka mainkan simbol **X** atau simbol **O**. Dalam game ini, terdapat pula opsi untuk mengatur ulang (*reset*) dan mengganti mode game.

Dalam game *Tic Tac Toe,* permainan dimulai dengan player 1 memilih kotak yang masih kosong dan ditandai dengan simbol **X** atau **O** sesuai pilihan *player*. Selanjutnya, *AI* akan bergantian memilih kotak yang masih kosong dan menandai dengan simbolnya. Dalam game ini, terdapat tiga kondisi. Kondisi pertama, yaitu kondisi menang, dimana salah satu *player* dapat memposisikan simbol **X** atau **O** berturut-turut secara *vertikal*, *horizontal*, dan diagonal. Pada kondisi ini, ketiga simbol berturut - turut akan ditandai dengan garis lurus. Kondisi kedua, yaitu kondisi kalah, dimana lawan *player* sudah memenangkan game. Kondisi ketiga, yaitu *draw*, dimana tidak ada *player* yang menang maupun kalah.



Gambar 1.1 Tampilan Game *Tic Tac Toe*.



Gambar 1.2 Game *Tic Tac Toe* saat dimulai.

2. Metode Permasalahan

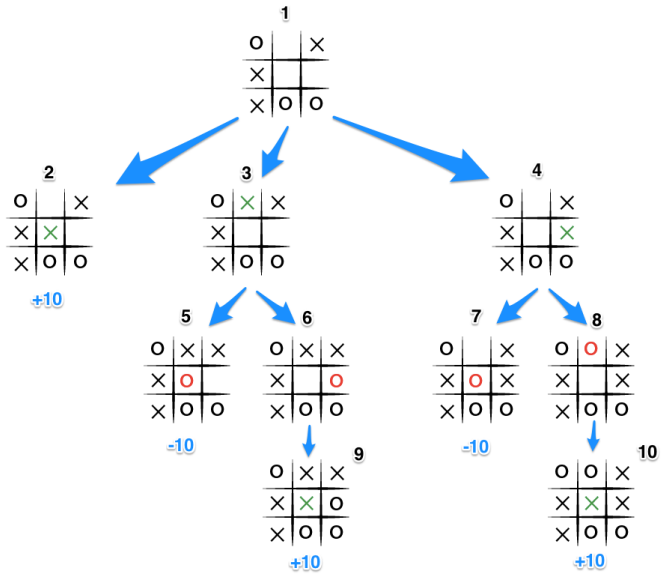
Game *Tic Tac Toe* pada proyek ini menggunakan metode *Adversarial Search*. *Adversarial Search* adalah istilah yang digunakan dalam teori permainan dan kecerdasan buatan untuk merujuk pada pendekatan pencarian yang digunakan untuk mengatasi situasi di mana ada dua pihak atau lebih yang bersaing dalam permainan atau keputusan. Dalam metode ini, terdapat dua peran penting, yaitu :

1. Pemain Utama (*Maximizer*)
   Pemain yang berusaha untuk memaksimalkan hasil atau skor yang menguntungkan mereka. Dalam banyak kasus, pemain ini mencoba untuk mencapai kemenangan atau hasil yang paling menguntungkan baginya.
2. Lawan (*Minimizer*)
   Pemain yang berusaha untuk meminimalkan hasil atau skor yang menguntungkan pemain utama. Lawan berusaha mencegah pemain utama mencapai kemenangan atau hasil yang menguntungkan pemain utama.

Pada metode ini, digunakan dua algoritma, yaitu Algoritma Minimax dan Algoritma Alpha Beta.

1. Algoritma Minimax
   Algoritma Minimax adalah algoritma yang digunakan dalam permainan seperti *Tic-Tac-Toe* untuk menghitung langkah terbaik yang harus diambil oleh pemain. Langkah - langkah algoritma minimax diantaranya adalah :

1. Inisialisasi pohon permainan : Mulai dengan membuat pohon permainan yang mewakili semua kemungkinan langkah di *Tic-Tac-Toe*, di mana setiap simpul mewakili keadaan permainan saat itu.
2. Evaluasi simpul daun : Evaluasi setiap simpul daun (akhir permainan) dengan memberikan skor berdasarkan hasil permainan. Misalnya, skor positif untuk kemenangan, skor negatif untuk kekalahan, dan nol untuk seri.
3. Penilaian pohon : Lakukan penilaian skor untuk setiap simpul dalam pohon permainan dengan menggunakan algoritma Minimax. Pemain 'X' berusaha untuk memaksimalkan skor, sedangkan pemain **'O'** berusaha untuk meminimalkannya.
4. Penyaringan skor : Dalam langkah rekursif, setiap tingkat alternatif dari pohon akan memaksimalkan atau meminimalkan skor di bawahnya, bergantung pada giliran pemain.
5. Pemilihan langkah terbaik : Setelah pohon dievaluasi, pemain akan memilih langkah terbaik dengan memilih langkah yang menghasilkan skor terbaik pada tingkat pohon awal.
6. Melakukan langkah terbaik : Pemain akan mengambil langkah terbaik yang telah dihitung oleh algoritma Minimax.



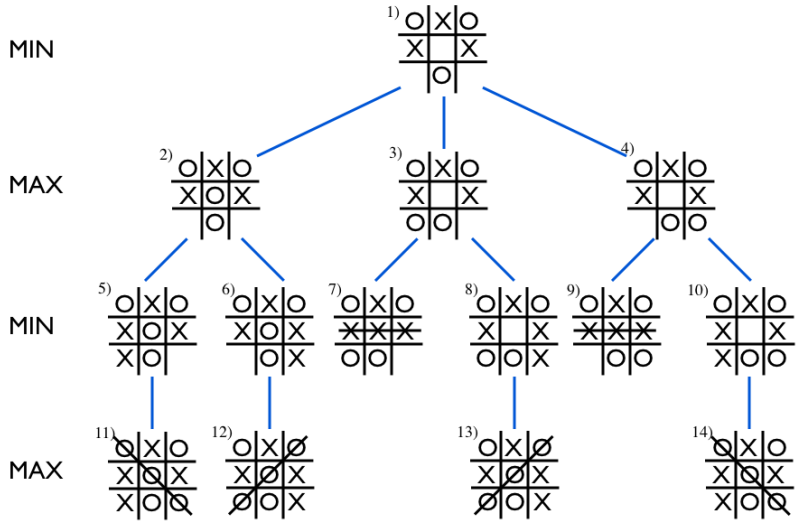
Gambar 2.1 *Tree* algoritma Minimax pada *Tic Tac Toe*.

2. Algoritma Alpha - Beta

   Algoritma Alpha-Beta adalah algoritma pencarian berkas yang digunakan dalam permainan dengan tujuan mengurangi jumlah simpul yang dieksplorasi, sehingga meningkatkan efisiensi pencarian. Tujuannya adalah untuk menemukan langkah terbaik untuk pemain dalam permainan giliran, yang mencoba memaksimalkan atau meminimalkan skor permainan. Berikut adalah langkah - langkah Algoritma Alpha - Beta :

1. Inisialisasi pohon permainan : Membuat pohon permainan yang mewakili semua kemungkinan langkah di *Tic-Tac-Toe,* di mana setiap simpul mewakili keadaan permainan saat itu.
2. Evaluasi simpul daun : Evaluasi setiap simpul daun (akhir permainan) dengan memberikan skor berdasarkan hasil permainan. Misalnya, skor positif untuk kemenangan, skor negatif untuk kekalahan, dan nol untuk seri.
3. Penilaian pohon dengan Alpha-Beta Pruning :
    a. Lakukan penilaian skor untuk setiap simpul dalam pohon permainan dengan menggunakan algoritma Alpha-Beta. Mulai dengan pemain **'X'** sebagai pemain maksimal dan pemain **'O'** sebagai pemain minimal.
    b. Selama pencarian, perbarui nilai alpha dan beta pada setiap simpul sesuai dengan nilai yang ditemukan. Jika pemain saat ini adalah pemain maksimal, perbarui alpha dengan nilai maksimum dari alpha dan nilai yang ditemukan. Jika pemain saat ini adalah pemain minimal, perbarui beta dengan nilai minimum dari beta dan nilai yang ditemukan.
    c. Selama pencarian, lakukan pemotongan alpha-beta. Jika pada suatu titik alpha menjadi lebih besar atau sama dengan beta, maka dapat menghentikan pencarian lebih lanjut dalam cabang pencarian ini karena hasil yang lebih baik tidak mungkin ditemukan.
4. Pemilihan langkah terbaik: Setelah pohon dievaluasi dengan Alpha-Beta, pemain **'X'** akan memilih langkah terbaik dengan memilih langkah yang menghasilkan skor terbaik pada tingkat pohon awal (akar).

5. Melakukan langkah terbaik: Pemain **'X'** akan mengambil langkah terbaik yang telah dihitung oleh algoritma Alpha-Beta, dan permainan akan melanjutkan dengan langkah tersebut.



Gambar 2.2 Tree algoritma Alpha - Beta pada *Tic Tac Toe.*

3. Analisis Hasil

A. Algoritma Minimax

1. Basis Kasus

   Algoritma memeriksa tiga basis kasus untuk mengakhiri rekursi:

   a. Jika pemain **X** menang (skor 1), maka dikembalikan skor 1.
   b. Jika pemain **O** menang (skor -1), maka dikembalikan skor -1.
   c. Jika permainan berakhir dengan seri (skor 0) karena semua sel sudah terisi, maka dikembalikan skor 0.

2. Rekursi

   a. Algoritma memasuki mode maksimisasi atau minimisasi tergantung pada giliran pemain yang sedang dievaluasi.
   b. Mode maksimisasi (**X**) bertujuan untuk memaksimalkan skor, sementara mode minimisasi (**O**) bertujuan untuk meminimalkan skor.
   c. Algoritma mencari setiap sel kosong yang tersedia dan mencoba setiap kemungkinan langkah yang mungkin.
   d. Untuk setiap kemungkinan langkah, algoritma membuat salinan papan permainan dan melakukan langkah (**X** atau **O**) pada sel yang sedang dievaluasi.
   e. Kemudian, algoritma melakukan rekursi pada papan permainan yang telah dimodifikasi.
   f. Setelah pemanggilan rekursif, algoritma memperoleh skor dari langkah tersebut (skor dari pemanggilan rekursif).

3. Perbandingan Skor

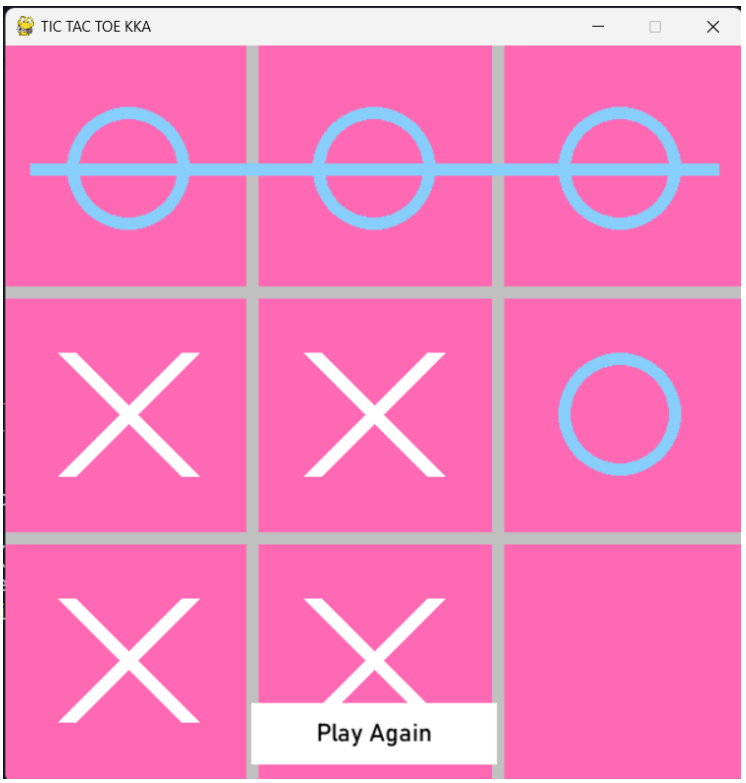
   a. Dalam mode maksimisasi, algoritma membandingkan skor saat ini dengan skor maksimum yang telah ditemukan. Jika skor saat

ini lebih tinggi dari skor maksimum sejauh ini, skor maksimum diperbarui, dan langkah terbaik diperbarui.
    b. Dalam mode minimisasi, algoritma membandingkan skor saat ini dengan skor minimum yang telah ditemukan. Jika skor saat ini lebih rendah dari skor minimum sejauh ini, skor minimum diperbarui, dan langkah terbaik diperbarui.
4. Kembalikan Hasil
    a. Setelah iterasi selesai, algoritma mengembalikan skor maksimum (dalam mode maksimisasi) atau skor minimum (dalam mode minimisasi) yang ditemukan selama pencarian dan langkah terbaik yang sesuai.
5. Jenis Kesulitan
    a. Algoritma memiliki opsi untuk mengatur kesulitan permainan ("*easy*" atau tidak).
    b. Dalam mode "*easy*", algoritma membatasi pencarian hanya pada lima sel kosong pertama. Ini dapat digunakan untuk membuat permainan kurang sulit.

6. Kompleksitas Waktu
   Dalam algoritma Minimax, setiap simpul dalam pohon pencarian harus dieksplorasi sepenuhnya, yang berarti bahwa kompleksitas waktu akan tumbuh eksponensial dengan kedalaman pohon pencarian. Untuk Tic Tac Toe, di mana kedalaman biasanya tidak terlalu dalam, ini mungkin masih dapat diatasi dengan baik.

7. Kelebihan:
    a. Implementasi algoritma yang relatif sederhana dan mudah dipahami.
    b. Secara konseptual, dapat memberikan solusi optimal dalam permainan papan catur.
8. Kekurangan:
    a. Tidak melakukan pemangkasan (*pruning*), sehingga mungkin menghasilkan banyak perhitungan yang tidak perlu.
    b. Pencarian penuh dari pohon permainan dapat memakan waktu, terutama dalam permainan yang lebih kompleks atau dengan kedalaman yang besar.

Berikut merupakan hasil permainan *Tic Tac Toe* dengan algoritma minimax dalam mode "*Easy*" dimana *AI* memainkan dengan simbol **O**. Dalam kondisi ini, *AI* memenangkan game dengan nilai eval yaitu -1.



Gambar 3.1 Hasil dari kondisi *AI* (simbol **O**) memenangkan game.

```
PS C:\Users\Acer\Documents\Semester 3\KKA\Final Project\Tic-Tac-Toe> python -u "c:\Users\Acer\Documents\Semeste
r 3\KKA\Final Project\Tic-Tac-Toe\main.py"
pygame 2.5.2 (SDL 2.28.3, Python 3.12.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
Minimax algorithm Chosen
AI Plays Easy
AI has chosen to mark the square in pos (0, 0) with an eval of: 0
AI has chosen to mark the square in pos (1, 2) with an eval of: 0
AI has chosen to mark the square in pos (0, 1) with an eval of: 0
AI has chosen to mark the square in pos (0, 2) with an eval of: -1
```

Gambar 3.2 Hasil tampilan program menunjukkan posisi dan nilai eval *AI*.

B. Algoritma Alpha - Beta

1. Basis Kasus

   Pada awal fungsi, terdapat pengecekan apakah permainan telah berakhir. Ini dilakukan dengan memeriksa apakah ada pemenang (**X** atau **O**) atau apakah permainan berakhir dengan seri (*draw*). Jika ada pemenang, fungsi mengembalikan 1 (kemenangan **X**), -1 (kemenangan **O**), atau 0 (seri), sesuai dengan aturan *Tic-Tac-Toe*.

2. Maximizing dan Minimizing

   Fungsi ini memiliki dua mode: *maximizing* (maksimisasi) dan *minimizing* (minimisasi), yang digunakan bergantian. Pemain **X** selalu berusaha untuk memaksimalkan skor, sementara pemain **O** berusaha untuk meminimalkan skor.

3. Iterasi

Selama mode maksimisasi, fungsi mencari langkah terbaik yang dapat diambil oleh pemain **X**. Selama mode minimisasi, fungsi mencari langkah terbaik yang dapat diambil oleh pemain **O**. Ini dilakukan dengan iterasi melalui semua kotak kosong yang tersedia pada papan permainan.

4. Pemangkasan Alpha-Beta
   Pada setiap iterasi, fungsi memanggil dirinya sendiri secara rekursif dengan perubahan papan yang disalin (untuk mencoba setiap langkah), dan memperbarui nilai alpha dan beta sesuai dengan mode yang berlaku.
   a. Selama mode maksimisasi, alpha adalah nilai terbaik yang ditemukan sejauh ini untuk pemain **X**, dan jika alpha lebih besar dari atau sama dengan beta, maka ada pemangkasan beta (*break*).
   b. Selama mode minimisasi, beta adalah nilai terbaik yang ditemukan sejauh ini untuk pemain **O**, dan jika beta kurang dari atau sama dengan alpha, maka ada pemangkasan alpha (*break*).

5. Print Statements
   Terdapat pencetakan pernyataan yang digunakan untuk melacak pemangkasan (cut) dalam algoritma. Ini digunakan untuk pemahaman lebih lanjut tentang bagaimana algoritma bekerja.

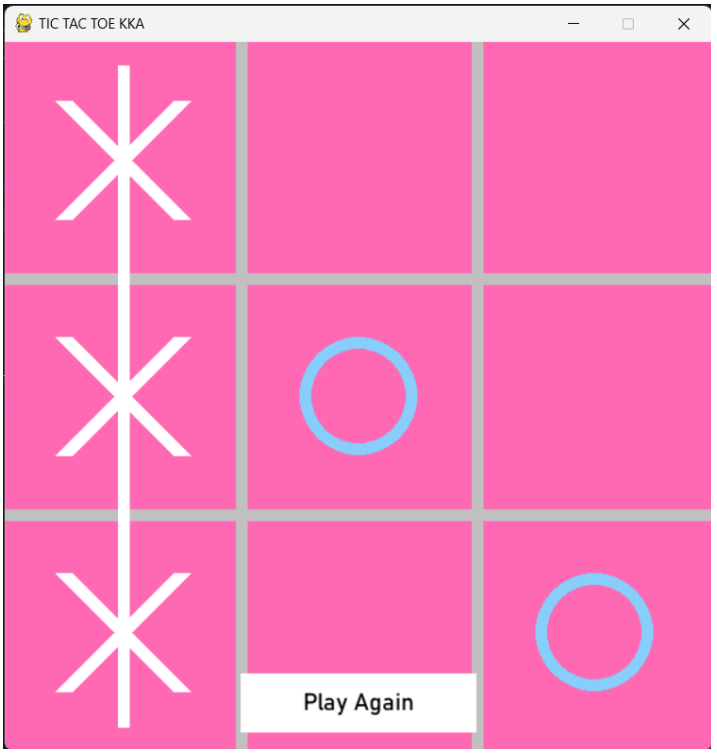
6. Return Values
   Fungsi mengembalikan nilai skor terbaik yang dapat dicapai dalam mode yang berlaku (maksimisasi atau minimisasi) dan langkah terbaik yang harus diambil oleh pemain pada saat itu.

7. Kompleksitas waktu
   Alpha-Beta Pruning berusaha untuk mengurangi jumlah simpul yang dieksplorasi dengan menghentikan pencarian lebih awal pada cabang tertentu jika sudah jelas bahwa hasilnya tidak akan mempengaruhi keputusan akhir. Oleh karena itu, alpha-beta pruning memiliki kompleksitas waktu yang lebih rendah dibandingkan Minimax, terutama jika pohon pencarian besar.

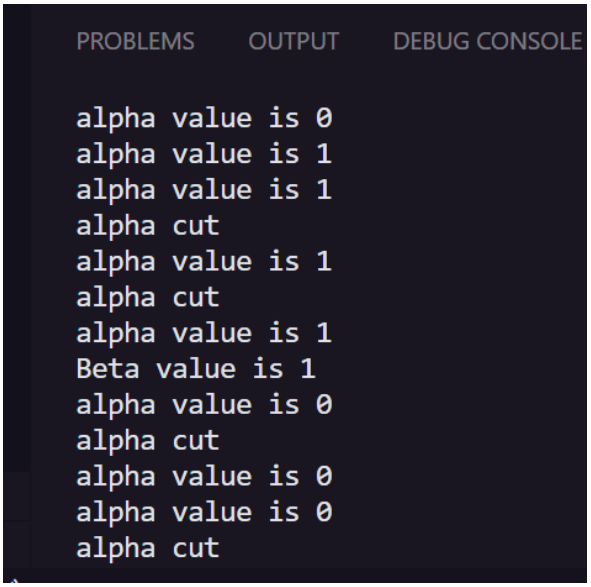
8. Kelebihan**:**
   a. Melibatkan pemangkasan (*pruning*), sehingga dapat mengurangi jumlah simpul yang dieksplorasi dan mempercepat pencarian.
   b. Efektif mengurangi cabang-cabang pohon yang tidak relevan untuk pencarian terbaik.

9. Kekurangan**:**

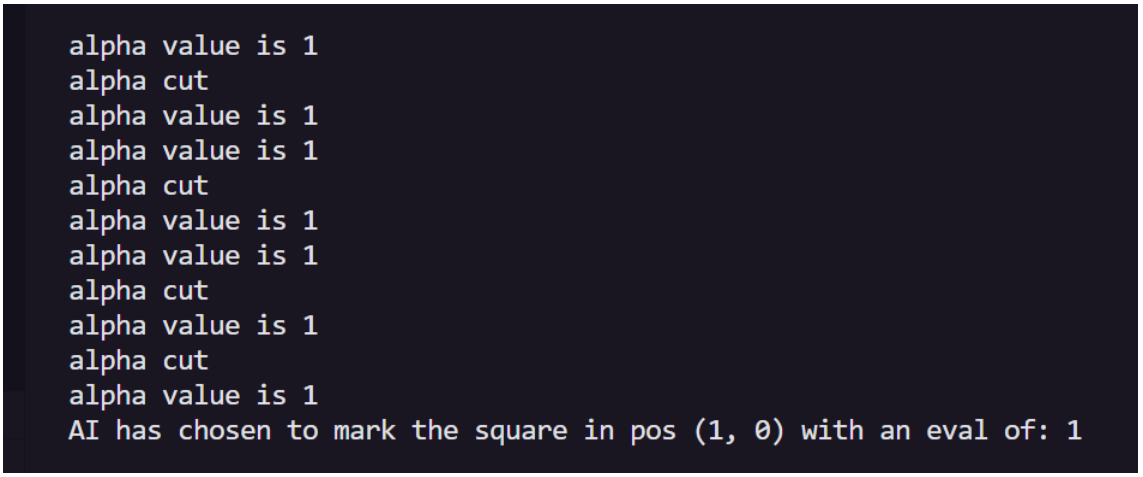
a. Lebih kompleks secara implementasi dibandingkan dengan *Minimax*.
b. Dalam beberapa kasus tertentu, mungkin tidak memberikan perbaikan signifikan dalam hal kinerja.



Gambar 3.3 Hasil dari kondisi *AI* (simbol X) memenangkan game.



Gambar 3.4 Hasil iterasi program menentukan nilai alpha dan melakukan pemangkasan alpha.



Gambar 3.2 Hasil tampilan program menunjukkan posisi dan nilai eval *AI*.

4. **Perbandingan Kedua Algoritma**
   - **Waktu Eksekusi:**
     - Alpha-Beta Pruning umumnya lebih cepat daripada Minimax karena melakukan pemangkasan.
     - Minimax dapat memerlukan lebih banyak waktu karena mencari semua kemungkinan langkah tanpa melakukan pemangkasan.
   - **Pemangkasan dan Efisiensi:**
     - Alpha-Beta Pruning efektif memotong cabang-cabang yang tidak relevan dari pohon permainan, mengurangi waktu eksekusi.
     - Minimax tidak melakukan pemangkasan dan secara potensial dapat memerlukan waktu yang lebih lama dalam pencarian pohon permainan yang besar.
   - **Kompleksitas Implementasi:**
     - Alpha-Beta Pruning memerlukan pemahaman yang lebih mendalam dan implementasi yang lebih rumit dibandingkan dengan Minimax.
     - Minimax lebih mudah diimplementasikan dan dipahami.

5. Kesimpulan

Dalam permainan *Tic Tac Toe* penggunaan algoritma Minimax dan Alpha-Beta Pruning memiliki kelebihan dan kekurangannya masing-masing. algoritma Minimax, dengan implementasi yang sederhana dapat memberikan solusi yang optimal, tetapi memiliki kekurangan dalam kompleksitas waktu atau kecepatannya karena algoritma ini tidak melakukan pemangkasan pohon saat proses pencarian. Sedangkan, algoritma Alpha-Beta Pruning memiliki implementasi yang lebih kompleks namun, mampu mengurangi jumlah simpul yang dieksplorasi dengan efisien, mempercepat pencarian, dan mengatasi kelemahan Minimax dalam hal waktu komputasi.