NOTES

- Laporan dalam format .pdf berisi informasi sbb:
    - Penjelasan mengenai *objective function* yang digunakan
    - Penjelasan proses pencarian dengan *minimax* dan *alpha beta pruning* yang dilakukan pada permainan Adjacency Strategy Game.
    - Jika mengimplementasikan, penjelasan algoritma *local search* yang digunakan. Berikan juga justifikasi mengapa algoritma tersebut dipilih dibandingkan dengan algoritma-algoritma yang lain.
    - Jika mengimplementasikan, penjelasan Genetic Algorithm yang digunakan.
    - Penjelasan hasil pertandingan yang dilakukan antara:
        - Bot *minimax* vs. manusia (sebanyak 5 kali)
        - Jika mengimplementasikan, Bot *local search* vs. manusia (sebanyak 3 kali)
        - Jika mengimplementasikan, Bot *minimax* vs bot *local search* (sebanyak 3 kali)
        - Jika mengimplementasikan, Bot *minimax* vs bot *genetic algorithm* (sebanyak 3 kali)
        - Jika mengimplementasikan, Bot *local search* vs bot *genetic algorithm* (sebanyak 3 kali)
        - Catat jumlah kemenangan dan kekalahan untuk setiap jenis pertandingan untuk mengetahui persentase kemenangan setiap jenis bot.
        - Untuk percobaan, minimal 1 percobaan dengan jumlah ronde maksimal atau sampai board penuh. Kemudian, untuk percobaan lainnya minimal 8 ronde. **Contoh**: Dalam 3 kali percobaan, direkam: 24 ronde, 8 ronde, 12 ronde.
    - Kontribusi setiap anggota dalam kelompok

Tugas Besar I IF3170 Inteligensi Buatan

Minimax Algorithm and Alpha Beta Pruning in Adjacency Strategy Game

Dibuat Oleh:

Salomo Reinhart Gregory Manalu	13521063
Margaretha Olivia Haryono	13521071
Muhammad Rifko Favian	13521075
Moch. Sofyan Firdaus	13521083

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2023

A. Objective Function

Misalkan, marka O adalah marka dari bot. Maka, objective function (f) adalah:

f = jumlah marka O pada papan - jumlah marka X pada papan

Fungsi ini menentukan kemenangan ataupun kekalahan dari Bot. Apabila f bernilai positif, maka Bot memenangkan game. Apabila f bernilai negatif, Bot kalah dalam game. Apabila f bernilai 0, maka permainan berakhir seri.

Alasan penggunaan rumus ini adalah karena rumus ini relevan dan sejalan dengan tujuan permainan, yaitu pemain (Bot) berusaha untuk memiliki lebih banyak marka daripada jumlah marka lawan.

B. Minimax Alpha Beta Pruning

Untuk melakukan pencarian, kita perlu menentukan Game Search Problem dari permainan Adjacency Strategy Game ini. Karena game ini hanya memiliki 2 pemain, maka bisa kita anggap pemain itu sebagai MAX dan MIN, MAX adalah agent yang kita buat untuk bisa memenangkan game nya, sementara MIN adalah lawan dari agent tersebut. Setelah melihat source code dan mencoba game ini, kami asumsi kemungkinan tujuan dari tugas ini adalah membuat bot yang bisa mengalahkan Player dengan algoritma Alpha Beta Pruning ini. Maka dari itu, kami simpulkan MAX adalah bot dan MIN adalah Player. Kemudian kita perlu menentukan Terminal State dari game ini. Dari aturan game ini, game akan selesai apabila papan penuh atau mencapai batas ronde yang telah ditetapkan di awal. Maka dari itu, Terminal State adalah state di mana papan sudah terisi penuh atau ronde sudah habis. Setelah itu kita menentukan nilai Utility saat mencapai Terminal State. Karena pemenang dari game ini adalah yang memiliki marka terbanyak pada papan, maka kami putuskan untuk menghitung utility dengan cara:

$$Utility = Mark\ MAX - Mark\ MIN$$

Apabila nilainya positif (+), maka bot menang; dan apabila nilainya negatif (-), maka bot akan kalah. Dari aturan permainan, di sini bot akan berusaha untuk mendapat marka sebanyak mungkin.

Permainan dimulai dengan adanya 4 X (Marka Player) di pojok kiri bawah, dan 4 O (Marka Bot) di pojok kanan atas. Hal ini adalah initial state dari game ini. Bot tidak selalu memulai permainan dari initial state, karena ada saatnya kemungkinan Player yang akan

memulainya. Kemudian Action dari game ini adalah mengisi salah satu dari semua kotak kosong oleh giliran pemain saat itu.

Di sini alpha dan beta dihitung layaknya dalam algoritma Alpha Beta Pruning biasa, tidak ada cara perhitungan yang berubah. Alpha dan Beta akan dihitung untuk melakukan Pruning apabila ada akar (branch) yang tidak mempengaruhi keputusan akhir dari MAX dan MIN.

Untuk feasibility dalam mengurangi kompleksitas, kami memutuskan untuk membatasi kedalaman penulusuran simpul menjadi maksimal sebesar 10. Hal ini dilakukan karena di awal game, akan membutuhkan waktu yang sangat lama untuk menulusuri semua simpul yang ada.

Proses kerja minimax yang dilakukan ialah:

1. Inisialisasi nilai α (alpha) dan β (beta), yaitu -∞ untuk alpha dan ∞ untuk beta.
2. Ekspansi Game-Search Tree mulai dari simpul awal, yaitu state saat ini, pohon keputusan permainan diperluas sampai kedalaman maksimal 10, ronde berakhir, atau kotak terisi penuh; kemudian menghitung utility di simpul-simpul terdalam tersebut. Di setiap tingkat pohon, satu pemain berusaha memaksimalkan utility (MAX) sementara pemain lainnya berusaha meminimalkan utility (MIN), diawali dengan bot sebagai MAX karena kita berusaha memenangkan bot, kemudian dilanjut dengan player sebagai MIN karena player akan berusaha mengurangi nilai utility.
3. Pruning: Saat memperluas pohon untuk pemain MAX, jika nilai utility dari salah satu simpul anak lebih kecil dari alpha, maka cabang tersebut dipotong dan tidak diperluas lebih lanjut karena kita tahu pemain MIN tidak akan pernah memilih jalan tersebut. Sebaliknya, saat memperluas pohon untuk pemain MIN, jika nilai utility dari salah satu simpul anak lebih besar dari beta, maka cabang tersebut dipotong dan tidak diperluas lebih lanjut karena kita tahu pemain MAX tidak akan memilih jalan tersebut.
4. Pada setiap langkah di mana pemain MAX berada di simpul, nilai alpha diperbarui menjadi maksimum dari nilai alpha saat ini dan nilai simpul anak yang baru dievaluasi. Selain itu, saat pemain MIN berada di simpul, nilai beta diperbarui menjadi minimum dari nilai beta saat ini dan nilai simpul anak yang baru dievaluasi.

C. Algoritma Local Search

Algoritma Local Search yang digunakan adalah Hill Climbing with Sideways Move.

Pada permainan ini, *agent* berperan sebagai *player* Bot yang bermain melawan *player* manusia. Initial state dari game ini adalah empat buah marka X di pojok kiri bawah dan empat buah marka O di pojok kanan atas. Asumsi Bot memainkan marka O. Selanjutnya, lawan akan menaruh satu marka X ditempat yang lawan inginkan. Satu ronde dari permainan ini dilakukan bergantian dengan bot hingga 30 kali setiap pihak atau hingga papan penuh.

Saat giliran Bot, Bot akan memeriksa *neighbor* mana yang memiliki nilai objektif tertinggi diantara semua *neighbor*. *Objective function* yang digunakan yaitu pengurangan jumlah marka yang dimiliki Bot dengan jumlah marka yang dimiliki lawan. Pada kasus ini, *neighbor* adalah status-status yang mungkin ketika Bot meletakkan markanya pada kotak kosong. Selanjutnya, Bot akan menaruh marka X di kotak *neighbor* yang memiliki nilai objektif tertinggi.

```
function HillClimbing(problem) return local maximum state
  currentState ← MakeNode(problem.initialState)
  loop do
    neighbor ← highest-valued successor of currentState
    currentState ← neighbor
```

Berikut contoh dari penerapan Hill-Climbing search pada permainan ini:

a. State Awal (Asumsi Bot memegang marka O, lawan marka X)

1	1	1	1	1	1	O	O
1	1	1	1	1	1	O	O
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
3	3	1	1	1	1	1	1
X	X	3	1	1	1	1	1
X	X	3	1	1	1	1	1

b. Giliran lawan untuk melangkah

-2	-2	-2	-2	-2	-2	O	O
-2	-2	-2	-2	-2	0	X	O
-2	-2	-2	-2	-2	0	X	0
-2	-2	-2	-2	-2	-2	0	-2
-2	-2	-2	-2	-2	-2	-2	-2
0	0	-2	-2	-2	-2	-2	-2
X	X	0	-2	-2	-2	-2	-2
X	X	0	-2	-2	-2	-2	-2

c. Giliran bot untuk melangkah

1	1	1	1	1	1	O	O
1	1	1	1	1	3	X	O
1	1	1	1	1	3	X	3
1	1	1	1	1	1	3	1
1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1
X	O	O	1	1	1	1	1
X	X	3	1	1	1	1	1

D. Genetic Algorithm

Genetic algorithm dapat diterapkan dalam permainan ini. Penerapan *Genetic Algorithm* pada game *Adjacency Strategy Game* adalah sebagai berikut:

1. Representasi Kromosom

   Setiap individu dalam populasi akan direpresentasikan sebagai kromosom yang berisi urutan langkah Bot dan lawan yang diwakili oleh angka-angka 2 digit nomor kotak tempat peletakan marka tersebut. Angka-angka ini mewakili nomor kotak pada papan permainan yang dimulai dari 01 hingga 64. Kromosom akan berubah seiring berjalannya permainan pada setiap langkah atau giliran. Langkah yang dilakukan oleh bot dan lawan

akan terus membuat panjang kromosom berkurang karena jumlah langkah yang semakin sedikit.

2. Populasi Awal

   Populasi awal akan dibuat dengan beberapa kromosom acak.

3. Fungsi Fitness

   Fungsi fitness akan menghitung jumlah marka bot O dikurangi jumlah marka pemain X setelah mengikuti langkah-langkah yang diwakili oleh kromosom. Tujuannya adalah memaksimalkan selisih jumlah marka O dan X.

4. Seleksi Parent dan Reproduksi (Cross Over)

   Parent akan dipilih secara acak berdasarkan nilai fitness dari masing-masing kromosom di populasi awal. Orang tua dengan nilai fitness lebih tinggi akan lebih cenderung dipilih untuk reproduksi. Setelah mendapatkan parent, maka akan dilakukan reproduksi atau *cross over*. Proses *cross over* akan menggabungkan atau menyilangkan komponen atau sebagian dari dua orang tua yang sudah dipilih. *Cross over* ini dilakukan untuk menghasilkan populasi baru.

5. Mutasi

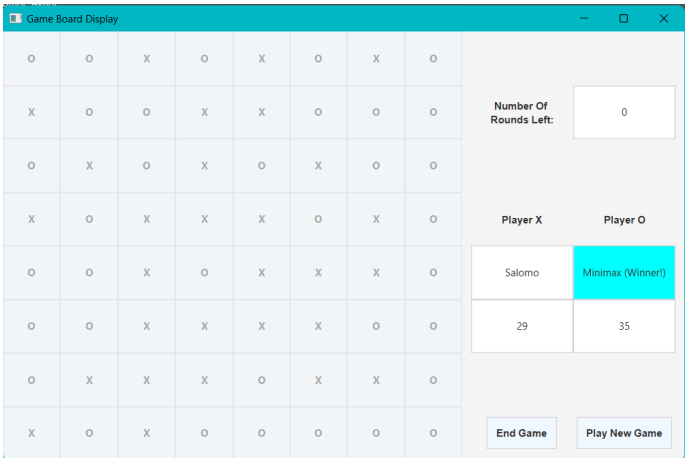
   Langkah terakhir adalah melakukan mutasi pada anak-anak baru atau populasi baru hasil *cross over* dari selected parent. Tujuan utama mutasi adalah untuk menghadirkan variasi dalam langkah-langkah yang diambil oleh bot selama permainan. Dengan demikian, bot dapat menjelajahi berbagai strategi untuk mencari tahu yang paling efektif dalam mencapai tujuan permainan, yaitu memiliki jumlah marka pemain yang lebih tinggi daripada marka lawan pada akhir permainan.

Proses-proses ini akan diulang terus menerus hingga mendapat solusi yang optimal sehingga Bot dapat memenangkan permainan Adjacency Strategy Game ini.

E. Uji Kasus

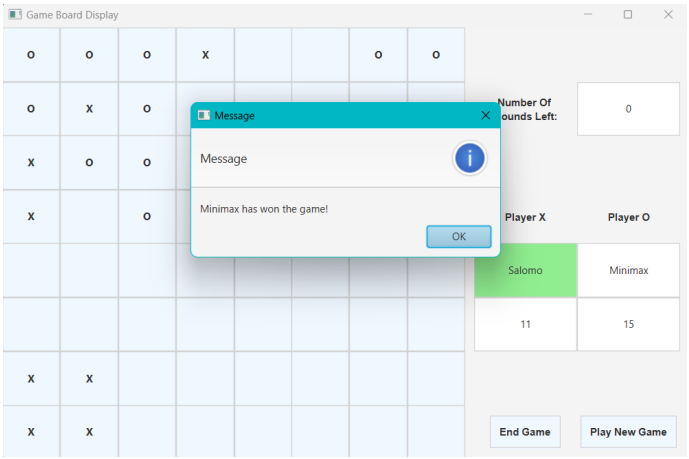
Untuk percobaan, minimal 1 percobaan dengan jumlah ronde maksimal atau sampai board penuh. Kemudian, untuk percobaan lainnya minimal 8 ronde. **Contoh**: Dalam 3 kali percobaan, direkam: 24 ronde, 8 ronde, 12 ronde.

1. Bot *minimax* vs. manusia
   a. Uji 1 (sampai board penuh)



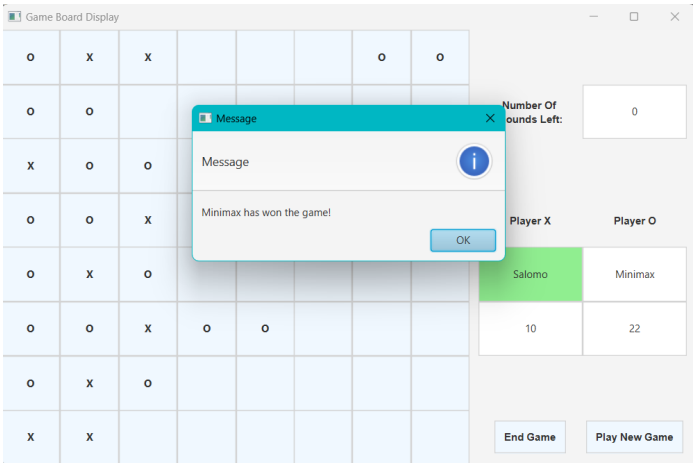
Pada percobaan pertama, hasil didapat bahwa permainan dimenangkan oleh bot.

b. Uji 2 (9 ronde)



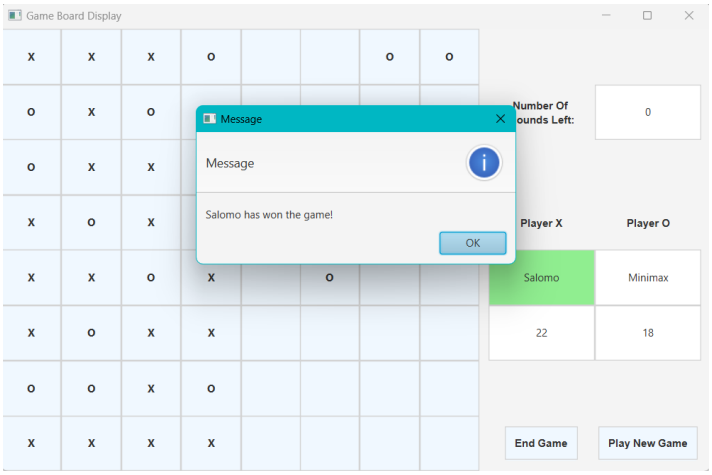
Pada percobaan kedua, hasil didapat bahwa permainan dimenangkan oleh bot.

c. Uji 3 (12 ronde)



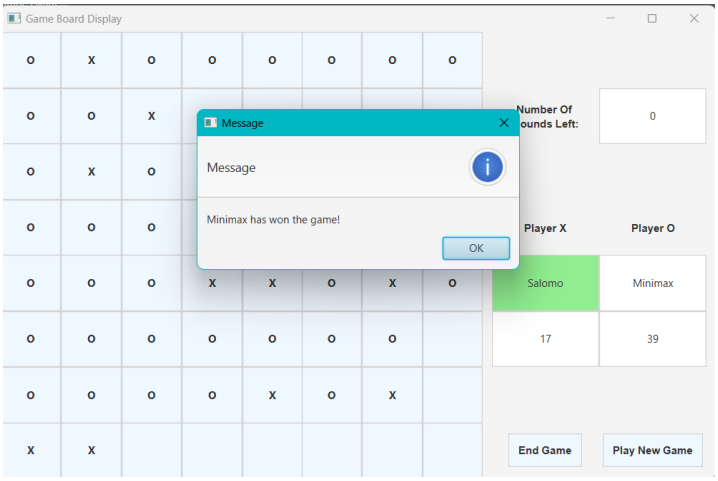
Pada percobaan ketiga, hasil didapat bahwa permainan dimenangkan oleh bot.

d. Uji 4 (16 ronde)



Pada percobaan keempat, hasil didapat bahwa permainan dimenangkan oleh manusia.

e. Uji 5 (24 ronde)

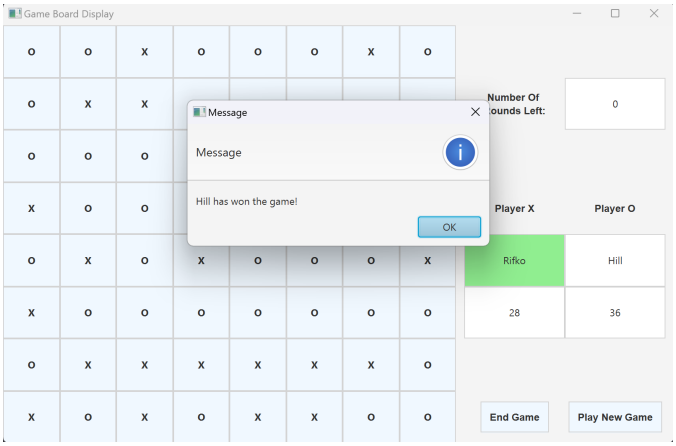


Pada percobaan keempat, hasil didapat bahwa permainan dimenangkan oleh bot.

Jumlah menang: 4, jumlah kalah: 1, persentase kemenangan: 80%

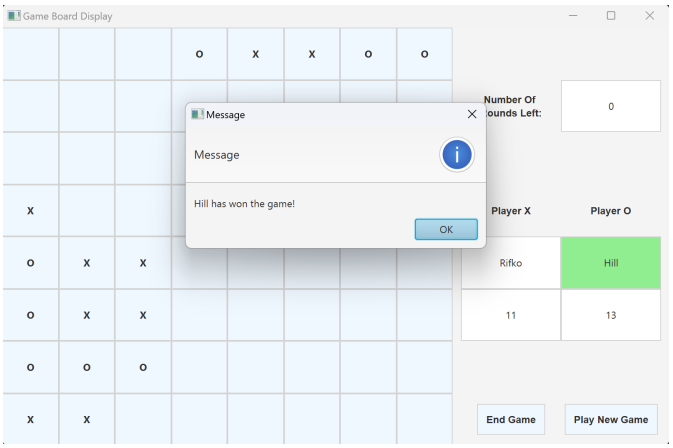
2. Jika mengimplementasikan, Bot *local search* vs. manusia

a. Uji 1 (sampai board penuh)



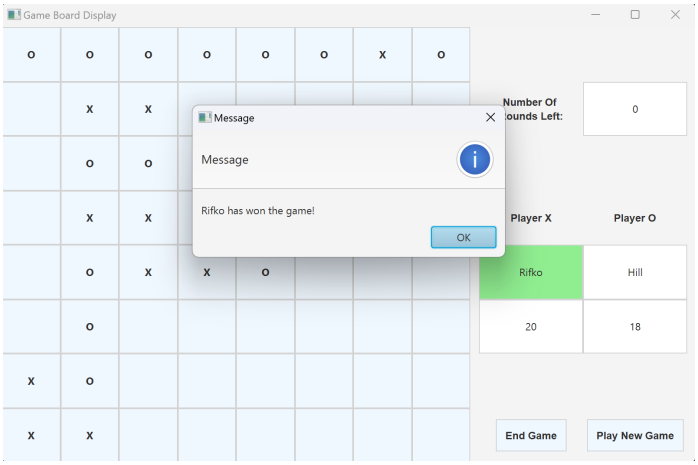
Pada percobaan pertama, hasil didapat bahwa permainan dimenangkan oleh bot.

b. Uji 2 (8 ronde)



Pada percobaan kedua, hasil didapat bahwa permainan dimenangkan oleh bot.

c. Uji 3 (15 ronde)

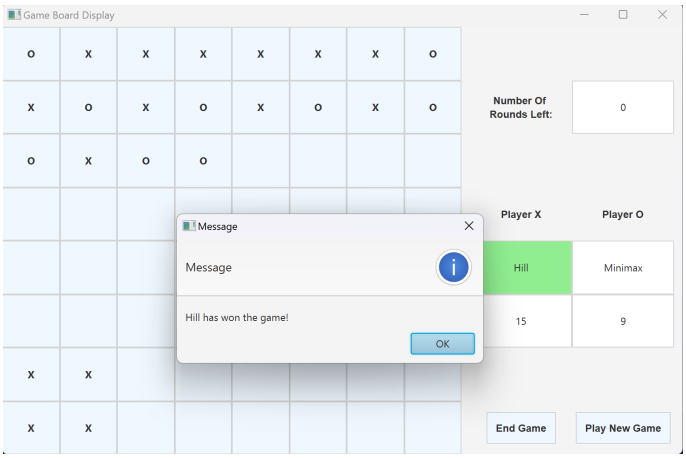


Pada percobaan ketiga, hasil didapat bahwa permainan dimenangkan oleh Player.

Jumlah menang: 2, jumlah kalah: 1, persentase kemenangan: 67%

3. Jika mengimplementasikan, Bot *minimax* vs bot *local search*
    a. Uji 1

    b. Uji 2 (8 ronde)



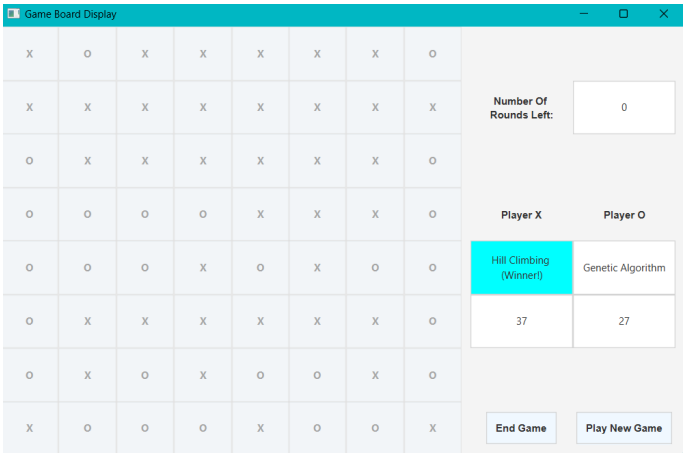
    c. Uji 3

Jumlah menang: <insert>, jumlah kalah: <insert>, persentase kemenangan: <insert>

4. Jika mengimplementasikan, Bot *minimax* vs bot *genetic algorithm*
    a. Uji 1
    b. Uji 2
    c. Uji 3

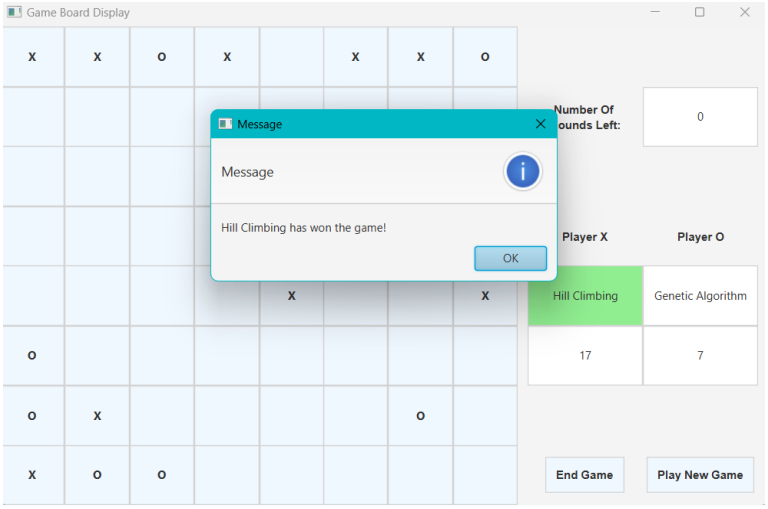
Jumlah menang: <insert>, jumlah kalah: <insert>, persentase kemenangan: <insert>

5. Jika mengimplementasikan, Bot *local search* vs bot *genetic algorithm*
    a. Uji 1 (sampai board penuh)



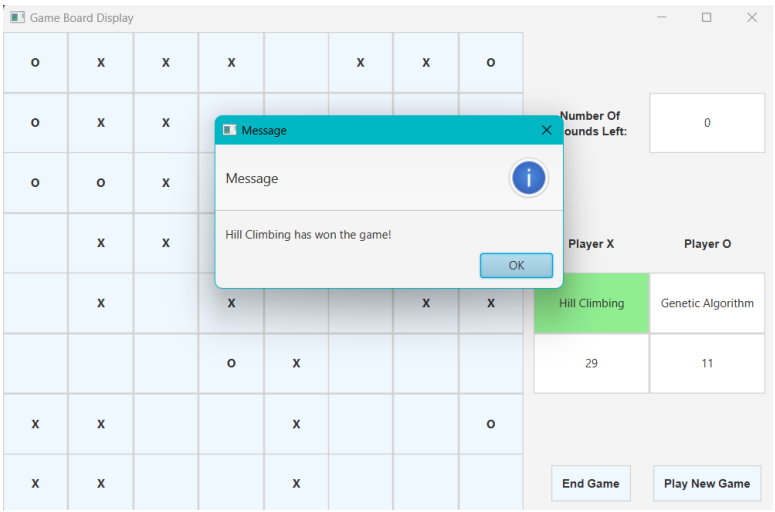
Pada percobaan pertama, hasil didapat bahwa permainan dimenangkan oleh bot Local Search.

b. Uji 2 (8 ronde)



Pada percobaan kedua, hasil didapat bahwa permainan dimenangkan oleh bot Local Search.

c. Uji 3



Pada percobaan ketiga, hasil didapat bahwa permainan dimenangkan oleh bot Local Search.

Jumlah menang: <insert>, jumlah kalah: <insert>, persentase kemenangan: <insert>

F. Pembagian Tugas

Nama	**NIM**	**Tugas**
Salomo Reinhart Gregory Manalu	13521063	Genetic Algorithm
Margaretha Olivia Haryono	13521071	Minimax
Muhammad Rifko Favian	13521075	Local Search
Moch. Sofyan Firdaus	13521083	Minimax, Handle Bot vs Bot