

## **LAPORAN TUGAS BESAR 1**

**“Implementasi *Minimax Algorithm* dan *Local Search* pada Permainan *Dots and Boxes*”**

**Laporan Ini Dibuat Untuk Memenuhi Tugas Perkuliahan**

**Mata Kuliah Inteligensi Buatan (IF3170)**



**DISUSUN OLEH:**

### **Kelompok 36**

- |                          |            |
|--------------------------|------------|
| 1. M. Risqi Firdaus      | (13520043) |
| 2. Rava Naufal Attar     | (13520077) |
| 3. Malik Akbar Hashemi R | (13520105) |
| 4. Hafidz Nur Rahman G   | (15320117) |

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

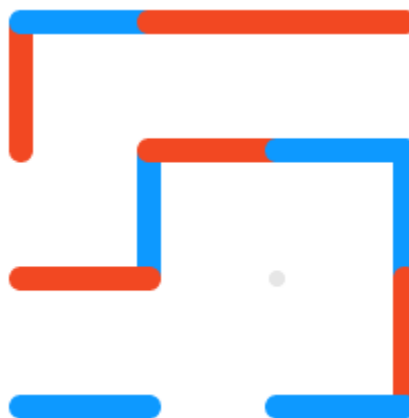
**SEMESTER I TAHUN 2022/2023**

## Objective Function

*Objective function* merupakan suatu fungsi yang dapat menilai semua *state* dalam *state space* berdasarkan kriteria tertentu. Fungsi ini mengacu ke fungsi yang mengeluarkan nilai *real* yang akan dimaksimalkan atau diminimalkan. Fungsi ini sangat berguna untuk penentuan keputusan karena kita dapat membandingkan *state* satu dengan *state* lainnya, dan biasa digunakan pula untuk optimalisasi.

*Objective function* yang digunakan pada permainan ini berguna dalam menentukan peletakan garis. *Objective function* pada bot ini dilakukan dengan mekanisme skoring untuk setiap kotak. Nilai dari setiap kotak ditentukan oleh berapa banyak garis yang ada di kotak tersebut, khusus untuk 4 garis, penentuan skor melibatkan siapa yang menaruh garis terakhir. Semua nilai untuk setiap kotak akan dijumlah dan hasilnya menjadi nilai *objective function* dari *state* tersebut. Jika suatu *state* terdapat suatu kotak dengan 3 garis, maka bagi pemain yang menaruh garis ketiga, *state* tersebut bernilai buruk karena pemain selanjutnya dapat menaruh garis satu lagi dan mendapatkan poin sehingga kotak dengan 3 garis bernilai negatif. Jika suatu *state* terdapat suatu kotak dengan 2 garis, maka bagi pemain yang menaruh 2 garis tersebut, *state* tersebut bernilai bagus karena pemain musuh terpaksa menaruh garis ketiga di kotak tersebut atau mencoba di kotak lain. Begitu pula untuk kotak dengan 1 garis. Pada bot ini kami menentukan nilainya adalah untuk 1 garis bernilai -5, 2 garis bernilai 10, dan 3 garis bernilai -20, serta 4 garis 100.

Objective function pada *state* di bawah bernilai sebesar 80.



## Minimax dan Alpha-Beta Pruning Algorithm

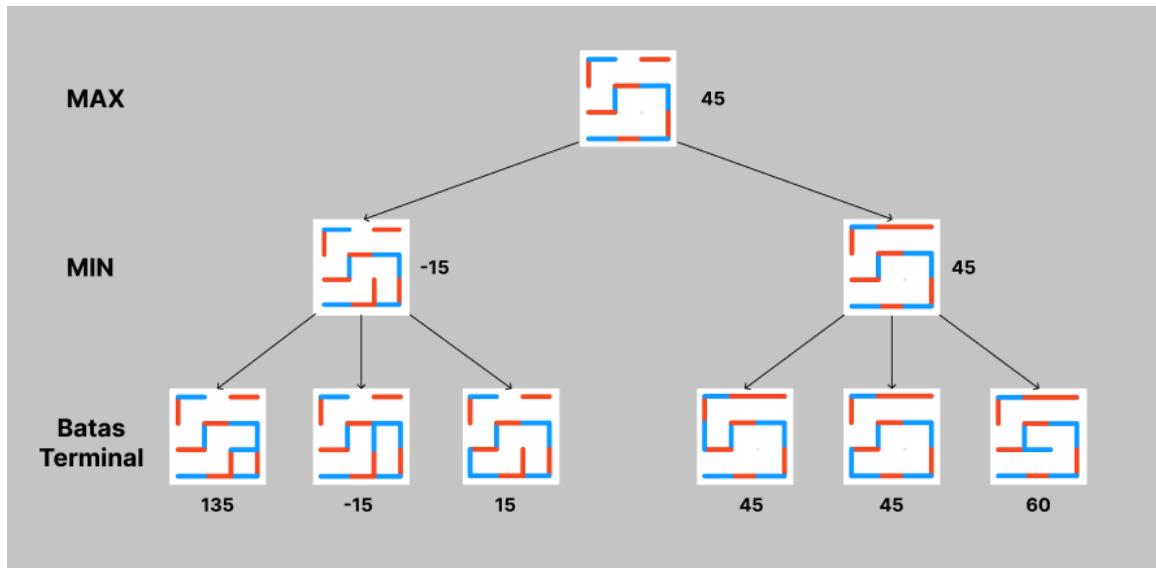
Algoritma Minimax merupakan sebuah algoritma yang digunakan untuk menentukan pilihan dengan mengevaluasi semua kemungkinan yang ada dan memprediksi langkah selanjutnya. Algoritma ini sangat cocok digunakan dalam permainan yang dimainkan oleh lebih dari 1 pemain. Algoritma Minimax akan melakukan pengecekan pada seluruh kemungkinan yang ada dan menghasilkan pohon ruang status yang lengkap. Kemudian, *state* daun akan dievaluasi nilai objektifnya hingga mendapatkan langkah paling optimal pada saat itu.

Pada program digunakan Minimax dengan optimasi menggunakan Alpha-Beta Pruning. Dalam mengkonstruksi program, bot ini dibuat dengan model rekursi. Untuk mendukung pembatasan waktu berpikir, digunakan multithreading dengan metode pembatasan waktu. Bot akan menyimpan best action sejauh waktu yang dilakukan. Best action yang didapat sejatinya adalah best action hingga ambang batas waktu berpikir.

Proses pencarian dimulai dengan mencari state yang dapat memaksimalkan nilai yang didapat oleh bot. Kemudian bot akan membangkitkan action yang dapat diambil. Setiap action akan ditelusuri hingga selesai atau sampai batas state tertentu. Setelah itu akan dilakukan evaluasi berdasarkan objective function yang telah dibuat. Setiap nilai pada state daun akan dievaluasi di state sebelumnya sesuai dengan kondisi maximum atau minimum. Setelah dievaluasi hingga state pemanggilan, bot akan mengetahui action yang mengembalikan nilai paling optimal yang dapat diraih. Metode alpha beta pruning digunakan untuk memangkas state berikutnya yang tidak akan dipilih.

Program yang dijalankan oleh bot sudah dimodifikasi agar dapat menyesuaikan aturan permainan. Jika pada algoritma dasar, evaluasi permainan dilakukan bergantian antarpemain. Pada algoritma modifikasi, ditambahkan aturan, jika pemain bisa mengonstruksi sebuah kotak, maka state permainan akan tetap, misal minimum-minimum atau maximum-maximum.

Selain itu, Bot Minimax kami buat dengan batasan kedalaman. Hal ini berguna untuk menyesuaikan waktu komputasi tiap gerakan yang terbatas, yaitu lima detik saja. Hasil temuan kami menunjukkan bahwa kedalaman optimum bagi bot minimax yang kami buat ialah 5. Jika kedalaman ditingkatkan, maka evaluasi yang dilakukan program belum maksimal karena terdapat batas waktu yang telah ditentukan. Akibatnya bot memiliki kemungkinan pengambilan keputusan buruk lebih besar.



Gambar di atas adalah contoh kondisi Minimax yang mungkin. Kondisi di atas hanya ilustrasi, tidak menggambarkan keseluruhan state yang dibangkitkan. Dengan menggunakan objective function yang telah didefinisikan, diperoleh nilai dari setiap state pada batas kedalaman tertentu. Oleh karena itu, langkah yang diambil pada state tersebut adalah menaruh stick pada posisi row di baris 1 kolom 2.

### Local Search Algorithm

Pada program diterapkan salah satu algoritma *local search* yaitu *hill-climbing search*. Algoritma ini paling memungkinkan untuk digunakan pada *Dots and Boxes Game (turn-based)* dibandingkan dengan algoritma-algoritma lainnya. *Hill-climbing search* memungkinkan *bot* untuk membuat keputusan terbaik dari instrumen-instrumen yang ada (*current state, successor, neighbor*) dengan tidak menyulitkan. Sedangkan jika menggunakan *simulated annealing algorithm*, diperlukan kalkulasi peluang  $e^{\Delta E/T}$  dan pendefinisian nilai dari  $T$  (*temperature*) itu sendiri. Selain itu, *genetic algorithm* tidak memungkinkan untuk dilakukan pada tahapan-tahapannya (*random selection, cross-over, mutation*). Hal ini dikarenakan permainan tidak *complete state* dan setiap giliran hanya dapat menempatkan satu garis dari setiap tempat kosong yang tersedia.

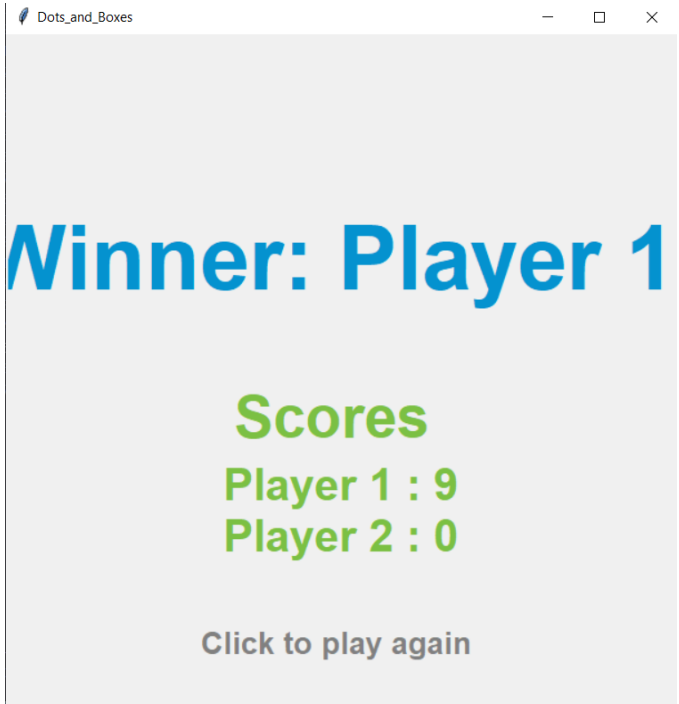
Akan tetapi, algoritma *hill-climbing* yang digunakan juga dimodifikasi berdasarkan aturan permainan yang berlaku. Karena tidak mungkin untuk tetap berada di *state* yang sama dan setiap giliran *player* harus menempatkan satu garis, maka algoritma akan memilih *neighbor*

dengan nilai tertinggi walaupun *neighbor value* tersebut lebih kecil daripada *current state value*. Selain itu, iterasi untuk mendapatkan *value state* terbaik setiap perpindahan *state* juga tidak dapat dilakukan. Hal ini dikarenakan setiap kali keputusan dibuat, maka *current state* akan berubah bergantung pada giliran *player* berikutnya. Algoritma juga akan memilih *neighbor* secara acak apabila terdapat lebih dari satu *highest-valued successor* dari *current state* agar pergerakan yang dilakukan lebih bervariasi dan tidak monoton.


## Hasil Pertandingan

Didapatkan hasil pertandingan sebagai berikut:

1. Bot Minimax vs Manusia
  - a. Bot Minimax sebagai player 1
    - i. Percobaan 1


Match Result	Winner
	Bot

ii. Percobaan 2

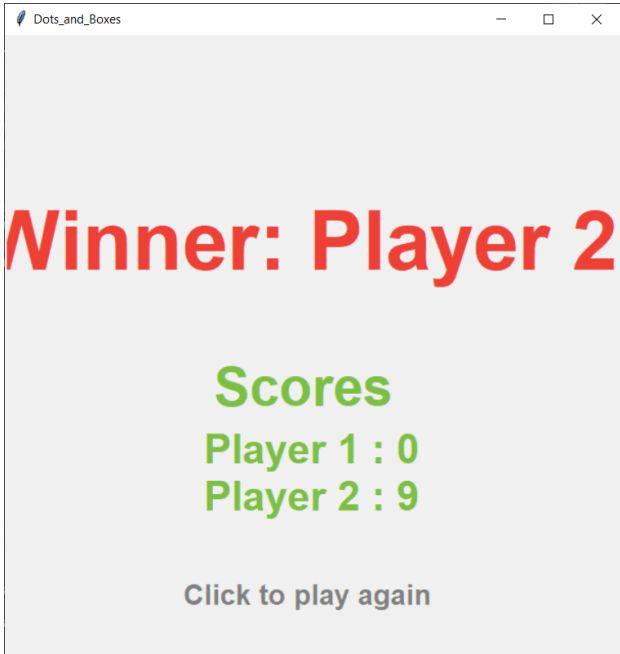
Match Result	Winner
	Bot

b. Bot Minimax sebagai player 2


i. Percobaan 3

Match Result	Winner
	Manusia

ii. Percobaan 4

Match Result	Winner
	Bot

iii. Percobaan 5

Match Result	Winner
	Bot

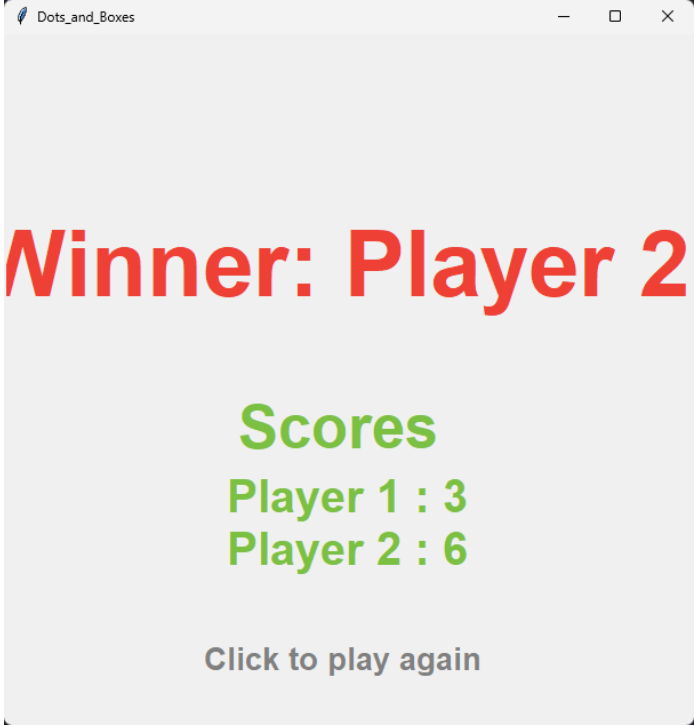


Bot	Win Status	Win Rate
Bot Minimax	4/5	80%


## 2. Bot Local Search vs Manusia

### a. Bot Local Search sebagai player 1

#### i. Percobaan 1

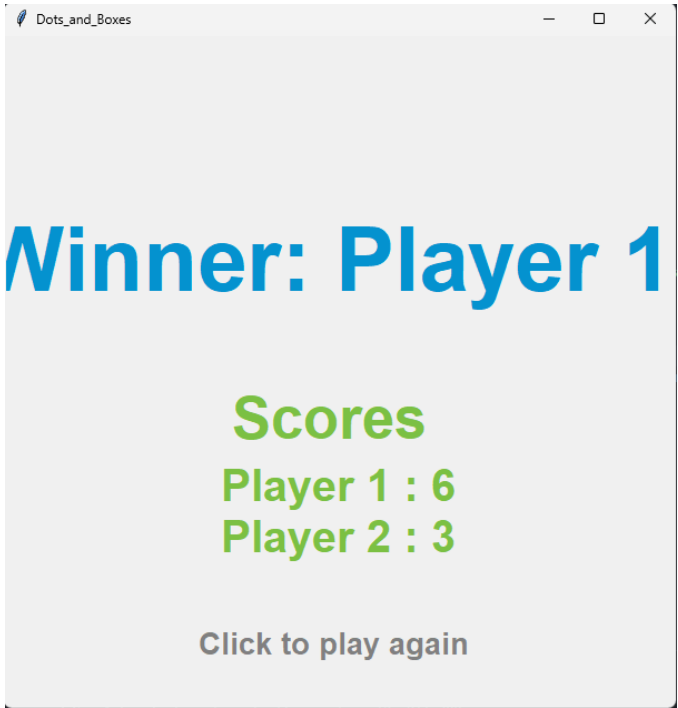
Match Result	Winner
	Manusia

ii. Percobaan 2

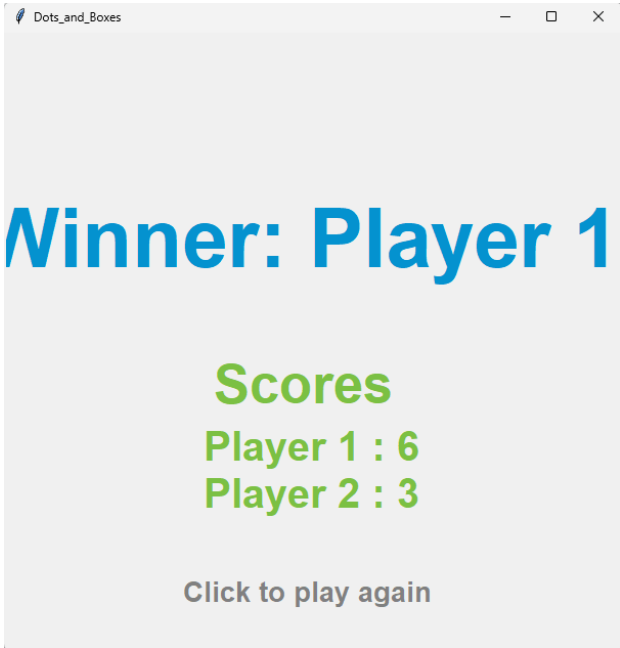
Match Result	Winner
	Bot

b. Bot Local Search sebagai player 2

i. Percobaan 3

Match Result	Winner
	Manusia

ii. Percobaan 4

Match Result	Winner
	Manusia

iii. Percobaan 5

Match Result	Winner
	Bot

Bot	Win Status	Win Rate
Bot Local Search	2/5	40%

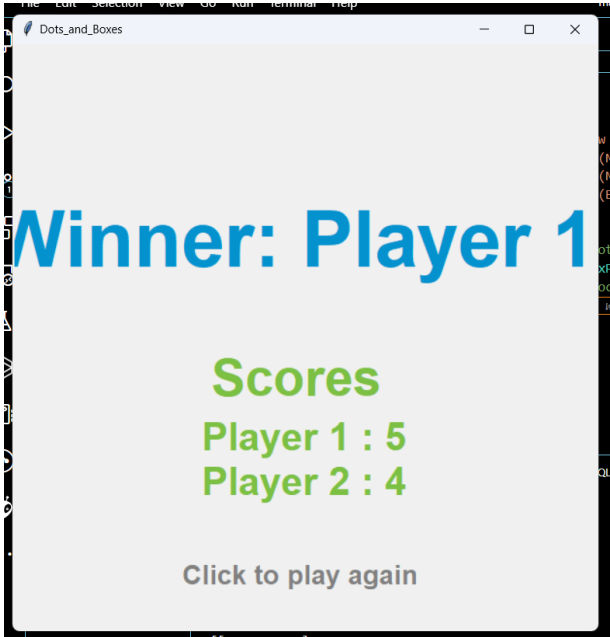
### 3. Bot Minimax vs Bot Local Search

#### a. Bot Minimax sebagai player 1

##### i. Percobaan 1

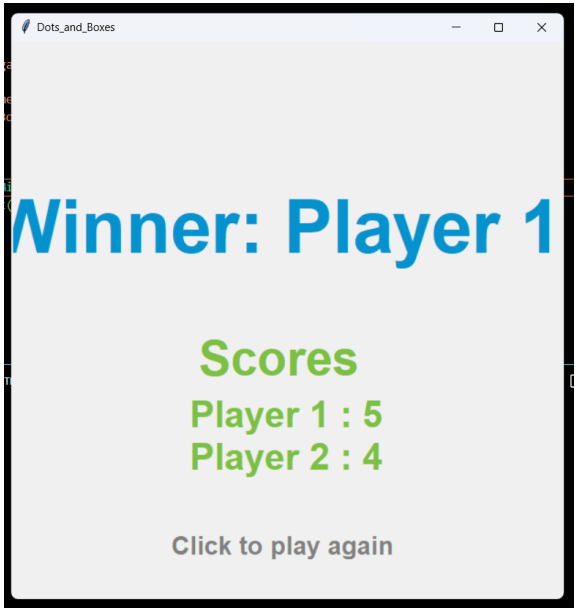
Match Result	Winner
	Bot Minimax

ii. Percobaan 2

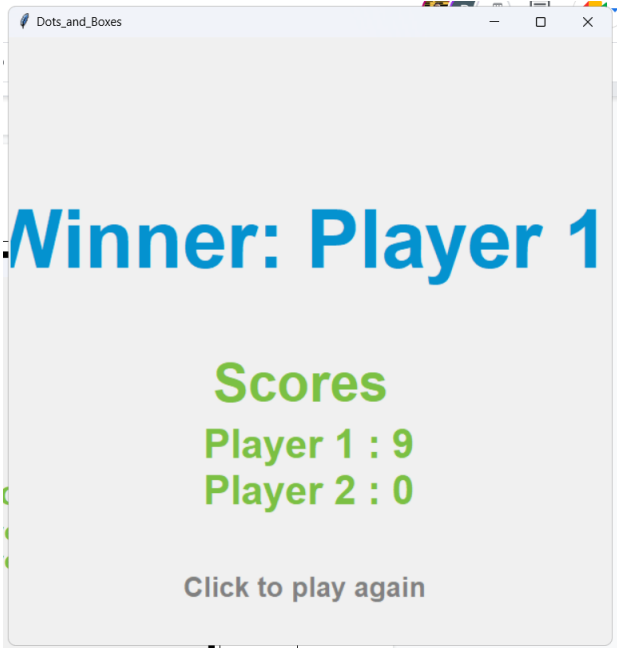
Match Result	Winner
	Bot Minimax

b. Bot Minimax sebagai player 2

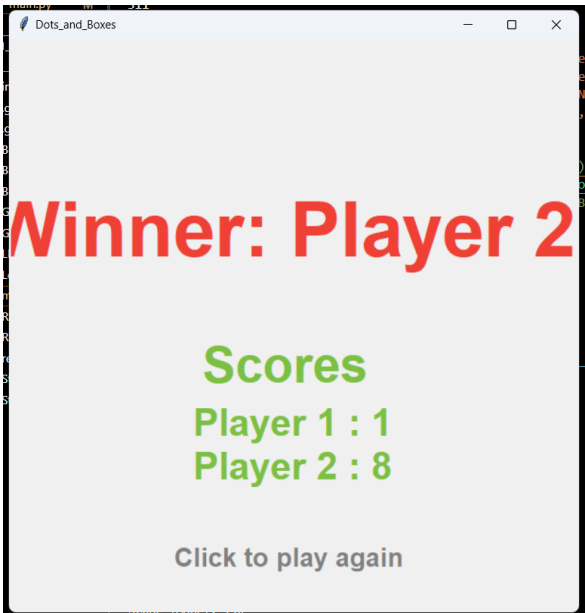
i. Percobaan 3

Match Result	Winner
	Bot LocalSearch

ii. Percobaan 4

Match Result	Winner
	Bot LocalSearch

iii. Percobaan 5

Match Result	Winner
	Bot Minimax

Bot	Win Status	Win Rate
Bot Minimax	3/5	60%
Bot Local Search	2/5	40%

## Saran

Berdasarkan bot yang telah kami buat, kami memiliki beberapa saran untuk pengembangan lebih lanjut, yaitu:

1. Riset lebih dalam mengenai *objective function* yang lebih optimal.
2. Urutan pembangkitan state selanjutnya pada Bot Minimax lebih dioptimalkan.
3. Mempercepat komputasi pada setiap pemanggilan fungsi minimax sehingga kedalaman yang dapat diperiksa oleh Bot Minimax dapat ditingkatkan.
4. Modifikasi untuk *local search algorithm* dapat di-*explore* lebih lanjut pada Bot Local Search.
5. Kedalaman dan waktu komputasi sangat mempengaruhi kemampuan bot minimax, maka minimax akan optimal jika tidak terlalu dalam, agar semua pilihan cabang dapat dievaluasi.
6. Mengoptimalkan algoritma minimax dengan cara tidak meng-*copy* state saat men-*transform*, tetapi melakukan transformasi tanpa *copy* dan melakukan *undo* ketika *backtrack*.

## Pembagian Tugas

No.	Pekerjaan	Penanggung Jawab
1	<i>Objective Function</i>	13520105
2	<i>Minimax and Alpha-Beta Pruning Algorithm</i>	13520043, 13520117
3	<i>Local Search Algorithm</i>	13520077