

Penerapan Algoritma Minimax pada Game Tic Tac Toe

Ramadhana Wahid Aji Pamungkas ^{a,1,*}, Husni Thamrin ^{a,2}

^a Universitas Muhammadiyah Surakarta, Indonesia;

¹ danawahidaji@gmail.com; ² husni.thamrin@ums.ac.id;

*Correspondent Author

Received:

Revised:

Accepted:

KATAKUNCI

Minimax
Kecerdasan buatan
Tic tac toe
Board

ABSTRAK

Algoritma minimax adalah algoritma yang digunakan untuk menentukan pilihan dengan memperkecil kemungkinan kehilangan nilai maksimal, hal ini dapat diartikan bahwa kemungkinan kekalahan akan diminimalkan dan kemenangan akan dimaksimalkan. Algoritma minimax dapat diterapkan dalam game berbasis papan permainan/board sehingga komputer akan berpikir seperti manusia dan memiliki kecerdasan buatan, seperti tic tac toe, catur, othello, dan lain lain. Game papan permainan/board yang akan diterapkan algoritma minimax ini adalah tic tac toe 4 x 4. Keuntungan dari penerapan algoritma minimax ini adalah mampu menganalisis segala kemungkinan posisi papan permainan/board untuk menghasilkan keputusan yang terbaik/bernilai maksimal karena algoritma minimax ini bekerja secara rekursif dengan mencari langkah yang akan membuat lawan mengalami kerugian minimum. Game akan lebih seru dan lebih menantang apabila diterapkan kecerdasan buatan melalui algoritma minimax ini. Game tic tac toe ini akan dirancang dengan 3 mode yaitu manusia vs manusia, manusia vs komputer yang memilih secara random, dan manusia vs komputer dengan diterapkan algoritma minimax. Hasil dari penerapan algoritma minimax pada game tic tac toe 4 x 4 untuk waktu eksekusi minimax atau waktu komputer mencari langkah terbaik sangat lama untuk 12 kotak kosong memerlukan waktu sampai 3 jam, ini belum 15 kotak kosong pasti akan lebih lama karena algoritma minimax mengecek semua kemungkinan pada papan permainan/board, berbeda dengan penerapan algoritma minimax pada game tic tac toe 3 x 3 untuk 9 kotak kosong menghasilkan waktu yang cepat sekitar 6 detik untuk komputer memilih langkah terbaik. dapat disimpulkan bahwa algoritma minimax bergantung pada banyaknya jumlah papan permainan/board.

KEYWORDS

Minimax
Artificial intelligence
Tic tac toe
Board

Application of the Minimax Algorithm to the Tic Tac Toe Game

Algorithm minimax is the algorithm used to make choices by minimizing the possibility of losing the maximum value, this can mean that the possibility of losing will be minimized and wins will be maximized. Algorithm minimax can be applied in game board game based/board so that computers will think like humans and have artificial intelligence, like tic tac toe, chess, othello, and others. Game board/board to which the algorithm will be applied minimax this is tic tac toe 4 x 4. The advantages of implementing the algorithm minimax is being able to analyze all possible positions of the game board/board to produce the best decision/maximum value because the minimax algorithm works recursively by looking for steps that will

make the opponent experience the minimum loss. Game will be more exciting and more challenging if artificial intelligence is applied through algorithms minimax. This game tic tac toe it will be designed with 3 modes namely human vs human, human vs computer that chooses automatically random, and humans vs computer with applied algorithms minimax. The result of applying the algorithm minimax on game tic tac toe 4 x 4 for the minimax execution time or the time the computer looks for the best move is very long for 12 empty squares it takes up to 3 hours, this is not yet 15 empty squares it will definitely take longer because of the algorithm minimax check all the possibilities on the game board/board, in contrast to the implementation of the algorithm minimax on game tic tac toe 3 x 3 for 9 empty squares results in a quick time of about 6 seconds for the computer to choose the best move. It can be concluded that the algorithm minimax depending on the number of game boards /board.

This is an open-access article under the [CC-BY-SA](#) license.



Pendahuluan

Game di era modern ini sangat populer karena pengaruh perkembangan teknologi yang sangat pesat dan semakin canggihnya peralatan teknologi. *Game* juga bisa menjadi teman di saat sendirian dan bisa sebagai obat refreking/penyegar otak agar tidak stress, sekarang banyak sekali jenis permainan/*game* yang dapat di pilih misalnya *game* pertarungan, *game* strategi, *game* casual, *game* papan dan lain lain. *Game* sekarang sudah banyak berkembang dan dapat dimainkan di mana saja karena sudah ada teknologi *handphone*, laptop, console dan alat lain yang dapat digunakan untuk bermain *game* [1].

Game memiliki manfaat juga harus diperhatikan tentang kerugian apabila terlalu sering bermain *game*/kecanduan bermain *game*, bahaya yang dapat di sebabkan karena kecanduan *game* adalah akan malas melakukan hal apapun, merusak mata karena terlalu sering melihat layar *handphone* atau komputer saat bermain komputer [2]. Kerugian ini dapat di atasi dengan cara membuat jadwal untuk bermain *game* dengan teratur misalnya bermain *game* hanya di hari sabtu dan minggu, dan ini harus ada komitmen agar mendapatkan manfaat dari bermain *game* [3].

Perkembangan *game* ini menunjukkan perkerjaan baru yaitu pembuat *game/game development* yang sekarang sudah banyak lapangan pekerjaannya oleh karena itu banyak sekali *game* yang beragam karena perkembangan ini, bahkan sudah ada *game* dengan kecerdasan buatan mirip dengan kecerdasan manusia ini dapat di buat dengan menggunakan algoritma tertentu misalnya algoritma *minimax* untuk *game* papan seperti *tic tac toe*, catur, dan lain lain [4]. Algoritma ini dimasukan ke dalam *game* agar komputer dapat memiliki kecerdasan mirip dengan manusia dan dapat mengambil keputusan seperti manusia [5]. Algoritma *minimax* merupakan algoritma yang mengecek semua kemungkinan langkah berikutnya dengan mempertimbangkan nilai maksimal dan minimal dari setiap kemungkinan sehingga komputer dapat memilih yang terbaik untuk memenangkan permainan. Algoritma *Minimax* ini dapat digunakan untuk membuat komputer memilih keputusan yang mirip dengan manusia saat bermain. Tujuan penggunaan algoritma ini agar permainan lebih menarik dan membuat permainan semakin seru [6].

Penggunaan algoritma ini membuat *game* menjadi lebih seru karena ada level/fase dimana pemain akan mudah menang, ada level/fase dimana komputer dapat menang dan komputer tidak terkalahkan atau hanya seri dengan komputer. *Game* seharusnya memiliki tingkat kesulitan yang beragam agar pemain tidak bosan dan akan terus memainkan *game* tersebut.

Game yang dapat diterapkan algoritma *minimax* ini misalnya *tic tac toe*, catur dan *game* papan lainnya. *Game Tic Tac Toe* ini akan sangat cocok karena akan membuat pemain seperti bermain dengan manusia lainnya yang digantikan oleh komputer, algoritma ini membuat komputer memiliki kecerdasan buatan mirip dengan manusia [7].

Algoritma *minimax* diterapkan dalam *game* diharapkan dapat membuat *game* yang lebih menantang karena melawan komputer dan dapat membuat *game* yang lebih menarik karena komputer tidak dapat kalah maksimal hasilnya adalah seri [8]. *Game Tic tac toe* dapat juga dibuat dengan lawan komputer yang memilih langkah secara *random* tanpa kecerdasan buatan, ini dapat membuat manusia merasa tertantang untuk mengalahkan komputer karena pada fase *random* komputer dapat mengalami kekalahan tidak seperti penerapan algoritma *minimax*. Pada penelitian ini dibuat *game* dengan jumlah papan permainan/*board* yang berbeda yaitu 4 x 4 dan akan diterapkan algoritma *minimax*, yang biasanya papan permainan/*board* dengan jumlah 3 x 3. Penelitian sebelumnya juga sudah ada yang membahas namun dengan papan permainan/*board* sama yaitu 3 x 3, dengan meneliti bagaimana jika komputer dengan algoritma *minimax* melawan komputer dengan algoritma *minimax*. [9]

Metode

1. Pengembangan Game

Tic Tac Toe 4 x 4 ini dibuat dengan menggunakan bahasa python dengan menggunakan modul *pygame* yang dapat digunakan untuk membuat *game*. *Game* akan disajikan seperti permainan pada umumnya dengan 3 level untuk yang pertama ada 2 *player* (manusia vs manusia), kemudian bermain dengan komputer masih memilih kotak secara *random* pada mode ini komputer dapat kalah, dan mode dimana komputer menerapkan Algoritma *Minimax* [10]. Pengembangan *Tic Tac Toe* 4 x 4 dimulai dengan membuat papan permainan/*board* 3 x 3 terlebih dahulu karena umumnya jumlah papan permainan/*board* 3 x 3 kemudian dirubah tampilannya menjadi papan permainan/*board* 4 x 4. Penggunaan algoritma *minimax* tidak dirubah karena penelitian ini ingin mengetahui efektifitas penerapan algoritma *minimax* dan ingin membuat *game* dengan tampilan berbeda. *Game Tic Tac Toe* 4 x 4 ini dibuat dengan mencari referensi dari dokumentasi, *google*, *youtube*, dan lain lain yang berhubungan dengan modul *pygame* dan bagaimana cara membuat *game Tic Tac Toe* dengan bahasa python.

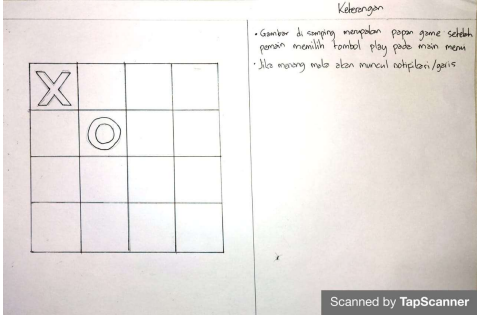
Game yang dikembangkan menerapkan algoritma *minimax*. Algoritma bekerja dengan prinsip menentukan pilihan dengan memperkecil kemungkinan kehilangan nilai maksimal. Hal ini dapat diartikan bahwa kemungkinan kekalahan akan diminimalkan dan kemenangan akan dimaksimalkan. Algoritma ini dapat diimplementasikan dalam *game* yang dimainkan dua pemain serta berbasis *zero-sum* seperti *game Tic Tac Toe* yang memiliki arti suatu kondisi dimana terdapat pemain yang memperoleh keuntungan, dan pemain lain memperoleh kerugian senilai dengan keuntungan yang diperoleh lawan dan sebaliknya [11]. Algoritma *minimax* melakukan pengecekan pada seluruh kemungkinan papan yang ada untuk mencari nilai maksimal, sehingga akan menghasilkan pohon *game* dari seluruh kemungkinan yang ada dan bekerja di belakang layar. Keuntungan yang di dapat dengan menggunakan algoritma *minimax* ini adalah mampu menganalisis segala kemungkinan posisi papan permainan untuk menghasilkan keputusan yang terbaik/bernilai maksimal karena algoritma *minimax* ini bekerja secara rekursif dengan mencari langkah yang akan membuat lawan mengalami kerugian minimum.

2. Storyboard

Storyboard/Papan cerita adalah sketsa awal yang digunakan sebagai alat perencanaan untuk menunjukkan bagaimana cerita akan terungkap. *Game Tic Tac Toe* sebelum di buat akan menggunakan *storyboard* agar peneliti memiliki pandangan seperti apa *game* yang akan dibuat, *storyboard* akan membantu proses coding atau pembuatan aplikasi karena sudah ada

bayangan *game* tersebut. Papan cerita/*storyboard Game Tic Tac Toe* dibuat sederhana dengan 3 mode karena penelitian ini hanya mencoba algoritma *minimax* pada *game Tic Tac Toe 4 x 4* yang biasanya pada papan/board 3 x 3, dengan tampilan pada gambar di tabel 1.

Table 1. *Storyboard*

Gambar	Penjelasan
	<p>Gambar disamping merupakan papan permainan ketika <i>Player</i> (manusia (x) & komputer (o)) memulai permainan. Pada <i>game</i> ini akan ada 3 tahap/mode yaitu bermain dengan sesama manusia, bermain dengan komputer mode <i>random</i> dan mode menggunakan <i>minimax</i>.</p> <p>Tombol g untuk mengganti mode dari <i>pvp</i>/bermain dengan manusia menjadi melawan komputer. Pada saat <i>Player</i> ingin bermain dengan komputer ada 2 mode yaitu Mode <i>random</i> dengan menekan tombol 0 pada <i>keyboard</i> dimana komputer dapat dikalahkan karena komputer hanya acak dalam menaruh tanda X atau O, sedangkan untuk Mode <i>Minimax</i> menekan tombol 1 pada <i>keyboard</i>, mode ini membuat komputer tidak akan pernah kalah pasti akan menang atau seri.</p>

3. Pengambilan Data

Penelitian ini mengambil data berupa waktu yang diperlukan komputer dalam menghitung satu langkah pada *game tic tac toe*, yaitu untuk mengisi tanda silang atau lingkaran ketika komputer mendapat giliran jalan. Data diambil baik untuk permainan papan 3 x 3 maupun 4 x 4. Dapat diduga bahwa ketika kotak kosong masih banyak akan diperlukan waktu yang lebih lama dalam menentukan langkah terbaik. Ketika kotak kosong tinggal satu, tidak diperlukan perhitungan algoritma untuk mengisi kotak tersebut. Penentuan waktu dalam menjalankan satu langkah *minimax* dilakukan dengan menyisipkan kode tertentu pada fungsi `eval()` dan proses tersebut memanfaatkan modul `time` di Python seperti tampak pada Gambar 1.

```
167 # --- MAIN EVAL ---
168
169 def eval(self, main_board):
170     if self.level == 0:
171         # random choice
172         eval = 'random'
173         move = self.rnd(main_board)
174     else:
175         # minimax algo choice
176         # Cara menghitung waktu eksekusi Minimax
177         startTime = time.perf_counter()
178         eval, move = self.minimax(main_board, False)
179         endTime = time.perf_counter()
180
181         print('Total waktu eksekusi program minimax : ', endTime - startTime)
182
183         print(f'AI has chosen to mark the square in pos {move} with an eval of: {eval}')
184
185     return move # row, col
```

Gambar 1. Kode Menghitung Eksekusi *Minimax*

Kode yang dimaksud yaitu instruksi pada baris 177 sampai 179. Pada baris 177 disimpan waktu awal proses menggunakan `perf_counter()`, baris 178 adalah penggunaan algoritma

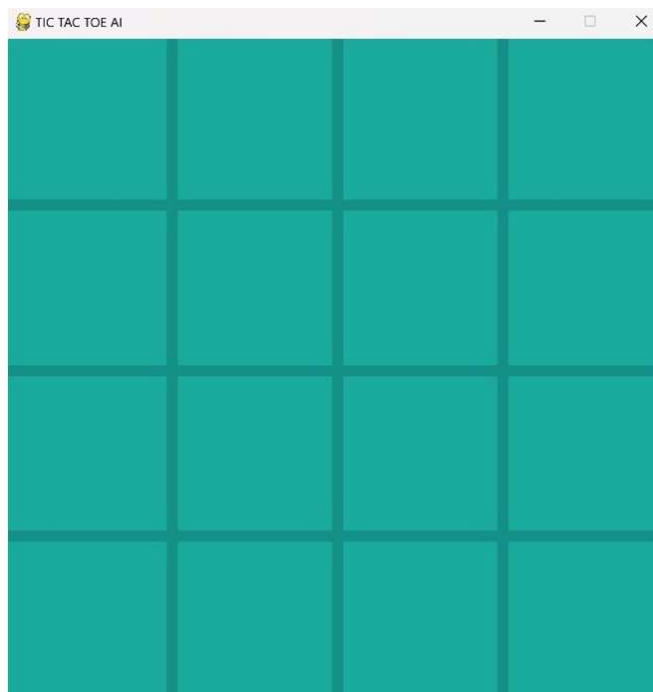
minimax, dan baris 179 menyimpan waktu akhir proses juga dengan *perf_counter()*. Selanjutnya di baris 181 ditampilkan rentang waktu eksekusi. Untuk laptop yang digunakan untuk melakukan pengujian memiliki spesifikasi sebagai berikut :

- *Prosesor* : 12th Gen Intel® Core™ i5-12500H 2.50 GHz
- *RAM* : 16 GB
- *Sistem Operasi* : Windows 11 Home Single Language
- *Storage* : 512 GB M.2 NVMe™ Pcle 3.0 SSD
- *Display* : 14-inch, 2.8K (2880 x 1800) OLED
- *Graphics* : Intel® Iris Xe Graphics
- *Memory* : 8 GB DDR4 on board + 8 GB DDR4 SO-DIMM

Hasil dan Pembahasan

Game Tic Tac Toe 4 x 4 ini dirancang menggunakan bahasa python dengan modul pygame [12]. Pada game ini ada 3 mode permainan yaitu manusia vs manusia, manusia vs komputer yang memilih secara random, dan manusia vs komputer dengan menggunakan algoritma *minimax*. Game Tic Tac Toe 4 x 4 ini dibuat untuk diambil data waktu pencarian langkah komputer dan dibandingkan dengan game Tic Tac Toe 3 x 3.

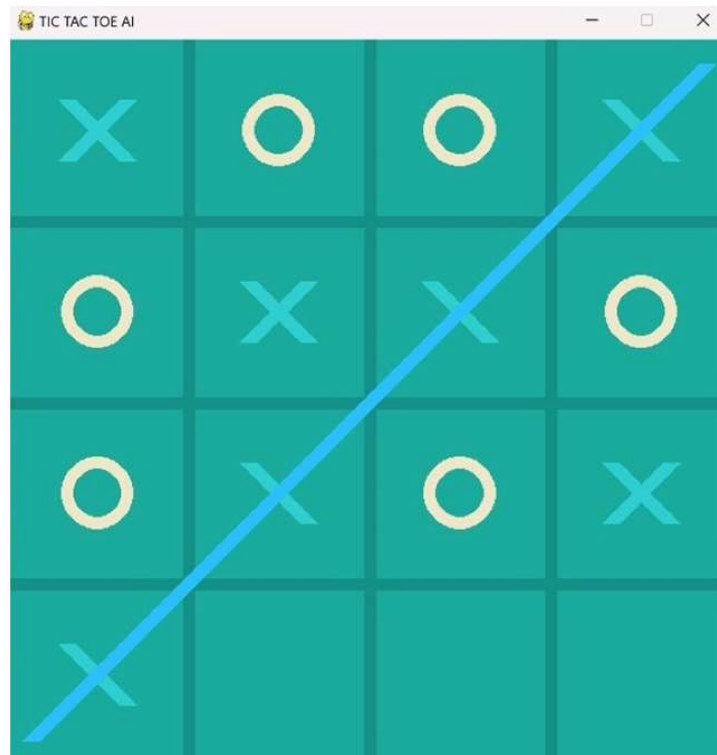
Tombol-tombol yang dapat digunakan dalam *game* ini ada beberapa misalnya untuk mereset *game* ada tombol R pada keyboard, untuk merubah mode melawan komputer ada tombol G serta jika ingin melawan komputer dengan mode random tekan tombol 0, jika ingin melawan komputer dengan kecerdasan buatan/algoritma *minimax* tekan tombol 1 [13]. *Game* ini dapat dijadikan bahan percobaan atau pengujian performa algoritma *minimax* karena diterapkan pada papan permainan yang berbeda yaitu 4 x 4, yang biasanya hanya 3 x 3. Gambar 2 menunjukkan game ketika dijalankan.



Gambar 2. Papan Tic Tac Toe 4 x 4

Proses mengecek kemenangan adalah dengan mencari satu deret baik vertikal, horizontal,

dan diagonal. Pada game ini ada 2 gambar untuk mewakili *player* (manusia dan komputer) yaitu X (*cross*) dan O (*circle*), apabila manusia dan komputer dapat membuat satu deret maka akan memenangkan permainan. *Game* di atas terbentuk ketika program dijalankan terdapat logo *pygame* pada *title bar* seperti ular *python*, karena program dibuat dengan modul *pygame*. Gambar 3 menunjukkan kemenangan manusia secara diagonal untuk vertikal dan horizontal sama jika mendapatkan satu deret dengan gambar yang sama maka akan terbentuk garis.



Gambar 3. Papan 4 x 4 Ketika Menang

Jika ingin membersihkan papan/mereset maka tekan tombol R, setelah menekan tombol papan akan bersih seperti Gambar 2, dan *game* siap dimulai lagi bisa manusia vs manusia atau manusia vs komputer. Algoritma *minimax* yang digunakan pada *game Tic Tac Toe 4 x 4* ini sama dengan yang diterapkan pada *game Tic Tac Toe 3 x 3* karena ingin mengetahui penerapan algoritma pada papan permainan/*board* yang berbeda.

1. Implementasi minimax

Kode program untuk Algoritma *Minimax* terlihat pada Gambar 4 dan Gambar 5.


```
114
115     # --- MINIMAX ---
116
117     def minimax(self, board, maximizing):
118
119         # terminal case
120         case = board.final_state()
121
122         # player 1 wins
123         if case == 1:
124             return 1, None # eval, move
125
126         # player 2 wins
127         if case == 2:
128             return -1, None
129
130         # draw
131         elif board.isfull():
132             return 0, None
133
134         if maximizing:
135             max_eval = -100
136             best_move = None
137             empty_sqr = board.get_empty_sqr()
138
139             for (row, col) in empty_sqr:
140                 temp_board = copy.deepcopy(board)
141                 temp_board.mark_sqr(row, col, 1)
142                 """Untuk mengecek data"""
143                 # print(self.n, temp_board.squares)
144                 # self.n += 1
145                 eval = self.minimax(temp_board, False)[0]
146                 if eval > max_eval:
147                     max_eval = eval
148                     best_move = (row, col)
149
150             return max_eval, best_move
151
152         elif not maximizing:
```

Gambar 4. Kode Program Minimax

```
127         if case == 2:
128             return -1, None
129
130         # draw
131         elif board.isfull():
132             return 0, None
133
134         if maximizing:
135             max_eval = -100
136             best_move = None
137             empty_sqr = board.get_empty_sqr()
138
139             for (row, col) in empty_sqr:
140                 temp_board = copy.deepcopy(board)
141                 temp_board.mark_sqr(row, col, 1)
142                 """Untuk mengecek data"""
143                 # print(self.n, temp_board.squares)
144                 # self.n += 1
145                 eval = self.minimax(temp_board, False)[0]
146                 if eval > max_eval:
147                     max_eval = eval
148                     best_move = (row, col)
149
150             return max_eval, best_move
151
152         elif not maximizing:
153             min_eval = 100
154             best_move = None
155             empty_sqr = board.get_empty_sqr()
156
157             for (row, col) in empty_sqr:
158                 temp_board = copy.deepcopy(board)
159                 temp_board.mark_sqr(row, col, self.player)
160                 eval = self.minimax(temp_board, True)[0]
161                 if eval < min_eval:
162                     min_eval = eval
163                     best_move = (row, col)
164
165             return min_eval, best_move
```

Gambar 5. Lanjutan Kode Program Minimax

Baris 117 merupakan awal pembuatan fungsi *Minimax* dengan perintah `def Minimax` dengan parameter *self*, *board*, *maximizing*. Parameter *board* untuk merepresentasikan *board* atau papan *Tic Tac Toe* 4 x 4 ditambah pada baris 120 ada kode program untuk mengecek kemenangan terdapat di fungsi *final_state*(). *Player* 1 (manusia) menang jika menghasilkan nilai evaluasi 1, sedangkan *player* 2 (komputer) akan menghasilkan nilai -1 dan jika komputer belum mendapatkan kemenangan maka akan menghasilkan nilai 0 kode ini terdapat di baris 122 sampai 132. Parameter *maximizing* digunakan untuk mencari nilai maksimum dan minimum yang menjadi acuan langkah komputer, jika *maximizing* maka nilai evaluasi lebih

besar dari max_eval sedangkan jika tidak maximizing nilai evaluasi lebih kecil dari min_eval .

Baris ke 130 sampai 132 merupakan kode apabila semua kotak terisi maka akan *draw* dan permainan akan berakhir dan untuk memulai lagi harus menekan tombol R untuk reset. Kode selanjutnya membuat pernyataan jika termasuk *maximizing* atau *not maximizing* dengan menggunakan fungsi *eval* untuk mengevaluasi pilihan komputer dalam memilih langkah. Pada baris ke 134 sampai 150 merupakan kode jika terjadi maximizing, terdapat beberapa *variable* seperti max_eval , $best_move$, dan $empty_sqrs$ yang digunakan untuk kode program di bawahnya, $empty_sqrs$ berisi kode untuk mengetahui bahwa *board*/papan permainan masih kosong.

Baris 139 merupakan kode pengulangan/pengecekan baris dan kolom pada $empty_sqrs$ yang terdapat *variable* $temp_board$ yang menyimpan papan/*board* imajinasi dengan menggunakan modul copy, selanjutnya membuat *variable* $eval$ yang berisi pemanggilan *minimax* dengan parameter papan/*board* imajinasi bernilai *False*. Jika nilai $eval$ lebih besar dari max_eval maka max_eval sama dengan $eval$ dan $best_move$ sama dengan baris dan kolom, langkah terakhir kembalikan nilai max_eval dan $best_move$ untuk komputer dapat memilih langkah. Kode untuk *not maximizing* hampir sama dengan *maximizing*, perbedaannya ada di nama *variable* min_eval , nilai papan/*board* imajinasi bernilai *True* dan nilai $eval$ lebih kecil dari min_eval

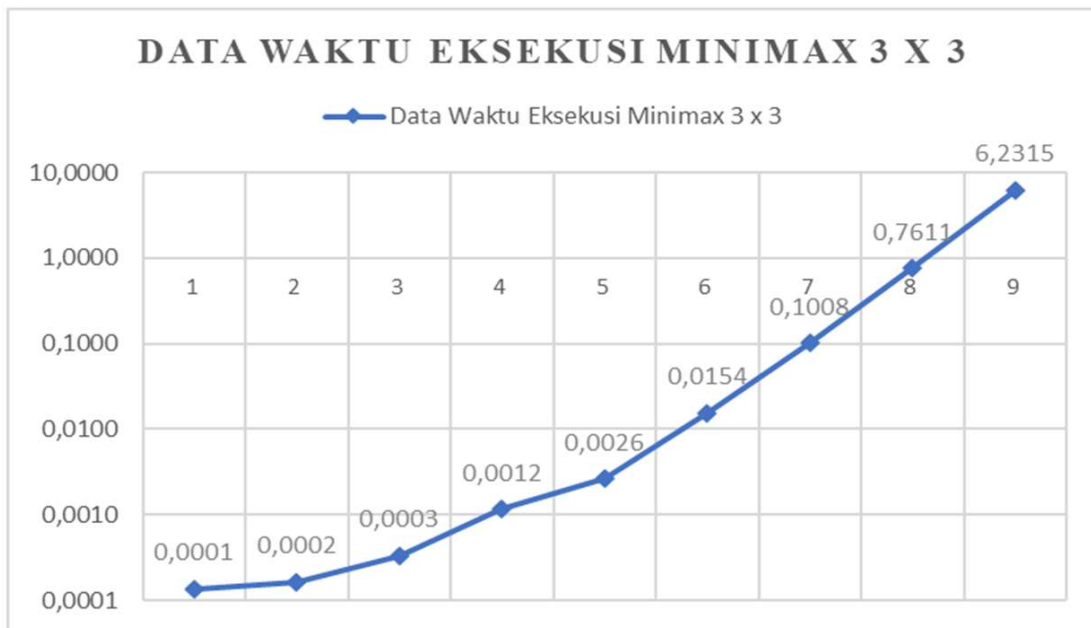
2. Pengujian

Kinerja algoritma *minimax* diuji dengan melihat waktu eksekusi algoritma untuk menjalankan satu langkah pada permainan *tic tac toe*.

Table 2. Data Waktu Eksekusi Minimax 3 x 3

Jumlah Kotak Kosong	Jumlah Percobaan Data Waktu Minimax Di Titik Berbeda (Detik)					Rata - Rata Data Waktu
	1	2	3	4	5	
1	0,0001281	0,0001213	0,0001481	0,0001751	0,000114	0,0001
2	0,0002799	0,0001248	0,0001696	0,0001246	0,000126	0,0002
3	0,0002812	0,0002938	0,0003451	0,0004602	0,0002766	0,0003
4	0,0008689	0,0009575	0,0023146	0,0008158	0,0009054	0,0012
5	0,0023911	0,0024172	0,0023204	0,0033052	0,0027079	0,0026
6	0,0108053	0,0126877	0,0209706	0,0194441	0,0132192	0,0154
7	0,0808614	0,1013003	0,1229037	0,1015065	0,0975181	0,1008
8	0,7556259	0,7948625	0,7342209	0,8368133	0,6840023	0,7611
9	6,6227222	5,6913418	6,761632	6,5179045	5,5637297	6,2315

Tabel 2, menunjukkan data waktu eksekusi algoritma *minimax* pada papan permainan (*board*) 3 x 3. Waktu eksekusi nyaris nol untuk mengisi satu kotak di antara satu atau dua kotak kosong. Waktu terlama diperlukan untuk mengisi salah satu kotak ketika masih ada 9 kotak kosong yaitu sebesar 6,2315 detik. Waktu sebesar ini diperlukan jika algoritma *minimax* mencari sendiri kotak terbaik di antara 9 kotak kosong yang ada. Jika waktu eksekusi tersebut digambarkan dalam grafik logaritmis, terlihat kurva seperti tampak pada Gambar 6.



Gambar 6. Grafik Data Waktu Eksekusi Minimax 3 x 3

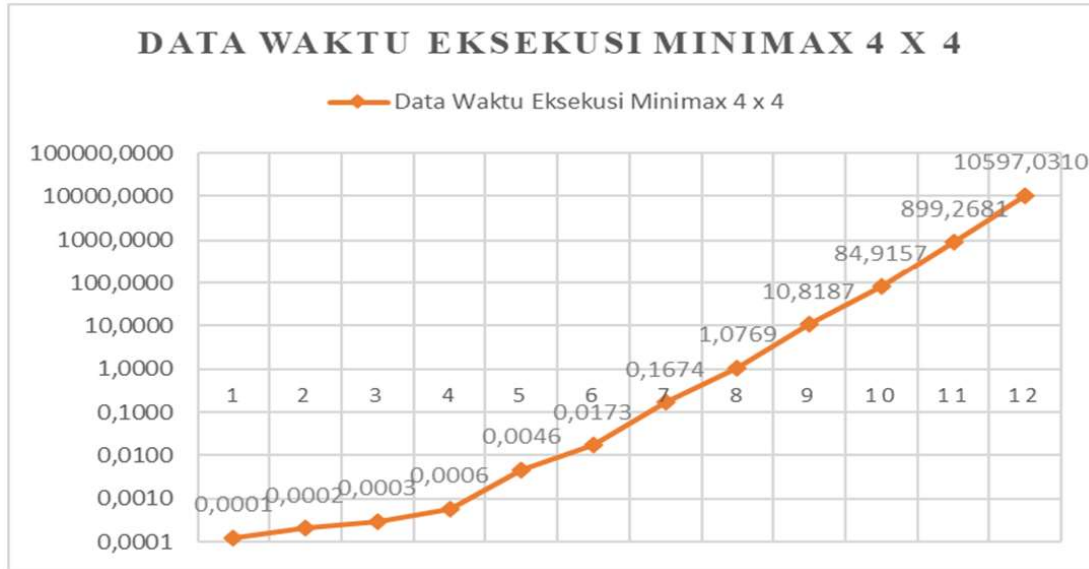
Table 3. Data Waktu Eksekusi Minimax 4 x 4

Jumlah Kotak Kosong	Data Waktu Eksekusi Minimax 4 x 4
1	0,0001
2	0,0002
3	0,0003
4	0,0006
5	0,0046
6	0,0173
7	0,1674
8	1,0769
9	10,8187
10	84,9157
11	899,2681
12	10597,0310

Data pada tabel 3, menunjukkan bahwa waktu eksekusi semakin meningkat ketika kotak kosong semakin banyak, hal ini karena algoritma minimax harus menguji setiap kemungkinan kotak yang ada. Ketika kotak kosong berjumlah 13, algoritma minimax memerlukan waktu lebih dari 10597 detik atau lebih dari 3 jam. Tentunya tidak ada yang mau menunggu komputer selama 3 jam untuk menjalankan satu langkah, hal ini sangat tidak cocok untuk sebuah game karena akan mengakibatkan rasa bosan dan tidak akan mau memainkan game tersebut kembali.

Kotak kosong berjumlah 13-16 tidak dilakukan pengujian karena waktu eksekusi yang diperlukan sangat lama. Berdasarkan hasil pengujian minimax ini dapat diketahui bahwa algoritma minimax hanya cocok dengan jumlah papan permainan (board) 3 x 3 dan tidak dapat diterapkan pada papan permainan 4 x 4 sebab terlalu lama menunggu komputer mencari langkah, mungkin dapat diterapkan namun dengan bantuan algoritma lain atau perubahan kode cara pengambilan langkah komputer bisa dengan komputer memilih dengan random

terlebih dahulu setelah itu menggunakan minimax agar game dapat dimainkan.



Gambar 7. Grafik Data Waktu Eksekusi Minimax

Gambar 7, merupakan grafik yang menampilkan data waktu eksekusi algoritma *minimax* ketika diteapkan pada papan 4 x 4. Gambar disajikan secara *logaritmis* agar data terlihat jelas karena kenaikan waktu eksekusi setiap pertambahan kotak kosong sangat besar. Bentuk grafik menunjukkan bahwa semakin banyak jumlah kotak kosong maka waktu semakin lama, sebaliknya semakin sedikit kotak kosong maka semakin cepat waktu untuk komputer memilih langkah.

3. Analisis Data

Pengujian algoritma *minimax* ini peneliti mencoba untuk mengisi beberapa kotak misalnya tinggal 1 kotak kosong kemudian lanjut 2 kotak kosong dan seterusnya, hal ini dikarenakan apabila langsung menguji dengan 15 kotak kosong maka komputer/laptop tidak akan kuat karena *resource* terlalu besar dan akan terjadi ledakan waktu karena algoritma *minimax* bekerja dengan mencari segala kemungkinan kotak yang ada [14]. Hasil dari pengujian *minimax* ini menunjukkan bahawa untuk 13 kotak kosong komputer sudah menghabiskan waktu yang lama dalam pencarian langkah terbaik, apabila pengujian dilakukan dari 15 kotak kosong maka waktu eksekusi algoritma *minimax* akan lebih lama karena semakin banyak kotak kosong maka komputer akan lebih lama dalam mencari langkah terbaik sebab masih banyak kemungkinan.

Data waktu eksekusi langkah komputer pada papan 4 x 4 dengan menggunakan algoritma *minimax* akan dibandingkan dengan data waktu eksekusi langkah komputer pada papan 3 x 3, dari percobaan ini akan menunjukkan kinerja algoritma *minimax* apakah menjadi lebih efektif dalam pencarian langkah komputer atau akan menyebabkan ledakan waktu yang lama dalam pencarian komputer karena jumlah kotak berbeda [11]. Performa algoritma *minimax* untuk papan permainan lebih dari 3 x 3 sangat bagus karena hanya membutuhkan waktu 6 detik dalam pencarian langkah pertama komputer sedangkan papan permainan 4 x 4 dalam pencarian langkah komputer untuk 12 kotak kosong membutuhkan waktu sampai sekitar 3 jam, hal ini dapat dijadikan acuan bahwa pencarian pertama dengan 13 kotak kosong, 14 kotak kosong dan 15 kotak kosong akan lebih lama dari 3 jam karena masih banyak kotak kosong/ banyak kemungkinan. Analisis data waktu ini berguna untuk menghitung kinerja algoritma *minimax* apakah cocok dengan papan permainan 4 x 4. Hasil dari pengujian ini menunjukkan bahwa algoritma *minimax* tidak cocok dengan papan permainan 4 x 4 karena untuk 12 kotak

kosong perlu waktu sekitar 3 jam dalam pengambilan langkah komputer, ini belum 15 kotak kosong pasti waktunya akan lebih dari 3 jam mungkin dapat di bantu algoritma lain atau dibuat komputer cari secara *random* setelah itu menggunakan *minimax* [15].

Simpulan

Setelah menerapkan Algoritma Minimax pada Game Tic Tac Toe 4 x 4, maka didapatkan kesimpulan yang pertama terciptanya Game Tic Tac Toe 4 x 4 dan 3 x 3 yang lebih menantang dengan memiliki 3 mode yaitu manusia vs manusia, manusia vs komputer yang memilih langkah secara random dan manusia vs komputer dengan algoritma minimax. Kesimpulan kedua yaitu pada level/fase komputer memilih langkah secara random maka dapat dikalahkan sedangkan jika komputer menerapkan algoritma minimax maka komputer tidak akan kalah dan hasil maksimal yang didapatkan manusia adalah seri. Kesimpulan kedua yaitu penerapan Algoritma Minimax pada Game Tic Tac Toe 4 x 4 merupakan hal yang kurang efektif karena dalam pencarian langkah komputer memerlukan waktu yang lama bahkan sampai 5 jam lebih untuk 15 kotak kosong namun untuk penerapan Algoritma Minimax pada Game Tic Tac Toe 3 x 3 sangat efektif karena hanya butuh 6 detik untuk melakukan langkah pertama. Kesimpulan keempat yaitu Algoritma Minimax bergantung kepada banyaknya kotak kosong apabila kotak kosong semakin banyak maka waktu yang diperlukan semakin lama contoh kasus untuk papan permainan/board 4 x 4

Daftar Pustaka

- [1] I. Rahmawati, T. S.-J. M. P. DAN, and U. 2021, "Penelitian Eksperimen Siswa Kelas Delapan SMP PGRI Kalimulya: Keuntungan Memanfaatkan Situs Game Pembelajaran Sebagai Media," *dinastirev.org*, vol. 2, no. 2, 2021, doi: 10.38035/jmpis.v2i2.
- [2] E. Sudarmilah and A. F. B. Arbain, "Using gamification to stimulate the cognitive ability of preschoolers," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 6, pp. 1250-1256, 2019.
- [3] E. Novrialdy, B. dan Konseling, and F. Ilmu Pendidikan, "Kecanduan game online pada remaja: Dampak dan pencegahannya," *core.ac.uk*, vol. 27, no. 2, pp. 148-158, 2019, doi: 10.22146/buletinpsikologi.47402.
- [4] A. Christopher, ... D. D.-J. T., and U. 2020, "Penerapan Algoritma Minimax Terhadap Permainan Tic-Tac-Toe Dengan Menggunakan Artificial Intelligence," *download.garuda.kemdikbud.go.id*, vol. 2020, no. 9, pp. 2657-1501, 2020.
- [5] N. Putra Sijabat, M. Alvin Riad, J. Sutoyo Muda Lumbantobing, and D. Batara Sanjaya, "Analisa Efektivitas Algoritma Minimax, Alpha Beta Pruning, dan Negamax dalam Penerapannya pada Permainan Papan (Board Game)," *ejournal.sisfokomtek.org*, vol. 3, no. 2, pp. 49-59, 2020.
- [6] M. Kurniawan, A. Pamungkas, S. H.-S. Online, and undefined 2016, "Algoritma Minimax Sebagai Pengambil Keputusan Dalam Game Tic-Tac-Toe," *ojs.amikom.ac.id*, pp. 6-7, 2016.
- [7] J. Setiawan *et al.*, "Desain Non-Player Character Permainan TIC-TAC-TOE dengan Algoritma Minimax," *eprints.ukmc.ac.id*, 2019, doi: 10.33557/jurnalmatrik.v21i3.715.
- [8] D. Perez-Liebana, S. Samothrakis, J. Togelius, S. M. Lucas, and T. Schaul, "General video game ai: Competition, challenges and opportunities," *ojs.aaai.org*, doi: 10.1109/TCIAIG.2015.2402393.
- [9] I. Lakhmani and V. Punjabi, "A Game Based Implementation of Minimax Algorithm Using AI Agents," vol. 6, no. 4, pp. 2662-2665, 2020.
- [10] T. Rončević, M. Rodić, and L. Despalatović, "Minimax and Monte Carlo Tree Search Implementations for Two- Player Game."
- [11] G. Emanuel, R. K. J. Bendi, and A. Arieffianto, "Desain Non-Player Character Permainan Tic-Tac-Toe Dengan Algoritma Minimax," *J. Ilm. Matrik*, vol. 21, no. 3, pp. 223-233, 2019, doi: 10.33557/jurnalmatrik.v21i3.725.
- [12] S. Kelly, "Python, PyGame and raspberry Pi game development," 2019, doi: 10.1007/978-1-4842-4533-0.

-
- [13] I. Uari, A. Muhazir, ... H. A.-S. N. T., and U. 2021, "Analisi Kecerdasan Buatan Pada Permainan Checker Menggunakan Optimasi Algoritma Minimax," *jurnal.uisu.ac.id*, 2021.
 - [14] H. Thamrin, "Efektivitas Algoritma Semantik dengan Keterkaitan Kata dalam Mengukur Kemiripan Teks Bahasa Indonesia," *Khazanah Inform. J. Ilmu Komput. dan Inform.*, vol. 1, no. 1, pp. 7-11, 2015, doi: 10.23917/khif.v1i1.1174.
 - [15] K. I. Santoso, F. Yunita, and N. P. Kusumo, "Penerapan Algoritma Minimax Pada Game Macan-macanan," *J. Sist. Inf. Bisnis*, vol. 6, no. 1, p. 21, 2016, doi: 10.21456/vol6iss1pp21-29.