ISSN : 1411-1624
e-ISSN: 2621-8089

IMPLEMENTASI ALGORITMA *MONTE CARLO TREE SEARCH* PADA PERMAINAN 2048 BERBASIS WEB

Alvin Kennedy[1], Nur Fanny Savira[2], Stefanus Andree Gunawan[3], Lukman Hakim[4]
Mahasiswa Universitas Bunda Mulia[1], Dosen Universitas Bunda Mulia[2,3,4]
Jalan Lodan Raya, RT.12/RW.2, Ancol, Kec. Pademangan, Kota Jakarta Utara, Daerah Khusus Ibukota Jakarta
Sur-el : alvinkennedy45@gmail.com[1], savirafanny@gmail.com[2], stefanusandreegunawan@gmail.com[3], lhakim@bundamulia.ac.id[4]

Abstract : *Artificial Intelligence is a new inovation in modern computer science that has been use in playing game. Game is very interested by many people, there is someone who make artificial intelligence to play a puzzle game that is 2048 that can be played automatically by computer using Monte Carlo Tree Search Algorithm through web-browser. Monte Carlo Tree Search is computation algorithm that using random sample. The website was design using flowchart and UML also made by using JavaScript for UX with the help of HTML for UI that can be played by AI with value runs input by user and doing test for comparing the difference result between each value runs by testing 5 times. Monte Carlo Tree Search Algorithm can be applied in playing game 2048.*

Keywords: *AI, Monte Carlo Tree Search, 2048, Game, Artificial Intelligence.*

Abstrak : *Kecerdasan Buatan adalah sebuah inovasi baru dalam ilmu pengetahuan komputer modern yang digunakan untuk bermain game. Game sangat diminati oleh banyak orang, ada juga seseorang yang membuat kecerdasan buatan untuk memainkan game puzzle yaitu 2048 yang dapat dimainkan secara otomatis oleh komputer dengan algoritma Monte Carlo Tree Search melalui web. Monte Carlo Tree Search merupakan algoritma komputasi yang menggunakan sampel acak. Website yang dirancang menggunakan flowchart dan UML serta dibuat menggunakan JavaScript sebagai UX dengan bantuan HTML sebagai UI yang dapat dimainkan oleh AI dengan nilai runsnya dimasukkan oleh user dan melakukan pengujian untuk membandingkan hasil perbedaan antar setiap nilai runs dengan pengujian sebanyak 5 kali. Algoritma Monte Carlo Tree Search dapat diterapkan dalam permainan game 2048.*

Kata kunci*: AI, Monte Carlo Tree Search, 2048, Game, Kecerdasan Buatan.*

1. PENDAHULUAN

Kecerdasan buatan (*Artificial Intelligence*) merupakan suatu inovasi baru dalam ilmu pengetahuan. Adanya kecerdasan buatan dimulai sejak munculnya komputer modern pada tahun 1940 dan tahun 1950 [1]. *AI* merupakan alat atau komputer yang dapat melakukan tugas yang dilakukan oleh manusia, oleh karena itu *AI* juga dapat dimanfaatkan dalam kegiatan bermain *game* [2].

Game saat ini sangat diminati oleh sebagian besar orang karena *gameplay* yang menarik. Terkadang, ada juga yang membuat kecerdasan buatan untuk memainkan suatu *game*. Kecerdasan buatan tersebut dibuat dengan tujuan agar *game* dapat diselesaikan secara otomatis / dimainkan oleh komputer. Penerapan kecerdasan buatan tersebut akan diterapkan pada game 2048, dimana komputer akan menyelesaikan *game* dengan kecerdasan yang telah diprogram yaitu *Monte Carlo*.

ISSN : 1411-1624
e-ISSN: 2621-8089

Monte Carlo merupakan algoritma komputasi yang dapat menggunakan sampel acak dalam menyelesaikan masalah. *Monte Carlo* dapat digunakan untuk menyelesaikan masalah pada beberapa bidang seperti fisika, *game,* matematika, dan bidang lainnya [3]. *Monte Carlo Tree Search* merupakan teknik pencarian *AI* yang digunakan secara luas, MCTS bisa dengan cepat menemukan langkah yang bagus dalam proses mengambil keputusan sekuensial yang besar dan kompleks dengan mengkombinasikan *search tree* tradisional dengan evaluasi *node* berdasarkan simulasi stokastik [4].

2. METODOLOGI PENELITIAN

Metode Pengembangan sistem yang penulis gunakan pada penelitian ini yaitu metode *extreme programing.*

2.1 Game

Game adalah suatu sistem dimana pemain mengambil keputusan melalui objek yang dikendalikan didalam permainan yang memiliki satu atau banyak objektif. Ada beberapa jenis permainan yang sudah beredar saat ini yaitu: *Role-Playing Game*, *First-Person Shooter*, *Puzzle Game*, *Adventure Game, Racing Game* [5].

2.2 *Game* 2048

Game 2048 dimainkan pada kotak berukuran 4x4. Tujuan dari *game* 2048 adalah untuk mencapai nilai 2048 dengan memindahkan dan menyatukan *tiles* yang terdapat pada *board* 4x4. 2 *tiles* secara acak berisi angka 2 atau 4. Pemain dapat menggerakkan angka tersebut (keatas, kebawah, kekanan, dan kekiri). Ketika 2 *tiles* dengan angka yang sama bertabrakan, maka *tiles* tersebut akan menambahkan nilainya dan nilai tersebut adalah skor pemain [6].

2.3 *AI*

AI merupakan sains yang digunakan untuk mengimitasi kepintaran mahkluk hidup untuk diterapkan kepada mesin dengan tujuan pemecahan masalah. *AI* memiliki beberapa metode yang dapat digunakan diantaranya *Fuzzy Logic, Evolutionary Computing, Machine Learning,* dalam penelitian ini penulis menggunakan metode *Monte Carlo Tree Search* [7].

2.4 *Monte Carlo Tree Search*

Monte Carlo Tree Search menggunakan *Monte Carlo rollouts* untuk mengestimasikan isi dari setiap bagian dari *search tree.* Semakin banyak simulasi yang dilakukan, *search tree* bertumbuh besar maka isi yang hasilkan lebih relevan [8]. *Monte Carlo Tree Search* adalah metode pencarian yang mengkombinasikan ketepatan pencarian *tree* dengan keumuman sampling acak [9].

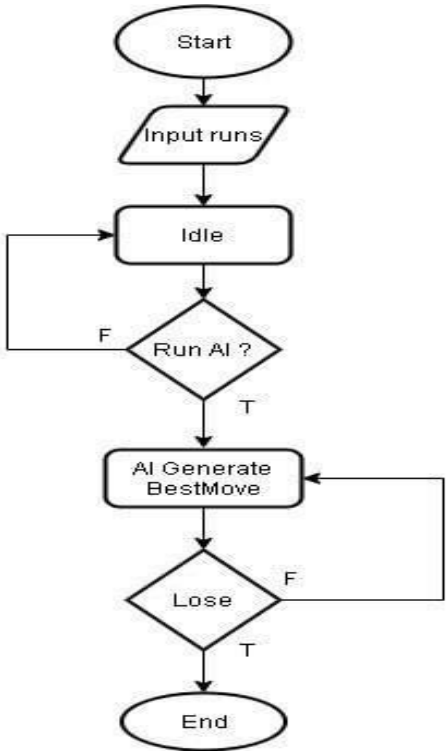
2.5 JavaScript

JavaScript merupakan Bahasa pemrograman yang digunakan untuk menyempurnakan tampilan sistem *web-based application* yang dikembangkan, berupa sekumpulan *script* yang berjalan pada dokumen HTML [10].

ISSN : 1411-1624
e-ISSN: 2621-8089

3. HASIL DAN PEMBAHASAN

Cara kerja sistem yang digambarkan pada *flowchart* pada gambar 1. adalah *user* menjalankan program setelah itu *user* memasukkan *input* berupa suatu angka, lalu program akan *idle* jika *user* tidak menekan tombol *run*, jika *user* menekan tombol *run* maka sistem akan menghasilkan *move* secara otomatis. Jika belum kalah maka akan dilakukan *move* sampai kondisi *lose* adalah *true*, dan program akan selesai.

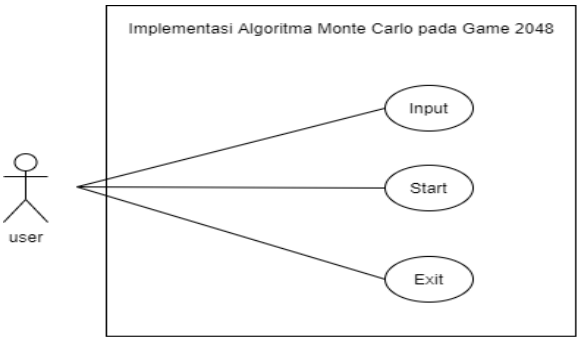


Gambar 1. *Flowchart* Program

Perancangan sistem yang dirancang penulis bertujuan untuk menggambarkan kondisi dan bagian-bagian yang berperan dalam sistem yang dirancang. Perancangan sistem dilakukan dengan menggunakan *use-case* diagram, *activity* diagram, *sequence diagram* dan *class* diagram.

1. *Use Case* Diagram

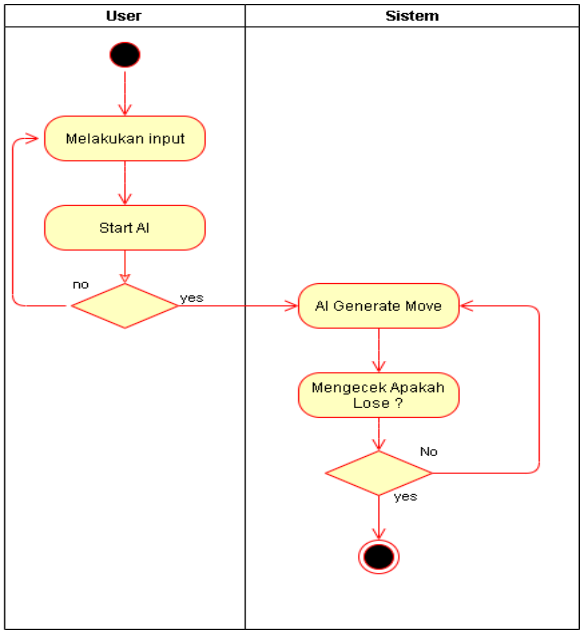
Ditunjukkan pada gambar 2. *User* dapat melakukan *Input* jumlah *runs*, *Start* program dan *Exit* untuk keluar dari program.



Gambar 2. *Use Case* Diagram

2. *Activity* Diagram

Ditunjukkan pada gambar 3. *User* melakukan *input* setelah itu jika *user* menekan tombol *Start AI* maka sistem akan melakukan *Generate Move*, jika tidak maka menunggu *user* sampai menekan tombol *Start*. Setelah itu sistem akan melakukan pengecekan apakah sudah kalah, jika sudah kalah maka sistem akan *end*, jika belum maka sistem akan melakukan *Generate Move*.

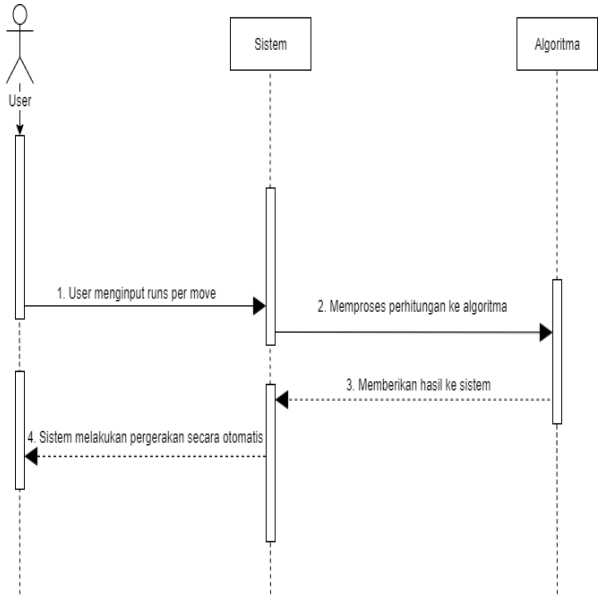


Gambar 3. *Activity* Diagram

ISSN : 1411-1624
e-ISSN: 2621-8089

3. *Sequence* Diagram

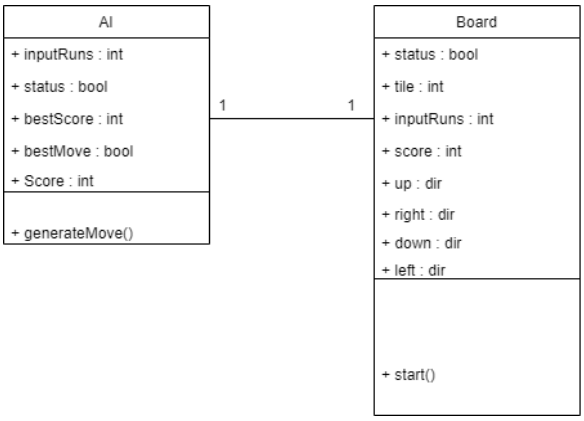
Ditunjukkan pada gambar 4. *User* melakukan *input*an nilai *runs* kepada sistem, setelah itu sistem akan mengirim suatu permintaan kepada algoritma untuk dilakukan proses perhitungan, jika algoritma sudah selesai melakukan perhitungan maka algoritma akan memberi respon balik ke sistem dengan memberikan hasil, dan setelah itu sistem akan melakukan pergerakan secara otomatis sesuai dengan nilai *input*an yang diterima oleh *user*.



Gambar 4. *Sequnce* Diagram

4. *Class* Diagram

Pada gambar 5. ditunjukkan bahwa ada kelas *AI* dimana merupakan kelas untuk melakukan proses *Generate Move*. Dimana kelas *AI* memiliki 3 *attribute* yaitu *InputRuns*, *Status* dan *Score* serta *method* yaitu *generateMove()*.



Gambar 5. *Class* Diagram

Fungsi *moveName* berguna untuk melakukan inisialisasi *move* berdasarkan nilai yang telah ditentukan.

```
function moveName(input move :
dir)
{menginisialisasi move berdasarkan
nilai yang ditentukan}

Deklarasi
Algoritma
return (0 <- 'up', 1 <- 'right', 2
<- 'down' , 3 <- 'left')[move]
```

Gambar 6. Fungsi moveName

Fungsi *AI_getBest* merupakan fungsi untuk memulai perhitungan proses algoritma *Monte Carlo* dan mendapatkan *move* terbaik berdasarkan perhitungan yang telah dilakukan.

```
Function AI_getBest(input grid :
grid)
{melakukan suatu pencarian fungsi
dan return best move}

Deklarasi
runs : integer

Algoritma
runs <-
document.getElementById('run-
count').value
return getBestMove(grid, runs)
```

Gambar 7. Fungsi *AI*_getBest

ISSN : 1411-1624
e-ISSN: 2621-8089

Fungsi *MonteSearch* berguna untuk melakukan pencarian mencari *bestMove* pada *tree* dan akan terus melakukan *looping* terhadap seluruh *move* yang sudah diinisialisasi dan setelah itu menentukan nilai tertinggi yang sudah didapatkan.

```
Function MonteSearch(input grid :
grid, runs : integer)
{melakukan proses pencarian untuk
menemukan bestMove}

Deklarasi
bestMove : integer
bestScore : double

Algoritma
for i <- 0 to 4 do
      res <- multiRandomRun(grid,
i, runs)
      score <- res.score
      if score >= bestScore then
            bestScore <- score
            bestMove <- i
      end if
End for
return move <- bestMove, score <-
bestScore
```

Gambar 8. Fungsi MonteSearch

Fungsi *multiRandomRun* berguna untuk melakukan suatu penambahan cabang *tree* pada algoritma *Monte Carlo* berdasarkan nilai *input* yang telah ditentukan oleh *user* yaitu *runs*. Dan masing-masing *move* akan dilakukan *looping* sebanyak *runs*.

```
Function multiRandomRun(input
grid : grid, move : integer, runs
: integer)
{melakukan proses pencarian untuk
mencari nilai bestScore}

Deklarasi
total : integer = 0
avg : double

Algortima
for i<-0 to runs then
      res <- randomRun(grid,move)
      s <- res.score
      if s == -1 then
            return -1
      end if
      total <- total + s
end for
avg <- total / runs
return score<-avg
```

Gambar 9. Fungsi multiRandomRun

Fungsi *randomRun* berguna untuk melakukan proses simulasi *tree* pada algoritma Monte Carlo. Dengan melakukan *random move* pada *move* yang sedang dijalankan serta *looping* pada papan yang telah dilakukan klon sampai tidak dapat melakukan *move* dan setelah itu akan melakukan *return score* untuk *update* nilai bestScore.

```
Function randomRun(input grid :
grid, move : integer)
{Melakukan proses random move
sampai tidak bisa move dan return
score tertinggi}

Deklarasi
g : grid = grid.clone()
score : integer = 0
moves : integer = 0

Algoritma
res <- moveAndAddRandomTiles(g,
move)
if !res.moved then
      return -1
end if
score <- score + res.score
while true do
      if !g.movesAvailable() then
            break
      end if

      rand <-
g.move(Math.floor(Math.random() *
4 ))
      if !rand.moved then
            continue
      end if
      score <- score + res.score
      g.addRandomTile()
      moves++
end while
return score
```

Gambar 10. Fungsi randomRun

ISSN : 1411-1624
e-ISSN: 2621-8089

Fungsi yang digunakan untuk melakukan proses *move* dan menambahkan *random tile* jika memungkinkan untuk melakukan pergerakan.

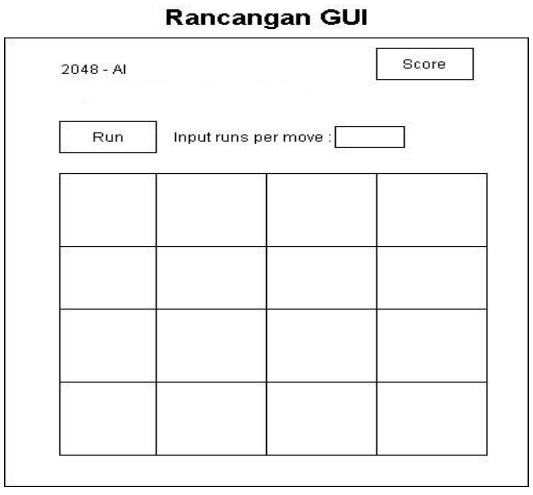
```
Function moveAndAddRandomTiles(input grid
: grid, direction : dir)
{melakukan proses move dan
menambah random tile}

Algoritma
res <- grid.move(direction)
if res.moved then
      grid.addRandomTile()
end if
return res
```

Gambar 11. Fungsi moveAndAdd

Berikut merupakan penjelasan dari gambar yang ada pada gambar 12 :

1. *Score* : merupakan tempat dimana skor akan ditampilkan.
2. *Run* : merupakan tombol yang berfungsi untuk menjalankan program.
3. *Input runs per move* : merupakan *input* yang akan diproses oleh algoritma sebagai total pergerakan yang akan dilakukan untuk setiap *move*.
4. *Tile* : merupakan tempat berupa kotak berukuran 4x4 yang berfungsi sebagai tempat untuk menjalankan permainan 2048.

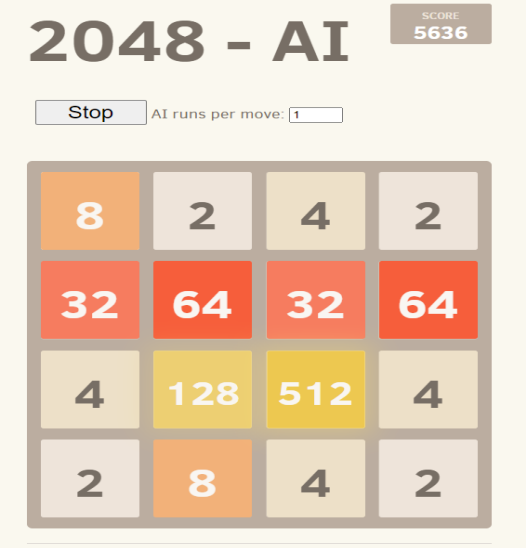


Gambar 12. Tampilan GUI

Pada gambar 13. Merupakan tampilan ketika pertama kali program dijalankan. *User* hanya perlu melakukan *input* berupa angka pada bagian sebelah kanan, lalu selanjutnya dapat dengan menekan tombol *run*. Lalu program akan memproses algoritma *monte carlo tree search* dan secara otomatis akan melakukan pergerakan sampai *button run* dihentikan atau permainan berakhir karena tidak dapat melakukan *move*.

Gambar 13. Tampilan Awal

Pada gambar 14, merupakan tampilan *game over*. Jika tidak terdapat gerakan yang dapat dilakukan maka tampilan hanya seperti itu.

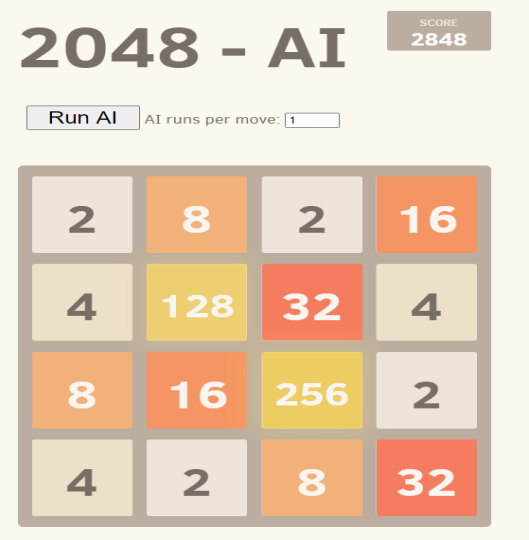


Gambar 14. Tampilan *GameOver*

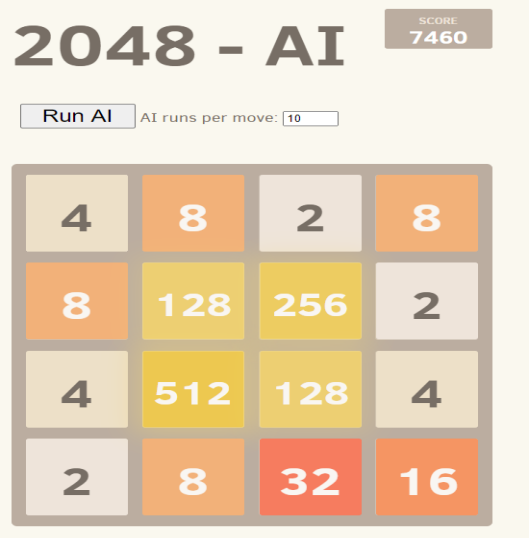
ISSN : 1411-1624
e-ISSN: 2621-8089

Pengujian dilakukan dengan cara membandingkan hasil dari perbedaan antara *input*an nilai *runs* dengan masing-masing diuji sebanyak 5 kali. Terdapat contoh nilai *runs* yang akan digunakan sebagai uji coba pada penelitian ini, yaitu :

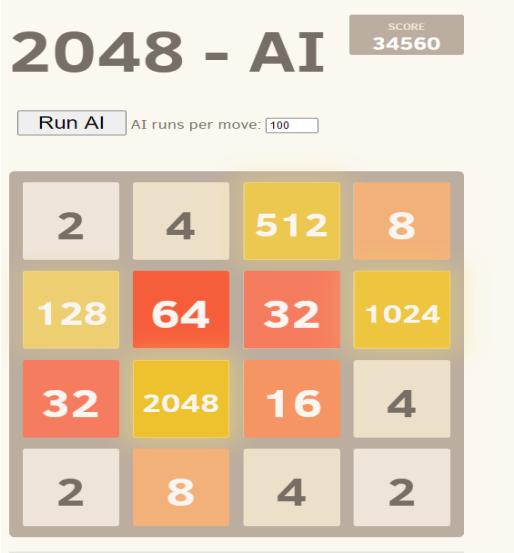
1. Nilai *runs* 1
2. Nilai *runs* 10
3. Nilai *runs* 100



Gambar 15. Percobaan 1



Gambar 16. Percobaan 6



Gambar 17. Percobaan 11

Pada tabel 1. Ditunjukkan bahwa terdapat 4 percobaan dengan nilai *runs* yang berbeda dan masing-masing percobaan dilakukan selama 5 kali. Didapati bahwa nilai rata-rata tersebut berbeda-beda, Terlihat jelas perbedaan antara masing-masing percobaan yaitu perbedaan *score*. Dan dapat dikatakan bahwa semakin besar nilai *runs* maka semakin besar juga nilai *score* yang bisa didapatkan.

Tabel 1. Analisis Hasil Percobaan

No	*Nilai Run*	*Score percobaan*					*Average score*
		1	*2*	*3*	*4*	*5*	
1	1	2848	6252	400	5012	5380	3978.4
2	10	7460	13544	16824	10520	15928	12855.2
3	100	34560	59720	51328	35408	56500	47503.2

4. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, diketahui bahwa algoritma *Monte Carlo Tree Search* dapat diterapkan pada permainan 2048 Dapat diterapkan dengan cara melakukan pengulangan sebanyak *input*an dari

ISSN : 1411-1624
e-ISSN: 2621-8089

nilai *runs* oleh *user* pada masing-masing *move*. Setiap *move* dilakukan pengecekan jika bisa bergerak maka akan melakukan *move* pada arah tersebut jika tidak maka akan dilewati dan akan mengecek *move* selanjutnya dan setelahnya akan dilakukan simulasi yaitu *move* secara acak sampai tidak dapat bergerak, setelah itu akan melakukan *return* nilai total yang didapat dari hasil simulasi tersebut lalu hasil akan dibandingkan untuk dijadikkan *move* yang akan menjadi pergerakan selanjutnya.

Cara kerja dari Algoritma *Monte Carlo Tree Search* pada permainan 2048 adalah dengan memilih *move* yang telah diinisialisasi dan akan dilakukan penelusuran cabang *tree* pada masing-masing *move* tersebut dan akan dilakukan pengecekan jika *move* tersebut dapat digerakkan. Jika dapat maka akan dilakukan simulasi yaitu melakukan pergerakan *move* secara *random* sampai tidak dapat melakukan *move* lagi sebanyak nilai yang telah di*input* oleh *user* dan setelah itu akan melakukan *backpropagation* ke cabang teratas dan akan *return* nilai total dari cabang tersebut. Lalu akan melakukan pengecekan sampai seluruh cabang telah diproses setelah itu baru akan ditentukan *move* berdasarkan cabang dengan nilai tertinggi.

DAFTAR PUSTAKA

[1] M. Yulfikar, U. Yudatama, and E. U. Artha, "REDIKSI KETERSEDIAAN STOK KAYU DENGAN METODE BACKPROPOGATION DAN JARINGAN KOHONEN (Studi Kasus Ud. Wahyu Nugroho Grabag Magelang)," *J. Komtika*, vol. 2, no. 2, pp. 115–124, 2019, doi: 10.31603/komtika.v2i2.2598.

[2] G. R. Meliani and A. Suryadi, "GAME ARTIFICIAL INTELEGENT: RAM CITY TOWER DENGAN ALGORITMA A*," *J. PETIK*, vol. 3, no. 2, pp. 31–38, May 2018, doi: 10.31980/jpetik.v3i2.148.

[3] B. Y. Geni, J. Santony, and Sumijan, "Prediksi Pendapatan Terbesar pada Penjualan Produk Cat dengan Menggunakan Metode Monte Carlo," *J. Inform. Ekon. Bisnis*, vol. 1, no. 4, pp. 15–20, Oct. 2019, doi: 10.37034/infeb.v1i4.5.

[4] P. Dong, H. Liu, and L. Xing, "Monte Carlo tree search -based non-coplanar trajectory design for station parameter optimized radiation therapy (SPORT)," *Phys. Med. Biol.*, vol. 63, no. 13, 2018, doi: 10.1088/1361-6560/aaca17.

[5] Y. Amrizal and R. Kurniati, "Game Aritmatika Berbasis Android," *INOVTEK Polbeng - Seri Inform.*, vol. 1, no. 2, p. 100, Nov. 2016, doi: 10.35314/isi.v1i2.121.

[6] N. Kondo and K. Matsuzaki, "Playing game 2048 with deep convolutional neural networks trained by supervised learning," *J. Inf. Process.*, vol. 27, no. Section 6, pp. 340–347, 2019, doi: 10.2197/ipsjjip.27.340.

[7] D. Putra and E. Triastuti, "Application of E-Learning and Artificial Intelligence in Education Systems in Indonesia," *Int. J. Comput. Appl.*, vol. 177, no. 27, pp. 16–22, Dec. 2019, doi: 10.5120/ijca2019919739.

[8] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and *tree search*," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016, doi: 10.1038/nature16961.

[9] C. Li, W. Xu, S. Huang, and L. Yang, "A Monte Carlo tree search-based method for online decision making of generator startup sequence considering hot start," *Int. J. Electr. Power Energy Syst.*, vol. 121, no. April, p. 106070, 2020, doi: 10.1016/j.ijepes.2020.106070.

[10] S. Mariko, "Aplikasi website berbasis HTML dan JavaScript untuk menyelesaikan fungsi integral pada mata kuliah kalkulus," *J. Inov. Teknol. Pendidik.*, vol. 6, no. 1, pp. 80–91, 2019, doi: 10.21831/jitp.v6i1.22280.