Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)
Edisi. .. Volume. ... Bulan 20.. ISSN : 2089-9033

IMPLEMENTASI ALGORITMA MTD(F) UNTUK MENENTUKAN SOLUSI PADA *BOARD GAME* 2048

Rizqiana Trinenda

Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
E-mail : rizqiana.trinenda@

ABSTRAK

Permainan 2048 adalah permainan yang diciptakan oleh pengembang web yang berasal dari Italia yaitu Gabriele Cirulli pada bulan Maret 2014. Permainan tersebut cukup mudah dimainkan karena tinggal menggeser ke kiri, kanan, atas atau bawah untuk menggabungkan kotak-kotak bernomor tertentu supaya membentuk hasil akhir 2048. Berbagai algoritma sudah diimplementasikan untuk menyelesaikannya, seperti dengan menggunakan algoritma greedy, backtracking, dan minimax. Dari hasil implementasi tersebut didapatkan bahwa algoritma greedy kurang optimal dalam menyelesaikan permainan 2048 karena dari 11 kali percobaan tidak bisa mendapatkan angka 2048 namun cepat dalam mengambil langkah sedangkan untuk algoritma backtracking dapat menyelesaikan permainan 2048 dalam 1 kali percobaan namun dengan waktu 23 menit 18 detik. Untuk algoritma minimax hasilnya ditentukan dari kedalaman dalam pencarian, semakin dalam mencari semakin besar kemungkinan untuk dapat menyelesaikan permainan 2048 namun membutuhkan waktu yang lebih lama sesuai kedalaman pencariannya. Untuk itu masih diperlukan penelitian terhadap algoritma lainnya untuk menyelesaikan permainan 2048.
Algoritma yang diteliti adalah algoritma MTD(f). Berdasarkan dari hasil penelitian, analisis perancangan sistem dan implementasi serta pengujian yang dilakukan dari algoritma MTD(f) untuk menyelesaikan permainan 2048, maka diperoleh kesimpulan bahwa algoritma MTD(f) dapat diterapkan pada permainan 2048 namun untuk mendapatkan tile 2048 agar bisa memenangkan permainan 2048 membutuhkan kedalaman pencarian minimal pencarian level tiga dan membutuhkan waktu sekitar 13 menit 59 detik. Tingkat akurasi pada algoritma MTD(f) pada pencarian dengan kedalaman level nol, satu dan dua adalah 0%, sedangkan pada level tiga adalah 20%, level empat adalah 30% dan level lima adalah 60%.

Kata kunci : Algoritma, MTD(F), *minimax*, permainan, 2048, pencarian

1. PENDAHULUAN

Permainan 2048 adalah permainan yang diciptakan oleh pengembang web yang berasal dari Italia yaitu Gabriele Cirulli pada bulan Maret 2014. Permainan tersebut cukup mudah dimainkan karena tinggal menggeser ke kiri, kanan, atas atau bawah untuk menggabungkan kotak-kotak bernomor tertentu supaya membentuk hasil akhir 2048 [1]. Permainan 2048 dirilis untuk berbagai platform sehingga banyak orang yang dapat memainkannya. Saat ini banyak orang menggunakan berbagai macam cara untuk menyelesaikan permainan tersebut. Berbagai algoritma sudah diimplementasikan untuk menyelesaikannya, seperti dengan menggunakan algoritma greedy, backtracking [2], dan minimax [3]. Dari hasil implementasi tersebut didapatkan bahwa algoritma greedy kurang optimal dalam menyelesaikan permainan 2048 karena dari 11 kali percobaan tidak bisa mendapatkan angka 2048 namun cepat dalam mengambil langkah sedangkan untuk algoritma backtracking dapat menyelesaikan permainan 2048 dalam 1 kali percobaan namun dengan waktu 23 menit 18 detik. Untuk algoritma minimax hasilnya ditentukan dari kedalaman dalam pencarian, semakin dalam mencari semakin besar kemungkinan untuk dapat menyelesaikan permainan 2048 namun membutuhkan waktu yang lebih lama sesuai kedalaman pencariannya. Untuk itu diperlukan penelitian terhadap algoritma lainnya untuk menyelesaikan permainan 2048.

Algoritma yang akan diteliti adalah algoritma MTD(f). Algoritma MTD(f) merupakan salah satu pengembangan dari algoritma minimax. MTD(f) adalah singkatan dari Memory-enhanced Test Driver with node n and value f. Dalam penelitiannya, Aske Plaat [4] menyatakan bahwa algoritma MTD(f) adalah algoritma minimax paling efisien dibandingkan algoritma-algoritma minimax sebelumnya. Dalam beberapa percobaan permainan komputer seperti catur, othello, dan checkers, algoritma MTD(f) mempunyai performa rata-rata lebih baik daripada algoritma-algoritma optimasi minimax lainnya seperti Alpha-Beta dan negascout [5], sehingga algoritma MTD(f) memungkinkan dapat menyelesaikan permainan 2048.

Dari uraian tersebut, maka akan dilakukan penelitian untuk mengimplementasikan algoritma MTD(f) dalam menyelesaikan permainan 2048. Diharapkan dengan mengimplementasikan algoritma MTD(f) dapat diketahui kemampuan algoritma MTD(f) untuk menyelesaikan permainan 2048.

1.1 Permainan 2048

Permainan 2048 merupakan permainan untuk pemain tunggal yang diciptakan pada bulan Maret 2014 oleh seorang programmer asal Italia yang bernama Gabriele Cirulli. Ketika pertama kali dirilisnya permainan ini, permainan ini langsung melesat ke peringkat 1 dunia dalam hanya waktu seminggu dan mencapai lebih dari 4 juta download dan hal tersebut merupakan rekor yang cukup fantastis. Pada kenyataannya, permainan ini hanyalah sebuah proyek "iseng" yang dikerjakan oleh programmer Italia tersebut yang hanya ingin mencoba apakah ia mampu membuat sebuah permainan dari awal dan ia berhasil menyelesaikan pembuatannya hanya dalam waktu seminggu. Permainan ini juga bukanlah yang pertama kali diciptakan, sebelumnya juga pernah ada permainan berjudul Three! yang memiliki tampilan serta cara bermain yang mirip, dan juga permainan 1024 yang cara bermainnya sama persis, hanya saja dengan tujuan permainan yang sedikit berbeda [1].



Gambar 1 Permainan 2048

1.2 Aturan Permainan 2048

Untuk aturan cara bermain permainan 2048 adalah sebagai berikut [1]:

1) Permainan 2048 dimainkan di papan abu-abu 4×4.
2) Di atasnya terdapat beberapa kotak dengan warna berbeda yang bergeser ketika pemain menekan empat tombol arah di keyboard.
3) Kotaknya dapat bergerak sejauh mungkin sampai bertemu kotak lain atau pinggir papan.
4) Apabila dua kotak bernomor sama bergabung saat bergeser, keduanya akan bergabung dan menghasilkan kotak baru yang nomornya merupakan hasil penjumlahan nomor di dua kotak yang bergabung tadi.
5) Misal pada Gambar 2 kotak yang bernilai 2 bisa digabung saat pemain menggeserkan ke arah kiri atau kanan dan akan menghasilkan kotak yang bernilai 4.
6) Kotak yang baru ini tidak bisa bergabung dengan kotak lain dengan arah yang sama.
7) Di setiap pergeseran akan ada satu kotak baru bernomor 2 atau 4 yang muncul secara acak pada ruang kosong di papan.
8) Pemain baru menang ketika sebuah kotak bernomor 2048 muncul di papan.
9) Saat pemain tidak bisa bergerak lagi maka permainan berakhir.
10) Skor didapatkan dari hasil penggabungan kotak.

1.3 Algoritma *Alphabeta Prunning*

Dalam algoritma *minimax*, pencarian dilakukan pada seluruh bagian pohon, sementara sebagian pohon tidak seharusnya diperiksa. *Alpha-beta pruning* merupakan modifikasi dari algoritma *minimax*, yang akan mengurangi jumlah node yang dievaluasi oleh pohon pencarian. Pencarian untuk node berikutnya akan dipikirkan terlebih dahulu. Algoritma ini akan berhenti mengevaluasi langkah ketika terdapat paling tidak satu kemungkinan yang ditemukan dan membuktikan bahwa langkah tersebut lebih buruk jika dibandingkan dengan langkah yang diperiksa sebelumnya. Sehingga, langkah berikutnya tidak perlu dievaluasi lebih jauh. Dengan algoritma ini hasil optimasi dari suatu algoritma tidak akan berubah [5]. Pada algoritma ini, terdapat dua nilai yang diatur, yaitu alpha dan beta, yang merepresentasikan nilai minum dari *max* yang diyakini dan nilai maksimum dari *min* yang diyakini. Nilai awal alpha adalah tak hingga negatif dan nilai awal beta adalah tak hingga positif. Sebagai hasil dari proses rekursif, area pencarian akan semakin kecil. Ketika beta menjadi lebih kecil dari alpha, akan berarti posisi saat itu tidak dapat menjadi hasil terbaik permainan untuk kedua pemain dan pencarian tidak perlu dilakukan lebih jauh. Pada Gambar 2 dapat dilihat *pseudocode* dari algoritma *alphabeta*.

```
function alphabeta(node, depth, α, β, maximizingPlayer)
    if depth = 0 or node is a terminal node
        return the heuristic value of node
    if maximizingPlayer
        v := -∞
        for each child of node
            v := max(v, alphabeta(child, depth - 1, α, β, FALSE))
            α := max(α, v)
            if β ≤ α
                break (* β cut-off *)
        return v
    else
        v := ∞
        for each child of node
            v := min(v, alphabeta(child, depth - 1, α, β, TRUE))
            β := min(β, v)
            if β ≤ α
                break (* α cut-off *)
        return v
```

Gambar **Error! No text of specified style in document.** *Pseudocode* Algoritma *Alphabeta*

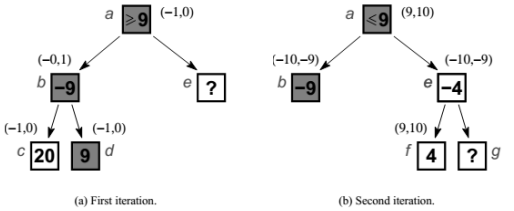
1.4 Algortima MTD(f)

Algoritma MTD(f) adalah sebuah algoritma optimasi Minimax baru yang lebih sederhana dan lebih sangkil daripada beberapa pendahulunya [4]. Nama dari algoritma ini adalah kependekan dari MTD(n,f), yang disingkat dari *Memory-enhanced Test Driver with node n and value f*. MTD adalah nama dari sekumpulan driver program yang mencari pohon Minimax menggunakan pemanggilan *zero-window AlphaBeta*. Dalam beberapa percobaan permainan komputer seperti catur, othello, dan checkers, algoritma ini mempunyai performa rata-rata lebih baik daripada Negascout (variasi dari *AlphaBeta* yang diimplementasikan dalam hampir semua permainan catur, checkers, dan othello). Salah satu program catur terkuat, Cilkchess milik MIT yang menggunakan metode komputasi pararel, juga menggunakan MTD(f) sebagai algoritma pencariannya menggantikan Negascout yang digunakan oleh program catur pendahulunya, StarSocrates. Algoritma MTD(f) terdiri dari 10 baris kode saja, yaitu seperti pada Gambar 3.

```
function MTDF(root, f, d)
g := f
upperBound := +∞
lowerBound := -∞
  while lowerBound < upperBound
    if g = lowerBound then
       β:= g+1
    else
       β := g
    g := AlphaBeta (root, β-1, β, d)
    if g < β then
       upperBound := g
    else
       lowerBound := g
return g
```

Gambar 3 *Psuedo Code* Algoritma MTD(f)

Algoritma MTD(f) tersebut memanggil fungsi *Alpha Beta* berkali kali dengan metode *zero-window* pencarian *Alpha-Beta*, tidak seperti Negascout yang menggunakan pencarian *wide-window*. Pemanggilan *Alpha Beta* mengembalikan batas dari nilai evaluasi Minimax. Batas dari nilai itu kemudian disimpan dalam upperbound (batas atas) dan lowerbound (batas bawah), membentuk sebuah interval yang melingkupi nilai Minimax yang sebenarnya pada pencarian dengan kedalaman tertentu. Positif dan negatif tak hingga adalah kependekan dari nilai diluar interval pada daun pohon. Ketika batas atas dan batas bawahnya bernilai sama atau batas bawah telah melampaui nilai batas atas, maka nilai Minimax telah ditemukan.



Gambar 4 Pohon Pencarian MTD(F)

2. ISI PENELITIAN

2.1 Analisis Masalah

Berdasarkan penelitian yang dilakukan oleh Stephen dari Institut Teknologi Bandung yang menerapkan algoritma greedy dan backtracking pada permainan 2048 [2] dan Vasilis Vryniotis yang menerapkan algoritma minimax pada permainan 2048 [3], didapatkan bahwa algoritma greedy kurang optimal dalam menyelesaikan permainan 2048 karena dari 11 kali percobaan tidak bisa mendapatkan angka 2048 namun cepat dalam mengambil langkah sedangkan untuk algoritma backtracking dapat menyelesaikan permainan 2048 dalam 1 kali percobaan namun dengan waktu 23 menit 18 detik. Untuk algoritma minimax hasilnya ditentukan dari kedalaman dalam pencarian, semakin dalam mencari semakin besar kemungkinan untuk dapat menyelesaikan permainan 2048 namun membutuhkan waktu yang lebih lama sesuai kedalaman pencariannya. Untuk itu diperlukan penelitian terhadap algoritma lainnya untuk menyelesaikan permainan 2048.

2.2 Analisis Permainan 2048

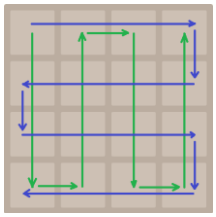
Permainan 2048 merupakan permainan yang dimainkan sendiri. Pada tugas akhir ini permainan akan dilakukan oleh algoritma yang akan diteliti yaitu MTD(f) untuk menentukan solusi dalam menyelesaikan permaian 2048. Permainan 2048 dimainkan dengan cara digeser ke kiri, kanan, atas dan bawah. Permainan akan berakhir bila tidak ada kotak yang bisa digeser atau digabungkan lagi. Permainan akan dimenangkan bila berhasil mendapatkan kotak yang bernilai 2048. Secara acak angka 2 atau 4 akan muncul bila sudah digeser. Bila ada kotak yang sama nilainya dan posisinya bersebelahan maka nilainya bisa digabungkan. Data masukkan untuk lawan data masukkan berupa angka 2 dan 4 dengan kemungkinan muncul untuk angka 2 adalah 90% dan angka 4 adalah 10%. Sedangkan untuk data masukkan pemain yaitu berupa arah untuk menggeser ke kiri, kanan, atas dan bawah . Bila salah satu arah tidak bisa digeser maka aplikasi tidak akan merespon dan papan permainan tidak berubah.

2.3 Nilai Evaluasi

Nilai evaluasi untuk permainan 2048 pada penelitian berdasarkan *snaking tactic* [1] yang dapat dihitung dengan jumlah dari nilai linier pada papan dikalikan dengan nilai dari barisan geometri dengan rasio < 1. Dapat juga ditulis sebagai berikut:

$$p_n \in Path_{0....N-1} \quad (1)$$

$$Score = \sum_{n=0}^{N-1} value(p_n) * r^n \quad (2)$$



Gambar 5 Snaking Tactic

2.4 Analisis Algoritma MTD(f)

Pada bagian ini akan dibahas tentang penyelesaian permainan 2048 dengan menggunakan algoritma MTD(f). Permainan 2048 dimainkan pada sebuah papan permainan dengan ukuran 4x4 sehingga total kotaknya adalah 16. Papan permainan tersebut direpresentasikan dalam matriks 4x4.

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

Gambar 6 Matriks Papan Permainan

Algoritma MTD(f) akan diimplementasikan sebagai pemain yang akan memaksimalkan skor yang didapat. Algoritma MTD(f) memanggil algoritma *AlphaBeta pruning* dalam pencarian solusinya.

1024	512	256	128
2	8	4	128
4	4	2	2
16			

Gambar 7 Keadaan awal Papan Permainan

Kemudian dicari langkah selanjutnya dengan menggunakan algoritma MTD(f) untuk mendapatkan nilai evaluasi tiap langkah. Tiap langkah yang telah dicari dapat dilihat pada tabel berikut:

Tabel 1 Hasil Pencarian Langkah Kiri

Iterasi Ke-	G	Beta	Lowerbound	Upperbound
1	0	0	-∞	+∞
	1170,5	0	1170,5	+∞
2	1170,5	1171,5	1170,5	+∞
	1172	1171,5	1172	+∞
3	1172	1172,5	1172	+∞
	1172	1172,5	1172,5	1172

Tabel 2 Hasil Pencarian Langkah Kanan

Iterasi Ke-	G	Beta	Lowerbound	Upperbound
1	0	0	-∞	+∞
	1170,5	0	1170,5	+∞
2	1170,5	1171,5	1170,5	+∞
	1172	1171,5	1172	+∞
3	1172	1172,5	1172	+∞
	1172	1172,5	1172,5	1172

Tabel 3 Hasil Pencarian Langkah Atas

Iterasi Ke-	G	Beta	Lowerbound	Upperbound
1	0	0	-∞	+∞
	1184	0	1184	+∞
2	1184	1185	1184	+∞
	1184	1185	1184	1184

Tabel 4 Hasil Pencarian Langkah Bawah

Iterasi Ke-	G	Beta	Lowerbound	Upperbound
1	0	0	-∞	+∞
	1025	0	1025	+∞
2	1025	1026	1025	+∞
	1172	1026	1172	+∞
3	1172	1173	1172	+∞
	1172	1173	1172	1172

Pada tabel diatas telah dilakukan pencarian menggunakan algoritma MTD(f). Variabel G adalah nilai pencarian dari pemanggilan fungsi alphabeta, variabel beta nilai beta pada MTD(f) untuk nilai upperbound pada fungsi alphabet, sedangkan lowerbound dan upperbound adalah batas atas dan batas bawah dari algoritma MTD(f) dalam melakukan pencarian. Kemudian didapatkan nilai tiap perpindahan sebagai berikut : Kiri = 1172, Kanan = 1172, Atas = 1184 dan Bawah =1172. Dari hasil tersebut didapatkan bahwa langkah selanjutnya adalah ke atas

2.5 Pengujian Sistem

Pengujian algoritma dilakukan untuk menguji tingkat akurasi dan kecepatan dari algoritma MTD(f) yang telah diimplementasikan pada permainan 2048. Permainan 2048 akan dimainkan oleh algoritma MTD(f) sebanyak 10 kali tiap tingkat kedalaman pencarian. Kemudian dihitung jumlah permainan yang bisa mencapai angka 2048 pada tiap tingkat kedalaman pencarian. Sehingga untuk mendapatkan hasil akurasi algoritma MTD(f) dengan membandingkan banyaknya jumlah permainan yang bisa mencapai angka 2048 dengan banyaknya permainan yang telah dilakukan. Pencarian akan

dilakukan sampai tingkat akurasi lebih besar sama dengan 50%. Sedangkan untuk kecepatannya dihitung berdasarkan waktu yang dibutuhkan untuk menyelesaikan permainan 2048.

Tabel 5 Hasil Pengujian Kedalaman Level 0

Percobaan ke-	Status	Tile Paling besar	Waktu (HH:mm:ss.SSS)	Skor
1	Gagal	256	00:00:05.627	3368
2	Gagal	256	00:00:05.452	3236
3	Gagal	128	00:00:04.053	1932
4	Gagal	256	00:00:04.695	2632
5	Gagal	512	00:00:07.212	5184
6	Gagal	64	00:00:01.734	668
7	Gagal	256	00:00:05.787	3472
8	Gagal	256	00:00:05.761	3364
9	Gagal	256	00:00:06.370	3884
10	Gagal	128	00:00:03.698	1796

Tabel 6 Hasil Pengujian Kedalaman Level 1

Percobaan ke-	Status	Tile Paling besar	Waktu (HH:mm:ss.SSS)	Score
1	Gagal	512	00:00:10.845	8548
2	Gagal	256	00:00:04.725	2928
3	Gagal	256	00:00:05.151	3140
4	Gagal	512	00:00:09.751	7108
5	Gagal	256	00:00:06.577	4200
6	Gagal	512	00:00:08.955	6580
7	Gagal	128	00:00:02.640	1292
8	Gagal	512	00:00:08.136	5728
9	Gagal	256	00:00:07.661	5012
10	Gagal	128	00:00:02.824	1380

Tabel 7 Hasil Pengujian Kedalaman Level 2

Percobaan ke-	Status	Tile Paling besar	Waktu (HH:mm:ss.SSS)	Score
1	Gagal	1024	00:07:50.290	16404
2	Gagal	1024	00:06:03.666	12588
3	Gagal	512	00:01:45.126	9352
4	Gagal	1024	00:04:01.744	14824
5	Gagal	512	00:00:46.471	7272
6	Gagal	512	00:01:18.127	8092
7	Gagal	1024	00:06:11.310	16848
8	Gagal	1024	00:08:14.355	18380
9	Gagal	512	00:01:49.780	9204
10	Gagal	512	00:01:10.856	7464

Tabel 8 Hasil Pengujian Kedalaman Level 3

Percobaan ke-	Status	Tile Paling besar	Waktu (HH:mm:ss.SSS)	Score
1	Berhasil	2048	00:15:20.262	24508
2	Gagal	1024	00:10:42.021	17788
3	Gagal	1024	00:11:36.187	16476
4	Gagal	1024	00:15:19.595	9552
5	Berhasil	2048	00:26:19.731	20176
6	Gagal	1024	00:09:35.158	15620
7	Gagal	512	00:07:35.641	8780
8	Gagal	1024	00:12:23.831	12456
9	Gagal	1024	00:12:21.244	12492
10	Gagal	512	00:09:37.873	7968

Tabel 9 Hasil Pengujian Kedalaman Level 4

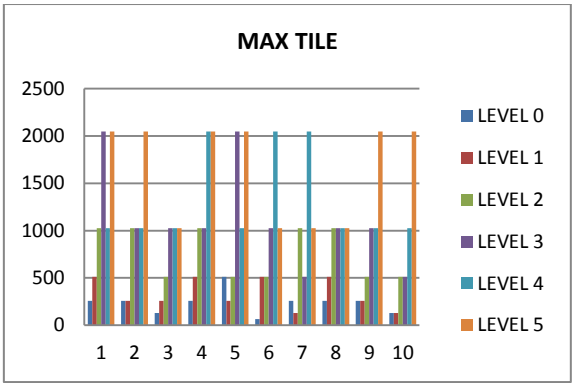
Percobaan ke-	Status	Tile Paling besar	Waktu (HH:mm:ss.SSS)	Score
1	Gagal	1024	00:31:49.967	16160
2	Gagal	1024	00:28:05.387	17748
3	Gagal	1024	00:39:40.811	12252
4	Berhasil	2048	00:43:53.370	20168
5	Gagal	1024	00:36:10.246	14040
6	Berhasil	2048	00:33:54.834	20256
7	Berhasil	2048	00:29:55.392	21124
8	Gagal	1024	00:34:29.883	15448
9	Gagal	1024	00:37:18.236	15640
10	Gagal	1024	00:42:23.269	10260

Tabel 10 Hasil Pengujian Kedalaman Level 5

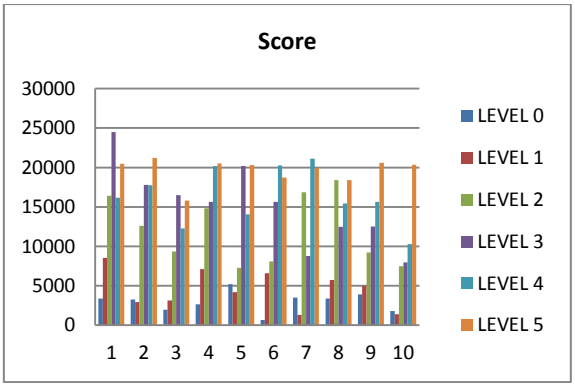
Percobaan ke-	Status	Tile Paling besar	Waktu (HH:mm:ss.SSS)	Score
1	Berhasil	2048	00:57:05.272	20472
2	Berhasil	2048	01:11:45.322	21220
3	Gagal	1024	01:08:59.489	15804
4	Berhasil	2048	01:27:40.496	20516
5	Berhasil	2048	01:13:29.821	20304
6	Gagal	1024	01:19:01.814	18708
7	Gagal	1024	01:25:12.240	19948
8	Gagal	1024	01:23:10.546	18412
9	Berhasil	2048	01:32:27.933	20572
10	Berhasil	2048	01:21:23.320	20344

2.6 Hasil Pengujian

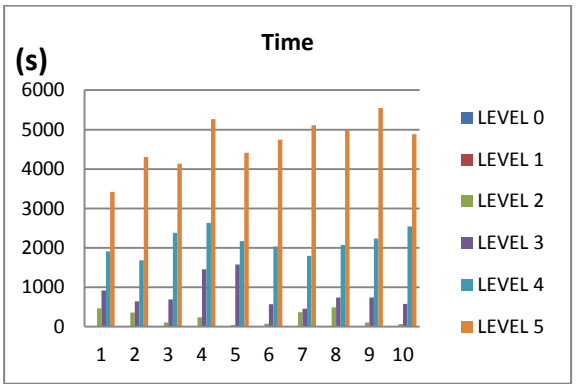
Hasil pengujian yang telah dilakukan sepuluh kali pada setiap level kedalaman dapat dilihat pada gambar.



Gambar 8 Hasil Pengujian Berdasarkan *Max Tile*



Gambar 9 Hasil Pengujian Berdasarkan *Score*



Gambar 10 Hasil Pengujian Berdasarkan Waktu

3. PENUTUP

3.1 Kesimpulan

Berdasarkan dari hasil penelitian, analisis perancangan sistem dan implementasi serta pengujian yang dilakukan dari algoritma MTD(f) untuk menyelesaikan permainan 2048, maka diperoleh kesimpulan bahwa algoritma MTD(f) dapat diterapkan pada permainan 2048 namun untuk mendapatkan tile 2048 agar bisa memenangkan permainan 2048 membutuhkan kedalaman pencarian minimal pencarian level tiga dan membutuhkan waktu sekitar 13 menit 59 detik. Tingkat akurasi pada algoritma MTD(f) pada pencarian dengan kedalaman level nol, satu dan dua adalah 0%, sedangkan pada level tiga adalah 20%, level empat adalah 30% dan level lima adalah 60%.

3.2 Saran

Dalam implementasi algoritma MTD(f) pada permainan 2048 ini masih jauh dari sempurna dan masih banyak kekurangan. Oleh karena itu perlu dilakukan analisis yang lebih dalam. Adapun saran yang dapat diberikan oleh peneliti pada penelitian ini adalah menerapkan algoritma lainnya yang dapat diterapkan pada permainan 2048 agar bisa lebih cepat dan akurat untuk menyelesaikan permainan 2048.

DAFTAR PUSTAKA

[1] K. Lu, "Game Theory of 2048," *Math 89s: Game Theory and Democracy*, 2014.

[2] Stepen, "Perbandingan Algoritma Greedy & Bactracking Dalam Penyelesaian Permainan 2048," *Makalah IF2211 Strategi Algoritma* , 2014.

[3] V. Vryniotis, "Using Artificial Intelligence to solve the 2048 Game (JAVA code) | Datumbox," 14 November 2014. [Online]. Available: http://blog.datumbox.com/using-artificial-intelligence-to-solve-the-2048-game-java-code/. [Accessed 20 September 2015].

[4] A. Plaat, "MTD (f) A Minimax Algorithm faster than NegaScout," Erasmus University, Rotterdam, 1997.

[5] M. Shalihin, I. Arwani and E. M. Adam, "Penerapan Algoritma Minimax Dengan Optimasi Mtd(F) Pada Permainan Dam-Daman," *Informatika*, 2014.

[6] R. S. Pressman, Software Engineering: A Practitioner's Approach, Seventh Edition, New York: McGraw-Hill, 2010.

[7] J. V. Neumann and O. Morgenstern, Theory of Games and Economic Behavior, Third Edition, New Jersey: Princeton, 1953.

[8] S. Russell and P. Norvig, Artificial Intelligence A Modern Approach, Third Edition, New Jersey: Pearson, 2010.

[9] A. Ilman, "Penerapan Algoritma Minimax Dengan Optimasi Mtd(F) Pada Permainan Catur," *Makalah IF2251 Strategi Algoritmik*, 2008.

[10] J. Lewis and W. Loftus, Java Software Solution Foundations of Programming Design, Sixth Edition, New Jersey: Pearson, 2008.

[11] D. A. Uston, Theory and Problems of Software Engineering, New York: McGraw-Hill, 2002.

[12] Microsoft, "Introduction to the C# Language and the .NET Framework," Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx. [Accessed 18 12 2015].

[13] Microsoft, "Visual Studio," Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/vstudio/aa718325.aspx. [Accessed 15 12 2015].