IMPLEMENTASI ALGORITMA PRIM SEBAGAI CREATOR JALUR PERMAINAN MAZE

Devian Ricko Hutama[1]
dvnrco@yahoo.com

R. Gunawan Santosa[2]
gunawan@ukdw.ac.id

Junius Karel[3]
karel@ukdw.ac.id

Abstrak

In this modern era, computer provides more than mere a computing machine. Game is one of the advancements made, and maze is among the popular computer games. While maze is having a simple goal to exit, creating the maze itself is a challenging matter: creating a single winding and confusing path which connects one entry and one exit
This research implements the Prim Algorithm as a maze generator. User will required to enter certain size of the maze, and the system will automatically create the grids, randoming the entry and exit, and providing the necessary weight data for the grid. The Prim Algorithm will then process the weight data, rendering a maze path. This will provide a variation of maze for each game play.
Results of the research shown that maze complexity is not affected by the randoming algorithm used to distribute the grid weights, but dependent to the maze size. Another result is that the number of path walls deconstructed in an already- formed maze in order to make the correct path is carried on a certain pattern (N x N) - 1, dependent to the size of the maze (N x N).

Kata kunci : *algoritma prim, maze generator, maze, pohon bentangan minimum*

1. Pendahuluan

Dengan adanya perkembangan teknologi yang semakin maju, penggunaan komputer di era modern ini tidak hanya sebatas menghitung secara otomatis, melainkan juga dimanfaatkan untuk bermain.Salah satu permainan komputer yang cukup terkenal yaitu *maze*. Pada permainan *maze* pemain diharapkan menemukan jalan keluar yang tepat, dengan menelusuri jalur-jalur yang bercabang dan dibatasi oleh tembok.Dimensi atau ukuran dari *maze* jugamenjadi faktor penting dari sebuah *maze*. Semakin besar dimensi dari sebuah *maze*, maka akan tingkat kesulitan untuk menemukan pintu keluar pun juga akan bertambah.

Berdasarkan masalah di atas, pada penelitian ini penulis akan membangun sebuah sistem yang mengimplementasikan algoritma Prim menjadi sebuah *creator* jalur permainan *maze* berupa jalur bercabang yang dibatasi tembok dengan dimensi *maze* yang bervariatif.

2. Tinjauan Pustaka

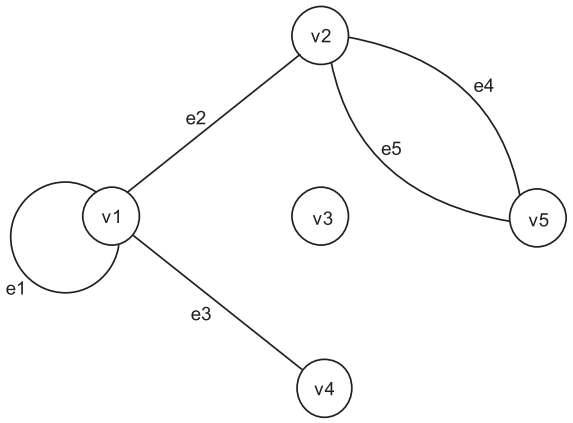
2.1. Graf

Menurut Even (1979), sebuah graf G(V, E) adalah sebuah struktur yang terdiri dari sebuah set verteks V = {v1, v2, …} dan sebuah set *edge* E ={e1, e2, …} dimana setiap *edge* e terhubung dengan elemen-elemen dari sepasang verteks {u, v} yang belum tentu berbeda.

[1]*Program Studi Teknik Informatika, Fakultas Teknologi Info rmasi, Universitas Kristen Duta Wacana*
[2]*Program Studi Teknik Informatika, Fakultas Teknologi Info rmasi, Universitas Kristen Duta Wacana*
[3]*Program Studi Teknik Informatika, Fakultas Teknologi Info rmasi, Universitas Kristen Duta Wacana*

Menurut Even (1979), 2 buah verteks yang terhubung dengan sebuah *edge* disebut *endpoints.* Dan apabila ada 2 buah *edge* yang memiliki *endpoints* yang sama, maka keduanya disebut *parallel edges*. Apabila sebuah *edge* memiliki *endpoints* berupa 2 verteks yang sama, maka *edge* tersebut dapat dikatakan *self-loop*.

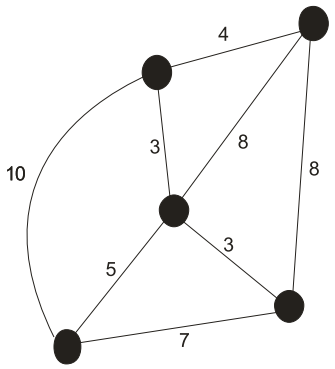
Gambar 1 merupakan sebuah contoh graf dengan V = {v1, v2, v3, v4, v5} dan E = {e1, e2, e3, e4, e5}. Pada graf tersebut, dapat terlihat bahwa e4 dan e5 merupakan *parallel edges*, dan e1 merupakan *self-loop*.



Gambar 1. Contoh Graf dengan 5 Verteks dan 5 *Edge*

2.2. Graf Berbobot (*Weighted Graph*)

Menurut Deo (1994), suatu graf G dikatakan sebagai graf berbobot apabila setiap *edge* dari G berasosiasi dengan sebuah bilangan asli (*real number*). Graf berbobot dapat digambarkan sebagai beberapa kota yang dihubungkan oleh jalan dengan panjang jalan tertentu. Dalam hal ini, kota adalah penggambaran dari verteks, dan jalan yang menghubungkan kota-kota tersebut sebagai *edge*. Pada Gambar 2, terlihat graf dengan 5 verteks dan 5 *edge* dengan setiap *edge* berasosiasi dengan sebuah bilangan yang kemudian disebut bobot dari *edge* tersebut.



Gambar 2. Graf Berbobot

2.3. Pohon (*Tree*)

Menurut Wilson (1985), sebuah hutan (*forest)* didefinisikan sebagai sebuah graf yang tidak memiliki *circuit*, dan sebuah *forest* yang terhubungkan disebut sebagai pohon (*tree*). Dalam bukunya, Gross dan Yellen (1999) mendefinisikan pohon (*tree*) sebagai graf terkoneksi yang tidak memiliki sirkuit (*cycle).*

Menurut Deo (1994), terdapat beberapa teorema sederhana mengenai *properties* dari sebuah *tree* sebagai berikut :

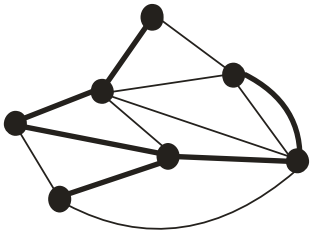
Hanya ada satu *path* untuk setiap pasangan verteks pada sebuah *tree* T.

Sebuah *tree* dengan verteks berjumlah n, akan memiliki *edge* berjumlah n-1.

Sebuah graf adalah juga sebuah *tree* jika dan hanya jika graf tersebut *minimally connected*. *Minimally connected* sendiri berarti jika menghilangkan satu buah *edge* manapun dari suatu graf akan memutuskan graf tersebut. Dan juga, *minimally connected* graf tidak boleh memiliki *circuit*.

2.4. Pohon Bentangan (*Spanning Tree*)

Menurut Deo (1994), sebuah *tree* T dikatakan sebuah *spanning tree* dari sebuah graf terkoneksi G jika T adalah sebuah subgraf dari G dan T memuat semua verteks dari G. Pada Gambar 3, terlihat sebuah *spanning tree* dari sebuah graf terkoneksi digambarkan dengan garis tebal.

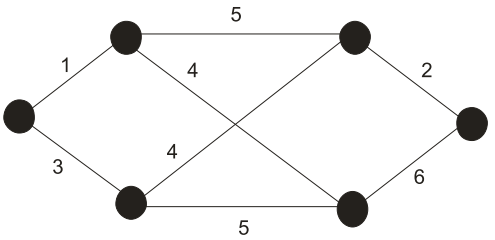


Gambar 3.Spanning Tree dari Sebuah Graf Terkoneksi

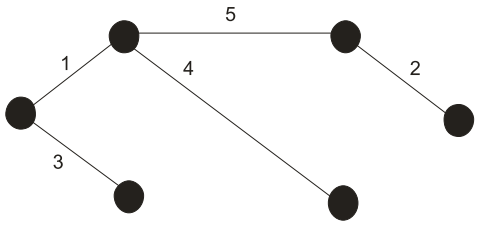
Pada sebuah graf terkoneksi, terdapat minimal satu buah *spanning tree* dari graf tersebut.Ini berarti, sebuah graf terkoneksi bisa memiliki lebih dari satu buah *spanning tree*.

2.5. Pohon Bentangan Minimum (*Minimum Spanning Tree*)

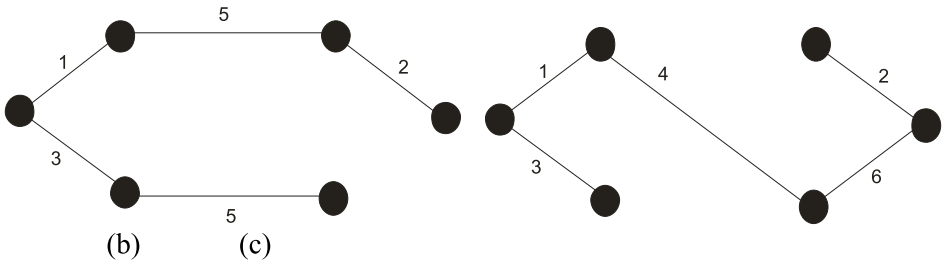
Menurut Deo (1994), sebuah *spanning tree* dengan bobot (*weight)* terkecil dalam sebuah graf berbobot disebut sebagai *minimum spanning tree* atau disebut juga *shortest spanning tree*. Pada Gambar 4, terlihat sebuah graf berbobot. Dari graf tersebut, didapatkan 3 buah *spanning tree* yang dapat dilihat pada gambar 5(a), 5(b), dan 5(c).Dari 3 buah *spanning tree* tersebut, didapati 3 jumlah bobot yang berbeda pula.Dengan demikian, yang merupakan *minimum spanning tree* dari graf pada Gambar 4 adalah *spanning tree* pada Gambar 5(a) karena memiliki bobot yang terkecil yaitu 15.



Gambar 4. Graf Berbobot



(a)



Gambar 5. Tiga Buah *Spanning Tree* dari Graf Berbobot

2.6. Algoritma Prim

Menurut Deo (1994), ada beberapa algoritma yang dapat digunakan untuk mencari *minimum spanning tree* dari sebuah graf. Algoritma tersebut antara lain algoritma Kruskal dan algoritma Prim. Dalam penelitian kali ini, penulis menggunakan algoritma Prim untuk menemukan *minimum spanning tree* dari sebuah graf.Algoritma Prim tidak mengharuskan kita untuk melakukan *listing* dari semua *edge* ke dalam urutan bobot ataupun melakukan pengecekan langkah apabila *edge* yang dipilih membentuk sebuah sirkuit. (Deo, 1994, hlm.62)

Menurut Gross dan Yellen (1999), ide dasar dari algoritma Prim adalah memulai dari verteks manapun dan 'menumbuhkan' sebuah pohon Prim dengan menambahkan *frontier edge* dari *edge* dengan bobot terkecil dalam setiap iterasinya. Dalam hal ini, *edge* dengan bobot terkecil menjadi prioritas utama dalam algoritma Prim. Yang dimaksudkan *frontier edge* adalah semua *edge* yang menghubungkan *vertex-vertex* yang sudah terpilih di dalam pohon Prim dengan *vertex* lain yang belum dikunjungi. Algoritma Prim untuk mencari *minimum spanning tree* dijelaskan sebagai berikut :

Input : sebuah graf terkoneksi berbobot G.
Ouput : sebuah pohon bentangan minimum T.
Pilih sembarang verteks s dari graf G.
Inisialisasi pohon Prim T sebagai verteks s.
Inisialisasi kumpulan *frontier edge* untuk pohon T sebagai 'kosong'.
While pohon Prim T belum mencakup semua verteks dalam graf G
Update kumpulan *frontier edge* untuk T.
e adalah *frontier edge* untuk T dengan bobot terkecil.
v adalah *non-tree endpoint* dari *edge* e.
Tambahkan *edge* e (dan verteks v) ke dalam pohon T.
Return pohon Prim T.

3. Hasil dan Pembahasan

Penelitian ini dibuat untuk melihat sejauh mana algoritma Prim dapat diimplementasikan sebagai *creator* jalur permainan *maze*. Pada penelitian ini, dilakukan 3 buah analisis mengenai pengujian perbandingan terhadap 2 buah algoritma *random* bobot untuk *vertex* pada *maze* terhadap kompleksitas *maze* yang dihasilkan, total penghancuran *wall* sesuai dengan ukuran *maze*, dan total waktu yang dibutuhkan sistem untuk menjalankan algoritma Prim hingga menjadi *maze*. Adapun penelitian mengenai pengukuran waktu, maka akan sangat tergantung pada spesifikasi *hardware* yang digunakan untuk pengujian. Berikut adalah spesifikai *hardware* yang digunakan :

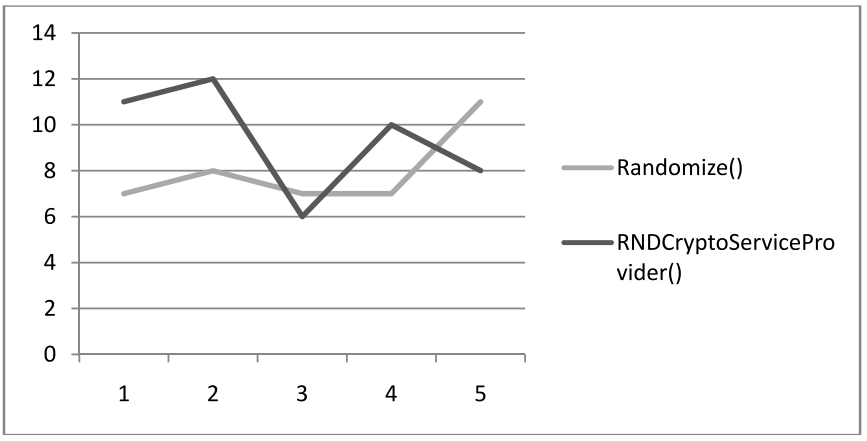
- *Processor* :Intel® Core™ i5-460M (2.53 GHz, 3MB L3 Cache)
- *Operating System* :Windows® 7
- *Memory* :2 GB DDR3

Tabel 1.
Hasil Perbandingan Kompleksitas *Maze* Menggunakan 2 Buah Algoritma *Random* Berbeda

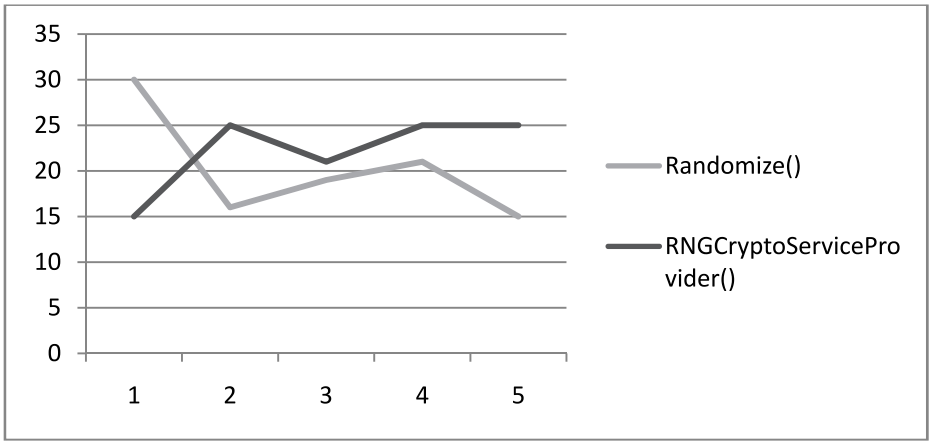
Ukuran Maze	Langkah Tercepat Mencapai Pintu Keluar									
	Randomize()					RNGCryptoServiceProvider()				
	1	2	3	4	5	1	2	3	4	5
5x5	7	8	7	7	11	11	12	6	10	8
6x6	7	11	9	10	10	16	16	12	13	13
7x7	12	15	11	11	11	10	13	13	16	15
8x8	21	14	14	16	13	16	16	13	14	12
9x9	17	16	18	17	17	15	21	14	24	15
10x10	30	16	19	21	15	15	25	21	25	25
11x11	23	21	21	20	20	24	17	14	15	25
12x12	24	20	27	38	22	21	28	27	34	27
13x13	33	30	31	39	22	22	29	21	23	34
14x14	29	42	22	23	24	29	23	28	45	29
15x15	30	35	29	34	30	32	25	39	25	28
16x16	31	53	32	28	24	26	44	23	27	41
17x17	43	47	36	31	29	28	33	26	47	37
18x18	53	35	25	42	70	38	40	47	56	35
19x19	31	27	34	65	45	30	41	41	40	34
20x20	33	40	49	38	45	36	56	41	36	49

Tabel 1 menunjukkan jumlah langkah tercepat yang diperlukan dari pintu masuk hingga mencapai pintu keluar pada *maze* hasil *generate* sistem. 2 buah algoritma *random* yang digunakan pada analisis ini merupakan algoritma *random* pada *software* Microsoft Visual Studio 2008. Algoritma pertama memanfaatkan sintaks Randomize()dengan nilai terkecil 1 dan nilai tertinggi 999, sedangkan algoritma kedua memanfaatkan fasilitas *Random Number Generator Crypto Service Provider*. Algoritma ini menggunakan 8 bit, sehingga akan menghasilkan nilai terkecil 1 dan nilai tertinggi 256. Pengujian dilakukan pada ukuran *maze* 5x5 hingga 20x20 sebanyak masing-masing 5 kali *generate*

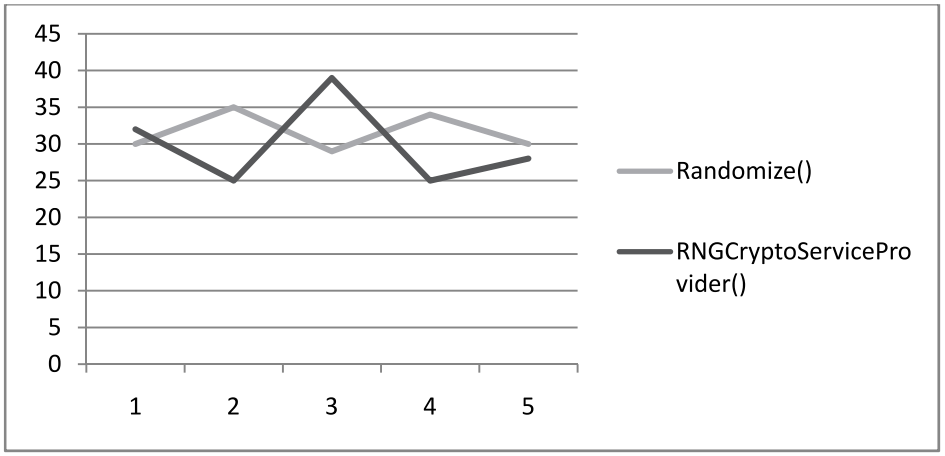
Dari data pada Tabel 1, diambil 4 buah sampel ukuran *maze* 5x5, 10x10, 15x15, dan 20x20 dan dilakukan analisis terhadap grafik peningkatan ataupun penurunan jumlah langkah tercepat untuk mencapai pintu keluar.Berdasarkan Gambar 6, Gambar 7, Gambar 8, dan Gambar 9, didapati hasil bahwa perbedaan 2 buah algoritma *random* yang digunakan tidak mempengaruhi tingkat kompleksitas dari *maze* yang dihasilkan sistem. Adapun yang berpengaruh adalah besarnya ukuran *maze* dengan terbukti bahwa semakin besar ukuran *maze*, akan semakin tinggi pula tingkat kompleksitas dari *maze*. Hal ini dapat dilihat dari *range* kompleksitas *maze* dengan ukuran 5 x 5 yaitu 6 – 12, ukuran 10 x 10 memiliki *range* dari 15 – 30, ukuran 15 x 15 memiliki *range* dari 25 – 40, dan ukuran 20 x 20 memiliki *range* dari 30 – 55.



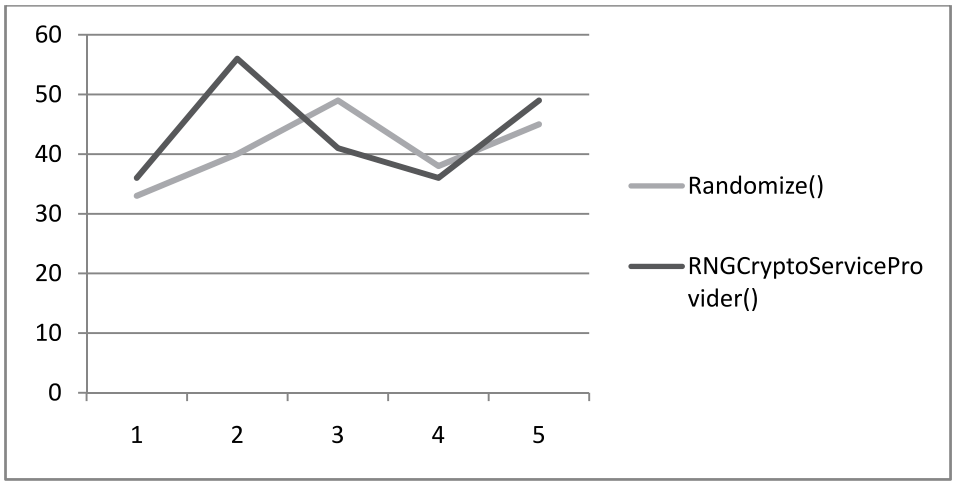
Gambar 6. Grafik Perbandingan Kompleksitas *Maze* Ukuran 5x5Menggunakan 2 Algoritma *Random* Berbeda



Gambar 7. Grafik Perbandingan Kompleksitas *Maze* Ukuran 10x10Menggunakan 2 Algoritma *Random* Berbeda



Gambar 8. Grafik Perbandingan Kompleksitas *Maze* Ukuran 15x15Menggunakan 2 Algoritma *Random* Berbeda



Gambar 9. Grafik Perbandingan Kompleksitas *Maze* Ukuran 20x20Menggunakan 2 Algoritma *Random* Berbeda

Tabel 2.
Jumlah Penghancuran *Wall* dan Total Waktu yang Dibutuhkan Sistem untuk Mengimplementasikan Algoritma Prim hingga Membentuk *Maze*

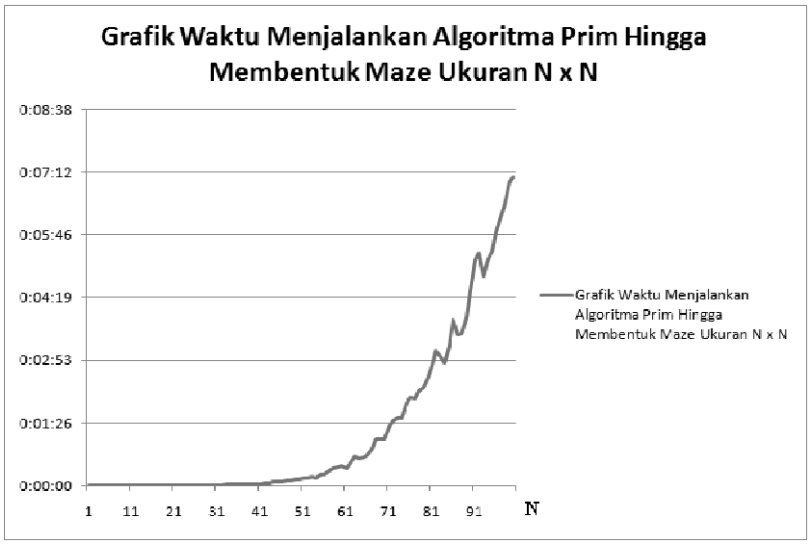
Ukuran Maze	Jumlah Penghancuran Wall	Total Waktu		Ukuran Maze	Jumlah Penghancuran Wall	Total Waktu
5x5	24	0:00:00		28x28	783	00:00.5
6x6	35	0:00:00		29x29	840	00:00.6
7x7	48	0:00:00		30x30	899	00:00.6
8x8	63	0:00:00		31x31	960	00:00.7
9x9	80	0:00:00		32x32	1023	00:00.7
10x10	99	0:00:00		33x33	1088	00:01.0
11x11	120	0:00:00		34x34	1155	00:01.2
12x12	143	0:00:00		35x35	1224	00:01.5
13x13	168	0:00:00		36x36	1295	00:01.6
14x14	195	0:00:00		37x37	1368	00:01.9
15x15	224	00:00.0		38x38	1443	00:01.7
16x16	255	00:00.0		39x39	1520	00:01.8
17x17	288	00:00.0		40x40	1599	00:02.5
18x18	323	00:00.0		41x41	1680	00:02.2
19x19	360	00:00.1		42x42	1763	00:03.7
20x20	399	00:00.1		43x43	1848	00:03.6
21x21	440	00:00.1		44x44	1935	00:04.5
22x22	483	00:00.1		45x45	2024	00:05.1
23x23	528	00:00.2		46x46	2115	00:05.7
24x24	575	00:00.2		47x47	2208	00:06.3
25x25	624	00:00.3		48x48	2303	00:06.9
26x26	675	00:00.3		49x49	2400	00:07.8
27x27	728	00:00.4		50x50	2499	00:08.7
51x51	2600	00:09.7		77x77	5928	02:00.3
52x52	2703	00:10.8		78x78	6083	02:09.5
53x53	2808	00:12.4		79x79	6240	02:14.8
54x54	2915	00:09.7		80x80	6399	02:25.4
55x55	3024	00:15.0		81x81	6560	02:45.6
56x56	3135	00:15.8		82x82	6723	03:06.3
57x57	3248	00:20.4		83x83	6888	02:56.8
58x58	3363	00:23.4		84x84	7055	02:49.5
59x59	3480	00:24.6		85x85	7224	03:12.0
60x60	3599	00:26.2		86x86	7395	03:47.0
61x61	3720	00:24.3		87x87	7568	03:27.5
62x62	3843	00:29.8		88x88	7743	03:29.1
63x63	3968	00:40.8		89x89	7920	03:50.7
64x64	4095	00:37.8		90x90	8099	04:28.6
65x65	4224	00:39.1		91x91	8280	05:10.0
66x66	4355	00:41.4		92x92	8463	05:20.5
67x67	4488	00:50.5		93x93	8648	04:47.8
68x68	4623	01:04.5		94x94	8835	05:08.7

Tabel 2. (lanjutan)
Jumlah Penghancuran *Wall* dan Total Waktu yang Dibutuhkan Sistem untuk Mengimplementasikan Algoritma Prim hingga Membentuk *Maze*

Ukuran Maze	Jumlah Penghancuran Wall	Total Waktu		Ukuran Maze	Jumlah Penghancuran Wall	Total Waktu
69x69	4760	01:03.9		95x95	9024	05:23.9
70x70	4899	01:04.1		96x96	9215	05:49.9
71x71	5040	01:20.5		97x97	9408	06:10.8
72x72	5183	01:30.2		98x98	9603	06:25.8
73x73	5328	01:33.0		99x99	9800	06:58.3
74x74	5475	01:32.4		100x100	9999	07:05.3
75x75	5624	01:52.4				
76x76	5775	02:00.7				

Tabel 2 menunjukkan jumlah penghancuran *wall* yang dilakukan sistem dalam melakukan proses *generate maze* dengan mengimplementasikan algoritma Prim untuk ukuran 5 x 5 hingga 100 x 100. Dari data tersebut, terlihat bahwa jumlah penghancuran *wall* yang dilakukan sistem bertambah banyak seiring dengan peningkatan ukuran *maze*. Untuk ukuran 5 x 5, terjadi penghancuran *wall* sebanyak 24 kali, dan untuk ukuran 100 x 100, terjadi penghancuran *wall* sebanyak 9.999 kali. Dapat dilihat bahwa jumlah penghancuran *wall* dengan ukuran n x n memenuhi rumus (n x n) – 1, hal ini tidak termasuk penghancuran *wall* sebagai pintu masuk maupun pintu keluar.

Tabel 2 menunjukkan total waktu yang dibutuhkan sistem untuk menjalankan algoritma Prim hingga membentuk sebuah *maze* untuk ukuran 5 x 5 hingga 100 x 100. Dari data tersebut, terlihat bahwa total waktu yang dibutuhkan sistem cenderung meningkat seiring dengan peningkatan ukuran *maze*. Pada Gambar 10, grafik yang dihasilkan dari data pada Tabel 2 membentuk sebuah grafik eksponensial. Hasil ini menunjukkan bahwa peningkatan waktu yang dibutuhkan sistem untuk mengimplementasikan algoritma Prim hingga membentuk *maze* semakin signifikan untuk ukuran *maze* yang semakin besar. Peningkatan yang semakin tinggi akan meningkatkan total waktu yang dibutuhkan jauh lebih lama.Dengan ini dapat disimpulkan bahwa algoritma Prim bukan merupakan algoritma terbaik untuk dimanfaatkan sebagai *creator* jalur permainan *maze*.



Gambar 10. Grafik Waktu yang Dibutuhkan Sistem untuk Menjalankan Algoritma Prim hingga Membentuk *Maze*

4. Kesimpulan

Kompleksitas *maze* yang dihasilkan sistem tidak tergantung pada algoritma pemberian bobot *vertex*, melainkan tergantung pada besarnya ukuran *maze*. Terkait dengan besarnya ukuran maze, Jumlah penghancuran *wall* pada *grid maze* dengan ukuran N x N memenuhi rumus (N x N) – 1.

Algoritma Prim bukan merupakan algoritma terbaik sebagai *creator* jalur permainan *maze*, karena algortima Prim hanya baik digunakan untuk ukuran *maze* kecil. Hal ini terbukti karena semakin besar peningkatan ukuran *maze*, maka total waktu yang dibutuhkan oleh sistem untuk membuat *maze* akan meningkat jauh.

Daftar Pustaka

Deo, N. 1994. *Graph Theory with Applications to Engineering and Computer Science.* New Delhi : Prentice-Hall of India.

Even, S. 1979.*Graph Algorithms*. Maryland : Computer Science Press, Inc.

Gross, J., & Yellen, J. 1999. *Graph Theory and It's Application* . Florida : CRC Press.

Turan, M., & Aydin, K. 2010 .A Dynamic Terrain-Spaced Maze Generation Algorithm.*Global Journal of Computer Science and Technology*, Vol. 10 No. 15, pp. 9 – 14.

West, D.B. 2001. *Introduction to Graph Theory.* Upper Saddle River : Prentice-Hall, Inc.

Wilson, R.J. 1985. *Introduction to Graph Theory.* Hongkong : Longman Scientific & Technical.

Xu, J., & Kaplan, C.S. 2007.Image-Guided Maze Construction.*ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2007,* Vol. 26 No. 3.