Aplikasi Pencarian Solusi Graf Labirin Menggunakan Algoritma Tremaux

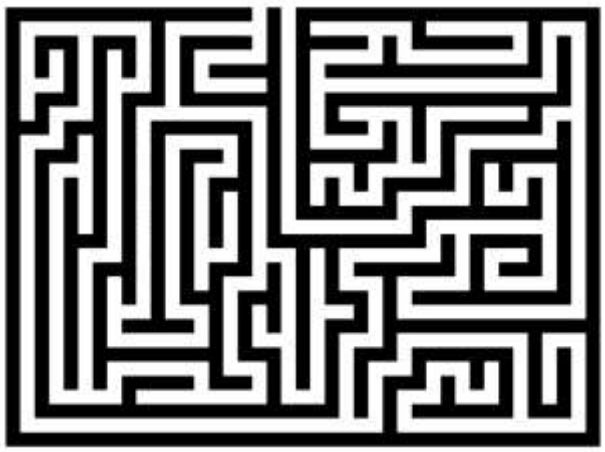
Naufal Malik Rabbani - 13514052
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514052@std.stei.itb.ac.id

***Abstrak*—Permainan labirin merupakan permainan yang menarik dan menantang bagi sebagian orang, terutama anak-anak. Dibalik uniknya bentuk-bentuk labirin, labirin dapat diubah bentuk menjadi sebuah graf. Dengan graf tersebut kita dapat menemukan solusi labirin dengan mudah. Terdapat banyak metode untuk melakukan pencarian solusi labirin, salah satunya dengan menggunakan algoritma Tremaux yang dikembangkan dari Pencarian Kedalaman-Utama (Depth-First Search) pada graf.**

***Kata kunci*—Permainan labirin, solusi labirin, algoritma Tremaux, *Depth-First Search*.**

I. PENDAHULUAN

Menurut KBBI *Online* (http://kbbi.web.id), labirin adalah tempat yang penuh dengan jalan dan lorong yang berliku-liku dan simpang siur; sesuatu yang sangat rumit dan berbelit-belit (tentang susunan, aturan, dan sebagainya); sistem rongga atau saluran yang berhubungan.



Gambar 1.1 Labirin

Seringkali dijumpai permainan labirin pada majalah anak-anak. Permainan ini ditujukan untuk menyegarkan pikiran mereka (anak-anak) dan juga sebagai salah satu permainan asah otak (*puzzle*). Namun selain terdapat di media cetak, labirin juga banyak terdapat pada permainan komputer ataupun pada permainan konsol. Bahkan adapula labirin yang dibuat nyata, biasanya dibentuk dari pepohonan ataupun tembok yang menjulang tinggi. Labirin nyata ini merupakan permainan labirin yang menggunakan fisik, manusia benar-benar memasuki labirin tersebut.

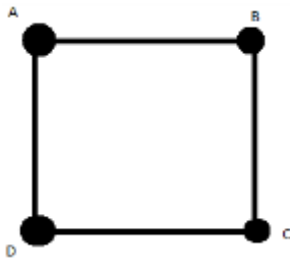
Gambar 1.2 Labirin nyata

Tidak jarang bahwa labirin yang dibuat baik didalam media cetak, media digital, ataupun labirin nyata, merupakan lebirin yang sulit untuk ditemukan solusinya menggunakan akal manusia biasa, dikarenakan labirin yang terlalu rumit ataupun yang terlalu panjang hingga melelahkan. Maka dari itu perlu lah dibuat metode untuk menemukan solusi labirin dengan capet. Salah satu metodenya dengan menggunakan algoritma Termaux yang memanfaatkan konsep-konsep pada graf.

II. DASAR TEORI

A. Definisi Graf

Graf G = (V, E)
V = himpunan tidak kosong dari simpul-simpul = {v1, v2, v3, ..., v_n}
E = himpunan sisi yang menghubungkan 2 simpul = {e1, e2, e3, ..., e_n}



Gambar 2.1 Graf G_1

G_1 = ({A, B, C, D}, {AB, BC, CD, DA}). G_1 merupakan sebuah graf dengan simpul A, B, C, dan D yang dihubungkan oleh garis-garis yang disebut sisi-sisi.

B. Jenis-jenis Graf

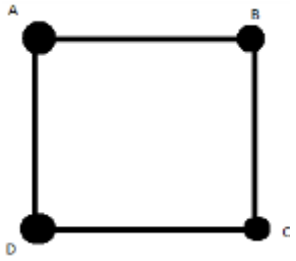
Berdasarkan sisi yang terdapat pada sebuah graf, membentuk gelang atau sisi ganda, terdapat 2 jenis graf. Pertama adalah graf sederhana, yaitu graf yang tidak mengandung sisi ganda maupun gelang. Kedua adalah graf tak-sederhana, yaitu graf yang memiliki sisi ganda ataupun gelang.

Berdasarkan orientasi sisinya graf dapat dibedakan menjadi 2. Pertama adalah graf tak-berarah, yaitu graf yang sisinya tidak memiliki arah. Kedua adalah graf berarah, yaitu graf yang sisinya memiliki orientasi arah sehingga sisi dari simpul A menuju simpul B berbeda dengan sisi dari simpul B menuju simpul A.

Terdapat juga graf semu, yaitu graf tak-berarah yang memiliki sisi ganda ataupun sisi gelang. Graf kosong adalah graf yang himpunan sisinya adalah kosong (tidak memiliki sisi).

C. Terminologi Graf

Dua buah simpul dikatakan bertetangga bila kedua simpul tersebut dihubungkan langsung oleh minimal 1 buah sisi. Sebuah simpul dikatakan bersisian dengan sebuah sisi jika sisi tersebut menghubungkan simpul tersebut dengan simpul lainnya. Derajat sebuah simpul adalah banyaknya sisi yang bersisian dengan simpul tersebut.



Gambar 2.2 Graf G_1

Pada graf G1 simpul A bertetangga dengan simpul B dan simpul C tetapi tidak bertetangga dengan simpul C. Simpul AB bersisian dengan sisi AB dan sisi AD namun tidak bersisian dengan sisi CD maupun dangan sisi CB. Derajat setiap simpul pada G1 adalah d(A)=2, d(B)=2, d(C)=2, d(D)=2.

Notasi derajat sebuah simpul adalah d(v). Jumlah derajat suatu graf yang memiliki beberapa simpul adalah genap, dengan kata lain jumlah simpul berderajat ganjil adalah genap.

Lintasan adalah sisi-sisi yang dilewati dari simpul awal ke simpul tujuan. Panjang lintasan adalah jumlah sisi pada lintasan tersebut. Sirkuit atau siklus adalah lintasan yang berawal dan berakhir pada simpul yang sama.

Sebuah graf dikatakan terhubung jika untuk setiap pasang simpul terdapat lintasan yang menghubungkan pasangan simpul tersebut. Jika ada yang tidak memiliki lintasan maka dikatakan graf tak-terhubung.

D. Graf Khusus

- Graf lengkap (K_n) adalah graf yang setiap simpulnya bertetangga dengan semua simpul lainnya, dengan kata lain memiliki sisi ke semua simpul lainnya. Jumlah sisi pada graf lengkap yang terdiri dari n buah simpul adalah n(n-2)/2.
- Graf lingkaran (C_n) adalah graf yang memiliki n simpul dengan setiap simpul masing-masing berderajat 2.
- Graf teratur adalah graf yang memiliki derajat yang sama. Misalkan d adalah derajat setiap simpul, maka jumlah sisi pada graf teratur dengan n simpul adalah nr/2. Graf lengkap termasuk kedalam graf teratur.
- Graf bipartite G(V1, V2) adalah graf dengan himpunan simpul yang dapat dibagi menjadi dua, misalkan Va dan Vb, sehingga setiap sisi yang terdapat pada G menghubungkan Va dan Vb, dengan simpul ke simpul.
- Graf berbobot adalah graf yang setiap sisinya memiliki harga atau bobot.

E. Lintasan dan Sirkuit

- Lintasan dan Sirkuit. Lintasan Euler adalah lintasan yang tepat satu kali melalui setiap sisi yang berada pada graf. Sedangkan sirkuit Euler adalah lintasan Euler yang memiliki simpul awal dan simpul akhir yang sama.
- Lintasan dan Sirkuit Hamilton. Lintasan Hamilton adalah lintasan yang melelui semua simpul di dalam graf tepat satu kali. Sedangkan sirkuit Hamilton adalah lintasan Hamilton yang memiliki simpul awal dan simpul akhir yang sama.

F. Pohon

Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Sekumpulan pohon tak-terhubung atau graf tak-terhubung yang tidak mengandung sirkuit dan mengandung pohon didalamnya disebut hutan.

G. Pohon Merentang

Pohon merentang dari sebuah graf terhubung adalah upagraf (*sub-graph*) merentang yang tidak mengandung sirkuit. Cara untuk membuat pohon merentang adalah dengan memutus sirkuit yang ada pada graf. Sebuah graf mungkin memiliki lebih dari

satu pohon merentang.

Pohon merentang minimum adalah pohon merentang dari sebuah graf terhubung-berbobot dengan jumlah bobot paling kecil dari semua pohon merentang yang ada dalam graf terhubung-berbobot tersebut.

H. Algoritma Prim

Salah satu cara untuk membuat pohon merentang dengan menggunakan algoritma Prim. Langkah-langkahnya adalah sebagai berikut :

- Pilih sisi dengan bobot minimum dalam graf terhubung-berbobot, lalu masukkan ke dalam T.
- Pilih sisi lain yang memiliki bobot minimum namun harus bersisian dengan T dan tidak membuat sirkuit. Lalu masukkan ke dalam T.
- Ulangi langkah pertama dan kedua sebanyak jumlah sisi dikurang 2.

I. Algoritma Kruskal

Selain menggunakan algritma Prim, cara lain untuk mendapatkan pohon merentang minimum adalah dengan algoritma Kruskal. Langkah-langkahnya adalah sebagai berikut :

- Siapkan sebuah pohon kosong T.
- Urutkan setiap sisi yang ada berdasarkan bobotnya.
- Masukkan sisi yang paling minimum ke dalam T dan pastikan tidak membuat sirkuit.
- Ulangi langkah kedua dan ketiga sebanyak jumlah sisi dikurang 1.

III. MENCARI SOLUSI LABIRIN MENGGUNAKAN ALGORITMA TREMAUX

A. Dasar Algoritma Tremaux

Terdapat banyak algoritma-algoritma yang digunakan untuk memecahkan (menemukan jalan keluar dari) sebuah labirin. Seperti *Wall Follower*, *Pledge Algorithm*, *Chain Algorithm*, *Recursive Backtracker*, *Tremaux Algorithm*, *Dead-end Filler*, *Cul-de-sac Filler*, *Blind Alley Filler*, *Blind Alley Sealer*, *Shortest Path Finder* (*A**), *Shortest Paths Finder*, *Collision Solver*, dan *Random Mouse*. Tiap-tiap algoritma memiliki keunggulan dan kelemahannya masing-masing.

Algoritma Tremaux, metode pemecahan labirin ini dirancang untuk dapat digunakan oleh manusia dalam labirin nyata. Metode ini mirip dengan (*Depth-First Search*) DFS pada graf, dan akan menemukan solusi untuk semua labirin. Berikut adalah ketentuan-ketentuan penggunaan algoritma Tremaux :

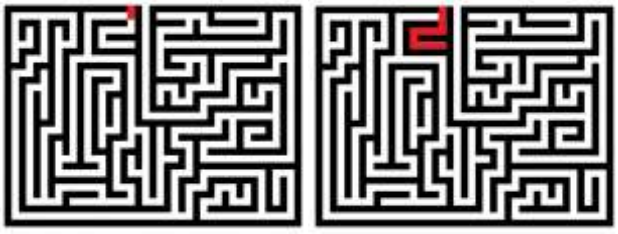
- Saat berjalan menyusuri sebuah bagian, tarik garis dibelakang untuk menandai jalan.
- Ketika menemui jalan buntu, balik arah dan kembali ke tempat semula datang. Tetap menarik garis, sehingga garis yang dilewati sekarang ada 2 buah garis (*backtrack*).
- Ketika menemui persimpangan yang belum pernah ditandai, ambillah prioritas berikut ini jika memungkinkan : kanan, lurus, kiri, atau *backtrack*.
- Ketika menyusuri lorong yang belum pernah ditandai dan ditemukan persimpangan yang sudah pernah ditandai, anggap itu sebagai jalan buntu.
- Ketika sedang melakukan *backtrack* dan menemui persimpangan yang sudah pernah ditandai, lanjutkan menentukan prioritas persimpangan.
- Arti dari belum ditandai adalah belum pernah dilewati. Arti dari tanda 1 kali adalah sudah pernah dilewati tepat 1 kali. Arti dari tanda 2 kali adalah sudah pernah dilewati 2 kali yang berarti jalan tersebut bukan solusi.
- Ketika sudah mencapai tempat tujuan, maka jalur nya adalah bagian-bagian yang ditandai 1 kali. Jika tidak ditemukan solusi maka akan kembali ke tempat semula dan semua bagian telah ditandai 2 kali.

B. Penggunaan Algoritma Tremaux

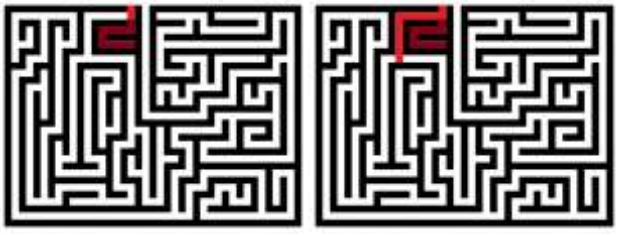
Setelah mengetahui ketentuan-ketentuan algoritma Tremaux, sekarang akan dibuktikan dengan menggunakan labirin yang terdapat di halaman awal. Berikut adalah gambar (a) sampai gambar (n) yang mewakili langkah demi langkah algoritma Tremaux berjalan. Gambar tersusun dari kiri ke kanan dan dari atas ke bawah.

Pada gambar terakhir, gambar (n), merupakan hasil dari algoritma Tremaux yaitu solusi dari labirin tersebut. Karena keterbatasan tempat, maka kami singkat keberjalanan algoritma nya hingga menemukan sebuah solusi.

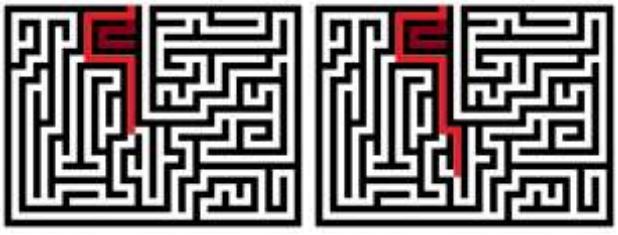
Berikut langkah-langkah algoritma Tremaux berjalan pada sebuah labirin :



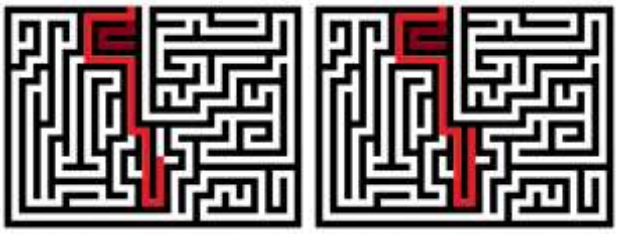
a. Langkah 1 *b. Langkah 2*



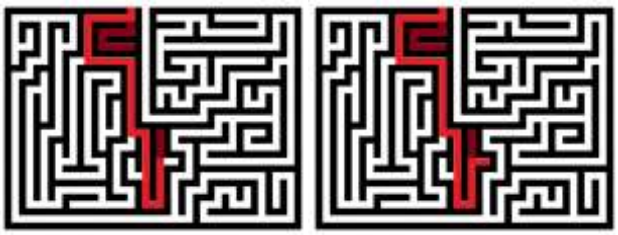
c. Langkah 3 *d. Langkah 4*



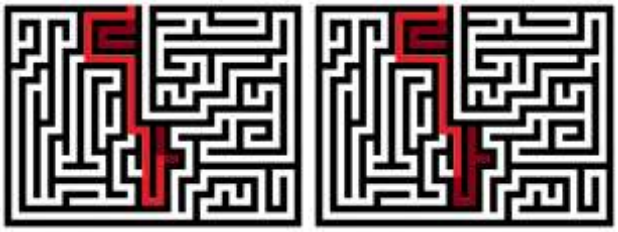
e. Langkah 5 *f. Langkah 6*



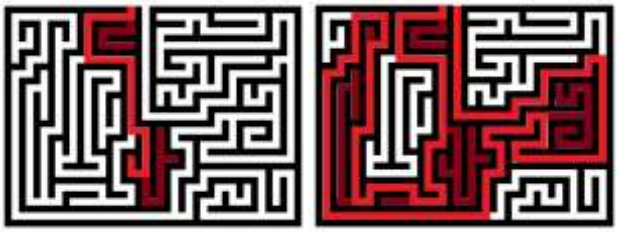
g. Langkah 7 *h. Langkah 8*



i. Langkah 9 *j. Langkah 10*



k. Langkah 11 *l. Langkah 12*



m. Langkah 13 *n. Langkah 39*

Sebagai keterangan, pintu masuk labirin berada di pintu sebelah atas kiri, sedangkan pintu keluar labirin berada di pintu sebelah atas kanan. Warna hitam merupakan tambok atau penghalang, warna putih merupakan jalur yang belum pernah dilewati, warna merah merupakan jalur yang sudah pernah dilewati 1 kali, sekaligus sebagai jalur solusi labirin, dan warna merah tua adalah jalur yang sudah dilewati 2 kali, yang berarti sudah dipastikan solusi bukan melewati jalur tersebut.

C. Analisis Algoritma Tremaux

Dilihat dari keberjalanan algoritma Tremaux pada gambar-gambar diatas, pertama berjalan menyusuri bagian awal sampai bertemu dengan pertigaan (lurus atau ke kanan), menurut prioritas harus disusuri yang lurus terlebih dahulu. Ternyata jalan tersebut adalah jalan buntu, maka harus dilakukan *backtrack* menuju ke pertigaan yang sebelumnya dilewati.

Lalu kita susuri ke kanan, dan menemukan pertigaan lagi (ke kiri atau ke kanan), menurut prioritas harus disusuri yang kiri terlebih dahulu. Ternyata bertemu dengan pertigaan lagi yang belum pernah ditandai, maka pilih sesuai prioritas lagi, dan begitu seterusnya algoritma ini berjalan.

Dengan menggunakan algoritma Tremaux ini, kita dapat menyelesaikan labirin tersebut 'hanya' dalam 39 langkah.

D. Kekurangan dan Kelebihan Algoritma Tremaux

Dapat kita lihat bahwa labirin yang sebelumnya kita kerjakan menggunakan algoritma Tremaux adalah labirin yang sempurna. Labirin sempurna berarti tidak terdapat graf yang berbentuk siklus. Sehingga jika labirin tersebut diubah menjadi sebuah graf pohon dengan 3-*ary*, pohon tersebut dapat membentuk pohon yang sempurna juga, dengan anak-anak pohon yaitu Kiri, Lurus, dan Kanan.

Meskipun contoh demikian sempurna, algoritma Tremaux kenyataannya juga bisa mencari solusi dari sebuah labirin walaupun labirin tersebut adalah labirin yang rumit dan penuh dengan siklus (jika diubah menjadi graf pohon, tidak akan membentuk pohon karena terdapat siklus antar daun-daunnya). Ini dikarenakan ketentun yang ada pada poin 4 dari penjelasan sebelumnya, yaitu "Ketika menyusuri lorong yang belum pernah ditandai dan ditemukan persimpangan yang sudah pernah ditandai, anggap itu sebagai jalan buntu."

Hal yang baru saja disebutkan adalah salah satu keuntungan algoritma Tremaux di banding dengan algoritma lainnya, contohnya adalah *Wall Follower*. *Wall Follower* tidak bisa menangani jika terdapat siklus di dalam labirin yang akan dipecahkan.

Keunggulan lainnya adalah algoritma Tremaux tidak perlu mengetahui dimana letak jalan keluar, atau apakah ada jalan keluar pada labirin tersebut. Karena sudah disebutkan pada ketentuan poin 7, yaitu "Ketika sudah mencapai tempat tujuan, maka jalur nya adalah bagian-bagian yang ditandai 1 kali. Jika tidak ditemukan solusi maka akan kembali ke tempat semula dan semua bagian telah ditandai 2 kali."

Berbeda dengan metode *A**, metode *A** memerlukan pengetahuan dimana letak jalan keluar, dan jika tidak ada jalan keluar metode ini tidak dapat bekerja. Karena metode *A** menggunakan tolak ukur terdekat antara posisi sekarang dengan pintu keluar. Meskipun pada kenyataannya, metode *A** adalah metode yang paling diminati karena lebih dinamis dan bisa digunakan tidak hanya untuk labirin tetapi juga untuk persoalan lainnya (terlebih lagi bagi permainan MMORPG).

Kekurangan dari algoritma Tremaux adalah, kompleksitas waktu yang cukup lama, dan juga pengalokasian memori yang cukup besar (jika di gunakan menggunakan komputer). Waktu yang cukup lama karena, bisa jadi solusi labirinnya terdapat pada jalur kanan nya, tetapi prioritas algoritma ini adalah kiri, lurus, dan kanan (seperti DFS). Sehingga bisa jadi algoritma ini menjelajahi seluruh bagian labirin dengan ratusan langkah, dan ternyata solusi nya hanya 1 langkah ke kanan. Memori yang cukup besar karena, di setiap persimpangan, algoritma Tremaux akan mem-*push stack* pada program, untuk mengingat persimpangan mana yang sedang di jalani. Juga membutuhkan memori yang besar karena digunakan untuk mengingat apakah jalan ini sudah pernah dilewati, sudah dilewati 1 kali, ataupun sudah dilewati 2 kali.

Kekurangan algoritma Tremaux lainnya adalah algoritma Tremaux tidak dapat menentukan jalar terpendek untuk menuju solusi. Dapat dilihat dari contoh keberjalanan algoritma Tremaux yang sudah kita kerjakan

tadi, karena labirin tersebut adalah labirin sempurna, maka hanya ada 1 solusi, sama halnya seperti pencarian pada pohon sempurna yang hanya terdapat 1 solusi.

Namun jika labirin yang akan dipecahkan solusinya adalah sebuah labirin yang rumit dan terdapat siklus, solusi dari labirin tersebut (dengan menggunakan metode apapun) bisa jadi terdapat lebih dari 1 solusi. Berbeda dengan metode *A**, yang mampu mendapatkan hanya 1 solusi dari berbagai kemungkinan solusi labirin yang ada.

IV. KESIMPULAN

Algoritma Tremaux merupakan salah satu metode untuk memecahkan solusi dari sebuah labirin. Algoritma Tremaux ini cukup mudah digunakan karena cara bekerja nya mirip seperti *Depth-First Search* (DFS) pada graf, yang juga mudah dimengerti oleh kita. Walaupun algoritma Tremaux ini terbilang memakan waktu yang cukup lama, tetapi algoritma ini merupakan salah satu yang dapat diterapkan kepada manusia langsung (permainan labirin nyata), dan mampu menghasilkan solusi yang pasti, dimana letak jalan keluar, atau tidak ada jalan keluar dari labirin tersebut.

V. UCAPAN TERIMA KASIH

Alhamdulillaahirabbil ‘aalamiin, rasa syukur saya curahkan kepada Allah SWT atas berkah dan rahmah-Nya. Ucapan terima kasih saya sampaikan tertuju kepada Bapak Rinaldi Munir selaku dosen IF2120 Matematika Diskrit yang telah membimbing, mendidik, dan meluapkan ilmunya kepada saya sebagai mahasiswanya, sehingga terbentuklah dasar-dasar ilmu pengetahuan dalam pembuatan makalah ini. Ucapan terima kasih juga saya sampaikan tertuju kepada keluarga saya, dan juga teman-teman saya yang tidak henti memberikan dorongan untuk menimba ilmu.

REFERENSI

[1] Diktat Kuliah IF2120 Matematika Diskrit, disusun oleh Dr. Ir. Rinaldi Munir, M.T. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung 2006

[2] http://2.bp.blogspot.com/-NbzkWury2zs/UU3BH4_L_jI/AAAAAAAAAMI/6E_AfiVnoN8/s1600/labirin-small.jpg

[3] http://ngapainaja.com/wp-content/uploads/2015/04/labirin5.jpg

[4] https://en.wikipedia.org/wiki/Maze_solving_algorithm pada 8 Desember 2015 pukul 20.59

[5] http://www.primaryobjects.com/2013/05/13/solving-mazes-with-ai-pathfinding-techniques-a-vs-tremaux/ pada 8 Desember 2015 pukul 21.12

[6] http://blog.jamisbuck.org/2014/05/12/tremauxs-algorithm.html pada 9 Desember 2015 pukul 08.32

[7] http://www.astrolog.org/labyrnth/algrithm.htm pada 9 Desember 2015 pukul 20.40

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Desember 2015

Naufal Malik Rabbani - 13514052