PERBANDINGAN ALGORITMA DEPTH FIRST SEARCH, BACKTRACKING DAN A STAR UNTUK MENCARI JALAN KELUAR SEBUAH LABIRIN

Yosafat Adiguna[1], Daniel Swanjaya, M. Kom.[2]
[1,2]Teknik Informatika, Fakultas Teknik, Universitas Nusantara PGRI Kediri
[3]Teknik Informatika, Fakultas Teknik, Universitas Nusantara PGRI Kediri
E-mail: [1]*yosafatadiguna30@gmail.com*, [2]*daniel@unpdkediri.ac.id*

Abstrak – Labirin merupakan sebuah teka-teki yang memiliki banyak jalan yang berliku-liku sehingga akan sulit untuk menemukan jalan keluar. Labirin seringkali digunakan dalam sebuah *game* dimana pemain harus mencari jalan keluar dari labirin yang ada. Dalam dunia komputer untuk mencari jalan keluar sebuah labirin, labirin harus dikonversi menjadi matriks dengan ukuran tertentu setelah itu dilakukan pencarian dengan menggunakan algoritma pencarian. Ada banyak sekali algoritma pencarian yang ditemukan sampai hari ini namun tidak semua algoritma pencarian dapat diterapkan untuk mencari jalan keluar sebuah labirin. Dari beberapa algoritma yang ada belum pernah dilakukan perbandingan untuk menentukan algoritma yang terbaik untuk mencari jalan keluar sebuah labirin, dalam penelitian ini penulis menggunakan tiga algoritma pencarian yaitu *Backtracking*, *Depth First Search* dan *A Star* untuk mencari jalan keluar sebuah labirin, dari hasil yang didapat akan disimpulkan manakah algoritma yang terbaik untuk mencari jalan keluar labirin dari tiga algoritma diatas.

Kata Kunci — *Backtracking, Depth First Search, A Star,* labirin, perbandingan.

1. PENDAHULUAN

Menurut KBBI labirin merupakan tempat yang penuh dengan lorong dan jalan yang berliku-liku dan simpang siur [1]. Menurut Lexico labirin adalah sebuah jaringan dari jalan-jalan atau batas-batas yang dibuat sebagai puzzle (teka-teki) yang akan dicari jalan keluarnya [2]. Menurut Cambridge Dictionary (Kamus Cambridge) labirin adalah kumpulan bagian-bagian atau jalan-jalan yang saling berhubungan dan membingungkan yang adalah mudah untuk tersesat di dalamnya [3]. Dari definisi-definisi diatas dapat disimpulkan bahwa labirin adalah tempat yang berliku-liku dan membingungkan.

Labirin dapat memiliki jalan bercabang yang bukan merupakan rute tercepatnya dan jalan buntu yang bukan merupakan jalan keluar. Dari ketiga algortima yang dipilih masing-masing algoritma memiliki tingkat kompleksitasnya masing-masing sehingga dengan membandingkan algoritma-algoritma yang ada penulis mengharapkan dapat menemukan agloritma terbaik yang dapat menemukan jalan keluar dan memiliki tingkat efisiensi yang maksimal.

Penelitian yang dilakukan oleh Sayed Fachrurrazi yang menggunakan algoritma Prim untuk mencari rute terpendek jalan keluar sebuah labirin. Algoritma Prim mencari *minimum spanning tree* (pohon perentangan minimum) dengan menggunakan graf berbobot yang dimulai secara acak lalu mencari nilai terkecilnya [4] merupakan salah satu penelitian pencarian jalan keluar labirin.

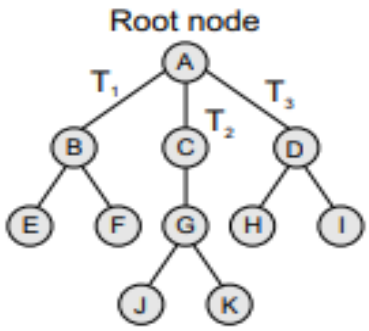
Penelitian lain menggunakan algoritma *Breadth First Search* (BFS) untuk mencari rute terpendek jalan keluar sebuah labirin dalam sebuah *game* berbasis android pernah dilakukan oleh Astrid Novita Putri. Penelitian ini dilakukan dengan cara menggunakan algoritma BFS untuk membantu *player* menuju jalan keluar dengan lebih cepat [5]. Algoritma BFS bekerja dengan cara memeriksa seluruh simpul anak yang ada dan mengulang proses tersebut sampai menemukan simpul tujuan.

Algoritma Dijkstra digunakan oleh Yusuf Anshori, A. Y. Erwin Dodu dan Fadli Kurniawan untuk diterapkan pada robot yang digunakan untuk mencari rute terpendek jalan keluar labirin yang sudah diidentifikasi oleh robot [6]. Algoritma ini bekerja dengan graf berbobot dan mempertimbangkan bobot terkecil dari simpul akar sampai simpul daun tujuan. Dengan demikian didapati rute terpendek.

2. METODE PENELITIAN

2.1 *Tree*

Menurut [7] *Tree* (pohon) secara rekursif dapat diartikan sebagai satu atau lebih kumpulan simpul yang salah satu simpulnya merupakan akar sedangkan simpul-simpul yang lain dapat menjadi kumpulan-kumpulan simpul yang masing-masing adalah sub-tree(bagian kecil pohon) dari akar. Berikut adalah gambar *tree:*



Gambar 1. Tree

Terminologi dasar sebuah *tree/* pohon:

1. Simpul akar (*root node*) adalah simpul yang berada di tempat paling atas sebuah *tree*. Jika simpul akar bernilai *null* maka *tree* yang ada kosong.
2. *Sub-tree*, jika simpul akar tidak bernilai *null* maka T_1, T_2 dan T_3 adalah *sub-tree* dari simpul akar.
3. Jalan (*Path)* adalah serangkaian sisi yang saling berhubungan.
4. Simpul orang tua *(Ancestor node*) adalah simpul manapun yang menjadi simpul pendahulu dari simpul akar akan menuju simpul.
5. Simpul anak (*Descendant node*) adalah simpul manapun yang menjadi simpul penerus dari simpul akar sampai simpul sekarang.
6. Nomer Level, setiap simpul dalam *tree* memiliki nomer level sedemikian rupa sehingga nomer level simpul akar adalah nol, simpul anak dari simpul akar bernilai 1. Lalu setiap simpul memilki nomer level yang lebih tinggi daripada simpul orang tuanya.
7. Derajat, derajat sebuah simpul adalah jumlah anak simpul yang dimiliki oleh simpul tersebut. Nilai derajat simpul daun adalah nol.

Adapun jenis-jenis *Tree* adalah:

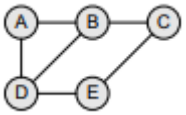
1. *Tree*
2. Hutan (*Forest*)
3. *Pohon biner (Binary Tree*)
4. Pohon Pencarian Biner (*Binary Search Tree*)
5. Pohon Ekspresi (*Expression Tree*)
6. Pohon Turnamen (*Tournament Tree*)

2.2 Graf

Menurut [8] Graf adalah sebuah struktur data yang abstraks yang digunakan untuk mengimplementasikan konsep matematis sebuah graf. Sebuah graf sering dipandang sebagai generalisasi dari sebuah *tree*/ pohon dimana daripada memiliki hubungan simpul *parent – child tree* graf dapat memiliki hubungan kompleks antar simpul yang dapat terjadi. Terminologi dasar dalam Graf:

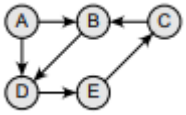
1. Tetangga (*neighbours*), untuk setiap sisi , e = (v, u) yang menghubungkan simpul u dengan simpul v, simpul v dan simpul u berdekatan atau bertetanggaan.
2. Derajat simpul (*degree of nodes*) adalah banyaknya sisi yang tersambung pada sebuah simpul.
3. Graf Reguler (*Regular Graph*) adalah graf di mana setiap simpulnya memiliki jumlah tetangga yang sama, dengan kata lain derajat simpul dalam sebuah graf reguler adalah sama.
4. Jalan (*Path)* adalah serangkaian sisi yang saling berhubungan.
5. Graf Tak Berarah *(Undirected Graph*) adalah sebuah graf yang sisi nya tidak memiliki arah.
6. Graf Berarah (*Directed Graph*) adalah sebuah graf yang sisi nya memiliki arah.

Sebuah graf dapat bearah atau tidak berarah, pada graf tidak berarah sisi tidak mempunyai arah manapun yang terasosikan dengan sisi tersebut. Oleh karena itu jika digambar diantara simpul A dan B sehingga simpul dapat mengarah dari A ke B atau sebaliknya dari B ke A. Berikut adalah gambar graf tak berarah:



Gambar 2. Graf Tak Berarah

Dalam Graf Berarah sisi membentuk sebuah pasangan yang berurutan. Terdapat sebuah jalan yang berasal dari A menuju B dan tidak dari B ke A. sisi (A, B) diinisialisasikan dari simpul A(disebut sebagai simpul awal) dan berhenti di simpul B (disebut sebagai simpul terminal). Berikut adalah gambar graf berarah sederhana:



Gambar 3. Graf Berarah

2.3 Algoritma *Backtracking*

Backtracking adalah cara sistematis untuk mengulang seluruh konfigurasi yang mungkin terjadi dari sebuah ruang pencarian. Konfigurasi ini dapat berupa semua susunan permutasi objek yang dapat terjadi atau seluruh cara yang mungkin untuk membuat kumpulan dari objek-objek tersebut [9].

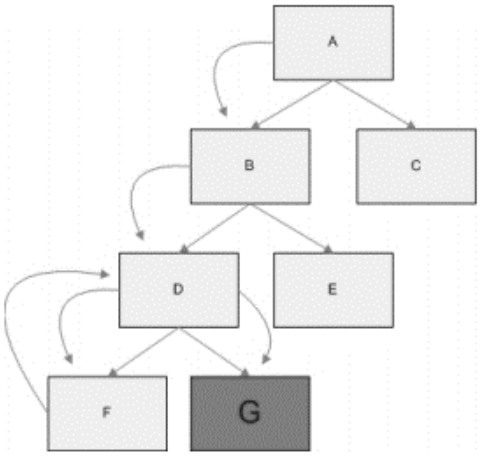
Menurut [10] *Backtracking* adalah algoritma yang berbasis pada *Depth First Search* (DFS) untuk mencari solusi persoalan lebih cepat. Runut balik yang merupakan perbaikan dari algoritma *brute-force*, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada. Dengan langkah ini tidak perlu memeriksa semua kemungkinan solusi yang ada. Akibatnya waktu pencarian dapat dihemat. *Backtracking* lebih alami disebut sebagai algoritma *rekursif*. Kadang disebutkan pula bahwa *Backtracking* merupakan bentuk tipikal dari algoritma *rekursif*. *Backtracking* pertama kali dikenalkan oleh D. H. Lehmer pada tahun 1950. R. J. Walker, Golomb, Baumert

menyajikan uraian umum tentang *Backtracking* dan penerapannya pada berbagai persoalan.

Pencarian dengan menggunakan algoritma *Backtracking* mempunyai langkah-langkah sebagai berikut:

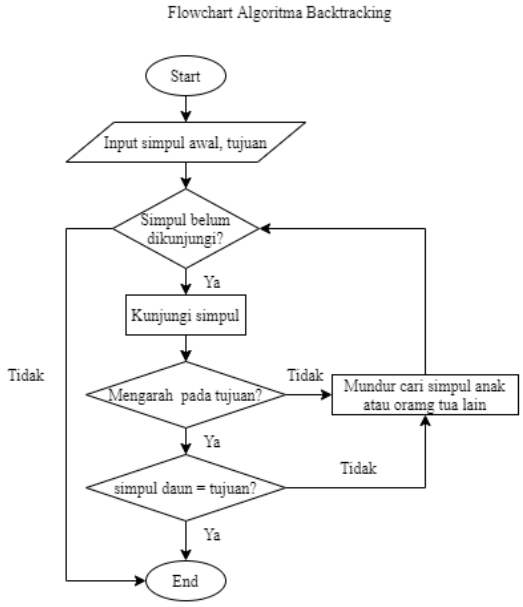
1. Solusi didapat dengan membentuk lintasan dari simpul akar sampai simpul daun. Simpul yang dilahirkan dinamakan simpul hidup dan simpul hidup yang diperluas dinakan simpul-E (*Expand node*).
2. Jika lintasan yang diperoleh dari perluasan simpul-E tidak mengarah pada solusi, maka simpul itu tidak akan diperluas lagi.
3. Jika posisi terakhir ada disimpul mati, maka pencarian dilakukan dengan membangkitkan simpul anak yang lainnya dan jika tidak ada simpul anak maka dilakukan *backtracking* ke simpul orang tua.
4. Pencarian dihentikan ketika menemukan solusi atau tidak ada simpul hidupyang dapat diperluas.

Gambar 4 adalah gambaran cara kerja a*lgoritma Backtracking*. Pada Gambar 4 simpul akar adalah A dan simpul tujuan adalah G. Pencarian dimulai dari memeriksa simpul anak dari simpul A yang adalah simpul B dan C. Lalu simpul B dipilih karena menuju pada simpul tujuan. Lalu diperiksa simpul anak dari simpul B yaitu D dan E. Simpul D lalu dipilih Karena mengarah pada simpul tujuan. Lalu diperiksa simpul anak dari D yaitu F dan G. lalu simpul F secara *default*, karena simpul F adalah simpul daun maka diperiksa apakah simpul F merupakan tujuan dan karena F bukan tujuan maka pencarian dilakukan dengan *backtrack* ke simpul D lalu memilih simpul anak lain dari simpul D. Simpul G lalu dipilih dan diperiksa apakah simpul G merupakan tujuan, karena simpul G merupakan simpul tujuan pencarian selesai.



Gambar 4. Contoh Algoritma *Backtracking*

Gambar 5 adalah gambar flowchart algoritma *Backtracking* untuk mencari jalan keluar sebuah labirin. Langkah pertama yang dilakukan dalam algoritma *Backtracking* seperti yang terlihat dalam gambar 5 adalah memasukkan simpul awal dan simpul tujuan. Lalu dilakukan pengecekan apakah ada simpul yang belum dikunjungi dari simpul aktif jika tidak maka pencarian dihentikan. Jika ada yang belum dikunjungi maka simpul dikunjungi lalu dilakukan pengecekan apakah jalan yang ditempuh mengarah pada simpul tujuan, jika tidak maka dilakukan *backtrack* ke simpul anak atau orang tua lain lalu pencarian dilanjutkan dari langkah pengecekan simpul yang belum dikunjungi. Jika simpul menuju pada simpul tujuan lalu dilakukan proses perulangan dengan memilih simpul yang sebagai simpul aktif. Jika mencapai simpul daun maka dilakukan pengecekan apakah simpul daun yang didapat adalah simpul tujuan dan jika tidak maka dilakukan *backtrack.* Jika simpul daun adalah tujuan maka pencarian selesai.



Gambar 5. Flowchart Algoritma *Backtracking*

2.4 *Depth First Search*

Menurut [11] DFS adalah algoritma graf traversal lain yang fundamental. Dimulai dengan simpul pilihan lalu memilih salah satu dari simpul tetangganya dan berjalan sejauh yang dapat dilakukan sebelum melakukan runut balik. DFS ditemukan oleh seorang matematikawan Perancis Charles Pierre Tremaux sebagai sebuah strategi untuk memecahkan sebuah labirin. DFS menyediakan alat yang bermanfaat untuk membuat simulasi yang yang memungkinkan dalam model skenario.

Sedangkan menurut [12] DFS berjalan dengan memperluas simpul awal dari sebuah graf dan berjalan semakin dalam sampai menemukan simpul tujuan atau sampai simpul tidak memiliki simpul anak yang dapat dikunjungi. Ketika berada di jalan buntu,

algoritma ini melakukan runut balik kembali ke simpul yang belum sepenuhnya di dilewati.

2.5 *A Star*

Menurut [13] Algoritma pencarian rute terpendek *A Star(A*)* berkembang dari algoritma Djikstra dengan menemukan rute terpendek lebih cepat. Hal itu dapat terjadi karena penambahan informasi yang algoritma dapat pakai sebagai bagian dari fungsi heuristik saat menentukan jalan mana yang akan dipilih selanjutnya.

3. HASIL DAN PEMBAHASAN

3.1. Kebutuhan Data

Kebutuhan data yang diperlukan dalam penelitian ini adalah data berupa labirin dalam bentuk matriks. Data tersebut di dapat dari [14].

.

Contoh matriks labirin yang digunakan dalam penelitian ini adalah sebagai berikut:

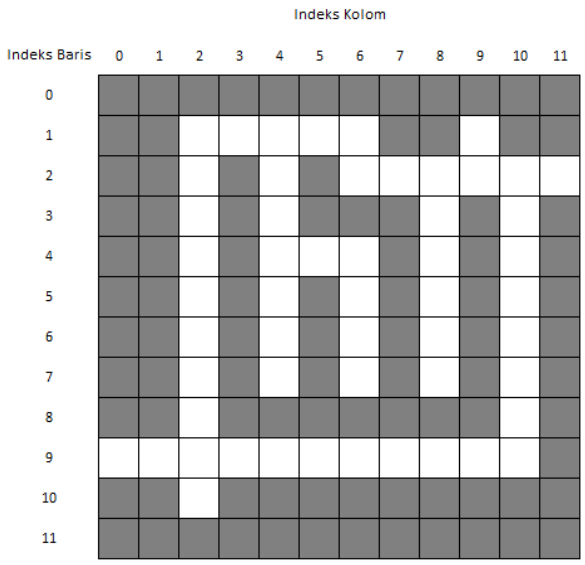
Indeks kolom

Indeks baris	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	0	0	1	0	0
2	0	0	1	0	1	0	1	1	1	1	1	1
3	0	0	1	0	1	0	0	0	1	0	1	0
4	0	0	1	0	1	1	1	0	1	0	1	0
5	0	0	1	0	1	0	1	0	1	0	1	0
6	0	0	1	0	1	0	1	0	1	0	1	0
7	0	0	1	1	1	0	1	0	1	0	1	0
8	0	0	1	0	0	0	0	0	0	0	1	0
9	1	1	1	1	1	1	1	1	1	1	1	0
10	0	0	1	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0

Gambar 6. Contoh Matriks Labirin

Gambar diatas merupakan matriks labirin yang akan digunakan untuk penelitian ini, matriks ini berisi angka 1 dan 0. Angka 1 mewakili jalan yang dapat dilalui sedangkan angka 0 mewakili tembok yang tidak dapat dilalui. Matriks memiliki ukuran 12 x 12, dengan 12 baris yang memiliki indeks 0 sampai 11 dimulai dari kiri atas dan 12 kolom dengan jumlah indeks yang sama.

Berikut adalah gambar visualisasi matriks labirin diatas:



Gambar 7. Visualisasi Contoh Matriks Labirin

3.2. Hasil Testing

Dengan menggunakan matriks labirin contoh kasus yang terdapat diatas dilakukan pencarian jalan keluar dengan menggunakan algoritma *Backtracking*.

Dengan simpul awal dari kasus tersebut adalah koordinat (9,0) dan simpul tujuan dari kasus tersebut adalah (2,11). Dari hasil percobaan ditemukan bahwa algoritma *Backtracking* berhasil menemukan jalan keluar pada labirin ini dan jalan keluar yang ditemukan merupakan rute tercepat. Hasil pencarian pada contoh kasus yang dipakai dapat dilihat dalam Tabel berikut ini:

Tabel 1. Data Pencarian Jalan Keluar Pada Labirin

Iterasi ke-	Lintasan Yang Di Lalui	Jumlah Langkah
1	(9,0)	1
2	(9,0).(9,1)	2
3	(9,0).(9,1),(9,2)	3
4	(9,0).(9,1),(9,2),(9,3)	4
5	(9,0).(9,1),(9,3),(9,4)	5
6	(9,0).(9,1),(9,3),(9,4),(9,5)	6
7	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6)	7
8	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7)	8
9	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8)	9
10	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8),(9,9)	10
11	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8),(9,9),(9,10)	11
12	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8),(9,9),(9,10),(8,10)	12

13	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8),(9,9),(9,10),(8,10),(7,10)	13
14	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8),(9,9),(9,10),(8,10),(7,10),(6,10)	14
15	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8),(9,9),(9,10),(8,10),(7,10),(6,10),(5,10)	15
16	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8),(9,9),(9,10),(8,10),(7,10),(6,10),(5,10),(4,10)	16
17	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8),(9,9),(9,10),(8,10),(7,10),(6,10),(5,10),(4,10),(3,10)	17
18	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8),(9,9),(9,10),(8,10),(7,10),(6,10),(5,10),(4,10),(3,10),(2,10)	18
19	(9,0).(9,1),(9,3),(9,4),(9,5),(9,6),(9,7),(9,8),(9,9),(9,10),(8,10),(7,10),(6,10),(5,10),(4,10),(3,10),(2,10),(2,11)	19
END		

Dari hasil pencarian diatas didapat bahwa untuk mencari jalan keluar pada contoh kasus diperlukan 19 langkah dan jalan keluar yang didapat merupakan rute terpendek. Dari percobaan juga didapat bahwa durasi pencarian adalah 6 milisekon. Dari data hasil pencarian ada beberapa hal yang perlu di perhatikan yaitu bahwa algoritma *Backtracking* mampu mencari rute terpendek saat pencarian labirin berlangsung. Dapat dilihat pada iterasi ke 4 di Tabel 1 bahwa setelah dari simpul (9,2) perulangan menuju (9,3) dan bukan menuju (10,2). Hal ini dapat terjadi karena algoritma *Backtracking* mempertimbangkan apakah simpul (10,2) mengarah pada simpul tujuan dan jawabannya adalah tidak, sehingga simpul (10.2) dimatikan dan tidak dilewati.

Berikut adalah tabel dari 10 percobaan penggunaan algoritma-algoritma yang ada untuk mencari jalan keluar labirin:

Tabel 3. Percobaan DFS

Labirin	Jumlah Langkah	Lintasan Terpendek	Durasi	Status
12A	18	Ya	89 ms	Sukses
12B	52	Tidak	78 ms	Sukses
12C	72	Ya	15 ms	Sukses
12D	24	Tidak	140 ms	Sukses
12E	58	Tidak	0	Sukses
12F	-	-	-	Gagal
12G	42	Tidak	15 ms	Sukses
12H	-	-	-	Gagal
12I	-	-	-	Gagal
12J	58	Tidak	38 ms	Sukses

Tabel 4. Percobaan Backtracking

Labirin	Jumlah Langkah	Lintasan Terpendek	Durasi	Status
12A	18	Ya	68 ms	Sukses
12B	28	Tidak	0 ms	Sukses
12C	28	Tidak	266 ms	Sukses
12D	24	Tidak	0 ms	Sukses
12E	54	Tidak	156 ms	Sukses
12F	20	Tidak	78 ms	Sukses
12G	34	Tidak	109 ms	Sukses
12H	44	Tidak	47 ms	Sukses
12I	20	Ya	62 ms	Sukses
12J	54	Tidak	63 ms	Sukses

Tabel 5. Percobaan A Star

Labirin	Jumlah Langkah	Lintasan Terpendek	Durasi	Status
12A	18	Ya	248 ms	Sukses
12B	20	Ya	172 ms	Sukses
12C	47	Ya	15 ms	Sukses
12D	24	Ya	250 ms	Sukses
12E	42	Tidak	101 ms	Sukses
12F	25	Tidak	47 ms	Sukses
12G	30	Ya	313 ms	Sukses
12H	37	Ya	250 ms	Sukses
12I	22	Tidak	265 ms	Sukses
12J	42	Tidak	297 ms	Sukses

3.3. Pembahasan

Dari ketiga algoritma yang ada menghasilkan hasil dari sepuluh labirin yang sama. Dengan menggunakan waktu, jumlah langkah, lintasan terpendek dan status sebagai parameter.

Algoritma DFS memiliki nilai durasi yang relatif lebih kecil dibanding dengan Backtracking atau A Star. Namun DFS memiliki 3 labirin yang menghasilkan jalan buntu atau jalan keluar tidak dapat ditemukan.

Algoritma Backtracking memiliki nilai status yang baik karena dapat menemukan jalan keluar semua labirin yang ada. Namun Backtracking memiliki nilai jalur terpendek yang kecil yaitu hanya satu dari sepuluh labirin yang ada.

Algoritma A Star juga memiliki nilai status yang baik karena dapat menemukan jalan keluar semua labirin yang ada. Algoritma ini juga memiliki nilai lintasan terpendek lebih baik dari pada algoritma-algoritma sebelumnya yaitu enam dari sepuluh labirin yang ada. Namun nilai durasi dari algoritma ini rata-rata lebih besar dari algoritma-algoritma sebelumnya.

4. SIMPULAN

1. Algoritma DFS memiliki kekurangan dalam bagian pencarian karena memiliki kemungkinan

untuk tidak menemukan jalan keluar dibandingkan dengan algoritma lain.

2. Algoritma Backtracking memiliki kekurangan dalam pencarian rute terpendek dibandingkan dengan algoritma lain.
3. Algoritma A Star memiliki kekurangan dalam durasi dibandingkan dengan algoritma lain.
4. Algoritma *Backtracking* memiliki keunggulan dalam mencari jalan keluar dibandingkan dengan DFS dan sama dengan A Star yang mampu mencari sepuluh jalan keluar dari sepuluh labirin.
5. Algoritma A Star memiliki keunggulan dalam mencari jalan keluar dibandingkan DFS dan sama dengan Backtracking yang dapat menemukan setiap jalan keluar dari sepuluh labirin yang ada.
6. Algoritma A Star memiliki keunggulan dalam pencarian rute tercepat dibandingkan dengan DFS atau Backtracking hal ini disebabkan karena A Star juga menggunakan fungsi heuristic sebagai bagian integral dalam pencarian rute tercepat.

5. SARAN

1. Algoritma yang digunakan dalam penelitian ini memiliki tingkat kerumitan yang berbeda-beda sehingga dapat diasumsikan bahwa algoritma dengan kerumitan tertinggi adalah algoritma terbaik, ada baiknya penelitian kedepan menggunakan algoritma dengan tingkat kerumitan yang setara.
2. Labirin yang digunakan dalam penelitian ini adalah matriks 2 dimensi berukuran n x n, untuk pengembangan penelitian ke depan dapat menggunakan matriks 3 dimensi yang berukuran n x n x n.

DAFTAR PUSTAKA

[1] Badan Pengembangan Bahasa dan Perbukuan, Kementrian Pendidikan dan Kebudayaan Republik Indonesia. 2016. https://kbbi.kemdikbud.go.id/entri/labirin diakses pada 23 Februari 2020.

[2] Dictionary.com dan Oxford University Press. 2020. https://www.lexico.com/en/definition/labyrinth diakses pada 23 Februari 2020.

[3] Cambridge University Press. 2020. https://dictionary.cambridge.org/dictionary/english/labyrinth diakses pada 23 Februari 2020.

[4] Fachrurrazi, S., 2018. Sistem Penentuan Rute Yang Tepat Dalam Sebuah Labirin Dengan Menerapkan Algoritma Prim. *Jurnal Sistem Informasi*. Vol. II, Np. 1: 51-67.

[5] Putri, N. A., 2016. Optimasi Alghoritma Breadth First Search Pada Game Engine Third Person Shooter Maze Berbasis Agen Cerdas Android. *Jurnal Transformatika*. Vol. XIV, No. 1: 50-55.

[6] Anshori, Y., Dodu, E. A. Y., Kurniawan, F. Perancangan Robot Penelusur Menggunakan Algoritma Dijkstra Dan Metode Maze Solver. *Techno.COM*. Vol. XVIII, No. 2: 166-177.

[7] Thareja, R. 2014. DATA STRUCTURES USING C. India: Oxford Unviersity Press. Hal. 279.

[8] Thareja, R. 2014. DATA STRUCTURES USING C. India: Oxford Unviersity Press. Hal. 383.

[9] Skiena, S. 2008. The Algorithm Design Manual. New York: Springer. Hal. 271.

[10] Siraif, Br. Rina., 2013. Perancangan Aplikasi Game Labirin Dengan Menggunakan Algoritma Backtracking. *Pelita Informasi Budi Darma*. Vol. V, N0. 2: 101-103.

[11] Needham, M., Hodler, Amy E. 2019. Graph Algorithms. O'Reilly Media, Inc. Hal 48.

[12]` Thareja, R. 2014. DATA STRUCTURES USING C. India: Oxford Unviersity Press. Hal. 397.

[13] Needham, M., Hodler, Amy E. 2019. Graph Algorithms. O'Reilly Media, Inc. Hal 56.

[14] https://github.com/AkshayGuptaK/maze-solvers/blob/master/Maze%20Solver%20Step%20Through.xlsx.