Pencarian Langkah Minimum dalam Penyelesaian Permainan Ular Tangga dengan Algoritma *Branch and Bound*

Hafidz Nur Rahman Ghozali – 13520117
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520117@std.stei.itb.ac.id

Abstract – Permainan Ular Tangga merupakan permainan papan yang bertujuan untuk mencapai kotak dengan nomor maksimum secepat mungkin. Permainan ini menggunakan guliran dadu untuk menjalankan bidak setiap pemain. Makalah ini membahas mengenai pencarian langkah minimum dalam penyelesaian permainan ular tangga dengan memanfaatkan Algoritma Branch and Bound.

Keywords—Ular Tangga, Algoritma Branch and Bound

I. PENDAHULUAN

Permainan Ular Tangga merupakan permainan yang biasa dimainkan oleh anak-anak. Permainan ini dapat dimainkan oleh dua orang atau lebih. Permainan ini menggunakan papan permainan dan sebuah dadu yang mempunyai enam sisi. Ular tangga merupakan permainan yang menarik karena langkah setiap pemain ditentukan oleh guliran dadu setiap pemain, tidak dapat dikontrol. Papan permainan ini dilengkapi dengan tangga dan ular sebagai bantuan dan rintangan dalam permainan.

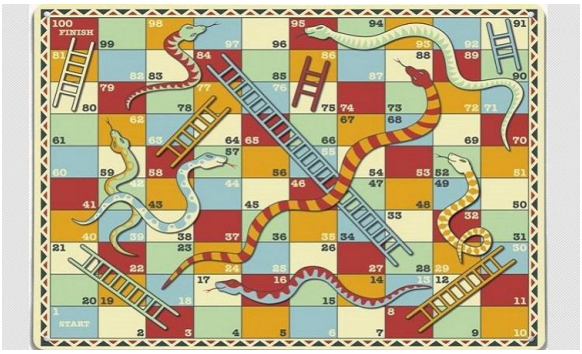
II. TEORI DASAR

A. Permainan Ular Tangga

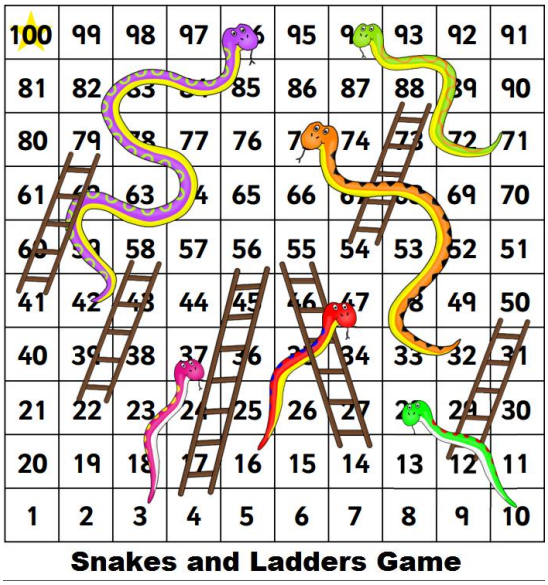
Permainan Ular Tangga merupakan sebuah permainan yang dimainkan pada papan kotak-kotak berukuran m baris dan n kolom. Biasanya, ukuran papan permainan adalah persegi, contohnya 10 baris x 10 kolom. Setiap kotak pada papan permainan memiliki angka menaik dari 1 hingga angka maksimum, bergantung pada ukuran papan permainan.

Permainan ular tangga dapat dimainkan oleh dua orang atau lebih. Biasanya, permainan ini dimainkan oleh 4 orang. Setiap pemain memiliki bidaknya masing-masing dengan warna yang berbeda. Pada awal permainan, semua bidak terletak di luar papan permainan atau bisa disebut dengan kotak 0. Untuk menggerakan bidak miliknya, pemain diharuskan untuk melempar dadu 6 sisi terlebih dahulu. Pemain dapat menggerakkan bidak sesuai dengan mata dadu yang muncul pada pelemparan tersebut. Setiap pemain mendapatkan giliran pelemparan dadu secara tetap dan hanya dapat menggerakkan bidak miliknya saja.

Tujuan dari permainan ular tangga adalah menjadi pemain tercepat dalam mencapai angka maksimum dalam papan permainan. Untuk mencapai tujuan tersebut, terdapat beberapa bantuan dan rintangan di dalam papan permainan. Bantuan tersebut berupa tangga yang menghubungkan antara kotak dengan angka yang kecil menuju kotak dengan angka yang lebih besar. Ketika seorang pemain berada pada kotak pangkal tangga, maka pemain tersebut dapat naik ke kotak di ujung tangga. Sebaliknya, terdapat ular sebagai rintangan dalam permainan ular tangga ini. Ular menghubungkan antara kotak dengan angka yang lebih besar dengan kotak dengan angka yang lebih kecil. Ketika seorang pemain berada pada kotak yang terdapat kepala ular, maka pemain tersebut harus turun menuju kotak yang ditunjuk oleh ekor ular yaitu kotak dengan angka yang lebih kecil.



Gambar 1. Papan Permainan Ular Tangga



Gambar 2. Papan Permainan Ular Tangga

B. Algoritma Branch and Bound

Algoritma Branch and Bound merupakan sebuah algoritma yang digunakan untuk persoalan optimasi, baik meminimasi atau memaksimasi suatu fungsi objektif dengan tidak melanggar batasan-batasan dalam suatu persoalan. Algoritma Branch and Bound memiliki kemiripan dengan Algoritma Breadth First Search (BFS). Algoritma Branch and Bound akan membangkitkan simpul baru berdasarkan cost paling kecil atau paling besar, sedangkan algoritma BFS akan membangkitkan simpul baru berdasarkan urutan pembangkitannya *(First In First Out)*. Dalam Algoritma Branch and Bound, simpul-simpul yang akan dibangkitkan akan disimpan di dalam struktur data *priority queue* atau antrian yang memiliki prioritas.

Setiap simpul memiliki nilai ongkos atau *cost*, yaitu taksiran biaya minimum atau maksimum dari simpul akar ke simpul tujuan melalui simpul tersebut. Pada persoalan minimasi, simpul yang akan diekspansi merupakan simpul dengan ongkos atau cost paling kecil. Sedangkan pada persoalan maksimasi, simpul dengan ongkos atau *cost* paling besar akan diekspansi terlebih dahulu.

Untuk menentukan cost pada setiap simpul, dibutuhkan teknik heuristik yang paling optimal berdasarkan intuisi. Dalam Algoritma Branch and Bound juga terdapat fungsi pembatas. Fungsi pembatas ini berfungsi untuk memangkas jalur atau simpul yang tidak mengarah ke solusi yang optimal. Fungsi pembatas digunakan ketika sudah menemukan satu solusi terbaik. Kemudian simpul-simpul yang ada di dalam antrian akan dimatikan atau dihapus apabila nilai costnya tidak lebih baik daripada cost solusi yang ditemukan.

Secara garis besar, Algoritma Branch and Bound dalam persoalan minimasi dapat dinyatakan sebagai berikut:

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar tersebut adalah simpul tujuan, maka solusi telah ditemukan. Hentikan jika hanya menginginkan satu solusi saja.
2. Hentikan jika Q kosong.
3. Jika Q tidak kosong, pilih satu simpul dari antrian Q dengan cost paling kecil.
4. Jika simpul yang terpilih merupakan simpul solusi, maka solusi telah ditemukan. Hentikan jika hanya menginginkan satu solusi saja.
5. Jika simpul yang terpilih bukan merupakan solusi, bangkitkan anak-anak dari simpul tersebut. Jika simpul tersebut tidak memiliki anak, kembali ke langkah 2.
6. Hitung ongkos atau *cost* dari anak-anak simpul tersebut dan masukkan ke dalam Q.
7. Kembali ke langkah 2

C. Priority Queue

Priority Queue merupakan pengembangan dari struktur data *Queue* dengan setiap elemen di dalamnya memiliki prioritas tertentu. Pada saat penambahan elemen baru, elemen tersebut akan berada di dalam antrian sesuai dengan prioritasnya. Biasanya, elemen paling depan dalam sebuah antrian adalah elemen yang memiliki prioritas tertinggi. Oleh karena itu, dalam pengambilan elemen akan selalu terambil elemen dengan prioritas terbesar.

Struktur data *priority queue* ini digunakan dalam menyimpan simpul-simpul yang akan diekspansi dalam Algoritma Branch and Bound. Prioritas yang digunakan berdasarkan *cost* atau ongkos dari setiap simpul. Semakin kecil ongkos suatu simpul, maka semakin besar prioritas dalam *priority queue*-nya.

D. Map

Map merupakan struktur data yang digunakan untuk memetakan suatu nilai dari nilai yang lain. Sebuah kunci atau *key* dapat menunjuk ke nilai atau *value* tertentu.

Dalam makalah ini, struktur data map digunakan dalam menyimpan informasi tangga dan ular yang ada di dalam papan permainan ular tangga. *Key* pada setiap elemen merupakan angka tempat pangkal tangga dan kepala ular berada. Sedangkan *value*-nya merupakan angka pada kotak yang ditunjuk oleh ujung tangga atau ekor ular.

III. APLIKASI ALGORITMA BRANCH AND BOUND PADA PERMAINAN ULAR TANGGA

A. Rumusan Masalah

Permainan ular tangga merupakan persoalan minimasi karena meminimumkan langkah untuk mencapai angka maksimum.

Papan permainan ular tangga yang tergambar pada gambar 1 dan 2 merupakan papan permainan yang akan dibahas pada makalah ini. Terdapat konfigurasi papan permainan berupa tangga dan ular. Konfigurasi tangga dan ular pada papan permainan 1 dapat dilihat pada tabel berikut

Tabel 1. Tabel Konfigurasi Ular dan Tangga

Tangga	Ular
(3,21)	(17,12)
(8,30)	(52,29)
(28,84)	(57,40)
(58,77)	(62,22)
(75,86)	(88,18)
(80,100)	(93,51)
(90,91)	(97,79)

Simpul akar pada persoalan ini adalah kondisi awal ketika memulai permainan. Seorang pemain hanya dapat bergerak 1 hingga 6 langkah sesuai dengan kemungkinan mata dadu. Oleh karena itu, pembangkitan ruang status pada setiap simpul juga didasarkan pada 6 kemungkinan tersebut. Simpul tujuan pada permainan ular tangga ini adalah kotak yang memiliki angka 100.

B. Teknik Heuristik

Dalam penyelesaian permainan ular tangga menggunakan algoritma branch and bound, state didasarkan pada posisi bidak pemain. Setiap simpul yang dibangkitkan akan memuat posisi bidak dan ongkos atau *cost* simpul tersebut. Ongkos simpul tersebut dikalkulasi menggunakan Teknik Heuristik yang mempertimbangkan jarak posisi bidak saat ini dengan kotak yang memiliki angka 100 dan 6 kemungkinan mata dadu. Ongkos simpul tersebut dihitung berdasarkan persamaan berikut

$$\hat{c}(i) = f(i) + \hat{g}(i)$$
$$\hat{g}(i) = \left\lceil \frac{100 - i}{6} \right\rceil$$

dengan

$\hat{c}(i)$ merupakan ongkos untuk simpul i dengan kotak bernomor i

$f(i)$ merupakan ongkos dari simpul akar ke simpul i, yaitu panjang lintasan yang dilalui

$\hat{g}(i)$ merupakan estimasi ongkos dari simpul i ke simpul tujuan

C. *Priority Queue*

Antrian dalam pembangkitan simpul dalam penyelesaian permainan ular tangga ini diimplementasikan dalam struktur data *priority queue* dengan prioritas ditentukan oleh ongkos setiap simpul. Semakin kecil ongkos suatu simpul maka semakin tinggi prioritas simpul tersebut. Apabila kedua simpul memiliki ongkos yang sama, simpul dengan nilai *f* lebih kecil akan berada di depan. Jika nilai *f* pada kedua simpul masih bernilai sama, maka simpul dengan angka terbesar akan berada di depan. Simpul yang berada di paling depan antrian adalah simpul dengan prioritas terbesar atau ongkos paling minimum.

D. Algoritma Branch and Bound

Terdapat beberapa fungsi/prosedur bantuan untuk menjalankan prosedur Branch and Bound, yaitu

1. `Node(value: int, length: int) → Node`
   Fungsi ini digunakan untuk membuat simpul baru dengan value dan panjang lintasan dari simpul akar
2. `isGoal() → boolean`
   Fungsi ini mengembalikan `true` jika simpul tersebut merupakan simpul tujuan. Jika tidak maka akan mengembalikan `false`
3. `kill()`
   Prosedur untuk menghapus simpul yang memiliki ongkos lebih dari solusi yang ditemukan
4. `getValue(node: Node) → int`
   Fungsi untuk mendapatkan value atau angka dari simpul node
5. `getLength(node; Node) → int`
   Fungsi untuk mendapatkan panjang lintasan dari simpul akar ke simpul node
6. `pop() → Node`
   Fungsi untuk mengambil elemen pertama dari sebuah *priority queue*
7. `push(newNode: Node)`
   Prosedur untuk menambahkan simpul baru ke dalam *priority queue*
8. `get() → int`
   Fungsi untuk mendapatkan nilai dari sebuah key dalam map

Algoritma Branch and Bound pada makalah ini dapat dirangkum dalam *pseudocode* berikut

Algorithm 1 Procedure Branch and Bound

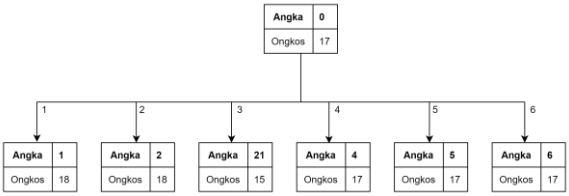
```
root ← Node(0, 0)
Q ← [root]
while Q is not empty do
    curr ← Q.pop()
    if curr isGoal() then
        solution ← current
        Q.kill()
    else
        i ← 1
        while i ≤ 6 do
            currVal ← getValue(curr) + i
            if currVal ≥ 100 then
                currVal ← 200 − currVal
            end if
            if currVal in config then
                currVal ← config.get(currVal)
            end if
            newNode ← Node(curr.getValue + i, curr.getLength + 1)
            if newNode.getLength ≤ solution.getLength then
                Q.push(newNode)
            end if
        end while
    end if
end while
```

Pada langkah pertama, simpul akar adalah simpul dengan angka 0 dan ongkos bernilai 17. Queue hanya berisi simpul akar.

Angka	**0**
Ongkos	17

Gambar 3. Simpul Akar

Karena simpul akar bukanlah simpul tujuan, maka simpul akar akan diekspansi dengan 6 kemungkinan berdasarkan angka mata dadu 1-6.

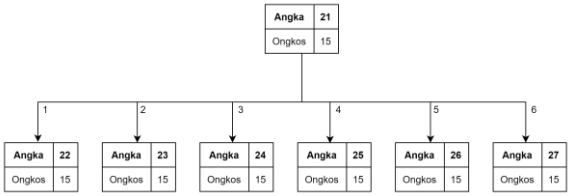


Gambar 4. Pembangkitan simpul akar

Ketika mendapatkan mata dadu 3, pemain dapat langsung berpindah ke kotak bernomor 21 dengan memanfaatkan tangga. Setelah simpul akar diekspansi, keenam simpul yang dihasilkan dimasukkan ke dalam Queue

Angka	21	6	5	4	2	1
Ongkos	15	17	17	17	18	18

Saat ini, Queue tidak kosong sehingga simpul dengan angka 21 dan ongkos 15 diambil. Simpul tersebut juga bukan merupakan simpul tujuan sehingga harus diekspansi.

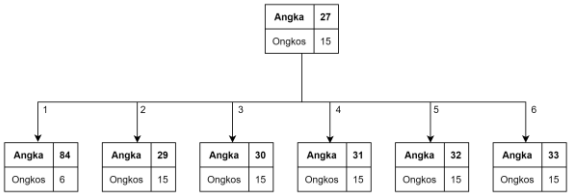


Gambar 5. Pembangkitan simpul angka 21

Kemudian keenam simpul hasil ekspansi dimasukkan ke dalam Queue.

Angka	27	26	25	24	23	22	6	...	1
Ongkos	15	15	15	15	15	15	17	...	18

Queue masih terisi, ambil simpul dengan angka 27 dan ongkos 15. Simpul tersebut juga bukan merupakan simpul tujuan sehingga perlu diekspansi lebih lanjut.

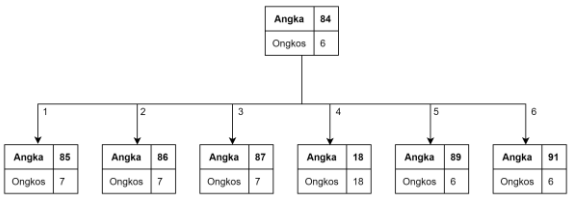


Gambar 6. Pembangkitan simpul angka 27

Apabila pemain mendapatkan mata dadu 1, pemain dapat berpindah ke kotak bernomor 84 dengan menggunakan tangga. Keenam simpul hasil ekspansi tersebut dimasukkan ke dalam Queue.

Angka	84	33	32	31	30	29	26	...	1
Ongkos	6	15	15	15	15	15	15	...	18

Queue masih terisi, ambil simpul dengan angka 84 dan ongkos 6. Simpul tersebut juga bukan merupakan simpul tujuan sehingga perlu diekspansi lebih lanjut.

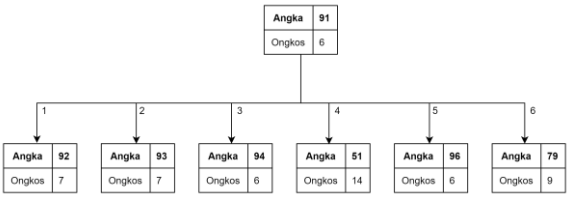


Gambar 7. Pembangkitan simpul angka 84

Apabila pemain mendapatkan mata dadu 4, pemain harus turun ke kotak bernomor 18. Keenam simpul hasil ekspansi tersebut dimasukkan ke dalam Queue.

Angka	91	89	87	87	85	33	32	...	1
Ongkos	6	6	6	7	7	15	15	...	18

Queue masih terisi, ambil simpul dengan angka 91 dan ongkos 6. Simpul tersebut juga bukan merupakan simpul tujuan sehingga perlu diekspansi lebih lanjut.

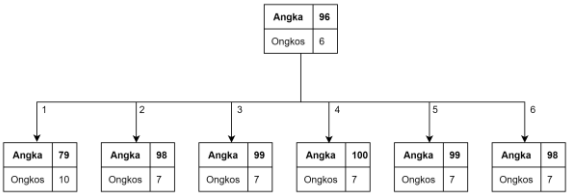


Gambar 8. Pembangkitan simpul angka 91

Keenam simpul hasil ekspansi tersebut dimasukkan ke dalam Queue.

Angka	96	94	89	87	93	92	86	...	1
Ongkos	6	6	6	6	7	7	7	...	18

Queue masih terisi, ambil simpul dengan angka 96 dan ongkos 6. Simpul tersebut juga bukan merupakan simpul tujuan sehingga perlu diekspansi lebih lanjut.

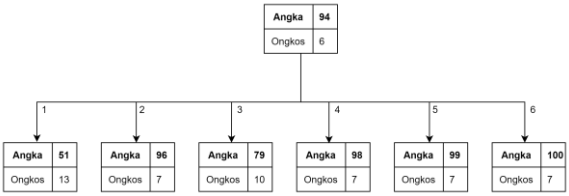


Gambar 9. Pembangkitan simpul angka 96

Keenam simpul hasil ekspansi tersebut dimasukkan ke dalam Queue.

Angka	94	89	87	100	99	99	98	...	1
Ongkos	6	6	6	7	7	7	7	...	18

Queue masih terisi, ambil simpul dengan angka 94 dan ongkos 6. Simpul tersebut juga bukan merupakan simpul tujuan sehingga perlu diekspansi lebih lanjut.

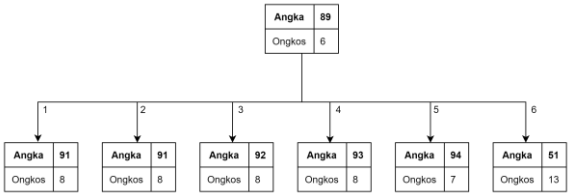


Gambar 10. Pembangkitan simpul angka 94

Keenam simpul hasil ekspansi tersebut dimasukkan ke dalam Queue.

Angka	89	87	100	100	99	99	98	...	1
Ongkos	6	6	7	7	7	7	7	...	18

Queue masih terisi, ambil simpul dengan angka 89 dan ongkos 6. Simpul tersebut juga bukan merupakan simpul tujuan sehingga perlu diekspansi lebih lanjut.

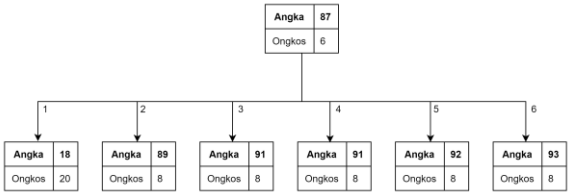


Gambar 11. Pembangkitan simpul angka 89

Keenam simpul tersebut dimasukkan ke dalam Queue.

Angka	87	100	100	99	99	99	98	...	1
Ongkos	6	7	7	7	7	7	7	...	18

Queue masih terisi, ambil simpul dengan angka 87 dan ongkos 6. Simpul tersebut juga bukan merupakan simpul tujuan sehingga perlu diekspansi lebih lanjut.



Gambar 12. Pembangkitan simpul angka 87

Keenam simpul tersebut dimasukkan ke dalam Queue.

Angka	100	100	99	99	99	98	98	...	1
Ongkos	7	7	7	7	7	7	7	...	18

Queue masih terisi, ambil simpul dengan angka 100 dan ongkos 7. Simpul yang diambil merupakan simpul tujuan sehingga pencarian dapat dihentikan.

Solusi tersebut dapat diperoleh dengan 6 langkah, yaitu 3 – 6 – 1 – 6 – 5 – 4. Solusi ini merupakan salah satu solusi yang paling optimal yang dapat ditemukan. Solusi tersebut dapat digambarkan sebagai berikut.

1. Bidak pemain belum masuk ke dalam board permainan, masih dalam simpul akar (angka 0).
2. Pemain berjalan 3 langkah menuju kotak nomor 3 dan menemukan tangga menuju kotak nomor 21. Saat ini pemain berada pada kotak nomor 21.
3. Pemain berjalan 6 langkah menuju kotak nomor 27.
4. Pemain berjalan 1 langkah menuju kotak nomor 28 dan dapat naik ke kotak nomor 84 melalui tangga yang menghubungkannya. Saat ini pemain berada pada kotak nomor 84.
5. Pemain berjalan 6 langkah menuju kotak nomor 90 dan naik ke kotak nomor 91 dengan tangga.
6. Pemain berjalan 5 langkah menuju kotak nomor 96.
7. Pemain berjalan 4 langkah menuju kotak tujuan bernomor 100.

Apabila algoritma ini dijalankan pada papan permainan ular tangga pada gambar 2 maka tidak akan menghasilkan solusi yang optimal. Solusi yang dihasilkan oleh algoritma branch and bound ini adalah 9 langkah, yaitu 4 – 6 – 6 – 6 – 6 – 6 – 6 – 6 – 2. Sedangkan solusi optimalnya hanya membutuhkan 6 langkah, yaitu 6 – 6 – 4 – 5 – 6 – 1. Algoritma Branch and Bound gagal mendapatkan langkah minimum yang paling optimal karena teknik heuristik dalam menghitung ongkos atau *cost* setiap simpul tidak tepat. Keberadaan tangga yang acak dan memiliki panjang yang berbeda-beda membuat pendekatan heuristik sulit dilakukan dalam penyelesaian permainan ular tangga dengan algoritma Branch and Bound.

IV. KESIMPULAN

Algoritma Branch and Bound dapat digunakan dalam mencari langkah penyelesaian pada permainan ular tangga. Pemenang dari permainan ular tangga ini adalah pemain yang tercepat dalam mencapai kotak yang memiliki angka maksimum. Pergerakan bidak ditentukan oleh guliran dadu. Terdapat 6 kemungkinan pergerakan yang bisa dilakukan oleh pemain. Hal ini menjadi dasar dalam pembentukan ruang status dalam Algoritma Branch and Bound. Ongkos setiap simpul dihitung dengan teknik heuristik. Ongkos dihitung berdasarkan banyaknya langkah dari simpul akar ke simpul tersebut ditambah dengan estimasi jumlah langkah dari simpul tersebut menuju simpul tujuan.

Algoritma Branch and Bound tidak menjamin memberikan solusi optimal pada penyelesaian permainan ular tangga. Hal ini disebabkan oleh ketidakoptimalan teknik heuristik yang digunakan. Keberadaan tangga yang acak dan memiliki panjang yang berbeda-beda membuat pendekatan heuristik sulit dilakukan.

Untuk mendapatkan solusi yang optimal, penulis menyarankan untuk menggunakan algoritma lain, seperti Algoritma Breadth First Search (BFS).

LINK VIDEO YOUTUBE

https://youtu.be/K-2FHiqSXVg

LINK SOURCE CODE

https://github.com/hafidznrg/Stima-UlarTangga

UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur kepada Allah Swt. karena atas rahmat dan karunia-Nya makalah ini dapat diselesaikan dengan baik dan tepat waktu. Penulis mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen mata kuliah IF2211 Strategi Algoritma Kelas K03. Tidak lupa, penulis juga mengucapkan terima kasih kepada kedua orang tua dan pihak-pihak yang telah membantu dalam pengerjaan makalah ini.

REFERENSI

[1] M. Rinaldi, M. Nur Ulfa, K. Masayu Leylia. 2021. *Algoritma Branch and Bound (Bagian 1)*. Diakses pada laman https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf pada tanggal 21 Mei 2022.

[2] W. Herbowo dan S. Bambang. 2020. *Permainan Ular Tangga dan Manfaatnya bagi Anak Usia Dini*. Diakses pada laman https://sabyan.org/permainan-ular-tangga-dan-manfaatnya-bagi-anak-usia-dini/ pada tanggal 21 Mei 2022.

[3] Anonim. Diakses pada laman https://www.howtofixx.com/snakes-and-ladders-template/ pada tanggal 21 Mei 2022.

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, 20 Mei 2022

Hafidz Nur Rahman Ghozali
13520117