Penggunaan Algoritma Branch And Bound Untuk Persoalan Teka-Teki River Crossing: Petani, Serigala, dan Domba

Rifaldy Aristya Kelana
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
rifaldyaristya23@gmail.com

***Abstrak*—Teka-teki adalah permainan yang membutuhkan kreativitas untuk dipecahkan sehingga teka-teki menuntut untuk dapat diselesaikan dengan cara yang optimal dan tepat. Salah satu teka-teki yang cukup populer adalah teka-teki river crossing yang banyak jenisnya. Oleh karena itu, untuk mencapai solusi dengan langka dan jumlah simpul ekspansi minimum dapat digunakan algoritma pada ilmu komputer untuk menyelesaikannya. Dalam makalah ini, penulis akan menyelesaikan salah satu teka-teki river crossing, yaitu teka-teki petani, domba, dan serigala dengan salah satu algoritma untuk persoalan optimasi yang cukup populer dalam ilmu komputer, yaitu branch and bound yang berbasis breadth first search .**

Keywords—teka-teki, optimasi, branch and bound, river crossing

I. Introduction

Teka-teki petani, domba, dan serigala adalah teka-teki yang berbentuk puzzle yaitu teka-teki yang solusinya berupa urutan langkah untuk mencapai keadaan solusi yang diharapkan. Pada teka – teki ini terdapat seorang petani yang mempunyai dua ekor domba dan seekor serigala dan berada pada tepi sungai. Petani ini ingin menyeberang ke tepi sungai yang lain dengan menggunakan perahu. Permasalahan dari teka-teki ini adalah perahu yang digunakan hanya dapat memuat maksimal 2 penumpang dan serigala akan memakan domba jika keduanya berada pada tempat yang sama tetapi di tempat tersebut tidak ada petani. Keadaan solusi yang dihasilkan dari teka-teki ini adalah petani, serigala, dan domba berada pada tepi sungai yang lain dengan tidak ada domba yang termakan oleh serigala dengan langkah seminimal mungkin.

Gambar 1. Ilustrasi Persoalan Teka Teki River Crossing

Sumber : https://brightside.me/wonder-quizzes/10-cunning-puzzles-to-mess-with-your-mind-663610/

II. Dasar Teori

A. *Persoalan Optimasi*

Persoalan Optimasi biasanya melibatkan pemecahan solusi dengan biaya seminimal atau keuntungan semaksimal mungkin namun tetap menghasilkan solusi yang tepat. Berikut adalah beberapa contoh persoalan optimasi adalah

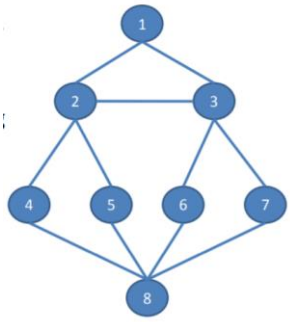
1. Knapsack Problem. Aspek yang ingin dioptimasi adalah keuntungan maksimum dari objek-objek yang dimasukkan ke dalam ransel. Batasan dari solusi adalah objek-objek yang dimasukkan ke dalam ransel jumlah seluruh beratnya tidak melebihi kapasitas maksimum berat yang dapat ditampung oleh ransel.
2. Travelling Salesperson Problem. Aspek yang ingin dioptimasi adalah jarak minimum yang ditempuh untuk mencapai solusi. Batasan dari solusi yang dihasilkan adalah seluruh simpul telah dikunjungi tepat sekali dan kembali ke simpul awal.
3. Graph Coloring Problem. Aspek yang ingin dioptimasi adalah jumlah warna yang dipakai untuk mewarnai seluruh simpul pada peta. Batasan dari solusi yang dihasilkan adalah seluruh simpul yang bertetangga diwarnai dengan warna yang berbeda.

Selain persoalan-persoalan di atas, kita dapat juga memandang persoalan optimasi dari sudut pandang yang lain, seperti jumlah simpul yang diekspansi minimal untuk menemukan solusi dengan cara memilih simpul yang paling mengarah ke solusi untuk diekspansi terlebih dahulu dan juga membunuh simpul yang sudah tidak mengarah ke solusi atau melanggar batasan dari solusi yang ditemukan

B. *Breadth First Search*

Breadth First Search atau pencarian melebar adalah penelusuran simpul dengan mengunjungi semua simpul yang bertetangga dengan simpul yang sedang diekspansi terlebih dahulu, berbeda dengan Depth First Search yang menelusuri satu simpul yang bertetangga dengannya lalu anaknya terus menerus hingga menemukan solusi.

Jika Breadth First Search digambarkan dengan antrian simpul hidup, dimana simpul-simpul dimasukkan dengan aturan First In First Out, dimana simpul-simpul yang pertama dibangkitkan simpul tersebutlah yang akan pertama kali diekspansi dengan kata lain simpul-simpul akan dimasukkan pada posisi terakhir dari antrian simpul hidup.



Gambar 2. Pohon Ruang Status Algoritma BFS beserta urutan ekspansi simpulnya
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/BFS-dan-DFS-(2020).pdf

Beberapa aspek yang dapat ditinjau dari algoritma BFS adalah sebagai berikut:

1. BFS menjamin solusi akan ditemukan jika solusi ada selama jumlah percabangan terbatas.
2. BFS menjamin langkah yang ditempuh untuk mendapatkan simpul solusi adalah minimum.
3. Kompleksitas waktu dari algoritma BFS adalah $O(b^d)$ dengan b adalah jumlah percabangan dari tiap simpul dan d adalah kedalaman pohon untuk mencapai simpul solusi.
4. Komplesitas ruang dari algoritma BFS juga adalah $O(b^d)$.

C. Branch And Bound

Branch and Bound adalah modifikasi dari algoritma BFS, dengan diterapkannya batasan (bound) pada setiap simpul. Batasan dari simpul dapat disebabkan oleh biaya simpul tersebut tidak memiliki biaya yang lebih baik daripada biaya solusi yang telah ditemukan atau juga pemangkasan karena simpul tersebut tidak mengarah pada solusi.

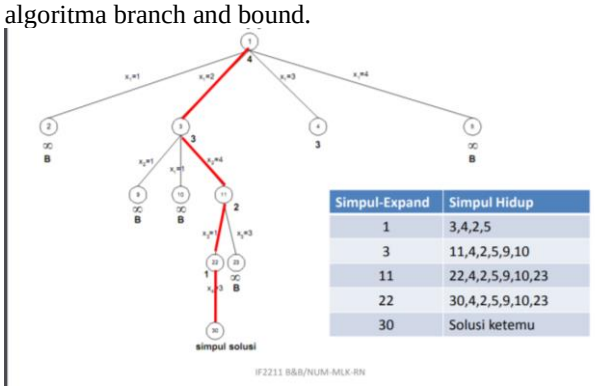
Selain digunakan untuk pemangkasan, nilai dari biaya tiap simpul juga dimanfaatkan untuk least cost search, sehingga penentuan simpul yang akan diekspansi tidak berdasarkan urutan masuk dari simpul tersebut ke dalam antrian, namun berdasarkan biaya dari simpul. Semakin kecil biaya dari suatu simpul, maka semakin dahulu simpul tersebut akan diekspansi.

Oleh karena itu terjadi modifikasi dari sistem antrian simpul hidup, yaitu dari antrian dengan prinsip First In First Out menjadi priority queue dengan prioritasnya adalah biaya dari masing-masing simpul yang berada di antrian tersebut. Dengan diterapkan prinsip pemangkasan dan least cost search maka akan dihasilkan solusi lebih cepat, atau dengan kata lain jumlah simpul yang diekspansi menjadi seminimal mungkin.

Beberapa persoalan yang solusinya didapatkan dengan menerapakan algoritma Branch and Bound adalah sebagai berikut:

1. Persoalan N-Ratu. Persoalan N-Ratu adalah pencarian solusi untuk meletakkan ratu pada papan 4x4 dimana ratu-ratu yang diletakkan tidak boleh berada pada satu baris atau 1 kolom atau diagonal yang sama dengan ratu-ratu lainnya. Prinsip branch and bound yang diterapkan adalah pemangkasan terhadap simpul-simpul yang melanggar batasan pada simpul solusi yaitu ditemukan minimal satu ratu lain yang berada pada satu baris atau satu kolom atau diagonal yang sama dengan posisi ratu pada simpul tersebut. Least Cost Search diterapkan dengan definisi biaya dari setiap simpul adalah panjang lintasan yang perlu ditempuh dari simpul tersebut untuk mencapai simpul solusi.
2. 15-Puzzle. Persoalan 15-Puzzle adalah menentukan urutan langkah pergeseran kotak kosong untuk menjadikan puzzle dengan posisi tiap nomor sesuai dengan posisi pada solusi. Prinsip branch and bound yang dipakai adalah least cost search dengan biaya dari tiap simpul ditentukan dari pertambahan jarak dari simpul akar ke simpul tersebut dan nilai perkiraan langkah dari simpul tersebut menuju simpul solusi. Perkiraan ini dihasilkan dari jumlah posisi nomor yang berbeda dengan posisi nomor tersebut pada puzzle solusi.
3. Travelling Salesperson Problem adalah persoalan untuk menentukan jarak terpendek untuk mengunjungi semua simpul dari simpul akar lalu kembali ke simpul asal. Prinsip branch and bound yang dipakai adalah least cost search dengan nilai biaya dari setiap simpul ditentukan dari jarak dari simpul akar ditambah pengurangan baris dan kolom yang diperlukan untuk membentuk reduced cost matrix dan jarak dari simpul tersebut dari simpul parentnya. Metode lain untuk menghitung biaya simpul adalah dengan menggunakan metode tur lengkap dengan menggunakan pertambahan dua sisi minimum dengan sisi yang dipilih wajib dimasukkan untuk menjadi nilai sisi yang dipilih lalu dibagi dua. Sedangkan pemangkasan dilakukan setelah ditemukan solusi terhadap simpul-simpul yang mempunyai biaya yang lebih besar dari biaya solusi terbaik.
4. Knapsack Problem adalah persoalan mengoptimalkan profit dengan tetap memenuhi bobot maksimum dari objek-objek yang dipilih. Prinsip branch and bound yang dipakai antara lain adalah prinsip least cost search dengan perhitungan biaya yang dipakai adalah kalkulasi biaya dari pemilihan objek dengan mengikutsertakan objek yang sedang dipilih. Pemangkasan dilakukan setelah menemukan solusi terhadap simpul-simpul yang memiliki biaya lebih besar daripada biaya simpul solusi yang telah dihasilkan atau total bobot objek yang dipilih melebihi kapasitas ransel.

Jadi dapat disimpulkan bahwa penerapan algoritma BFS memang menjamin bahwa jumlah langkah yang dari solusi akan minimal, dengan demikian menjamin jika ditinjau dari segi langkah solusi maka sudah optimal, namun pencarian dengan BFS dapat memakan waktu sangat lama, karena jumlah simpul yang akan diekspansi akan sangat banyak sehingga pencarian simpul solusi menjadi tidak optimal. Permasalahan inilah yang dapat diselesaikan oleh algoritma branch and bound.



Simpul-Expand	Simpul Hidup
1	3,4,2,5
3	11,4,2,5,9,10
11	22,4,2,5,9,10,23
22	30,4,2,5,9,10,23
30	Solusi ketemu

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf

Gambar 3. Ilustrasi algoritma Branch and Bound (pada persoalan 4-Ratu)

III. Implementasi Algoritma Branch and Bound pada Penyelesaian Teka-teki River Crossing

A. *Komponen-komponen branch and bound yang dipakai*

Pengaplikasian algoritma branch and bound untuk penyelesaian teka – teki river crossing dipilih karena permasalahan teka-teki ini memiliki komponen pemangkasan dan least cost search yang adalah ciri khas dari algoritma branch and bound. Ditinjau dari segi pemangkasan, pemangkasan dilakukan karena ada dua alasan:

1. Batasan dari solusi. Batasan yang diterapkan terhadap solusi adalah solusi yang didapatkan untuk menyeberangi semua makhluk hidup dari satu tepi sungai ke tepi sungai yang lain adalah tidak ada satu pun dari antaranya yang termakan oleh yang lainnya. Hal ini dapat terjadi jika pada rute yang ditempuh terdapat langkah yang membuat pada satu sisi sungai terdapat domba dan juga serigala namun di sisi tersebut tidak terdapat petani.
2. Jika pada suatu tahap ingin menyeberangi hewan tertentu dengan perahu namun pada sisi tersebut tidak terdapat hewan tersebut atau dengan kata lain hewan tersebut sudah berada pada sisi lain. Syarat ini adalah syarat yang bersifat hal umum.

Ditinjau dari segi least cost search, simpul dengan biaya lebih kecil akan diletakkan lebih depan pada priority queue, dengan kata lain simpul tersebut akan terlebih dahulu diekspansi.

Penghitungan biaya dari setiap simpul merupakan perkiraan jarak untuk sampai ke simpul solusi. Cara menghitung perkiraan ini adalah sebagai berikut:

1. Jika perahu berasal dari tepi asal menuju tepi tujuan, maka biaya dihitung dari jumlah komponen pada tepi asal dikurangi jumlah komponen dalam perahu
2. Jika perahu berasal dari tepi tujuan, maka biaya dihitung dari jumlah komponen pada tepi asal ditambah jumlah komponen dalam perahu.

Simpul pertama kali yang menjadi simpul akar diasumsikan adalah simpul dengan jumlah komponen dalam perahu 0 dan berasal dari tepi tujuan. Simpul tersebut adalah simpul dummy.

B. *Pohon ruang status untuk mencapai solusi*

Notasi yang dipakai untuk penulisan simpul adalah (<a,b,c> <d,e,f> <g,h,i> <id> <dariAsal> <cost> [ ]). a, b, dan c berurutan adalah jumlah domba, jumlah serigala, dan jumlah petani dari tepi sungai awal. d, e, f berurutan adalah jumlah domba, jumlah serigala, dan jumlah petani dari perahu. g, h , i berurutan adalah jumlah domba, jumlah serigala, dan jumlah petani dari tepi sungai tujuan. id adalah nomor untuk dari setiap simpul yang dimaksudkan agar penulisan simpul orang tua dari suatu simpul menjadi lebih mudah. dariAsal adalah True jika dia berangkat dari tepi asal ke tepi tujuan dan bernilai False jika berasal dari tepi tujuan ke tepi asal. [ ] adalah himpunan id dari simpul-simpul orang tua dari simpul yang bersangkutan.

Iterasi	Simpul Ekspan	Simpul Hidup	Status
1	(<2,1,1><0,0,0><0,0,0><A> <False><4>[ ])	(<2,1,1>,<0,1,1>,<0,0,0><C><True><2>[A]) (<2,1,1>,<1,0,1>,<0,0,0><D><True><2>[A]) (<2,1,1><0,0,1><0,0,0><B><True><3>[A])	Pass
2	(<2,1,1>,<0,1,1>,<0,0,0><C><True><2>[A])	(<2,1,1>,<1,0,1>,<0,0,0><D><True><2>[A]) (<2,0,0>,<0,0,1>,<0,1,1><E><False><3>[A-C]) (<2,1,1><0,0,1><0,0,0><B><True><3>[A]) (<2,0,0>,<1,0,1>,<0,1,1><F><False><4>[A-C])	Pass
3	(<2,1,1>,<1,0,1>,<0,0,0><D><True><2>[A])	(<2,0,0>,<0,0,1>,<0,1,1><E><False><3>[A-C]) (<2,1,1><0,0,1><0,0,0><B>	Bound

		<True><3>[A]) (<2,0,0>,<1,0,1>,<0,1,1><F><False><4>[A-C])	
4	(<2,0,0>,<0,0,1>,<0,1,1><E><False><3>[A-C])	(<2,0,1>,<0,1,1>,<0,1,0><G><True><1>[A-C-E]) (<2,0,1>,<1,0,1>,<0,1,0><H><True><1>[A-C-E]) (<2,1,1><0,0,1><0,0,0><B><True><3>[A]) (<2,0,0>,<1,0,1>,<0,1,1><F><False><4>[A-C])	Pass
5	(<2,0,1>,<0,1,1>,<0,1,0><G><True><1>[A-C-E])	(<2,0,1>,<1,0,1>,<0,1,0><H><True><1>[A-C-E]) (<2,1,1><0,0,1><0,0,0><B><True><3>[A]) (<2,0,0>,<1,0,1>,<0,1,1><F><False><4>[A-C])	Bound
6	(<2,0,1>,<1,0,1>,<0,1,0><H><True><1>[A-C-E])	(<1,0,0>,<0,0,1>,<1,1,1><I><False><2>[A-C-E-H]) (<1,0,0>,<0,1,1>,<1,1,1><J><False><3>[A-C-E-H]) (<2,1,1><0,0,1><0,0,0><B><True><3>[A]) (<2,0,0>,<1,0,1>,<0,1,1><F><False><4>[A-C])	Pass
7	(<1,0,0>,<0,0,1>,<1,1,1><I><F><2>[A-C-E-H])	(<1,0,0>,<0,1,1>,<1,1,1><J><False><3>[A-C-E-H]) (<2,1,1><0,0,1><0,0,0><B><True><3>[A]) (<2,0,0>,<1,0,1>,<0,1,1><F><False><4>[A-C])	Bound
8	(<1,0,0>,<0,1,1>,<1,1,1><J><False><3>[A-C-E-H])	(<1,1,1>,<1,0,1>,<1,0,0><L><True><1>[A-C-E-H-J]) (<1,1,1>,<0,0,1>,<1,0,0><K><True><2>[A-C-E-H-J]) (<2,1,1><0,0,1><0,0,0><B><True><3>[A]) (<2,0,0>,<1,0,1>,<0,1,1><F	Pass

		><False><4>[A-C])	
9	(<1,1,1>,<1,0,1>,<1,0,0><L><True><1>[A-C-E-H-J])	(<0,1,0>,<0,0,1>,<2,0,1><M><False><2>[A-C-E-H-J-L]) (<1,1,1>,<0,0,1>,<1,0,0><K><True><2>[A-C-E-H-J]) (<0,1,0>,<0,1,1>,<2,0,1><N><False><3>[A-C-E-H-J-L]) (<2,1,1><0,0,1><0,0,0><B><True><3>[A]) (<2,0,0>,<1,0,1>,<0,1,1><F><False><4>[A-C])	Pass
10	(<0,1,0>,<0,0,1>,<2,0,1><M><False><2>[A-C-E-H-J-L])	(<0,1,1>,<0,1,1>,<2,0,0><O><True><0>[A-C-E-H-J-L-M]) (<0,1,1>,<1,0,1>,<2,0,0><P><True><0>[A-C-E-H-J-L-M]) (<1,1,1>,<0,0,1>,<1,0,0><K><True><2>[A-C-E-H-J]) (<0,1,0>,<0,1,1>,<2,0,1><N><False><3>[A-C-E-H-J-L]) (<2,1,1><0,0,1><0,0,0><B><True><3>[A]) (<2,0,0>,<1,0,1>,<0,1,1><F><False><4>[A-C])	Pass
11	(<0,1,1>,<0,1,1>,<2,0,0><O><True><0>[A-C-E-H-J-L-M])	(<0,1,1>,<1,0,1>,<2,0,0><P><True><0>[A-C-E-H-J-L-M]) (<1,1,1>,<0,0,1>,<1,0,0><K><True><2>[A-C-E-H-J]) (<0,1,0>,<0,1,1>,<2,0,1><N><False><3>[A-C-E-H-J-L]) (<2,1,1><0,0,1><0,0,0><B><True><3>[A])	Solusi ketemu

		(<2,0,0>,<1,0,1>,<0,1,1><F><False><4>[A-C])	

Tabel 1. Pohon ruang status

Rekonstruksi solusi dari pohon ruang status yang terbentuk. Dari pohon ruang status yang telah terbentuk, didapatkan bahwa N adalah simpul solusi. Dengan menelusuri simpul-simpul orang tua dari N maka didapatkan Langkah solusi. Solusinya adalah A-C-D-G-I-K-L-N, yaitu

Langkah	Tepi Sungai Asal	Perahu	Tepi Sungai Tujuan	Arah
A (dummy)	domba : 2 serigala: 1 petani: 1	domba : 0 serigala: 0 petani: 0	domba : 0 serigala: 0 petani: 0	Tujuan-Awal
C	domba : 2 serigala: 1 petani: 1	domba : 0 serigala: 1 petani: 1	domba : 0 serigala: 0 petani: 0	Awal-Tujuan
E	domba : 2 serigala: 0 petani: 0	domba : 0 serigala:0 petani: 1	domba : 0 serigala: 1 petani: 1	Tujuan-Awal
H	domba : 2 serigala: 0 petani: 1	domba : 1 serigala: 0 petani: 1	domba : 0 serigala: 1 petani: 0	Awal-Tujuan
J	domba : 1 serigala: 0 petani: 0	domba : 0 serigala: 1 petani: 1	domba : 1 serigala: 1 petani: 1	Tujuan-Awal
L	domba : 1 serigala: 1 petani: 1	domba : 1 serigala: 0 petani: 1	domba : 1 serigala: 0 petani: 0	Awal-Tujuan
M	domba : 0 serigala: 1 petani: 0	domba : 0 serigala: 0 petani: 1	domba : 2 serigala: 0 petani: 1	Tujuan-Awal
O	domba : 0 serigala: 1 petani: 1	domba : 0 serigala: 1 petani: 1	domba : 2 serigala: 0 petani: 0	Awal-Tujuan

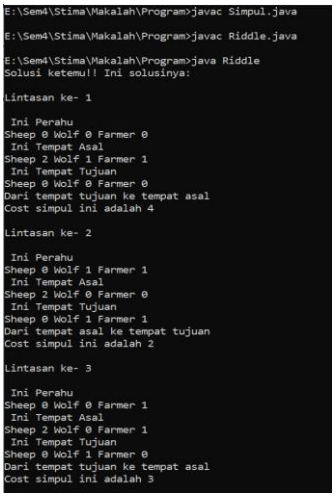
Tabel 2. Solusi lintasan

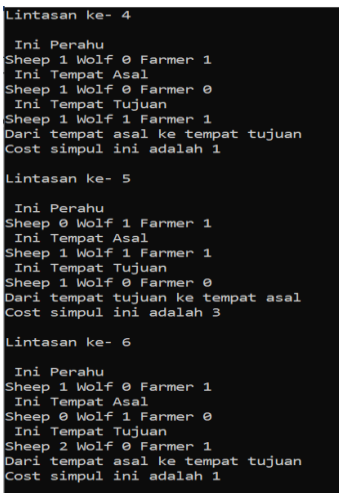
C. Fungsi- fungsi penting beserta penjelasannya

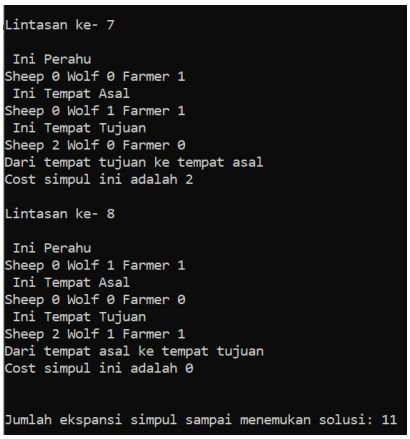
Ada beberapa hal penting dalam pengimplementasian algoritma branch and bound dalam menemukan solusi dari persoalan teka-teki river crossing ini:

1. Fungsi validasi yang digunakan untuk mengecek kevalidan suatu simpul sebelum simpul tersebut di ekspansi. Validasi terdiri dari pengecekan apakah dengan pemindahan komponen-komponen tersebut, nilai dari komponen pada suatu tempat menjadi negatif atau jumlah domba pada satu tempat tidak nol dan jumlah serigala pada tempat tersebut juga tidak nol, namun jumlah petani nol.
2. Fungsi ekspansi adalah fungsi yang dipanggil setelah simpul tersebut divalidasi (pass). Fungsi ini akan memperbaharui data jumlah tiap komponen di kedua sisi sesuai dengan data komponen dari perahu dan memanggil fungsi generateChild
3. Fungsi generateChild adalah fungsi yang dipanggil untuk membangkitkan anak-anak dari suatu simpul. Simpul-simpul dibangkitkan dengan kombinasi komponen-komponen pada perahu. Urutan kombinasi tersebut adalah
    1. Jumlah domba : 0
Jumlah serigala : 0
Jumlah petani : 1
    2. Jumlah domba : 0
Jumlah serigala : 1
Jumlah petani : 1
    3. Jumlah domba : 1
Jumlah serigala : 0
Jumlah petani : 1
4. Fungsi compareTo adalah fungsi untuk mengurutkan simpul. Pada fungsi ini, simpul-simpul pada array simpul hidup diurutkan berdasarkan biayanya. Untuk simpul dengan biaya yang sama diurutkan berdasarkan jumlah simpul orang tuanya. Semakin kecil biaya dan semakin banyak jumlah simpul orang tua suatu simpul, semakin tinggi prioritas simpul tersebut untuk diekspansi.
5. Fungsi sort akan dipanggil setiap suatu simpul selesai membangkitkan seluruh simpul-simpul anaknya.

D. Output dari program penyelesaian teka teki beserta penjelasannya







Gambar 4. Screenshoot output dari program

Program dapat dilihat pada https://github.com/rifaldyaristya/RiverCrossing

Seperti dapat dilihat dalam program, jumlah ekspansi simpul adalah 11 sesuai dengan yang tertera pada Tabel 1, namun ada sedikit perbedaan pada hasil solusi dengan solusi yang tertera pada tabel 2. Pada tabel 2 simpul tertera nilai-nilai saat komponen dalam perahu belum dipindahkan, sedangkan keluaran yang tertera dalam program adalah keadaan dimana komponen dalam perahu sudah dipindahkan. Meskipun demikian, keduanya menggambarkan langkah-langkah solusi yang sama.

LINK VIDEO DI YOUTUBE

https://youtu.be/hXapKl6t6oo

UCAPAN TERIMA KASIH

Saya mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena dengan anugerah-Nya, saya berhasil menyelesaikan makalah ini. Selain itu, saya mengucapkan terima kasih kepada dosen Strategi Algoritma saya, Dr. Masayu Leylia Khodra., ST., MT. dan seluruh tim dosen Strategi Algoritma. Terakhir, saya mau mengucapkan terima kasih untuk beberapa sumber yang telah menjadi referensi atas makalah yang saya buat ini, yaitu referensi-referensi yang tercantum pada bagian referensi.

REFERENSI

[1] https://brightside.me/wonder-quizzes/10-cunning-puzzles-to-mess-with-your-mind-663610/

[2] https://www.youtube.com/watch?v=HCp_eN6JSac&t=185s

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/BFS-dan-DFS-(2020).pdf

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, 4 Mei 2020

Rifaldy Aristya Kelana - 13518082