Pembuatan Permainan Sudoku Acak Menggunakan Algoritma Decrease and Conquer Untuk Memastikan Keunikan Solusi

Michael Utama - 13521137
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521137@std.stei.itb.ac.id

***Abstract*— Sudoku merupakan permainan logika sederhana yang menuntut pemain mengisikan angka satu sampai sembilan supaya tiap baris, kolom, dan sembilan subkotak berukuran 3x3 mengandung angka satu sampai sembilan. Meskipun tidak ada peraturan yang mengharuskan keunikan solusi, pada sudoku klasik hampir seluruh teka-teki yang dibuat memiliki solusi unik. Oleh karena itu, perlu dirancang algoritma yang efisien dalam pencarian teka-teki sudoku yang memiliki solusi tunggal secara acak. Dalam penelitian ini, akan digunakan algoritma *backtracking* untuk mencari solusi akhir, dan pencarian banyak kotak yang dapat dikosongkan menggunakan algoritma *decrease and conquer* dengan metode *decrease by a constant* dan *decrease by a constant factor*.**

Keywords—sudoku; puzzle; decrease and conquer; uniqueness

I. PENDAHULUAN

Sudoku adalah permainan logika yang mengharuskan pemain mengisi angka satu sampai sembilan (1-9) pada seluruh kotak kosong sehingga setiap baris, kolom, dan seluruh subkotak berukuran 3x3 berisi seluruh nilai dari satu sampai sembilan. Akibatnya, pada sebuah baris, kolom, atau subkotak berukuran 3x3 tidak boleh terdapat satupun nilai duplikat.

			2		9	5		
	5	3				4		
9								
6						1		4
				3		2	8	
5			8			3	9	
4				2				
3		2	4	5		6		
					8			

Gambar 1. Contoh permainan sudoku

Sumber: https://www.puzzles.ca/sudoku_puzzles/sudoku_hard_1033.html

Permainan sudoku dibagi menjadi beberapa tingkat kesulitan. Surat kabar umumnya membagi tingkat kesulitan sudoku berdasarkan banyak kotak yang perlu diisi pemain. Hal tersebut umumnya benar khususnya bagi penyelesaian sudoku menggunakan komputer, namun tidak dapat menggambarkan kesulitan sudoku bagi pemain manusia. Saat ini, banyak penyedia *puzzle* juga memasukkan kesulitan teknik yang diperlukan dalam perhitungan tingkat kesulitan sudoku.

Secara umum, sebuah puzzle sudoku memiliki solusi tunggal. Artinya, ada tepat satu kombinasi angka 1-9 pada kotak kosong yang dapat memenuhi aturan permainan sudoku. Jika sudoku memiliki solusi tidak tunggal, pemain akan dipaksa menebak jawaban yang merupakan cara kurang baik dalam penyelesaian sudoku. Meskipun tidak ditulis pada aturan secara eksplisit, keunikan *puzzle* sudoku telah dianggap pasti berlaku oleh pemain sehingga muncul teknik-teknik penyelesaian yang memanfaatkan keunikan tersebut.

Sudoku yang memiliki tingkat kesulitan tinggi dan hanya memiliki solusi tunggal sulit ditemukan secara manual karena keduanya memiliki ciri-ciri umum yang bertolak belakang. Di satu sisi, sudoku yang memiliki tingkat kesulitan tinggi umumnya memiliki banyak kotak yang harus diisi pemain, sedangkan semakin banyak kotak kosong pada sebuah sudoku, semakin tinggi kemungkinan permainan tersebut memiliki banyak solusi. Akibatnya, perlu dilakukan pencarian terus-menerus agar permainan yang dihasilkan memiliki kesulitan tinggi dan memiliki solusi unik.

Perlunya pencarian terus menerus tersebut membuat algoritma efisien yang dapat mencari permainan sudoku dengan solusi tunggal semakin penting. Algoritma *backtracking* dapat digunakan untuk mencari solusi, membuat papan solusi acak, serta memastikan solusi permainan unik, sedangkan untuk membuat sebuah permainan sudoku dengan beberapa kotak kosong diperlukan algoritma tambahan, seperti algoritma *decrease and conquer* yang penggunaannya akan dibahas dalam makalah ini.

II. LANDASAN TEORI

A. Algoritma Runut-balik

Algoritma runut-balik merupakan modifikasi *exhaustive search* yang lebih mangkus, karena pencarian yang tidak mengarah ke solusi tidak akan dilanjutkan [1]. Algoritma yang pertama kali diperkenalkan oleh Derrick H. Lehmer ini dapat digunakan baik pada persoalan optimasi maupun non-optimasi. Properti umum yang dimiliki algoritma runut-balik:

1. Solusi persoalan

   Umumnya dinyatakan sebagai vektor n-dimensi. Contoh: $X = (x_1, x_2, ..., x_n)$ dengan $x_i \in S_i$

2. Fungsi pembangkit

   Umumnya dinyatakan sebagai predikat $T(x_1, x_2, ..., x_{n-1})$ yang membangkitkan nilai-nilai untuk x_n.

3. Fungsi pembatas

   Umumnya dinyatakan sebagai predikat $B(x_1, x_2, ..., x_k)$ yang bernilai true jika dan hanya jika $(x_1, x_2, ..., x_k)$ mungkin mengarah ke sebuah solusi. Kata mungkin dapat didefinisikan tergantung model yang dibuat.

Selain itu, terdapat beberapa terminologi seputar algoritma runut-balik seperti

1. Ruang solusi (*solution space*)

   Ruang solusi merupakan seluruh kemungkinan solusi dari persoalan.

2. Pohon ruang status

   Pohon ruang status merupakan sebuah pohon yang simpulnya menandakan status persoalan, sedangkan sisi-sisinya diberi label sesuai hasil fungsi pembangkit.

3. Simpul hidup (*live node*)

   Simpul hidup merupakan simpul yang sudah dibangkitkan.

4. Simpul ekspan (*expand node*)

   Simpul ekspan merupakan simpul hidup yang sedang diperluas. Perluasan tersebut akan menambah jumlah simpul hidup.

5. Simpul mati (*dead node*)

   Simpul mati merupakan simpul hidup yang telah dimatikan oleh fungsi pembatas.

Pencarian solusi dengan algoritma runut-balik mengikuti aturan pembangkitan simpul sesuai *depth first search* (DFS) [1]. Sebelum diperluas, simpul ekspan diperiksa terlebih dahulu dengan fungsi pembatas. Apabila hasil fungsi pembatas *false*, pembangkitan simpul tidak dilanjutkan pada simpul ekspan tersebut. Apabila hasil fungsi pembatas *true*, perluasan simpul akan dilanjutkan seperti aturan DFS umumnya.

Algoritma runut-balik dapat digunakan untuk mencari seluruh kemungkinan solusi jika ruang solusi berhingga. Algoritma ini juga dapat digunakan untuk mencari hanya satu solusi, dengan menghentikan pencarian apabila ditemukan satu solusi persoalan.

B. Algoritma Decrease and Conquer

Algoritma *decrease and conquer* merupakan algoritma penyelesaian masalah dengan mereduksi persoalan menjadi beberapa *subproblem*, lalu memproses salah satunya [2]. Algoritma *decrease and conquer* terdiri atas dua tahap, yaitu *decrease* (mereduksi persoalan) dan *conquer* (menyelesaikan beberapa, namun tidak semua *subproblem*). Algoritma *decrease and conquer* terbagi menjadi tiga jenis sesuai ukuran reduksinya:

1. *Decrease by a constant*

   Ukuran persoalan direduksi berdasarkan sebuah konstanta yang umumnya bernilai satu. Contoh algoritma: *insertion sort.*

2. *Decrease by a constant factor*

   Ukuran persoalan direduksi sebesar faktor konstan, umumnya bernilai 2. Contoh algoritma: *binary search, ternary search.*

3. *Decrease by a variable size*

   Ukuran persoalan direduksi, besar reduksi berbeda tergantung kondisi pada iterasi. Contoh algoritma: *interpolation search*.

C. Permainan Sudoku

Menurut kamus Britannica, permainan sudoku dalam versi paling sederhana terdiri atas kotak 9x9 yang sebagian terisi dengan angka; objektif sudoku adalah mengisi kotak yang belum terisi sehingga tiap baris, kolom, dan sembilan *subgrid* 3x3 berisi angka 1-9 tepat sekali.

Pada permainan sudoku, terdapat beberapa tingkat kesulitan yang dipengaruhi kesulitan teknik yang diperlukan untuk memecahkan sebuah permainan [3], yaitu

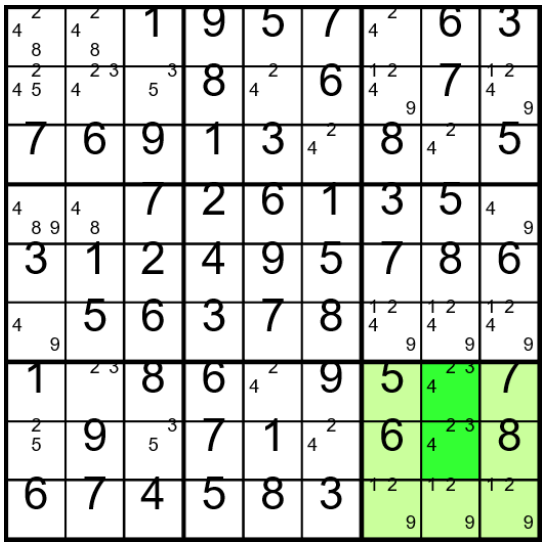
1. Tingkat mudah

   Umumnya permainan sudoku pada tingkat mudah dapat diselesaikan dengan hanya teknik *single position* dan *single candidate*. Kedua teknik tersebut bekerja dengan mengurangi kemungkinan posisi atau kandidat menjadi satu.

2. Tingkat sedang

   Permainan sudoku tingkat sedang dapat diselesaikan dengan teknik pada tingkat mudah, ditambah *candidate lines*, *double pairs*, dan *multiple lines*. Ketiga teknik tersebut melakukan eliminasi *possible values* pada sebuah kotak dengan memanfaatkan sifat *possible values* pada kotak lain.

   Sebagai contoh, pada gambar 2 kotak 3x3 bawah kanan, nilai 4 hanya mungkin terdapat pada kolom tengah (kolom ke-2 dari kanan). Akibatnya, nilai 4 tidak mungkin dimiliki kotak lain pada kolom ke-2 dari kanan selain pada kotak 3x3 di bawah kanan [4].

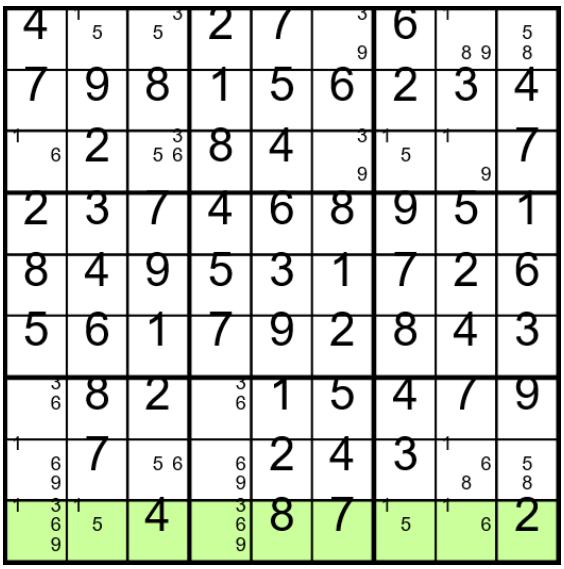


Gambar 2. Penggunaan teknik *candidate lines*

Sumber: https://www.sudokuoftheday.com/techniques/candidate-lines

3. Tingkat lanjut (*advanced*)

   Permainan sudoku tingkat lanjut merupakan permainan sudoku yang dapat diselesaikan dengan teknik yang digunakan pada tingkat sedang ditambah teknik *naked pairs* dan *hidden pairs*. Mirip seperti tingkat sebelumnya, kedua teknik melakukan eliminasi *possible values*.



Gambar 3. Penggunaan teknik *naked pairs*

Sumber: https://www.sudokuoftheday.com/techniques/naked-pairs-triples

Penggunaan naked pairs pada gambar 3 dapat dilihat di baris paling bawah. Terdapat dua kotak dengan *possible values* 1 dan 5. Jika terdapat kotak lain pada baris tersebut dengan nilai 1 atau 5, otomatis salah satu dari kedua kotak ini tidak memiliki nilai yang mungkin. Artinya, jika terdapat *naked pairs* seperti ini, pemain dapat menghilangkan kedua angka dari posisi lain untuk melakukan eliminasi nilai. Teknik ini memiliki cukup banyak variasi, seperti *naked triple* yang memiliki ide dasar sama namun berasal dari triplet dengan tiga kemungkinan nilai.

4. Tingkat ahli (*master*)

   Permainan sudoku tingkat ahli memerlukan teknik dari tingkat sebelumnya, ditambah *x-wing*, *swordfish*, dan *forcing chains*. Teknik x-wing merupakan eliminasi nilai-nilai yang masih mungkin pada sebuah kotak kosong dengan melihat pola empat *possible values* yang membentuk persegi panjang. Teknik swordfish bekerja dengan cara yang sama seperti x-wing, namun dengan enam atau lebih kotak dalam sebuah pola.

5. Tingkat sangat sulit

   Pada tingkat ini, pemain diharuskan menebak jawaban hingga menemukan jawaban akhir. Pada kasus khusus, dapat digunakan teknik *nishio* yang merupakan proses menebak untuk mencari kontradiksi.

Selain teknik yang disebutkan di atas, terdapat teknik penyelesaian sudoku yang muncul akibat asumsi bahwa solusi sebuah sudoku pasti unik. Teknik tersebut dinamakan sudoku *uniqueness test*.

III. Pembuatan Permainan Sudoku Acak Menggunakan Algoritma Decrease and Conquer Untuk Memastikan Keunikan Solusi

A. Deskripsi Umum Solusi

Pada permasalahan ini, perlu dibuat papan sudoku yang lengkap berisi 81 angka dan memenuhi aturan permainan. Pembuatan papan lengkap akan memanfaatkan algoritma runut-balik dan *random number generator*. Kotak dari permainan yang sudah selesai tersebut akan dikosongkan sebanyak mungkin dengan tetap menjaga keunikan solusi permainan tersebut.

B. Algoritma Pembangkit Solusi

Untuk melakukan pembangkitan solusi, penulis menggunakan algoritma runut-balik dengan properti sebagai berikut

1. Ruang solusi

   Ruang solusi pada permasalahan ini adalah

   $$X = (x_1, x_2, ..., x_{81})$$

   $$x_i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

2. Fungsi pembangkit

   Fungsi pembangkit untuk membuat solusi lengkap adalah membangkitkan seluruh kemungkinan nilai x_n yang mungkin, yaitu seluruh nilai antara 1 dan 9 inklusif. Untuk

memastikan papan yang dibuat acak, fungsi pembangkit membangkitkan nilai tersebut dengan urutan yang acak.

3. Fungsi pembatas

Fungsi pembatas dalam pembuatan solusi akan mengembalikan false jika x_n telah ditulis pada baris, kolom, atau kotak 3x3 yang sama dengan posisi ke-n, selebihnya mengembalikan true.

C. *Algoritma Pembangkit Permainan Sudoku*

Untuk melakukan penghapusan sebanyak mungkin nilai kotak sudoku, penulis menggunakan algoritma *decrease and conquer* untuk mengurangi kemungkinan banyak kotak yang dapat dihilangkan.

Untuk memilih kotak yang dihapus, dibuat sebuah larik berisi 81 elemen yang menandakan urutan kotak yang akan dihapus, lalu larik tersebut diacak menggunakan *random shuffle*. Karena urutan penghapusan tetap, akan berlaku

1. Jika permainan dengan n kotak kosong memiliki satu solusi, maka untuk setiap i < n, permainan dengan i kotak kosong juga memiliki satu solusi.
2. Jika permainan dengan n kotak kosong memiliki lebih dari satu solusi, maka untuk setiap i > n, permainan dengan i kotak kosong juga memiliki satu solusi.

Kedua properti di atas menyebabkan pencarian yang dilakukan tidak harus menyeluruh, namun dapat dihentikan ketika mencapai batasan tertentu, yaitu unik terakhir atau non-unik pertama. Beberapa alternatif algoritma *decrease and conquer* yang memanfaatkan properti tersebut adalah

1. Menggunakan *decrease by a constant* dimulai dari kotak permainan penuh
2. Menggunakan *decrease by a constant* dimulai dari kotak permainan kosong
3. Menggunakan *decrease by a constant factor*, secara default diatur *constant factor* = 2 dengan cara kerja seperti pada algoritma *binary search the answer*.

D. *Algoritma Pemeriksa Keunikan Solusi*

Untuk melakukan pemeriksaan terhadap keunikan solusi, penulis menggunakan algoritma runut-balik dengan properti:

1. Ruang solusi

   Untuk papan permainan $Y = (y_1, y_2, ..., y_n)$, ruang solusi pada permasalahan ini adalah

   $$X = (x_1, x_2, ..., x_{81})$$

   $$\text{Jika } y_i = 0,\ x_i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\};$$

   $$\text{Jika } y_i \neq 0,\ x_i = y_i$$

2. Fungsi pembangkit

   Fungsi pembangkit untuk membuat solusi lengkap adalah membangkitkan seluruh kemungkinan nilai x_n yang mungkin, yaitu y_n jika $y_n \neq 0$ dan seluruh nilai antara 1 dan 9 inklusif untuk $y_n = 0$.

3. Fungsi pembatas

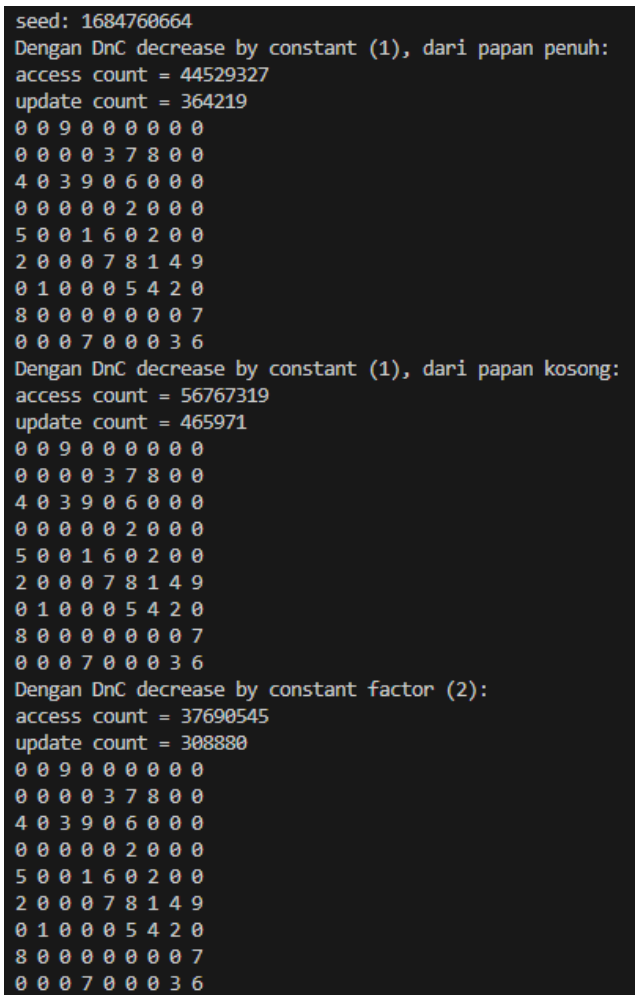
   Fungsi pembatas dalam pembuatan solusi akan mengembalikan false jika x_n telah ditulis pada baris, kolom, atau kotak 3x3 yang sama dengan posisi ke-n, selebihnya mengembalikan true. Fungsi pembatas hanya dijalankan pada posisi n dengan $y_n = 0$.

E. *Simulasi*

Ketiga alternatif algoritma pada poin C yang dibuat di bahasa c++ dapat menghasilkan contoh tampilan seperti pada gambar 4. Pada gambar tersebut, terlihat ketiga alternatif mampu menemukan jawaban papan sudoku yang sama, yaitu papan minimal yang masih memiliki solusi unik.

Terdapat perbedaan *access count* dan *update count* untuk tiap alternatif algoritma, meskipun ketiga algoritma diberikan papan mulai dan urutan penghapusan yang sama sesuai *seed* yang diberikan.

Untuk mendapat data dari percobaan yang cukup, maka percobaan ini akan dilakukan sebanyak 50 kali dan hasilnya akan disimpan ke dalam sebuah file untuk dilakukan pengamatan.



Gambar 4. Contoh hasil pembuatan permainan sudoku acak

Sumber: dokumentasi pribadi

TABLE I. JUMLAH AKSES DAN UPDATE TIAP METODE DECREASE AND CONQUER

No	Jumlah akses dan update		
	constant, dari penuh	*constant, dari kosong*	*constant factor (2)*
1	121853	83912570	148593
2	1130058	15801816	1062999
3	413355	34032863	2639487
4	201642	7949056	202613
5	237597	10035189	143355
6	311724	19552451	240910
7	224063	321943697	145179
8	422937	5194416	194841
9	774036	8226922	475149
10	244787	5003607	339747
mean	408205,2	51165259	559287,3
deviation	312448,7	98083893	782502,4

Tabel 1 menunjukkan bahwa metode DnC decrease by a constant dari papan kosong memerlukan jumlah akses yang jauh lebih banyak dibanding metode lain. Metode tersebut juga membuat perangkat kesulitan dalam melakukan simulasi, seperti yang dibuat pada tabel 1 dengan seed 1684764855. Oleh karena itu, perlu dibuat simulasi tersendiri bagi metode DnC *decrease by a constant* dari papan penuh *dan decrease by a constant factor*.

	constant	constant factor	empty cells
count	200.00	200.00	200.00
mean	712529.43	9273277.07	36.42
std	1261325.34	79698299.23	8.54
min	56658.00	66662.00	14.00
25%	203777.75	143730.25	32.00
50%	359820.50	251164.00	39.00
75%	647123.50	1021733.75	43.00
max	13559811.00	909232782.00	52.00

Gambar 5. Hasil simulasi dua metode *decrease and conquer*

Sumber: dokumentasi pribadi

Simulasi kedua metode *decrease and* conquer yang tersisa memiliki asumsi bahwa kompleksitas akses dan update sama, sehingga kedua variabel dapat disatukan. Seperti simulasi sebelumnya, simulasi dua metode ini dilakukan dengan seed yang sama pada tiap iterasi untuk membandingkan keefektifan kedua algoritma.

Hasil analisis dari 200 data simulasi yang diperoleh berada pada gambar 5. Nilai rerata pada gambar 5 menunjukkan bahwa algoritma *decrease and conquer* dengan metode *decrease by a constant* dimulai dari papan sudoku penuh lebih efisien dibandingkan metode *decrease by a constant factor*, dengan perbedaan yang cukup besar. Akan tetapi, untuk 50% kasus, perhitungan dengan metode *decrease by a constant factor* akan lebih cepat dilihat dari median kedua data, meskipun perbedaannya tidak signifikan.

Perbedaan tersebut tampak tidak intuitif, sebab algoritma *decrease and conquer* dengan reduksi faktor konstan umumnya memiliki kompleksitas lebih rendah dibanding dengan reduksi konstan. Salah satu kemungkinan penyebab terjadinya anomali tersebut adalah pada algoritma yang digunakan untuk memeriksa keunikan papan sudoku yang menggunakan algoritma runut-balik. Semakin banyak kotak kosong pada sudoku, jumlah pengecekan yang harus dibuat semakin meningkat. Artinya, algoritma *decrease and conquer* dengan metode *decrease by a constant factor* yang memeriksa posisi tengah (lebih banyak kotak kosong) memerlukan resource lebih

banyak dibandingkan dengan metode *decrease by a constant* yang memeriksa mulai posisi sudoku penuh.

IV. Penutup

A. Kesimpulan

Algoritma *decrease and conquer* dibantu dengan algoritma runut-balik dapat digunakan untuk menciptakan permainan sudoku yang memiliki solusi unik secara acak. Pilihan opsi algoritma *decrease and conquer* yang paling efisien adalah *decrease by constant factor*, terurut dari kondisi permainan masih penuh. Opsi tersebut menjadi pilihan paling efisien karena waktu menjalankan algoritma runut-balik lebih cepat jika semakin banyak kotak yang terisi pada sudoku.

B. Saran

Algoritma pembangkit solusi lengkap dan algoritma untuk memeriksa jawaban yang menggunakan algoritma runut-balik memakan porsi waktu lebih besar dibanding algoritma pencari jumlah kotak kosong terbanyak yang menggunakan algoritma *decrease and conquer*. Pengamatan kedepannya diharapkan dapat memperhatikan algoritma pembangkit dan pemeriksa jawaban agar memperoleh hasil lebih baik.

References

[1] R. Munir, "Algoritma Runut-balik (backtracking), bagian 1," 2021

[2] R. Munir, "Algoritma Decrease and Conquer," 2021

[3] Astraware Ltd., "Techniques for Solving Sudoku," https://www.sudokuoftheday.com/techniques, diakses 22 Mei 2023

[4] Astraware Ltd, "Sudoku techniques: Candidate Lines," https://www.sudokuoftheday.com/techniques/candidate-lines, diakses 22 Mei 2023

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, 22 Mei 2023

Michael Utama - 13521137