

IF2211 Startegi Algoritma

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Laporan Tugas Kecil 3

Disusun untuk memenuhi tugas mata kuliah Strategi Algoritma
pada Semester 2 (dua) Tahun Akademik 2024/2025



Oleh:

Rhio Bimo Prakoso Sugiyanto (13523123)

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2025**

DAFTAR ISI

DAFTAR ISI	2
BAB 1	3
BAB 2	4
BAB 3	5
3.1 Implementasi Brute Force.....	5
3.1.1 Inisialisasi Board dan Piece:.....	5
3.1.2 Pencarian Posisi yang Tepat:.....	5
3.1.3 Penyesuaian Orientasi:.....	5
3.1.4 Backtracking:.....	5
3.1.5 Proses Berulang:.....	5
3.2 Implementasi GUI.....	5
3.2.1 GUIPieces.java:.....	6
3.2.2 LoadingPanel.java:.....	6
3.2.3 MainPanel.java:.....	6
3.2.4 SolvePanel.java:.....	6
3.3 Penjelasan Class.....	6
3.4 Pengujian.....	8
3.4.1 Pengujian GUI.....	8
3.4.2 Pengujian Config.....	10
3.4.3 Pengujian Brute Force (Accepted).....	10
3.4.4 Pengujian Brute Force (No Solution).....	11
3.4.5 Debug Mode.....	12
BAB 4	14
LAMPIRAN	15

BAB 1

Deskripsi Tugas

IQ Puzzler Pro adalah permainan papan inovatif yang diproduksi oleh Smart Games. Permainan ini dirancang untuk menguji kemampuan logika, pemecahan masalah, dan kreativitas pemain melalui tantangan menyusun potongan puzzle yang memiliki bentuk unik ke dalam sebuah papan.

Komponen Utama dalam Permainan:

- **Board (Papan):**
Papan adalah elemen utama yang harus diisi sepenuhnya oleh potongan puzzle. Desain papan yang unik menuntut strategi penempatan yang tepat agar tidak ada ruang kosong yang tersisa. Setiap posisi di papan memiliki peran penting dalam menentukan validitas solusi yang ditemukan.
- **Blok/Piece:**
Blok adalah potongan puzzle yang digunakan untuk mengisi papan. Setiap blok memiliki bentuk yang unik dan dapat dirotasikan maupun dicerminkan, sehingga membuka berbagai kemungkinan penempatan. Tantangan utama permainan ini adalah memastikan bahwa semua blok digunakan dengan sempurna untuk mengisi papan tanpa adanya tumpang tindih (kecuali pada varian 3D).

Permainan dimulai dengan papan yang kosong. Pemain kemudian menempatkan blok satu per satu dengan hati-hati agar setiap potongan mengisi area yang tepat, menghindari penempatan yang menyebabkan tumpang tindih. Variasi dalam orientasi setiap blok, baik dengan rotasi maupun pencerminan, menambah kompleksitas dan memperkaya strategi penyusunan puzzle. Puzzle dianggap selesai jika papan terisi penuh dan setiap blok telah ditempatkan secara tepat.

Tugas utama program ini adalah untuk menemukan setidaknya satu solusi yang valid menggunakan algoritma Brute Force. Pendekatan ini melibatkan pencarian melalui setiap kemungkinan kombinasi penempatan blok untuk memastikan bahwa seluruh area papan dapat diisi tanpa celah dan tanpa adanya tumpang tindih. Jika setelah melalui seluruh kemungkinan solusi tidak ditemukan kombinasi yang memenuhi kriteria, maka sistem harus menampilkan bahwa solusi tidak ditemukan.

BAB 2

Landasan Teori

Brute force adalah salah satu metode penerapan algoritma yang sering digunakan untuk menyelesaikan permasalahan melalui pengujian semua kemungkinan solusi secara menyeluruh. Salah satu contoh penerapan metode ini dapat ditemukan dalam puzzle game IQ Puzzler Pro, di mana setiap kombinasi penempatan potongan puzzle dievaluasi untuk menemukan solusi yang valid.

Dalam konteks IQ Puzzler Pro, permainan ini menuntut pemain untuk mengisi seluruh papan dengan potongan puzzle yang unik. Setiap potongan dapat dirotasikan dan dicerminkan, sehingga menghasilkan banyak kemungkinan konfigurasi penempatan. Algoritma brute force bekerja dengan cara menguji setiap kemungkinan penempatan tersebut secara sistematis, sehingga jika terdapat solusi yang memenuhi syarat, yaitu papan terisi penuh tanpa adanya tumpang tindih—maka solusi tersebut akan ditemukan.

Meskipun metode brute force sering dianggap kurang efisien karena waktu komputasinya yang tinggi pada ruang solusi yang besar, dalam kasus IQ Puzzler Pro ruang solusi yang terbatas memungkinkan penerapan pendekatan ini secara efektif. Keunggulan brute force terletak pada kesederhanaan implementasinya dan jaminan bahwa solusi akan ditemukan apabila memang ada. Pendekatan ini tidak memerlukan strategi heuristik yang kompleks, melainkan mengandalkan evaluasi menyeluruh terhadap setiap kemungkinan.

BAB 3

Implementasi dan Pengujian

3.1 Implementasi Brute Force

3.1.1 Inisialisasi Board dan Piece:

Program memulai dengan memeriksa board yang telah dibuat berdasarkan dimensi N dan M yang diambil dari file teks input. Selanjutnya, setiap potongan puzzle (piece) diubah menjadi representasi matriks dan dimasukkan ke dalam daftar (list) yang berisi semua bentuk potongan, termasuk semua variasi rotasi dan pencerminan.

3.1.2 Pencarian Posisi yang Tepat:

Proses brute force dimulai dengan mengecek apakah potongan puzzle pertama dapat ditempatkan langsung pada board. Jika potongan tersebut dapat ditempatkan, maka program menempatkannya pada board. Namun, jika tidak memungkinkan, program akan mencoba menempatkan potongan tersebut di posisi lain dengan menggeser koordinat pada board secara sistematis.

3.1.3 Penyesuaian Orientasi:

Jika seluruh posisi pada board telah diperiksa dan potongan belum dapat ditempatkan, program akan mencoba menempatkan potongan yang sama dalam orientasi berbeda. Orientasi yang diuji meliputi setiap rotasi dan pencerminan potongan tersebut.

3.1.4 Backtracking:

Apabila semua kemungkinan penempatan (baik dari segi posisi maupun orientasi) untuk sebuah potongan tidak menghasilkan solusi yang valid, program akan melakukan backtracking. Proses backtracking dilakukan dengan cara menghapus potongan sebelumnya dari board dan mencoba memindahkannya ke kanan. Jika langkah tersebut masih belum berhasil, program akan mencoba kombinasi penempatan dan orientasi lain untuk potongan yang dihapus tersebut.

3.1.5 Proses Berulang:

Langkah pencarian, penyesuaian orientasi, dan backtracking diulang terus hingga semua potongan berhasil ditempatkan secara lengkap pada board atau hingga seluruh kemungkinan telah diuji dan tidak ditemukan solusi yang valid.

3.2 Implementasi GUI

Dalam implementasi bonus GUI, program ini memanfaatkan Java Swing untuk membangun antarmuka yang interaktif dan menarik. Seluruh komponen GUI tersusun rapih di dalam folder gui pada repository, dengan file-file utama sebagai berikut:

3.2.1 GUIPieces.java:

File ini bertanggung jawab untuk mengonversi setiap potongan puzzle yang terdapat dalam list menjadi representasi grafis berbentuk bola dengan warna dan ID (huruf) mereka masing-masing, sehingga memudahkan identifikasi dan visualisasi.

3.2.2 LoadingPanel.java:

Panel ini ditampilkan ketika program sedang dalam proses kalkulasi untuk menyelesaikan puzzle. Panel ini memberikan indikasi bahwa program sedang "loading", terutama ketika mode debugging tidak diaktifkan.

3.2.3 MainPanel.java:

Panel utama ini menampilkan nama program beserta dua tombol penting, yaitu 'DEBUG' dan 'Upload a File'. Tombol 'Upload a File' membuka antarmuka untuk memasukkan file teks input yang diperlukan agar program dapat memulai proses penyelesaian puzzle, sedangkan tombol 'DEBUG' berfungsi sebagai switch untuk memungkinkan pengguna melihat langkah demi langkah proses penyelesaian yang dilakukan oleh program.

3.2.4 SolvePanel.java:

Setelah puzzle berhasil dipecahkan, panel ini akan menampilkan konfigurasi salah satu penempatan potongan pada board sebagai bukti penyelesaian. Selain itu, file ini juga menampilkan informasi mengenai jumlah langkah (steps) yang ditempuh oleh metode brute force dan waktu eksekusi dalam satuan milidetik (ms). Tersedia pula dua tombol, 'Save Board' dan 'Back', yang memungkinkan pengguna menyimpan hasil penyelesaian puzzle dalam bentuk gambar atau teks, atau kembali ke halaman utama.

Dengan pembagian peran yang jelas pada setiap file tersebut, implementasi GUI tidak hanya meningkatkan estetika program, tetapi juga memberikan kemudahan navigasi dan interaksi bagi pengguna.

3.3 Penjelasan Class

No	Class/Method	Keterangan
1.	Main	Class yang menjalankan program secara keseluruhan
2.	DefaultSolver	Class yang melakukan <i>solving</i> dengan cara DEFAULT
	setDebugPanel(javax.swing.JPanel debugPanel)	Menginisiasi Debug interface pada GUI
	solve() & solveHelper(int index)	Melakukan algoritma brute force secara rekursif untuk memecahkan puzzle
3.	Piece	Class yang handle

		pembacaan piece dari teks beserta memanipulasinya, terdiri dari id dan shape
	fromString(String s)	Convert piece dari string file teks menjadi matriks
	getRotation()	Memasukan seluruh kemungkinan rotasi dan pencerminan piece kedalam list
	rotate(boolean[][] matrix)	Merotasikan matriks piece
	mirror(boolean[][] matrix)	Mencerminkan matriks piece
	matrixToString(boolean[][] matrix)	Mengubah matrix menjadi string
4.	Board	Class yang handle pembacaan kondisi board, terdiri dari row, cols, dan grid
	canPlace(boolean[][] shape, int row, int col)	Mengambil matrix piece dan melakukan check terhadap board, apakah piece dapat masuk atau tidak
	placePiece(boolean[][] shape, int row, int col, char id)	Memasukan matriks piece kedalam grid dari board
	removePiece(boolean[] shape, int row, int col)	‘Mengeluarkan’ piece dari board pada row dan col
	print()	Aku lupa hapus ini saat buat GUI wkwk, ini asalnya untuk print board di terminal dengan ini piece berwarna
	isComplete()	Melakukan check terhadap seluruh grid row dan col grid board, apakah masih kosong atau tidak
	isSolved(List<Piece> pieces)	Melakukan check apakah board sudah penuh semua dan membandingkan apakah total piece di board sama dengan banyaknya cell di grid yang sudah terisi
	getGrid()	Mengembalikan nilai grid board
5.	SolvePanel	Class yang handle GUI pada

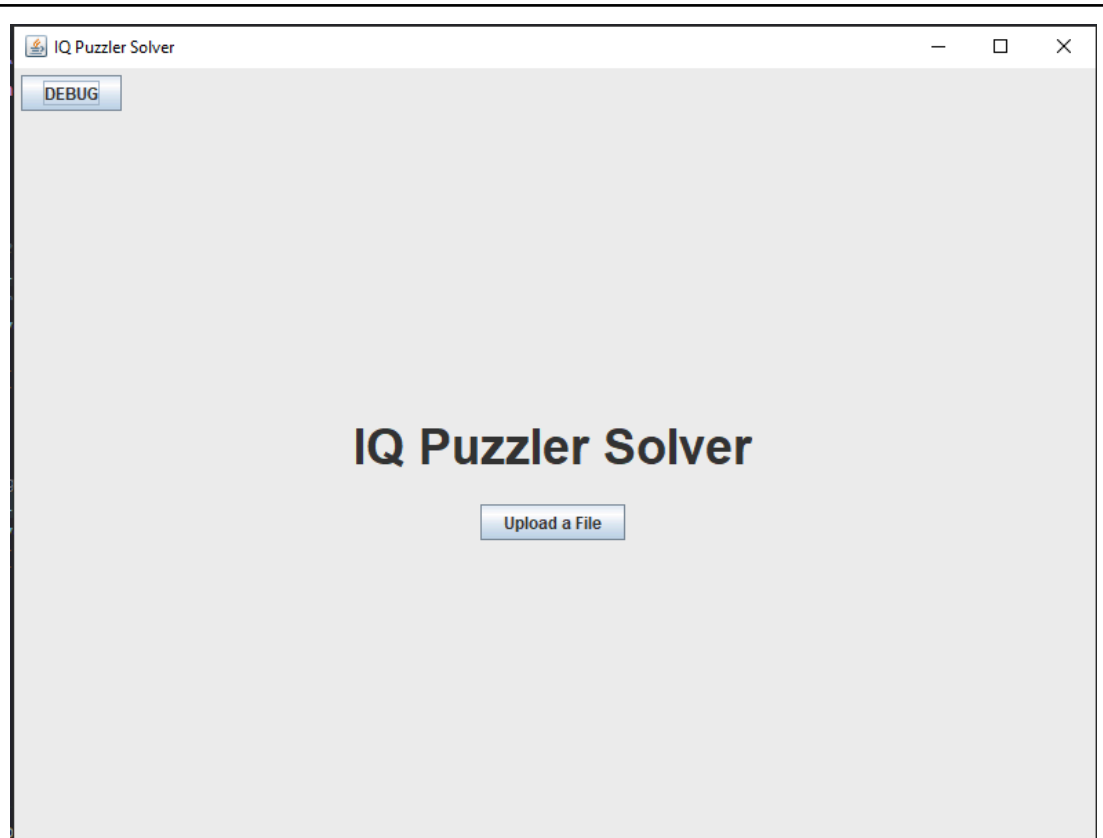
		saat puzzle telah berhasil dipecahkan
	saveAsText()	Melakukan penyimpanan kondisi puzzle yang telah dipecahkan ke dalam sistem dengan format file teks
	saveAsImage()	Melakukan penyimpanan kondisi puzzle yang telah dipecahkan ke dalam sistem dengan format file gambar
6.	LoadingPanel	Class yang handle GUI pada saat puzzle sedang dipecahkan (dan tidak sedang melakukan 'DEBUG')
7.	GUIPieces	Class yang handle bagaimana piece berbentuk di GUI
	paintComponent(Graphics g)	Mewarnai GUI dari piece dan dari grid kosong di board
8.	MainPanel	Class yang handle seluruh GUI dan input user pada keadaan awal program
	runPuzzle (File file)	Membaca input file dari user, melakukan validasi input tersebut, dan menginisialisasi kan class solve.
	getFirstNonWhitespaceChar (String line)	Helper function untuk handle ' ' (spasi) pada saat check piece
	backToMain()	Mengembalikan <i>current view</i> menjadi main panel kembali.
	main(String[] args)	Entry point dari Java application :v

3.4 Pengujian

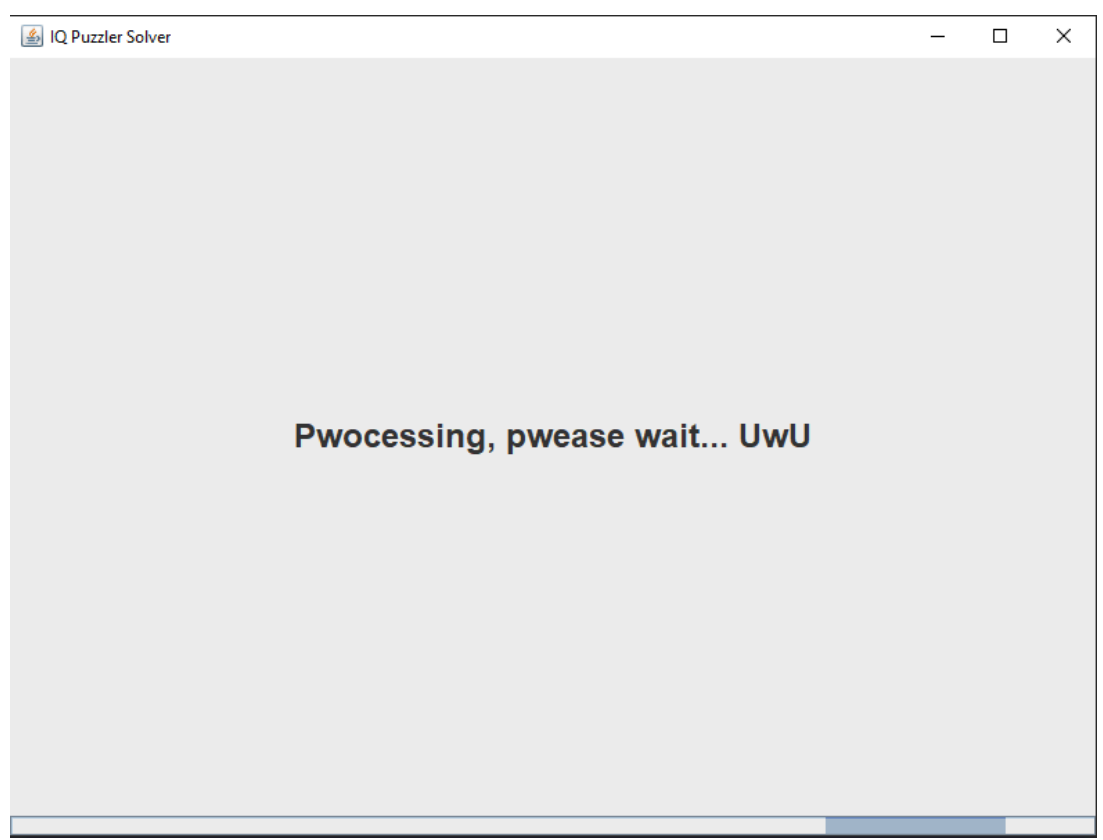
3.4.1 Pengujian GUI

No	Pengujian
----	-----------

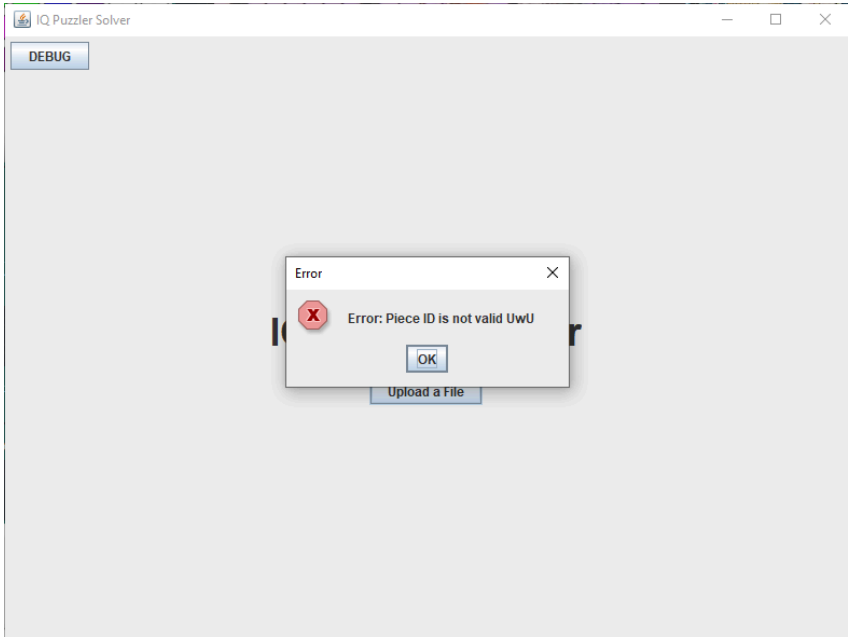
1.



2.

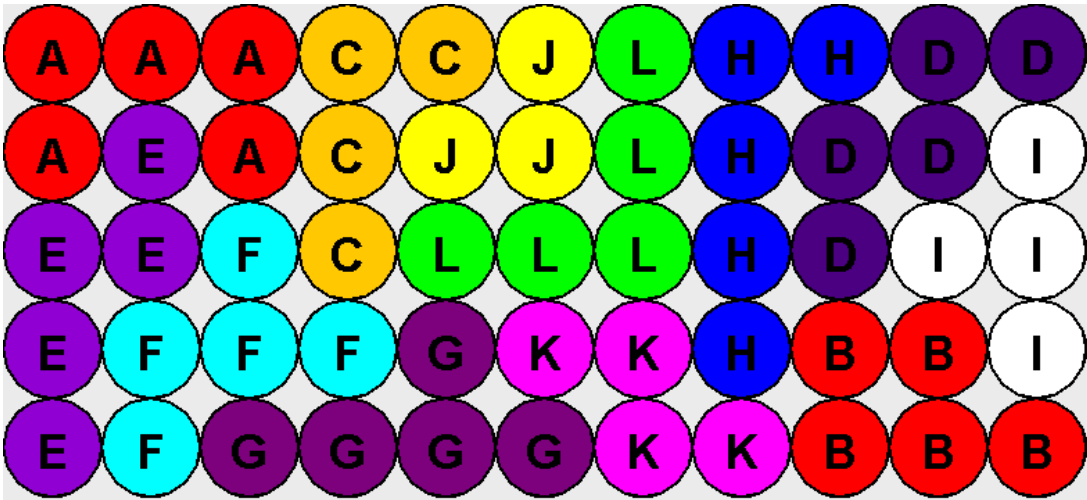


3.4.2 Pengujian Config

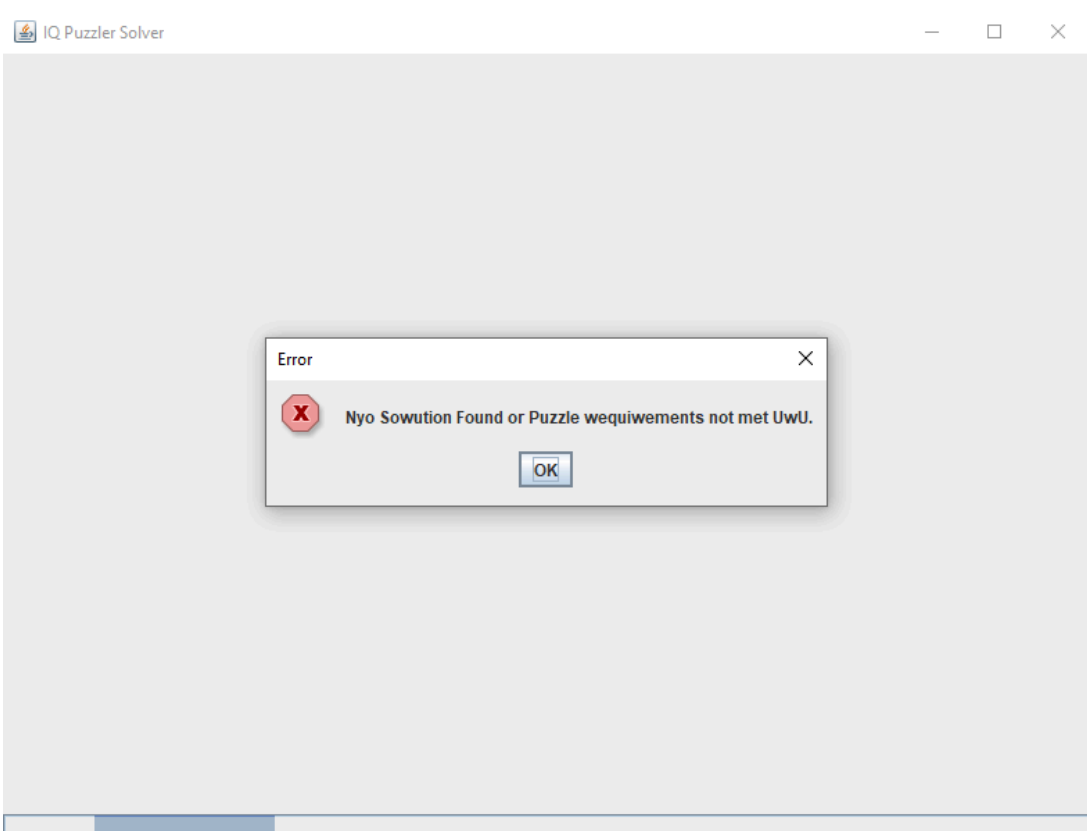
No	Pengujian
1.	 <p>The screenshot shows a window titled "IQ Puzzler Solver" with a "DEBUG" button. An error dialog box is displayed in the center, containing a red 'X' icon and the text "Error: Piece ID is not valid UwU". Below the error message is an "OK" button. At the bottom of the solver window, there is an "Upload a File" button.</p>

3.4.3 Pengujian Brute Force (Accepted)

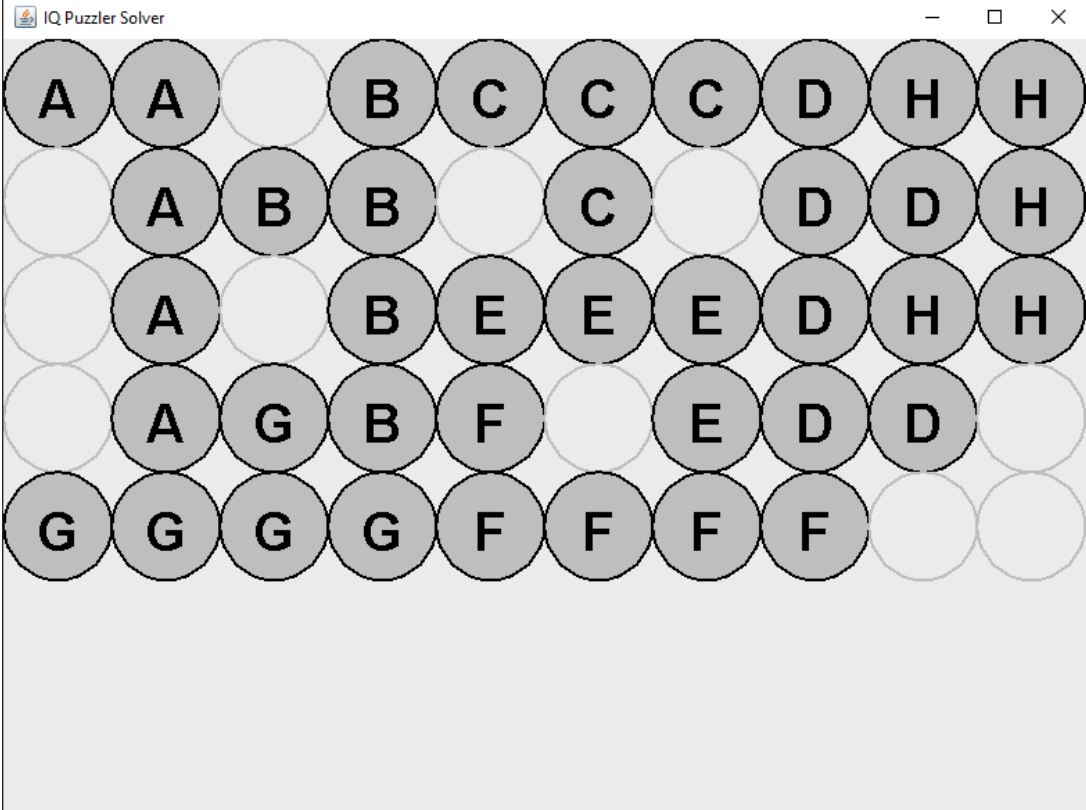
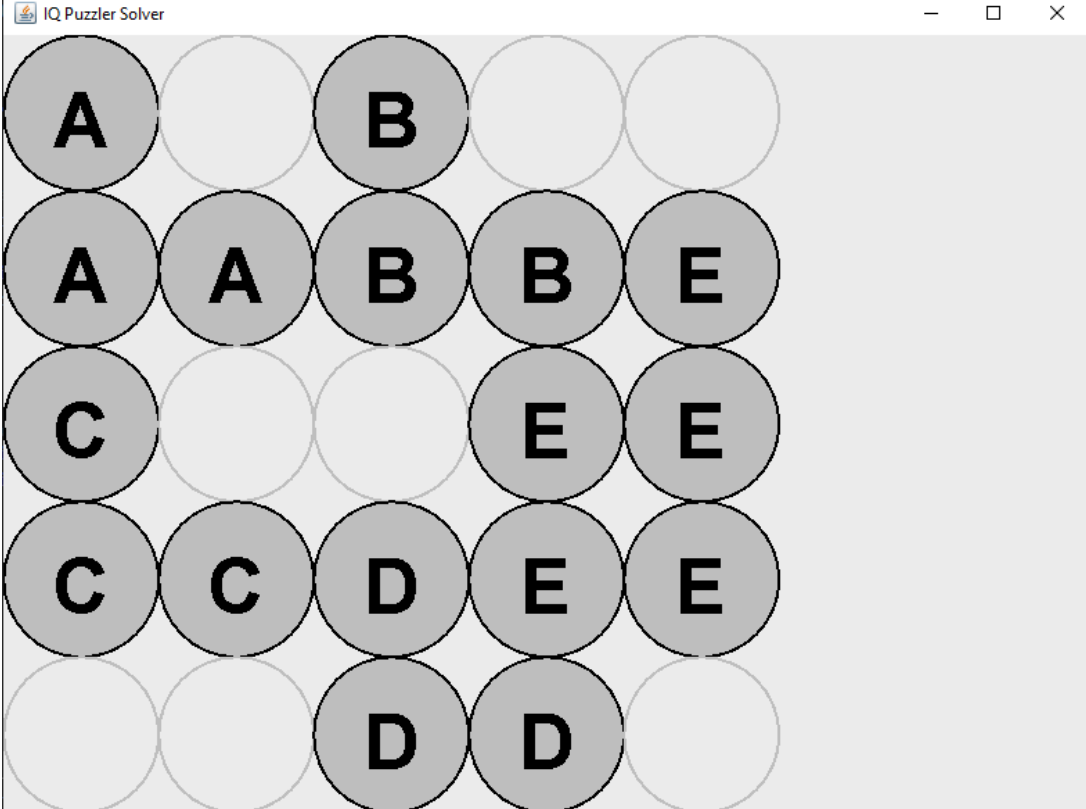
No	Pengujian
1.	<div><div><div><div><div>A</div><div>G</div><div>G</div><div>G</div><div>C</div></div><div><div>A</div><div>A</div><div>B</div><div>C</div><div>C</div></div><div><div>E</div><div>E</div><div>B</div><div>B</div><div>F</div></div><div><div>E</div><div>E</div><div>D</div><div>F</div><div>F</div></div><div><div>E</div><div>D</div><div>D</div><div>F</div><div>F</div></div></div></div><div><div>Elapsed time: 70 ms</div><div>Steps: 7166</div></div></div>

2.	 <p>Elapsed time: 2159132 ms Steps: 672325274</p>
----	---

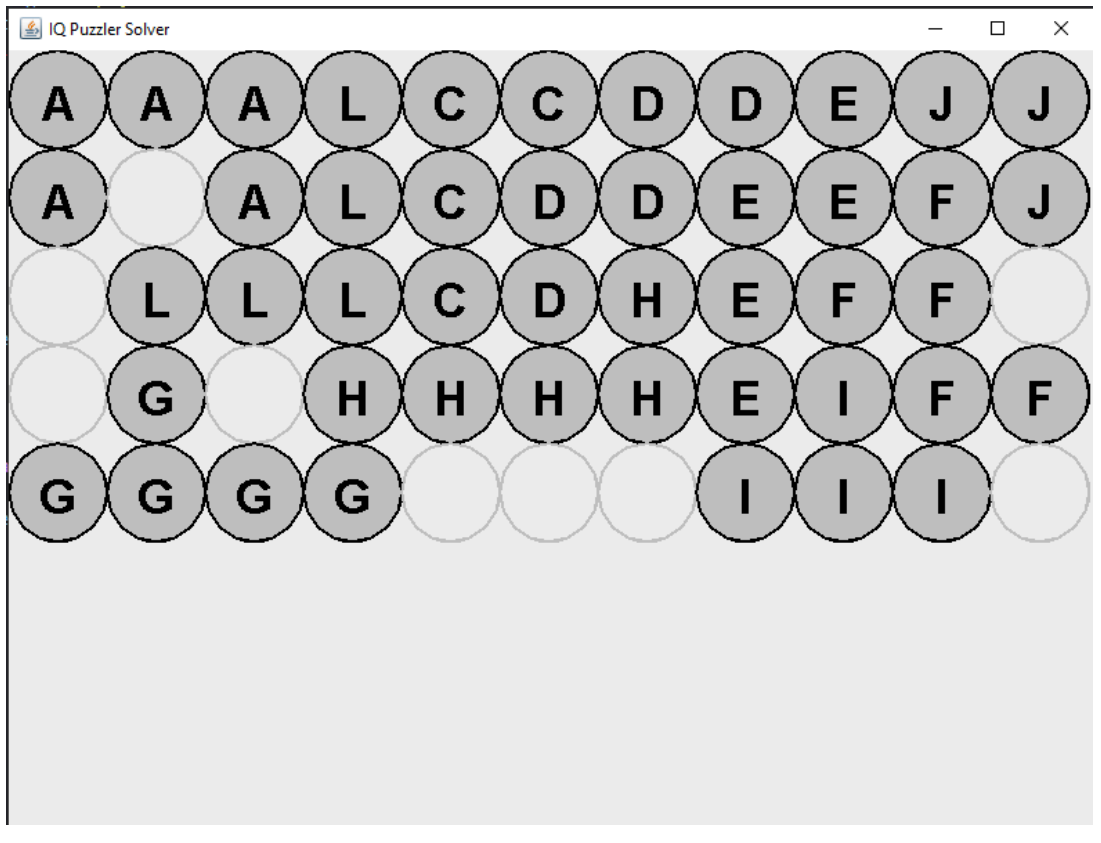
3.4.4 Pengujian Brute Force (No Solution)

No	Pengujian
1.	

3.4.5 Debug Mode

No	Pengujian
1.	 <p>The screenshot shows a 10x10 grid puzzle titled "IQ Puzzler Solver". The grid contains the following letters in the following positions (row, column):</p> <ul style="list-style-type: none"> Row 1: (1,1) A, (1,2) A, (1,4) B, (1,5) C, (1,6) C, (1,7) C, (1,8) D, (1,9) H, (1,10) H Row 2: (2,2) A, (2,3) B, (2,4) B, (2,6) C, (2,8) D, (2,9) D, (2,10) H Row 3: (3,2) A, (3,4) B, (3,5) E, (3,6) E, (3,7) E, (3,8) D, (3,9) H, (3,10) H Row 4: (4,2) A, (4,3) G, (4,4) B, (4,5) F, (4,7) E, (4,8) D, (4,9) D Row 5: (5,1) G, (5,2) G, (5,3) G, (5,4) G, (5,5) F, (5,6) F, (5,7) F, (5,8) F
2.	 <p>The screenshot shows a 5x5 grid puzzle titled "IQ Puzzler Solver". The grid contains the following letters in the following positions (row, column):</p> <ul style="list-style-type: none"> Row 1: (1,1) A, (1,3) B Row 2: (2,1) A, (2,2) A, (2,3) B, (2,4) B, (2,5) E Row 3: (3,1) C, (3,4) E, (3,5) E Row 4: (4,1) C, (4,2) C, (4,3) D, (4,4) E, (4,5) E Row 5: (5,3) D, (5,4) D

3.



BAB 4

Analisis Hasil Uji

Metode brute force merupakan pendekatan pencarian solusi yang dilakukan dengan menguji setiap kemungkinan kombinasi secara menyeluruh hingga ditemukan solusi yang valid. Dengan kompleksitas Big $O(n!)$, metode ini menyuruh program untuk mengevaluasi setiap iterasi yang memungkinkan, sehingga perbedaan waktu eksekusi bisa sangat signifikan tergantung pada bagaimana file konfigurasi puzzle ditulis. Misalnya, pengujian brute force pada puzzle IQ Puzzler Pro menunjukkan bahwa urutan dan orientasi potongan yang disimpan dalam file konfigurasi dapat menghasilkan variasi waktu eksekusi yang drastis, meskipun secara konseptual puzzle tersebut identik.

Sebagai ilustrasi, jika sebuah puzzle memiliki hanya satu solusi dan file konfigurasi menyimpan potongan yang seharusnya ditempatkan di akhir puzzle, dengan orientasi rotasi 270 derajat dan pencerminan, di bagian awal file konfigurasi, program harus menguji seluruh kombinasi rotasi dan pencerminan untuk setiap potongan sebelum akhirnya mencoba menggeser potongan pertama ke posisi yang sesuai. Jika penempatan potongan pertama hanya memungkinkan terjadi pada cell terakhir papan, proses iterasi yang panjang dan kompleks ini akan mengakibatkan waktu eksekusi yang sangat lama. Kondisi inilah yang menyoroti betapa pentingnya penulisan file konfigurasi yang optimal dalam meminimalkan jumlah iterasi yang harus dilakukan oleh program.

Faktor-faktor seperti urutan penyimpanan potongan, orientasi awal, dan cara pengaturan konfigurasi memiliki dampak langsung pada performa algoritma brute force. Walaupun pendekatan ini memiliki jaminan untuk menemukan solusi apabila solusi tersebut memang ada, kinerjanya sangat bergantung pada input yang diberikan. Oleh karena itu, pemahaman mendalam mengenai bagaimana brute force bekerja, serta bagaimana cara terbaik menulis dan mengatur file konfigurasi, menjadi aspek penting dalam mengembangkan solusi yang efisien untuk puzzle kompleks seperti IQ Puzzler Pro.

LAMPIRAN

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	✓	
2	Program berhasil dijalankan	✓	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	✓	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	✓	
5	Program memiliki <i>Graphical User Interface</i> (GUI)	✓	
6	Program dapat menyimpan solusi dalam bentuk file gambar	✓	
7	Program dapat menyelesaikan kasus konfigurasi <i>custom</i>		✓
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		✓
9	Program dibuat oleh saya sendiri	✓	