Laporan Tugas Kecil 1
IF2211 Strategi Algoritma

Disusun oleh:
Aryo Wisanggeni (13523100)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

1. Algoritma

Penyelesaian puzzle IQ Puzzler Pro dengan algoritma brute force dibuat dengan bahasa pemrograman java.Proses input dan output diproses melalui GUI dengan library swing. Program di-compile menjadi class masing-masing di folder bin dan selanjutnya dijadikan file jar yang digunakan untuk eksekusi program.

PseudoCode

Mulai

Deklarasi piecesList sebagai List
Deklarasi puzzleBoard sebagai papan

Untuk setiap piece dalam piecesList (loop 1)
Tempatkan piece pada puzzleBoard

Jika piece tidak muat
Pindahkan posisi piece

Jika piece muat
Kembali ke langkah loop 1 untuk piece berikutnya

Jika semua posisi telah habis (langkah 4)
Kembali ke posisi awal dan putar potongan 90 derajat searah jarum jam
Lakukan langkah loop 1 lagi

Lakukan langkah 4 sebanyak 3 kali untuk mencoba semua rotasi potongan

Jika semua rotasi sudah dicoba
Refleksikan piece, kemudian lakukan langkah 4 lagi
Lakukan rotasi 3 kali dan coba semua posisi untuk setiap rotasi

Jika pada langkah proses atas papan penuh dan tidak ada piece yang tersisa
Kembalikan solusi sebagai berhasil

Jika pada langkah proses atas papan tidak penuh, tetapi tidak ada pieceyang tersisa

Kembalikan bahwa tidak ada solusi yang ditemukan
Selesai

Deskripsi Algoritma

Algoritma bruteforce adalah algoritma yang menyelesaikan masalah dengan cara paling sederhana atau paling lempeng dan jelas dari awal. Keunikan algoritma ini adalah dapat menyelesaikan semua masalah, hanya saja bisa sangat menghabiskan waktu.

Pada tugas kecil ini, puzzle diselesaikan sesuai dengan deskripsi pseudocode di atas, yaitu dengan cara rekursi untuk implementasi brute force. Hal ini dilakukan karena dari setiap kemungkinan piece kejadian berikutnya bisa bercabang banyak. Dengan demikian, untuk mencoba cabang-cabang kemungkinan konfigurasi piece puzzle pada papan, rekursi merupakan metode implementasi yang bagus untuk kasus ini.

Bruteforce yang diimplementasi adalah sebagai berikut.

Fungsi yang digunakan untuk implementasi menerima kondisi papan dan juga list dari piece yang tersisa.

Sebagai basis fungsi, ada dua kondisi yang mungkin terjadi:

1. List dari piece yang tersisa habis dan papan terpenuhi, maka return papan yang berisi solusinya dan jumlah hitungan kasus = 1
2. List dari piece yang tersisa habis dan papan tidak terpenuhi, maka return null (indikasi tidak ada solusi) dan jumlah hitungan kasus = 1

Sebagai rekursi, algoritmanya adalah sebagai berikut:

1. Ambil suatu piece dari list of pieces
2. Taruh piece di suatu tempat di papan, jika tidak bisa diletakkan, coba posisi berikutnya.
3. Jika piece dapat diletakkan ambil piece lagi dari list of pieces dan coba letakkan di papan lagi untuk semua posisi yang mungkin.
4. Jika posisi sudah dicoba dan tidak ada yang mungkin untuk meletakkan piece, rotasi piece sejauh 90 derajat searah jarum jam sebanyak 3 kali (untuk semua kombinasi bentuk yang berbeda dari satu piece) dan cek semua posisi lagi.
5. Jika masih belum menemukan posisi yang valid, cerminkan/balikkan piece dan cobakan semua posisi lagi dan rotasi 90 derajat searah jarum jam sebanyak 3 kali lagi.
6. Untuk setiap kasus piece tidak bisa diletakkan naikkan jumlah kasus sebanyak 1

7. Fungsi akan berhenti ketika semua kombinasi berakhir ke basis 2 yang berarti tidak ada solusi atau salah satu kombinasi mendapat basis 1 yang berarti solusi ditemukan dan fungsi langsung berhenti tanpa memproses kombinasi lain.

Algoritma ini menangani semua kasus sehingga pasti solusi akan ditemukan jika ada. Selain itu, algoritma ini juga akan menangani kasus jika papan terpenuhi dan terdapat piece terpisah. Hal ini bisa terjadi karena ketika papan penuh, karena belum memenuhi kondisi basis “list piece tersisa habis”, piece berikutnya akan masih dicoba dimuatkan pada papan penuh yang akan berakhir dengan piece tidak dapat dimuatkan untuk setiap posisi. Ini akhirnya akan merambat ke piece lain yang sudah diletakkan di papan hingga memang ditemukan bahwa tidak ada kombinasi yang memenuhi.

2. Source Code

Struktur Program

Struktur file src pada tugas kecil ini adalah sebagai berikut.

Pada src, terdapat Main.java yang merupakan driver utama untuk menjalankan program termasuk GUI.Pada src juga terdapat tiga package buatan. Package pertama adalah **custom**, ini adalah package yang menyimpan class yang bersifat sebagai struct atau format input dan output, tetapi juga terdapat beberapa metode pada class-class dalam package tersebut. Package kedua adalah **helpercomp**, dalam package ini berisi program-program yang berfungsi untuk memroses input menjadi output atau dalam kata lain program-program pembantu yang

jika disatukan akan menjadi program Main. Package terakhir **gui**, sesuai namanya, adalah package yang mengendalikan Graphical User Interface (GUI) pada program ini, dengan MainFrame sebagai window awal untuk input dan Solution sebagai window yang muncul untuk menunjukkan hasil pemrosesan brute force.

Program **Main** yang berisi test-test per komponen sewaktu development hanya perlu memanggil GUI untuk memulai program pada versi final program ini

Tucil1_13523100 - Main.java

```java
import helpercomp.*;
import custom.*;
import gui.*;
import java.util.*;
import jdk.jshell.spi.SPIResolutionException;

public class Main {
    public static void main(String[] args) {
        // InputFormat form = Input.readTxt("../test/test1default.txt");
        // InputPuzzlerPro input = form.input;

        // INPUT TEST
        // input.printDetails();

        // PIECE VARIATION TEST
        // List<char[][]> var = BruteForce.getPieceVariations(input.piecesList.get(6));
        // input.printPieces(var);

        // MAIN PROCESS TEST
        // OutputPuzzlerPro output = BruteForce.getOutput(input);
        // output.printDetails();

        // TEXT OUTPUT TEST
        // System.out.println("Current working directory: " + System.getProperty("user.dir"));
        // OutputPuzzlerPro output = BruteForce.getOutput(input);
        // String filename = "../test/solution.txt";
        // OutputTxt.output(input, output, filename);

        // IMAGE OUTPUT TEST
        // OutputPuzzlerPro output = BruteForce.getOutput(input);
        // String filename = "../test/solutionBoard1.png";
        // output.printDetails();
        // OutputImage.solutionImage(output.filledBoardSolution, filename, "DARK");

        // GUI
        new MainFrame();
    }
}
```

Package Custom

File InputPuzzlerPro

Atribut dan konstruktor InputPuzzlerPro

Tucil1_13523100 - InputPuzzlerPro.java

```java
public class InputPuzzlerPro {
    public int height;
    public int width;
    public int numPieces;
    public String mode;
    public char[][] customBoard;
    public List<char[][]> piecesList;

    public InputPuzzlerPro(int height, int width, int numPieces, String mode,
                           char[][] customBoard, List<char[][]> piecesList) {
        this.height = height;
        this.width = width;
        this.numPieces = numPieces;
        this.mode = mode;
        this.customBoard = customBoard;
        this.piecesList = piecesList;
    }
```

Metode printDetails untuk memperlihatkan semua atribut InputPuzzlerPro pada CLI

Tucil1_13523100 - InputPuzzlerPro.java

```java
public void printDetails() {
        System.out.println("Height: " + height);
        System.out.println("Width: " + width);
        System.out.println("Number of Pieces: " + numPieces);
        System.out.println("Mode: " + mode);
        printBoard(customBoard);
        System.out.println("");
        printPieces(piecesList);
    }
```

File OutputPuzzlerPro

Atribut dan konstruktor OutputPuzzlerPro

Tucil1_13523100 - OutputPuzzlerPro.java

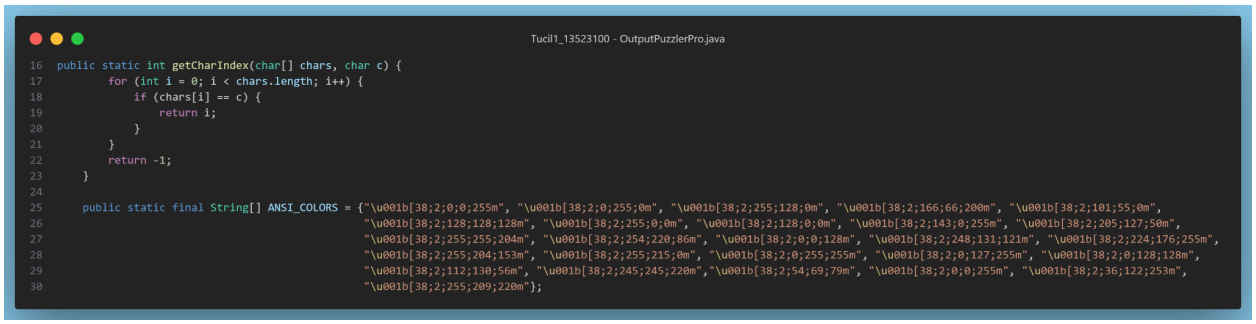
```java
public class OutputPuzzlerPro {
    public char[][] filledBoardSolution;
    public long executionTime;
    public int casesExplored;

    public OutputPuzzlerPro(char[][] filledBoardSolution, long executionTime, int casesExplored) {
        this.filledBoardSolution = filledBoardSolution;
        this.executionTime = executionTime;
        this.casesExplored = casesExplored;
    }
```

Metode untuk menampilkan papan solusi

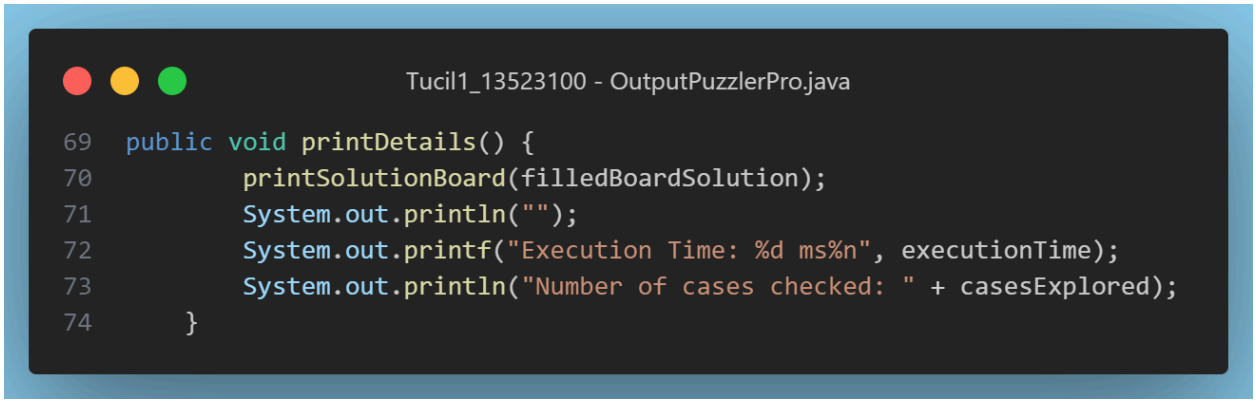
Tucil1_13523100 - OutputPuzzlerPro.java

```java
public static void printSolutionBoard(char[][] boardSolution) {
        if (boardSolution == null) {
            System.out.println("Solution: No solution");
        } else {
            char[] chars = new char[26];
            int charCount = 0;
            System.out.println("Solution:");
            for (int i = 0; i < boardSolution.length; i++) {
                for (int j = 0; j < boardSolution[i].length; j++) {
                    char currentChar = boardSolution[i][j];

                    if (!Character.isUpperCase(currentChar)) {
                        System.out.print("  ");
                        continue;
                    }

                    if (getCharIndex(chars, currentChar) == -1) {
                        // Add the character to the chars array
                        chars[charCount] = currentChar;
                        charCount++;
                    }

                    int charIndex = getCharIndex(chars, currentChar);

                    if (charIndex != -1 && charIndex < ANSI_COLORS.length) {
                        System.out.print(ANSI_COLORS[charIndex] + currentChar + " ");
                    } else {
                        // If no color exists for the charIndex, use the default color
                        System.out.print("\033[0m" + currentChar + " ");
                    }
                }
                System.out.println();
            }
            System.out.print("\033[0m");
        }
    }
```

Metode pembantu metode lain dan list ANSI_COLORS untuk print warna ke CLI



Metode untuk print semua atribut OutputPuzzlerPro



Metode untuk memberi luaran sebagai .txt

Tucil1_13523100 - OutputPuzzlerPro.java

```java
public void exportDetailsToTxt(String filename) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
            // Write solution board to file
            if (filledBoardSolution == null) {
                System.out.println("Solution: No solution");
                return;
            } else {
                writer.write("Solution:\n");
                for (int i = 0; i < filledBoardSolution.length; i++) {
                    for (int j = 0; j < filledBoardSolution[i].length; j++) {
                        char currentChar = filledBoardSolution[i][j];

                        if (!Character.isUpperCase(currentChar)) {
                            writer.write("  ");
                        } else {
                            writer.write(currentChar + " ");
                        }
                    }
                    writer.newLine();
                }
            }

            writer.newLine();

            // Write execution time and number of cases at the end
            writer.write("Execution Time: " + executionTime + " ms\n");
            writer.write("Number of cases checked: " + casesExplored + "\n");

            System.out.println("Solution saved to " + filename);
        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("An error occurred while saving the solution.");
        }
```

File InputFormat

Keseluruhan class InputFormat, class ini merupakan extension dari InputPuzzlerPro karena GUI yang memerlukan errorMessage yang terjadi pada proses input.

Tucil1_13523100 - InputFormat.java

```java
package custom;
import custom.InputPuzzlerPro;

public class InputFormat {
    public InputPuzzlerPro input;
    public String errorMessage;

    public InputFormat(InputPuzzlerPro input, String errorMessage) {
        this.input = input;
        this.errorMessage = errorMessage;
    }

    public InputPuzzlerPro getInput() {
        return input;
    }

    public String getErrorMessage() {
        return errorMessage;
    }
}

```

Package HelperComp

File Input

Bagian pertama menginput width, height, jumlah piece, dan mode

Tucil1_13523100 - Input.java

```java
public class Input {
    public static InputFormat readTxt(String filename) {
        try {
            File file = new File(filename);
            Scanner reader = new Scanner(file);

            // Reading N M P
            int height = -1;
            int width = -1;
            int numPieces = -1;
            if (reader.hasNextLine()) {
                String line = reader.nextLine().strip();
                String[] parts = line.split("\\s+");  // Split the line into parts by spaces

                if (parts.length == 3) {
                    height = Integer.parseInt(parts[0]);
                    width = Integer.parseInt(parts[1]);
                    numPieces = Integer.parseInt(parts[2]);

                    if (height <= 0 || width <= 0) {
                        String errormessage = "Error: matrix dimension zero or negative";
                        System.out.println(errormessage);
                        InputFormat err = new InputFormat(null, errormessage);
                        return err;
                    }

                    if (numPieces <= 0 || numPieces > 26) {
                        String errormessage = "Error: matrix dimension zero or negative";
                        System.out.println(errormessage);
                        InputFormat err = new InputFormat(null, errormessage);
                        return err;
                    }

                } else {
                    String errormessage = "Error: first line does not contain three integers";
                    System.out.println(errormessage);
                    InputFormat err = new InputFormat(null, errormessage);
                    return err;
                }
            }

            // Reading mode
            String mode = "";
            if (reader.hasNextLine()) {
                String line = reader.nextLine().strip();

                switch (line) {
                    case "DEFAULT":
                        mode = "DEFAULT";
                        break;
                    case "CUSTOM":
                        mode = "CUSTOM";
                        break;
                    default:
                        String errormessage = "Error: " + line + " is not a valid case";
                        System.out.println(errormessage);
                        InputFormat err = new InputFormat(null, errormessage);
                        return err;
                }
            }
```

Bagian kedua membaca papan custom dan piece-piece

Tucil1_13523100 - Input.java

```java
// Reading matrix if mode == "CUSTOM"
            char[][] customBoard = null;
            String lineAfterMatrix = "";

            if (mode.equals("CUSTOM")) {
                boolean inMatrix = false;
                ArrayList<String> matrixLines = new ArrayList<>();

                while (reader.hasNextLine()) {
                    String line = reader.nextLine();

                    if (line.contains("X") && line.strip().length() == width) {
                        inMatrix = true;
                    }

                    if (inMatrix) {
                        matrixLines.add(line.strip());
                        inMatrix = false;
                    } else {
                        if (matrixLines.isEmpty()) {
                            String errormessage = "Error: no custom matrix provided OR matrix provided does not match given width";
                            System.out.println(errormessage);
                            InputFormat err = new InputFormat(null, errormessage);
                            return err;
                        }
                        lineAfterMatrix = line;
                        break;
                    }
                }

                customBoard = new char[matrixLines.size()][];
                for (int i = 0; i < matrixLines.size(); i++) {
                    String row = matrixLines.get(i);
                    customBoard[i] = row.toCharArray();
                }

                if (customBoard.length != height) {
                    String errormessage = "Error: Board height does not match given height";
                    System.out.println(errormessage);
                    InputFormat err = new InputFormat(null, errormessage);
                    return err;
                }
            }

            // Reading pieces
            String line;
            ArrayList<String> piecesCandidate = new ArrayList<>();

            if (mode.equals("CUSTOM")) {
                line = lineAfterMatrix;
            } else {
                line = reader.nextLine();
            }

            while (true) {
                if (isPiecePartValid(line)){
                    piecesCandidate.add(line);
                } else {
                    String errormessage = "Error: invalid piece detected -> has different characters in one row OR not a capital alphabet, error piece: " + line;
                    System.out.println(errormessage);
                    InputFormat err = new InputFormat(null, errormessage);
                    return err;
                }

                if (reader.hasNextLine()) {
                    line = reader.nextLine();
                } else {
                    break;
                }
            }
```

Bagian ketiga, block yang dikomen adalah filter untuk mencegah piece yang tidak terhubung

Tucil1_13523100 - Input.java

```
List<char[][]> pieces = new ArrayList<>();
            ArrayList<String> currentPiece = new ArrayList<>();

            for (int i = 0; i < piecesCandidate.size(); i++) {
                String piecePart = piecesCandidate.get(i);

                if (currentPiece.isEmpty() || getFirstNonSpaceChar(piecePart) == getFirstNonSpaceChar(currentPiece.get(0))) { // have not handled if starts with space
                    currentPiece.add(piecePart);
                } else {
                    // If we encounter a different character, create a matrix from the current group
                    pieces.add(createMatrixFromPiece(currentPiece));
                    currentPiece.clear();
                    currentPiece.add(piecePart);

                }

                // add last piece
                if (i == piecesCandidate.size()-1) {
                    pieces.add(createMatrixFromPiece(currentPiece));
                }
            }

            if (pieces.size() != numPieces) {
                String errormessage = "Error: pieces detected: " + pieces.size() + ", expected number of pieces: " + numPieces;
                System.out.println(errormessage);
                InputFormat err = new InputFormat(null, errormessage);
                return err;
            }

            // if (!isAllPiecesValid(pieces)) {
            //     String errormessage = "Error: Invalid piece detected, cells of the piece are not connected";
            //     System.out.println(errormessage);
            //     InputFormat err = new InputFormat(null, errormessage);
            //     return err;
            // }

            InputPuzzlerPro res = new InputPuzzlerPro(height, width, numPieces, mode, customBoard, pieces);
            if (mode.equals("CUSTOM")) {
                res.customBoard = res.matrixTurnXtoUnd(res.customBoard);
            }

            InputFormat finalRes = new InputFormat(res, "");

            return finalRes;
        } catch (FileNotFoundException e) {
            String errormessage = "Error: File not found - " + e.getMessage();
            System.out.println(errormessage);
            InputFormat err = new InputFormat(null, errormessage);
            return err;
        } catch (NumberFormatException e) {
            String errormessage = "Error: File not found - " + e.getMessage();
            System.out.println(errormessage);
            InputFormat err = new InputFormat(null, errormessage);
            return err;
        }
```

Method pembantu input

Tucil1_13523100 - Input.java

```java
public static char[][] createMatrixFromPiece(List<String> piece) {
        int maxLength = piece.stream().mapToInt(String::length).max().orElse(0);
        char[][] matrix = new char[piece.size()][maxLength];

        for (int i = 0; i < piece.size(); i++) {
            String piecePart = piece.get(i);
            for (int j = 0; j < piecePart.length(); j++) {
                matrix[i][j] = (piecePart.charAt(j) == ' ') ? '_' : piecePart.charAt(j);
            }
            for (int j = piecePart.length(); j < maxLength; j++) {
                matrix[i][j] = '_';
            }
        }
        return matrix;
    }

    public static boolean isPiecePartValid(String piecePart) {
        if (piecePart.isEmpty()) {
            return false;
        }

        char alph = getFirstNonSpaceChar(piecePart);

        if (!Character.isUpperCase(alph)) {
            return false;  // If the first character is not a capital letter, return false
        }

        for (int i = 1; i < piecePart.length(); i++) {
            if (piecePart.charAt(i) != alph && piecePart.charAt(i) != ' ') {
                return false;
            }
        }

        return true;
    }

    public static char getFirstNonSpaceChar(String line) {
        for (int i = 0; i < line.length(); i++) {
            char currentChar = line.charAt(i);

            if (currentChar != ' ') {
                return currentChar;
            }
        }

        return '\0';
    }
```

File BruteForce

Fungsi utama untuk memroses puzzle

Tucil1_13523100 - BruteForce.java

```java
public class BruteForce {
    public static OutputPuzzlerPro getOutput(InputFormat form) {
        char[][] beginningMat;
        if (form.input.mode.equals("DEFAULT")) {
            beginningMat = createEmptyMatrix(form.input.height, form.input.width);
        } else if (form.input.mode.equals("CUSTOM")) {
            beginningMat = form.input.customBoard;
        } else {
            System.out.printf("Error: %s is not a valid case%n", form.input.mode);
            return null;
        }

        long startTime = System.currentTimeMillis();
        OutputPuzzlerPro res = processPuzzle(beginningMat, form.input.piecesList);
        long endTime = System.currentTimeMillis();
        long duration = endTime - startTime;
        res.executionTime = duration;

        return res;
    }
```

Fungsi rekursi brute force

Tucil1_13523100 - BruteForce.java

```java
public static OutputPuzzlerPro processPuzzle(char[][] currentMatrix, List<char[][]> pieces) {
        if (pieces.isEmpty()) {
            if (matrixIsFull(currentMatrix)) {
                OutputPuzzlerPro res = new OutputPuzzlerPro(currentMatrix, 0, 1);
                // res.printSolutionBoard(currentMatrix);
                return res;
            } else {
                OutputPuzzlerPro res = new OutputPuzzlerPro(null, 0, 1);
                // res.printSolutionBoard(currentMatrix);
                return res;
            }
        }

        char[][] currentPiece = pieces.get(0);
        // printMatrix(currentPiece);
        List<char[][]> currentVariations = getPieceVariations(currentPiece);
        int countCases = 0;

        // Try all piece variation with rotate and flip
        for (int i = 0; i < currentVariations.size(); i++) {
            int currentWidth = currentVariations.get(i)[0].length;
            int currentHeight = currentVariations.get(i).length;

            int height = currentMatrix.length - currentHeight + 1;
            int width = currentMatrix[0].length - currentWidth + 1;

            // if doesn't fit because larger than matrix, don't process because it fails here
            if (height <= 0 || width <= 0) {
                countCases += 1;
                continue;
            }

            // Checking if the current piece position variation fits for every possible location in matrix
            for (int j = 0; j < height; j++) {
                for (int k = 0; k < width; k++) {
                    boolean canFit = true;

                    // determining if currentVariation fits
                    for (int x = 0; x < currentHeight; x++) {
                        for (int y = 0; y < currentWidth; y++) {
                            if (currentVariations.get(i)[x][y] != '_' && currentMatrix[j + x][k + y] != '_') {
                                canFit = false;
                                break;
                            }
                        }
                        if (!canFit) break;
                    }

                    // If it fits, place the variation into currentMatrix and continue to the next piece
                    if (canFit) {
                        char[][] newMatrix = placeVariation(currentMatrix, currentVariations.get(i), j, k);
                        OutputPuzzlerPro res = processPuzzle(newMatrix, listWithoutFirstElement(pieces));
                        if (res.filledBoardSolution != null) {
                            res.casesExplored += countCases;
                            return res;
                        } else {
                            countCases += res.casesExplored;
                        }
                    } else { // count case and continue to the next position
                        countCases += 1;
                    }
                }
            }
            // if all positions fail, try another variation
        }
        // if all variation fail, return null matrix and how many cases were explored
        OutputPuzzlerPro res = new OutputPuzzlerPro(null, 0, countCases);
        return res;
    }
```

Fungsi pertama akan membuat papan kosong dari input dan fungsi kedua adalah fungsi yang cek apakah papan sudah terpenuhi atau tidak

Tucil1_13523100 - BruteForce.java

```java
public static char[][] createEmptyMatrix(int height, int width) {
        char[][] matrix = new char[height][width];

        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                matrix[i][j] = '_';
            }
        }

        return matrix;
    }


    public static boolean matrixIsFull(char[][] m) {
        for (int i = 0; i < m.length; i++) {
            for (int j = 0; j < m[0].length; j++) {
                if (m[i][j] == '_') {
                    return false;
                }
            }
        }

        return true;
    }
```

Fungsi untuk mendapatkan variasi rotasi dan cermin dari suatu piece

Tucil1_13523100 - BruteForce.java

```java
public static List<char[][]> getPieceVariations(char[][] piece) {
        List<char[][]> variations = new ArrayList<>();
        variations.add(piece);

        // flip
        char[][] mirroredPieceHorizontal = new char[piece.length][piece[0].length];

        for (int i = 0; i < piece.length; i++) {
            for (int j = 0; j < piece[0].length; j++) {
                mirroredPieceHorizontal[i][piece[0].length - j - 1] = piece[i][j];
            }
        }
        variations.add(mirroredPieceHorizontal);

        // rotate
        char[][] oldVar = piece;
        for (int i = 0; i < 3; i++) {
            int rows = oldVar.length;
            int cols = oldVar[0].length;
            char[][] var = new char[cols][rows];

            for (int j = 0; j < rows; j++) {
                for (int k = 0; k < cols; k++) {
                    var[k][rows - 1 - j] = oldVar[j][k];
                }
            }

            variations.add(var);
            oldVar = var;
        }

        oldVar = mirroredPieceHorizontal;
        for (int i = 0; i < 3; i++) {
            int rows = oldVar.length;
            int cols = oldVar[0].length;
            char[][] var = new char[cols][rows];

            for (int j = 0; j < rows; j++) {
                for (int k = 0; k < cols; k++) {
                    var[k][rows - 1 - j] = oldVar[j][k];
                }
            }

            variations.add(var);
            oldVar = var;
        }

        return variations;
    }
```

Fungsi untuk meletakkan piece pada matrix papan

Tucil1_13523100 - BruteForce.java

```java
public static char[][] placeVariation(char[][] currentMatrix, char[][] currentVariation, int startX, int startY) {
        char[][] newMatrix = new char[currentMatrix.length][currentMatrix[0].length];

        for (int i = 0; i < currentMatrix.length; i++) {
            System.arraycopy(currentMatrix[i], 0, newMatrix[i], 0, currentMatrix[i].length);
        }

        for (int i = 0; i < currentVariation.length; i++) {
            for (int j = 0; j < currentVariation[0].length; j++) {
                if (currentVariation[i][j] != '_') {
                    newMatrix[startX + i][startY + j] = currentVariation[i][j];
                }
            }
        }

        return newMatrix;
    }
```

Method untuk mengambil piece dari list

Tucil1_13523100 - BruteForce.java

```java
public static List<char[][]> listWithoutFirstElement(List<char[][]> originalList) {
        if (originalList.isEmpty()) {
            return new ArrayList<>();
        }

        return new ArrayList<>(originalList.subList(1, originalList.size()));
    }
```

File OutputTxt

Bagian pertama

Tucil1_13523100 - OutputTxt.java

```java
public class OutputTxt {
    public static void output (InputPuzzlerPro input, OutputPuzzlerPro output, String filename) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
            // Write input board to file
            char[][] inputBoard = input.createEmptyMatrix();;
            if (input.mode.equals("CUSTOM")) {
                if (input.customBoard == null){
                    inputBoard = null;
                } else {
                    inputBoard = input.matrixTurnXtoUnd(input.customBoard);
                }
            }

            if (inputBoard == null) {
                writer.write("Input Board: null");
                writer.newLine();
            } else {
                writer.write("Input Board:");
                writer.newLine();
                for (int i = 0; i < inputBoard.length; i++) {
                    for (int j = 0; j < inputBoard[i].length; j++) {
                        if (inputBoard[i][j] != '_') {
                            writer.write("  ");
                        } else {
                            writer.write(inputBoard[i][j] + " ");
                        }
                    }
                    writer.newLine();
                }
            }
            writer.newLine();

            // Write input pieces
            if (input.piecesList.isEmpty()) {
                writer.write("Pieces List: empty");
                writer.newLine();
            } else {
                writer.write("Pieces List:");
                writer.newLine();
                for (int index = 0; index < input.piecesList.size(); index++) {
                    writer.write("Piece " + (index + 1) + ":");
                    writer.newLine();
                    if (input.piecesList.get(index) == null) {
                        writer.write("null");
                    } else {
                        for (int i = 0; i < input.piecesList.get(index).length; i++) {
                            for (int j = 0; j < input.piecesList.get(index)[i].length; j++) {
                                writer.write(input.piecesList.get(index)[i][j] + " ");
                            }
                            writer.newLine();
                        }
                    }
                }
            }
            writer.newLine();
```

Bagian kedua

Tucil1_13523100 - OutputTxt.java

```java
// Write input pieces
            if (input.piecesList.isEmpty()) {
                writer.write("Pieces List: empty");
                writer.newLine();
            } else {
                writer.write("Pieces List:");
                writer.newLine();
                for (int index = 0; index < input.piecesList.size(); index++) {
                    writer.write("Piece " + (index + 1) + ":");
                    writer.newLine();
                    if (input.piecesList.get(index) == null) {
                        writer.write("null");
                    } else {
                        for (int i = 0; i < input.piecesList.get(index).length; i++) {
                            for (int j = 0; j < input.piecesList.get(index)[i].length; j++) {
                                writer.write(input.piecesList.get(index)[i][j] + " ");
                            }
                            writer.newLine();
                        }
                    }
                }
            }
            writer.newLine();

            // Write solution board to file
            if (output.filledBoardSolution == null) {
                writer.write("Solution: No solution");
                writer.newLine();
            } else {
                writer.write("Solution:\n");
                for (int i = 0; i < output.filledBoardSolution.length; i++) {
                    for (int j = 0; j < output.filledBoardSolution[i].length; j++) {
                        char currentChar = output.filledBoardSolution[i][j];

                        if (!Character.isUpperCase(currentChar)) {
                            writer.write("  ");
                        } else {
                            writer.write(currentChar + " ");
                        }
                    }
                    writer.newLine();
                }
            }

            writer.newLine();

            // Write execution time and number of cases at the end
            writer.write("Execution Time: " + output.executionTime + " ms\n");
            writer.write("Number of cases checked: " + output.casesExplored + "\n");

            System.out.println("Solution saved to " + filename);
        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("An error occurred while saving the solution.");
        }
```

File InputImageGenerator

Fungsi untuk save input sebagai image untuk GUI

Tucil1_13523100 - InputImageGenerator.java

```java
public static void saveInput (InputPuzzlerPro input) {
        if (input != null) {
            // setting up tempFolder
            String currentDirectory = System.getProperty("user.dir");

            File currentDirFile = new File(currentDirectory);
            String fullPath = currentDirectory + File.separator + "temp";

            File tempFolder = new File(fullPath);

            // Create tempFolder if it does not exist
            if (!tempFolder.exists()) {
                if (tempFolder.mkdirs()) {
                    System.out.println("Temporary folder created: " + tempFolder.getAbsolutePath());
                } else {
                    System.out.println("Failed to create temporary folder.");
                }
            } else {
                System.out.println("Temporary folder already exists.");
            }

            // Generating empty board
            if (input.mode.equals("DEFAULT")) {
                String boardName = fullPath + File.separator + "emptyBoard.png";
                char[][] formattedBoard = createEmptyMatrix(input.height, input.width);

                boardGenerator(formattedBoard, boardName, "LIGHT");
            } else if (input.mode.equals("CUSTOM")) {
                String boardName = fullPath + File.separator + "emptyBoard.png";
                char[][] formattedBoard = input.matrixTurnXtoUnd(input.customBoard);

                boardGenerator(formattedBoard, boardName, "LIGHT");
            }

            // Generating pieces
            for (int i = 0; i < input.piecesList.size(); i++) {
                String pieceName = fullPath + File.separator + "piece" + (i+1) + ".png";

                pieceGenerator(input.piecesList.get(i), pieceName, "LIGHT");
            }

        }
    }
```

File OutputImage

List warna untuk gambar

Tucil1_13523100 - OutputImage.java

```
12  private static final Color[] COLORS = {
13          new Color(45, 70, 84),     // charcoal
14          new Color(139, 69, 19),    // saddlebrown
15          new Color(0, 100, 0),      // darkgreen
16          new Color(128, 128, 0),    // olive
17          new Color(72, 61, 139),    // darkslateblue
18          new Color(60, 179, 113),   // mediumseagreen
19          new Color(0, 128, 128),    // teal
20          new Color(0, 0, 128),      // navy
21          new Color(154, 205, 50),   // yellowgreen
22          new Color(139, 0, 139),    // darkmagenta
23          new Color(176, 48, 96),    // maroon3
24          new Color(255, 0, 0),      // red
25          new Color(255, 165, 0),    // orange
26          new Color(255, 218, 34),    // school bus yellow
27          new Color(127, 255, 0),    // chartreuse
28          new Color(138, 43, 226),   // blueviolet
29          new Color(0, 255, 127),    // springgreen
30          new Color(220, 20, 60),    // crimson
31          new Color(0, 255, 255),    // aqua
32          new Color(0, 191, 255),    // deepskyblue
33          new Color(0, 0, 255),      // blue
34          new Color(255, 127, 80),   // coral
35          new Color(255, 0, 255),    // fuchsia
36          new Color(30, 144, 255),   // dodgerblue
37          new Color(96, 58, 64),  // Eggplant
38          new Color(221, 160, 221),   // plum
39
40      };
```

Fungsi menyimpan output sebagai image

Tucil1_13523100 - OutputImage.java

```java
public static void solutionImage(char[][] solutionBoard, String filename, String mode) {
        int cellSize = 80; // Size of each cell
        int padding = 5;  // Padding between cells
        int edgePadding = 40; // Padding at edge of image
        int fontSize = 40; // Font size for letters inside of the cells

        int rows = solutionBoard.length;
        int cols = solutionBoard[0].length;

        // Calculate the width and height of the image based on the matrix size
        int imageWidth = (cols * cellSize) + ((cols - 1) * padding) + 2 * edgePadding;
        int imageHeight = (rows * cellSize) + ((rows - 1) * padding) + 2 * edgePadding;

        BufferedImage image = new BufferedImage(imageWidth, imageHeight, BufferedImage.TYPE_INT_ARGB);
        Graphics2D g2d = image.createGraphics();
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
        g2d.setRenderingHint(RenderingHints.KEY_INTERPOLATION, RenderingHints.VALUE_INTERPOLATION_BILINEAR);

        if (mode.equals("DARK")) {
            g2d.setColor(Color.BLACK);
        } else {
            g2d.setColor(Color.WHITE);
        }

        g2d.fillRect(0, 0, imageWidth, imageHeight);

        // Set font and stroke for circle cell
        g2d.setFont(new Font("Arial", Font.PLAIN, fontSize));
        g2d.setStroke(new BasicStroke(2));

        // Calculate the starting positions to center the matrix
        int startX = edgePadding;
        int startY = edgePadding;

        // + (imageWidth - (cols * (cellSize + padding) + edgePadding)) / 2;
        // + (imageHeight - (rows * (cellSize + padding) + edgePadding)) / 2;

        // Iterate through the matrix and draw circles for capital letters
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                char currentChar = solutionBoard[i][j];
                int centerX = startX + j * (cellSize + padding) + cellSize / 2;
                int centerY = startY + i * (cellSize + padding) + cellSize / 2;

                // Draw a circle only for capital letters
                if (Character.isUpperCase(currentChar)) {
                    int charIndex = currentChar - 'A';
                    g2d.setColor(COLORS[charIndex]);

                    g2d.fillOval(centerX - cellSize / 2, centerY - cellSize / 2, cellSize, cellSize); // Draw circle
                    g2d.setColor(Color.WHITE); // Draw letter in white inside the circle

                    FontMetrics fontMetrics = g2d.getFontMetrics();
                    int charWidth = fontMetrics.charWidth(currentChar);
                    int charX = centerX - (charWidth / 2);
                    int baselineY = centerY + (fontMetrics.getAscent() - fontMetrics.getDescent()) / 2;

                    g2d.drawString(String.valueOf(currentChar), charX, baselineY); // Draw the letter

                    // Draw Links
                    // Up (i-1, j)
                    if (i > 0 && solutionBoard[i-1][j] == currentChar) {
                        drawLinkBetweenCells(g2d, centerX, centerY - cellSize/2, centerX, centerY - (cellSize/2 + padding), COLORS[charIndex]);
                    }
                    // Down (i+1, j)
                    if (i < rows - 1 && solutionBoard[i+1][j] == currentChar) {
                        drawLinkBetweenCells(g2d, centerX, centerY + cellSize/2, centerX, centerY + (cellSize/2 + padding), COLORS[charIndex]);
                    }
                    // Left (i, j-1)
                    if (j > 0 && solutionBoard[i][j-1] == currentChar) {
                        drawLinkBetweenCells(g2d, centerX - cellSize/2, centerY, centerX - (cellSize/2 + padding), centerY, COLORS[charIndex]);
                    }
                    // Right (i, j+1)
                    if (j < cols - 1 && solutionBoard[i][j+1] == currentChar) {
                        drawLinkBetweenCells(g2d, centerX + cellSize/2, centerY, centerX + (cellSize/2 + padding), centerY, COLORS[charIndex]);
                    }
                } else {
                    if (mode.equals("DARK")) {
                        g2d.setColor(Color.BLACK);
                    } else {
                        g2d.setColor(Color.WHITE);
                    }
                    g2d.fillRect(centerX - cellSize / 2, centerY - cellSize / 2, cellSize, cellSize); // Clear the area
                }

            }
        }

        g2d.dispose();

        try {
            ImageIO.write(image, "PNG", new File(filename));
            System.out.println("Image saved as " + filename);
        } catch (IOException e) {
            System.out.println("Error saving the image: " + e.getMessage());
        }

    }
```

Package GUI

File MainFrame

File ini menyimpan window awal ketika memulai program dan bertugas untuk menerima input dari user. Di bawah adalah fungsi untuk memanggil mainframe, selain untuk inisiasi window, juga terdapat fungsi untuk delete folder temp ketika program selesai digunakan

Tucil1_13523100 - MainFrame.java

```java
public class MainFrame extends JFrame {

    private JButton chooseFileButton;
    private JButton uploadFileButton;
    private JButton solveButton;
    private File selectedFile = null;

    public MainFrame() {
        String currentDirectory = System.getProperty("user.dir");
        File currentDirFile = new File(currentDirectory);
        String tempString = currentDirFile + File.separator + "temp";

        initialize();

        this.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                deleteTempFolder(new File(tempString));
            }
        });
    }
```

Implementasi inisiasi MainFrame (1)

Tucil1_13523100 - MainFrame.java

```
36  public void initialize() {
37          setTitle("Peak Brute Force IQ Puzzler Pro Solver (Def not brain rot fr fr)");
38          setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
39
40          getContentPane().setBackground(new Color(240, 240, 240));
41
42          JPanel mainPanel = new JPanel(new BorderLayout(20, 20));
43          mainPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
44          mainPanel.setBackground(new Color(230, 230, 230));
45
46          // Left panel (dynamic)
47          JPanel leftPanel = new JPanel(new BorderLayout(20, 20));
48          leftPanel.setBackground(new Color(230, 230, 230));
49
50          leftPanel.setLayout(new BoxLayout(leftPanel, BoxLayout.Y_AXIS));
51          leftPanel.setBackground(Color.WHITE);
52
53          JPanel topLeftPanel = new JPanel();
54          topLeftPanel.setLayout(new BoxLayout(topLeftPanel, BoxLayout.Y_AXIS));
55          topLeftPanel.setBackground(Color.WHITE);
56
57          JLabel boardLabel = new JLabel("Empty Board");
58          boardLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
59          topLeftPanel.add(boardLabel);
60
61          JPanel emptyBox = new JPanel();
62          emptyBox.setAlignmentX(Component.CENTER_ALIGNMENT);
63          emptyBox.setBackground(Color.WHITE);
64
65          topLeftPanel.add(emptyBox);
66
67          JPanel bottomLeftPanel = new JPanel();
68          bottomLeftPanel.setLayout(new BoxLayout(bottomLeftPanel, BoxLayout.Y_AXIS));
69          bottomLeftPanel.setBackground(Color.WHITE);
70
71          JLabel piecesLabel = new JLabel("Pieces");
72          piecesLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
73          bottomLeftPanel.add(piecesLabel);
74
75          JPanel[] imagePanels = new JPanel[26];
76          JLabel[] imageLabels = new JLabel[26];
77
78          JPanel gridPanel = new JPanel();
79          gridPanel.setLayout(new GridLayout(0, 7, 10, 10));  // 0 rows (dynamic), 6 columns, 10px gap
```

Implementasi inisiasi MainFrame (2)

Tucil1_13523100 - MainFrame.java

```
81  for (int i = 0; i < 26; i++) {
82              imagePanels[i] = new JPanel();
83              imagePanels[i].setBackground(Color.WHITE);
84              imagePanels[i].setPreferredSize(new Dimension(150, 150));
85
86              imageLabels[i] = new JLabel("Piece " + (i + 1));
87              imageLabels[i].setHorizontalAlignment(SwingConstants.CENTER);
88              imageLabels[i].setVerticalAlignment(SwingConstants.CENTER);
89
90              imagePanels[i].add(imageLabels[i]);
91
92              gridPanel.add(imagePanels[i]);
93          }
94
95          bottomLeftPanel.add(gridPanel);
96
97          leftPanel.add(topLeftPanel);
98          leftPanel.add(bottomLeftPanel);
99
100         // marginPanel to match mainPanel padding
101         JPanel marginPanelLeft = new JPanel();
102         marginPanelLeft.setLayout(new BorderLayout());
103         marginPanelLeft.setBorder(BorderFactory.createEmptyBorder(0, 20, 0, 0));
104         marginPanelLeft.add(leftPanel, BorderLayout.CENTER);
105
106         mainPanel.add(marginPanelLeft, BorderLayout.CENTER);
107
108         // Right panel (buttons)
109         JPanel rightPanel = new JPanel();
110         rightPanel.setPreferredSize(new Dimension(300, 150));
111         rightPanel.setLayout(new BoxLayout(rightPanel, BoxLayout.Y_AXIS));
112         rightPanel.setBackground(new Color(230, 230, 230));
113
114         JPanel fixedPanel = new JPanel();
115         fixedPanel.setLayout(new BoxLayout(fixedPanel, BoxLayout.Y_AXIS));
116         fixedPanel.setBackground(Color.WHITE);
117
118         Border paddingBorder = BorderFactory.createEmptyBorder(20, 20, 20, 20);
119         fixedPanel.setBorder(paddingBorder);
```

Implementasi inisiasi MainFrame (3)

Tucil1_13523100 - MainFrame.java

```java
// File buttons
        chooseFileButton = new JButton("Choose File");
        uploadFileButton = new JButton("Upload File");

        chooseFileButton.setPreferredSize(new Dimension(200, 40));
        uploadFileButton.setPreferredSize(new Dimension(200, 40));
        chooseFileButton.setMaximumSize(new Dimension(200, 40));
        uploadFileButton.setMaximumSize(new Dimension(200, 40));

        chooseFileButton.setBackground(Color.BLACK);
        uploadFileButton.setBackground(Color.BLACK);
        chooseFileButton.setFont(new Font("Arial", Font.PLAIN, 15));
        uploadFileButton.setFont(new Font("Arial", Font.PLAIN, 15));
        chooseFileButton.setForeground(Color.WHITE);
        uploadFileButton.setForeground(Color.WHITE);
        chooseFileButton.setFocusPainted(false);
        uploadFileButton.setFocusPainted(false);

        chooseFileButton.setAlignmentX(Component.CENTER_ALIGNMENT);
        uploadFileButton.setAlignmentX(Component.CENTER_ALIGNMENT);

        // Filename text
        JLabel fileName = new JLabel("No file selected", SwingConstants.CENTER);
        fileName.setAlignmentX(Component.CENTER_ALIGNMENT);

        // Choose file button functionality
        chooseFileButton.addActionListener((ActionEvent e) -> {
            JFileChooser fileChooser = new JFileChooser();
            String currentDirectory = System.getProperty("user.dir");
            File currentDirFile = new File(currentDirectory);

            File initialDirectory = new File(currentDirectory);

            if (initialDirectory.exists() && initialDirectory.isDirectory()) {
                fileChooser.setCurrentDirectory(initialDirectory);
            }

            FileNameExtensionFilter txtFilter = new FileNameExtensionFilter("Text Files (.txt)", "txt");
            fileChooser.setFileFilter(txtFilter);

            int result = fileChooser.showOpenDialog(MainFrame.this);

            if (result == JFileChooser.APPROVE_OPTION) {
                selectedFile = fileChooser.getSelectedFile();
                fileName.setText("Selected file: " + selectedFile.getName());
            } else {
                System.out.println("No file chosen.");
            }
        });
```

Implementasi inisiasi MainFrame (4)

Tucil1_13523100 - MainFrame.java

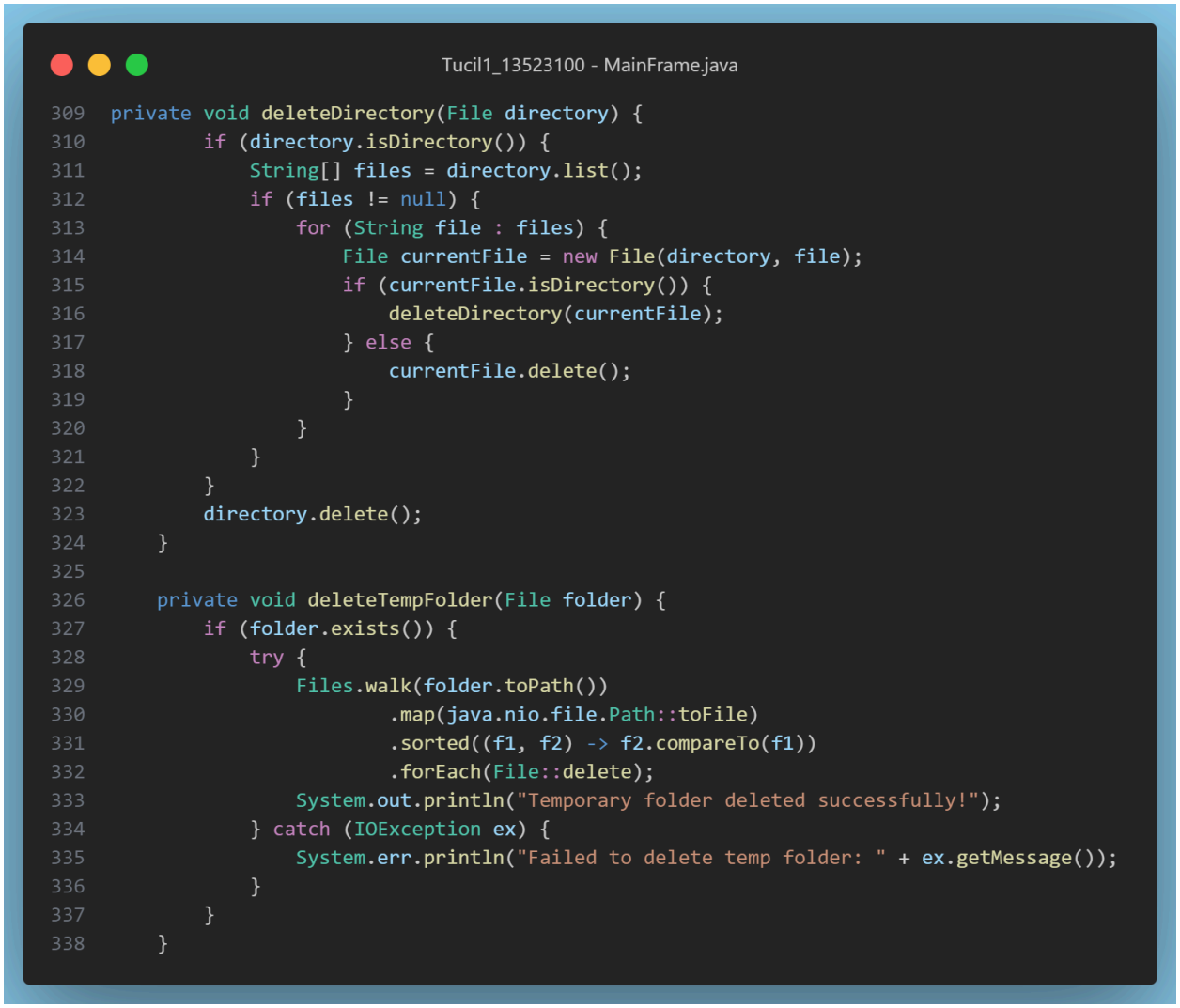
```java
AtomicReference<InputFormat> formRef = new AtomicReference<>(null);

        // Upload file button functionality
        uploadFileButton.addActionListener((ActionEvent e) -> {
            if (selectedFile != null) {
                System.out.println("Uploading file: " + selectedFile.getAbsolutePath());
                InputFormat form = Input.readTxt(selectedFile.getAbsolutePath());
                formRef.set(form);
                InputPuzzlerPro input = form.input;

                if (input == null) {
                    JOptionPane.showMessageDialog(this, form.errorMessage,
                        "Gooner input detected!", JOptionPane.ERROR_MESSAGE);
                } else {
                    String currentDirectory = System.getProperty("user.dir");
                    File currentDirFile = new File(currentDirectory);
                    String tempString = currentDirFile +  File.separator + "temp";
                    File tempFolder = new File(tempString);

                    if (tempFolder.exists()) {
                        for (File file : tempFolder.listFiles()) {
                            if (file.isDirectory()) {
                                deleteDirectory(file);
                            } else {
                                file.delete();
                            }
                        }
                        tempFolder.delete();
                    }

                    InputImageGenerator.saveInput(input);

                    if (!tempFolder.exists()) {
                        JOptionPane.showMessageDialog(this, "Input process failed, input image not created",
                        "Error", JOptionPane.ERROR_MESSAGE);
                    } else {
                        File emptyBoardFile = new File(tempFolder, "emptyBoard.png");
                        if (emptyBoardFile.exists()) {
                            ImageIcon emptyBoardIcon = new ImageIcon(emptyBoardFile.getAbsolutePath());
                            JLabel emptyBoardLabel = new JLabel(emptyBoardIcon);

                            SwingUtilities.invokeLater(() -> {
                                emptyBox.removeAll();
                                emptyBox.add(emptyBoardLabel, BorderLayout.CENTER);
                                emptyBox.revalidate();
                                emptyBox.repaint();
                            });
                        }

                        for (int i = 0; i < input.piecesList.size(); i++) {
                            String pieceFileName = "piece" + (i + 1) + ".png";
                            File pieceFile = new File(tempFolder, pieceFileName);

                            if (pieceFile.exists()) {
                                ImageIcon pieceIcon = new ImageIcon(pieceFile.getAbsolutePath());
                                JLabel pieceLabel = new JLabel(pieceIcon);
                                pieceLabel.setHorizontalAlignment(SwingConstants.CENTER);
                                pieceLabel.setVerticalAlignment(SwingConstants.CENTER);

                                int panelIndex = i;
                                SwingUtilities.invokeLater(() -> {
                                    JPanel panelToUpdate = imagePanels[panelIndex];
                                    panelToUpdate.removeAll();
                                    panelToUpdate.setLayout(new BorderLayout());
                                    panelToUpdate.add(pieceLabel, BorderLayout.CENTER);
                                    panelToUpdate.revalidate();
                                    panelToUpdate.repaint();
                                });
                            }
                        }
                    }
                }
            } else {
                JOptionPane.showMessageDialog(this, "No file selected to upload.",
                        "Error", JOptionPane.ERROR_MESSAGE);
            }
        });
```

Implementasi inisiasi MainFrame (5)

Tucil1_13523100 - MainFrame.java

```
fixedPanel.add(chooseFileButton);
        fixedPanel.add(Box.createVerticalStrut(10));
        fixedPanel.add(uploadFileButton);
        fixedPanel.add(Box.createVerticalStrut(10));
        fixedPanel.add(fileName);

        rightPanel.add(fixedPanel);

        mainPanel.add(rightPanel, BorderLayout.EAST);
        mainPanel.setBorder(BorderFactory.createEmptyBorder(20, 0, 20, 20));

        // Bottom Section
        JPanel bottomPanel = new JPanel(new BorderLayout());
        bottomPanel.setBackground(new Color(230, 230, 230));

        solveButton = new JButton("Solve");
        solveButton.setPreferredSize(new Dimension(500, 50));
        solveButton.setBackground(new Color(0, 122, 255));
        solveButton.setForeground(Color.WHITE);
        solveButton.setFocusPainted(false);
        solveButton.setFont(new Font("Arial", Font.BOLD, 18));

        // Solve button functionality
        solveButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (selectedFile != null && formRef.get() != null) {
                    InputFormat form = formRef.get();
                    if (form.input == null) {
                        JOptionPane.showMessageDialog(MainFrame.this, form.errorMessage,
                        "Error", JOptionPane.ERROR_MESSAGE);
                    } else {
                        new Solution(form);
                    }

                } else {
                    JOptionPane.showMessageDialog(MainFrame.this, "No file selected, can't solve",
                        "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        bottomPanel.add(solveButton);

        JPanel marginPanelBottom = new JPanel();
        marginPanelBottom.setLayout(new BorderLayout());
        marginPanelBottom.setBorder(BorderFactory.createEmptyBorder(0, 20, 0, 0));

        marginPanelBottom.add(bottomPanel, BorderLayout.CENTER);

        mainPanel.add(marginPanelBottom, BorderLayout.SOUTH);

        add(mainPanel);
        setMinimumSize(new Dimension(1200, 800));
        pack();

        // Display the frame
        setVisible(true);
```

Fungsi pembantu untuk file management



File Solution

Fungsi yang dipanggil untuk membuka solution dan juga delete folder temp kalau program ditutup

Tucil1_13523100 - Solution.java

```java
public Solution(InputFormat form ) {
        String currentDirectory = System.getProperty("user.dir");
        File currentDirFile = new File(currentDirectory);
        String tempString = currentDirFile + File.separator + "temp";

        initialize(form);

        this.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                deleteTempFolder(new File(tempString));
            }
        });
    }
```

Implementasi Solution (1)

Tucil1_13523100 - Solution.java

```java
public void initialize(InputFormat form) {
        OutputPuzzlerPro output = BruteForce.getOutput(form);
        output.printDetails();

        // Temporary folder for saving the solution
        String currentDirectory = System.getProperty("user.dir");
        File currentDirFile = new File(currentDirectory);
        String tempString = currentDirFile + File.separator + "temp";
        File tempFolder = new File(tempString);

        if (!tempFolder.exists()) {
            if (tempFolder.mkdirs()) {
                System.out.println("Temporary folder created: " + tempFolder.getAbsolutePath());
            } else {
                System.out.println("Failed to create temporary folder.");
            }
        } else {
            System.out.println("Temporary folder already exists.");
        }

        // Define the file path for the solution image
        String filename = tempFolder.getAbsolutePath() + File.separator + "solution.png";

        JPanel topPanel = new JPanel();
        topPanel.setLayout(new BoxLayout(topPanel, BoxLayout.Y_AXIS));
        topPanel.setBackground(Color.WHITE);

        JLabel boardLabel = new JLabel("Solution Board");
        boardLabel.setFont(new Font("Arial", Font.BOLD, 24));
        boardLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        boardLabel.setBorder(BorderFactory.createEmptyBorder(20, 0, 0, 0));
        topPanel.add(boardLabel);

        // Check if the solution board is available
        if (output.filledBoardSolution != null) {
            OutputImage.solutionDisplayImage(output.filledBoardSolution, filename, "LIGHT");

            ImageIcon boardImageIcon = new ImageIcon(filename);
            JLabel boardImageLabel = new JLabel(boardImageIcon);

            boardImageLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
            topPanel.add(boardImageLabel);
        } else {
            JPanel noSolutionPanel = new JPanel();
            noSolutionPanel.setPreferredSize(new Dimension(400, 400));
            noSolutionPanel.setBackground(Color.WHITE);
            JLabel noSolutionLabel = new JLabel("No Valid Solution");
            noSolutionLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
            noSolutionPanel.add(noSolutionLabel);
            noSolutionPanel.setAlignmentX(Component.CENTER_ALIGNMENT);
            topPanel.add(noSolutionPanel);
        }
```

Implementasi Solution (2)

Tucil1_13523100 - Solution.java

```
// Panel to house infoPanel
        JPanel centerPanel = new JPanel();
        centerPanel.setBackground(new Color(240, 240, 240));

        // Execution time and number of cases explored stored in infoPanel
        JPanel infoPanel = new JPanel();
        infoPanel.setLayout(new BoxLayout(infoPanel, BoxLayout.Y_AXIS));
        infoPanel.setBackground(Color.WHITE);
        infoPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        infoPanel.setPreferredSize(new Dimension(600, 100));
        infoPanel.setMaximumSize(new Dimension(600, 100));

        // Create the execution time label
        JLabel executionTimeLabel = new JLabel("Execution Time = " + output.executionTime + " ms");
        executionTimeLabel.setFont(new Font("Arial", Font.BOLD, 20));
        executionTimeLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
        infoPanel.add(executionTimeLabel);

        infoPanel.add(Box.createVerticalStrut(10));

        // Create the cases explored label
        JLabel casesExploredLabel = new JLabel("Number of Cases Explored = " + output.casesExplored);
        casesExploredLabel.setFont(new Font("Arial", Font.BOLD, 20));
        casesExploredLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
        infoPanel.add(casesExploredLabel);

        centerPanel.add(infoPanel);

        // Bottom Panel with buttons
        JPanel bottomPanel = new JPanel();
        bottomPanel.setLayout(new GridLayout(1, 2, 10, 10));  // 1 row, 2 columns, 10px horizontal and vertical gap
        bottomPanel.setBackground(Color.WHITE);
        bottomPanel.setBorder(new EmptyBorder(20, 20, 20, 20));

        // Button for saving the solution as text and image
        JButton saveTextButton = new JButton("Save Solution Text");
        JButton saveImageButton = new JButton("Save Solution Image");

        saveTextButton.setMaximumSize(new Dimension(300, 100));
        saveImageButton.setMaximumSize(new Dimension(300, 100));

        saveTextButton.setBackground(Color.BLACK);
        saveImageButton.setBackground(Color.BLACK);
        saveTextButton.setFont(new Font("Arial", Font.PLAIN, 15));
        saveImageButton.setFont(new Font("Arial", Font.PLAIN, 15));
        saveTextButton.setForeground(Color.WHITE);
        saveImageButton.setForeground(Color.WHITE);
        saveTextButton.setFocusPainted(false);
        saveImageButton.setFocusPainted(false);
```

Implementasi Solution (3)

Tucil1_13523100 - Solution.java

```java
// Save solution text button functionality
        saveTextButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser(currentDirFile);
                fileChooser.setDialogTitle("Choose a location to save the solution text");
                fileChooser.setSelectedFile(new File("solution.txt"));

                int returnValue = fileChooser.showSaveDialog(null);
                if (returnValue == JFileChooser.APPROVE_OPTION) {
                    File selectedFile = fileChooser.getSelectedFile();
                    // Ensure the file has the ".txt" extension
                    if (!selectedFile.getName().endsWith(".txt")) {
                        selectedFile = new File(selectedFile.getAbsolutePath() + ".txt");
                    }
                    String fname = selectedFile.getAbsolutePath();
                    OutputTxt.output(form.input, output, fname);  // Save the text file
                }
            }
        });
        bottomPanel.add(saveTextButton);

        // Save solution as image button functionality
        saveImageButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser(currentDirFile);
                fileChooser.setDialogTitle("Choose a location to save the solution image");
                fileChooser.setSelectedFile(new File("solution.png"));

                int returnValue = fileChooser.showSaveDialog(null);
                if (returnValue == JFileChooser.APPROVE_OPTION) {
                    File selectedFile = fileChooser.getSelectedFile();
                    // Ensure the file has the ".png" extension
                    if (!selectedFile.getName().endsWith(".png")) {
                        selectedFile = new File(selectedFile.getAbsolutePath() + ".png");
                    }
                    String fname = selectedFile.getAbsolutePath();
                    OutputImage.solutionImage(output.filledBoardSolution, fname, "LIGHT");  // Save the image
                }
            }
        });
        bottomPanel.add(saveImageButton);

        // mainPanel
        JPanel mainPanel = new JPanel(new BorderLayout(20, 20));
        mainPanel.setBackground(new Color(240, 240, 240));
        mainPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        // Add all the components to the main panel
        mainPanel.add(topPanel, BorderLayout.NORTH);
        mainPanel.add(centerPanel, BorderLayout.CENTER);
        mainPanel.add(bottomPanel, BorderLayout.SOUTH);

        JScrollPane scrollPane = new JScrollPane(mainPanel);
        scrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
        scrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);

        setTitle("Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        getContentPane().setBackground(new Color(240, 240, 240));
        add(mainPanel);
        setMinimumSize(new Dimension(1200, 800));
        pack();
        setVisible(true);
```

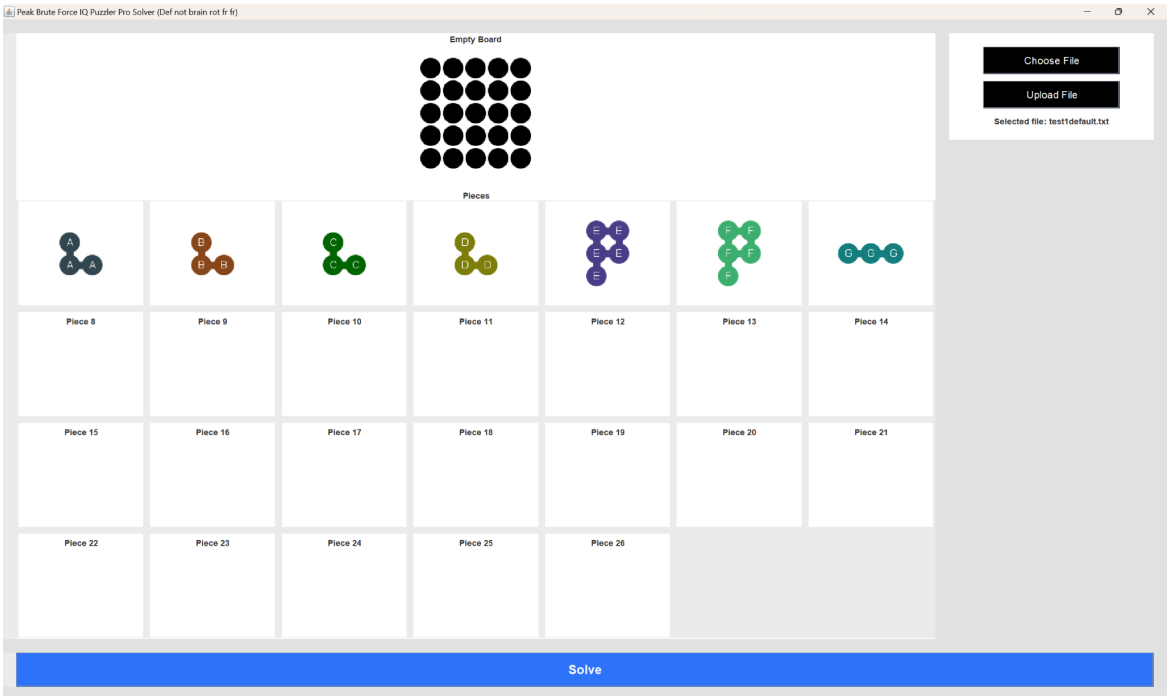
3. Test Case

Test Case 1 (default)

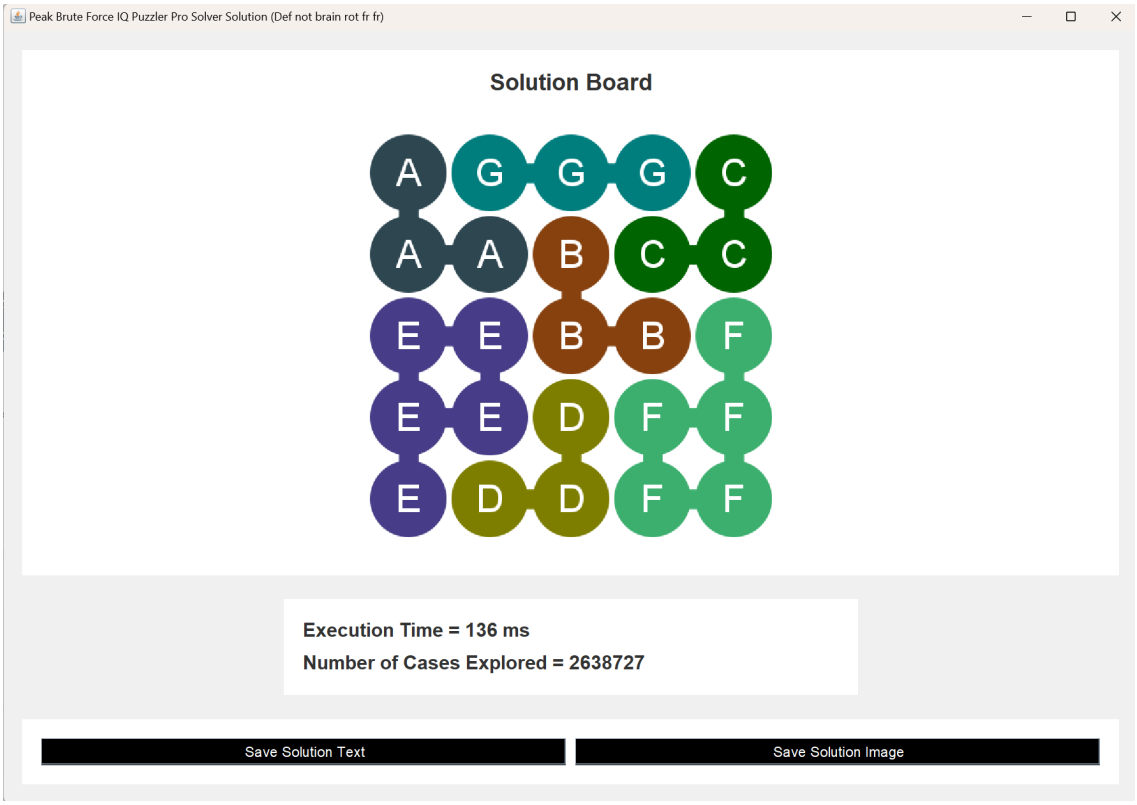
Input

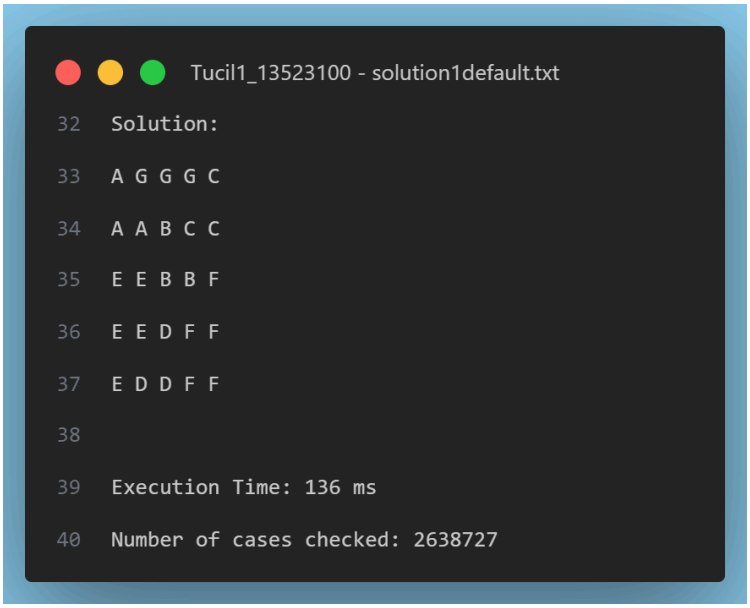
test > test1default.txt

```
5 5 7
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
```


Peak Brute Force IQ Puzzler Pro Solver (Def not brain rot fr fr)
Empty Board
Choose File
Upload File
Selected file: test1default.txt
Pieces
Piece 8
Piece 9
Piece 10
Piece 11
Piece 12
Piece 13
Piece 14
Piece 15
Piece 16
Piece 17
Piece 18
Piece 19
Piece 20
Piece 21
Piece 22
Piece 23
Piece 24
Piece 25
Piece 26
Solve

Output


Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
A G G G C
A A B C C
E E B B F
E E D F F
E D D F F
Execution Time = 136 ms
Number of Cases Explored = 2638727
Save Solution Text
Save Solution Image

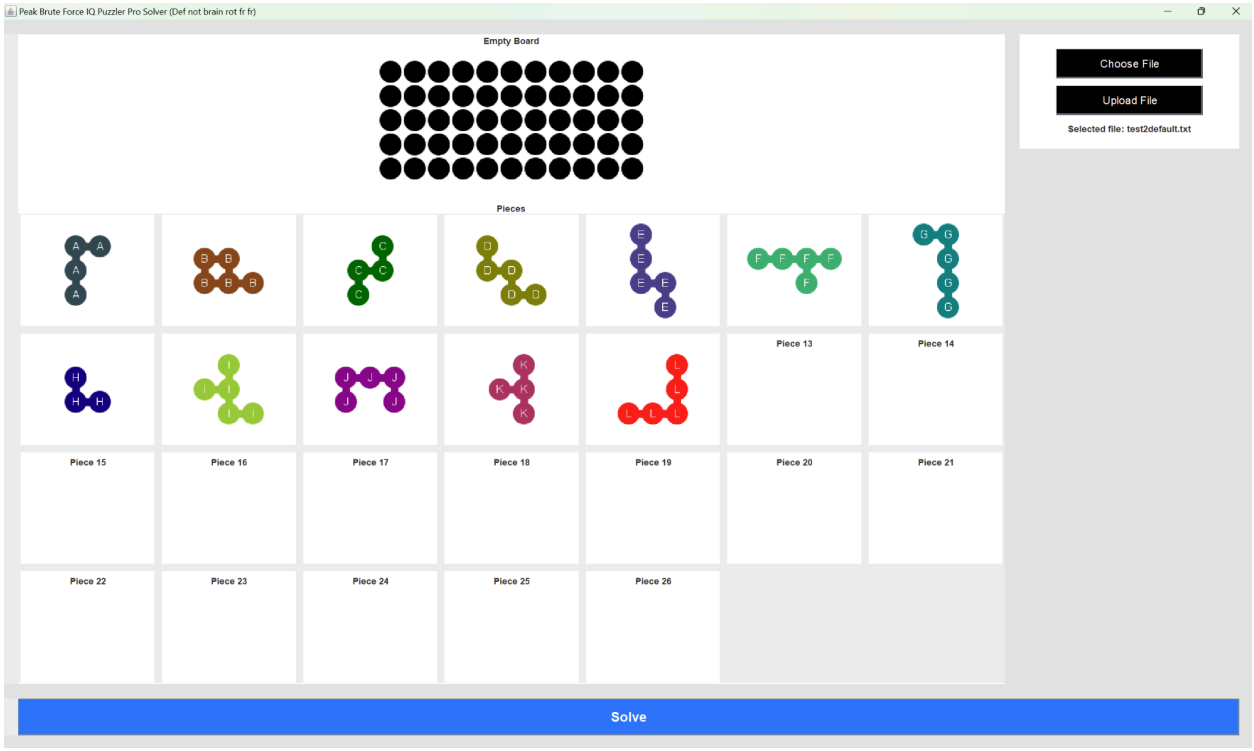


Test Case 2 (default)

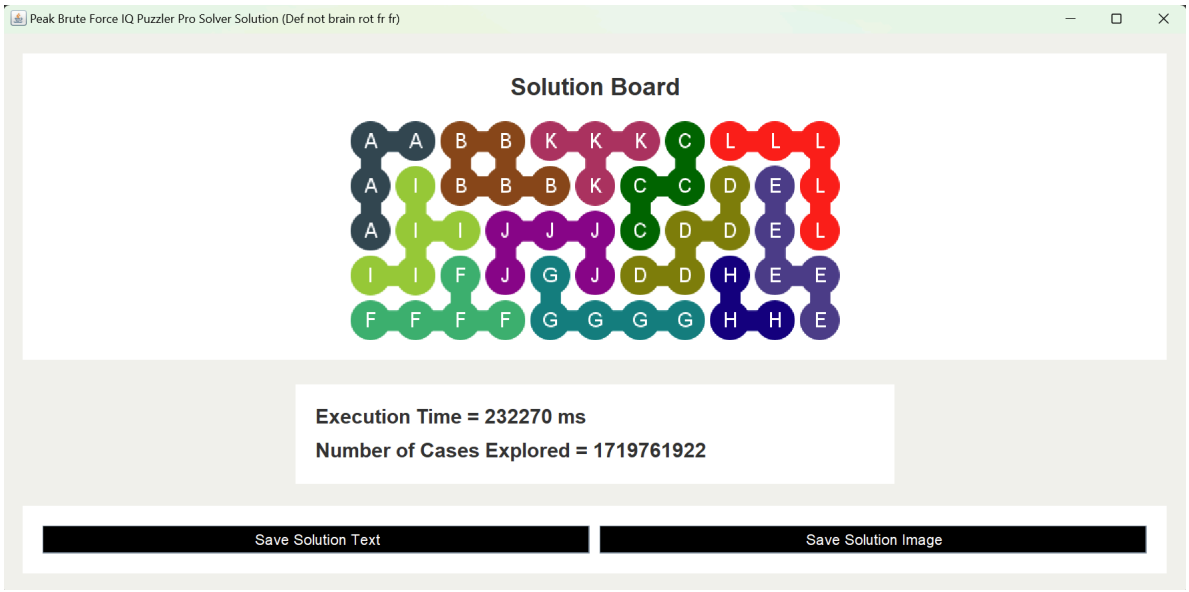
Input

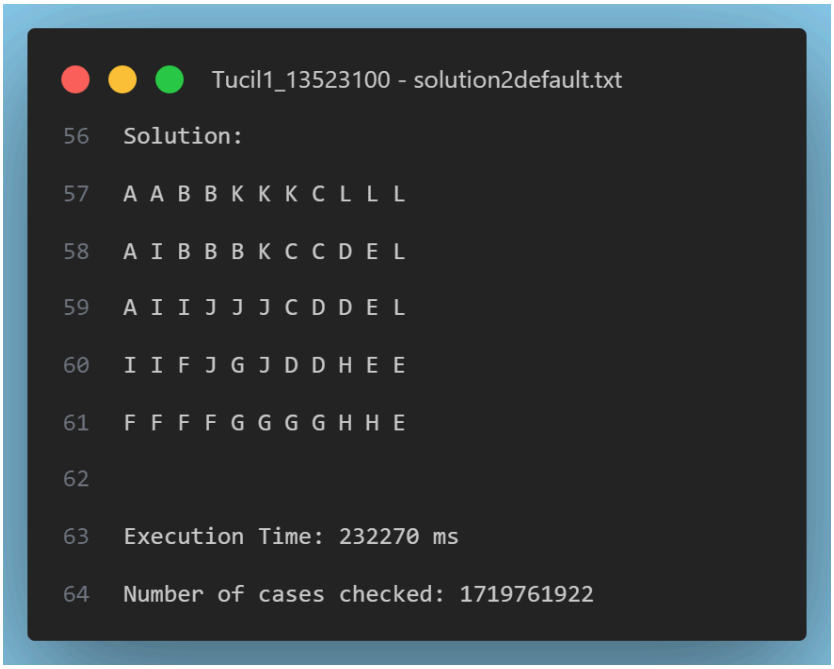
test > test2default.txt

```
5 11 12
DEFAULT
AA
A
A
BB
BBB
 C
CC
C
D
DD
 DD
E
E
EE
 E
FFFF
  F
GG
 G
 G
 G
H
HH
 I
II
 II
JJJ
J J
 K
KK
 K
  L
  L
LLL
```

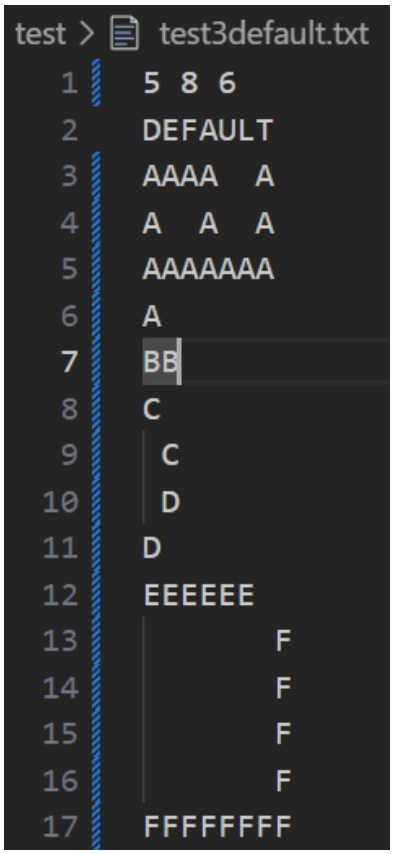

Peak Brute Force IQ Puzzler Pro Solver (Def not brain rot fr fr)
Empty Board
Choose File
Upload File
Selected file: test2default.txt
Pieces
Piece 13
Piece 14
Piece 15
Piece 16
Piece 17
Piece 18
Piece 19
Piece 20
Piece 21
Piece 22
Piece 23
Piece 24
Piece 25
Piece 26
Solve

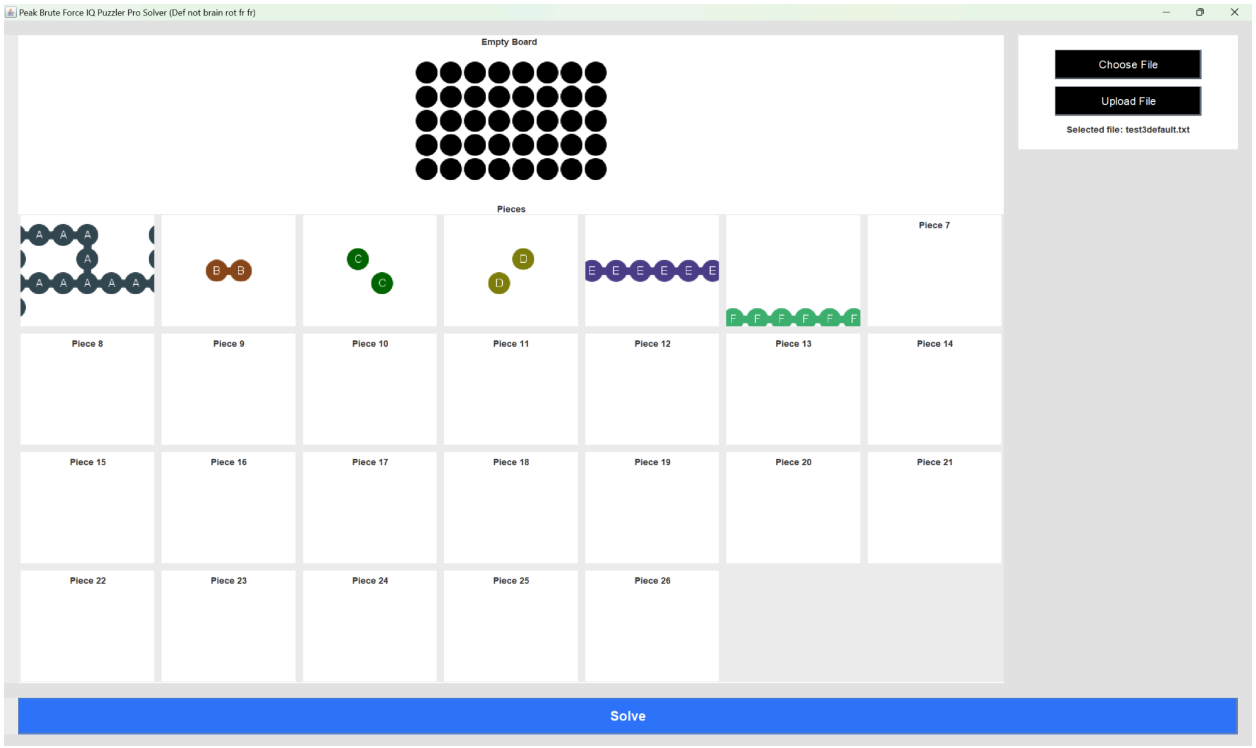
Output


Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
Execution Time = 232270 ms
Number of Cases Explored = 1719761922
Save Solution Text
Save Solution Image

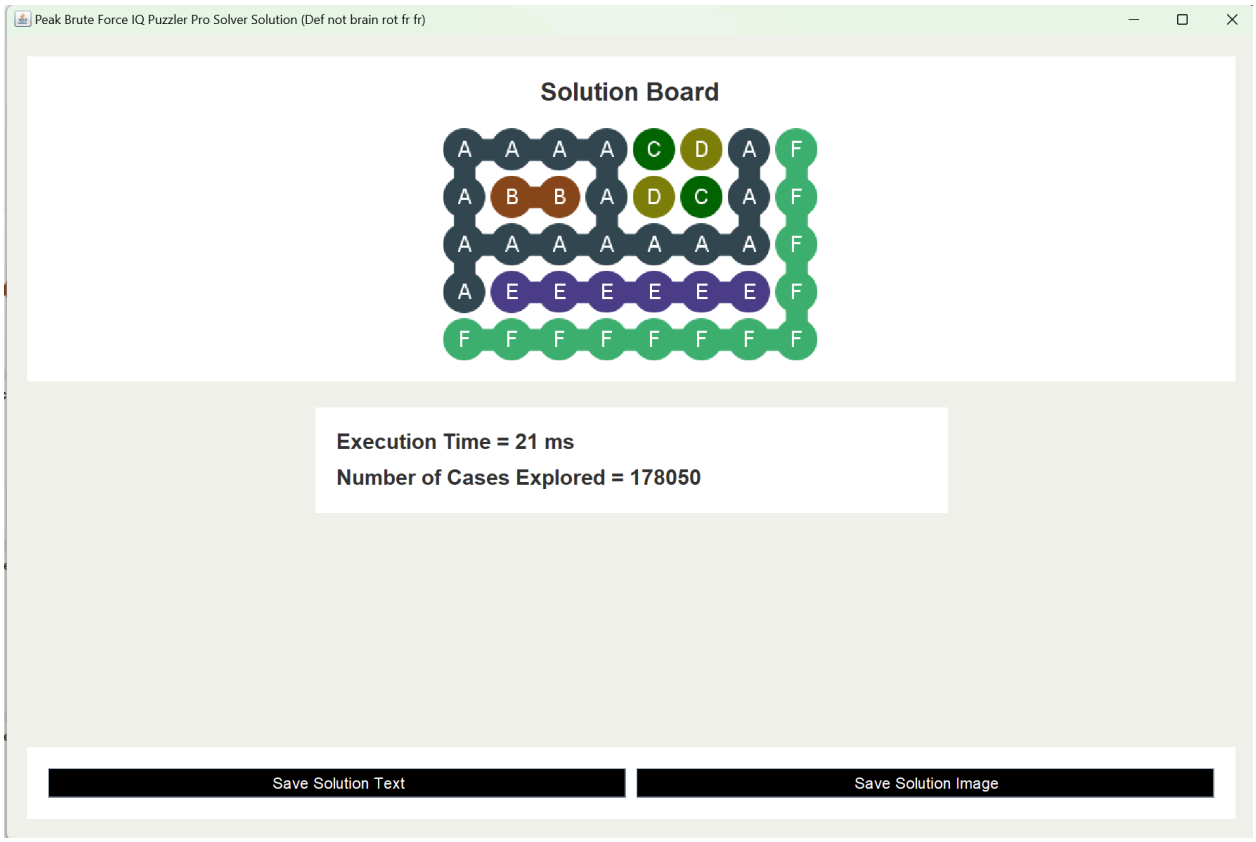


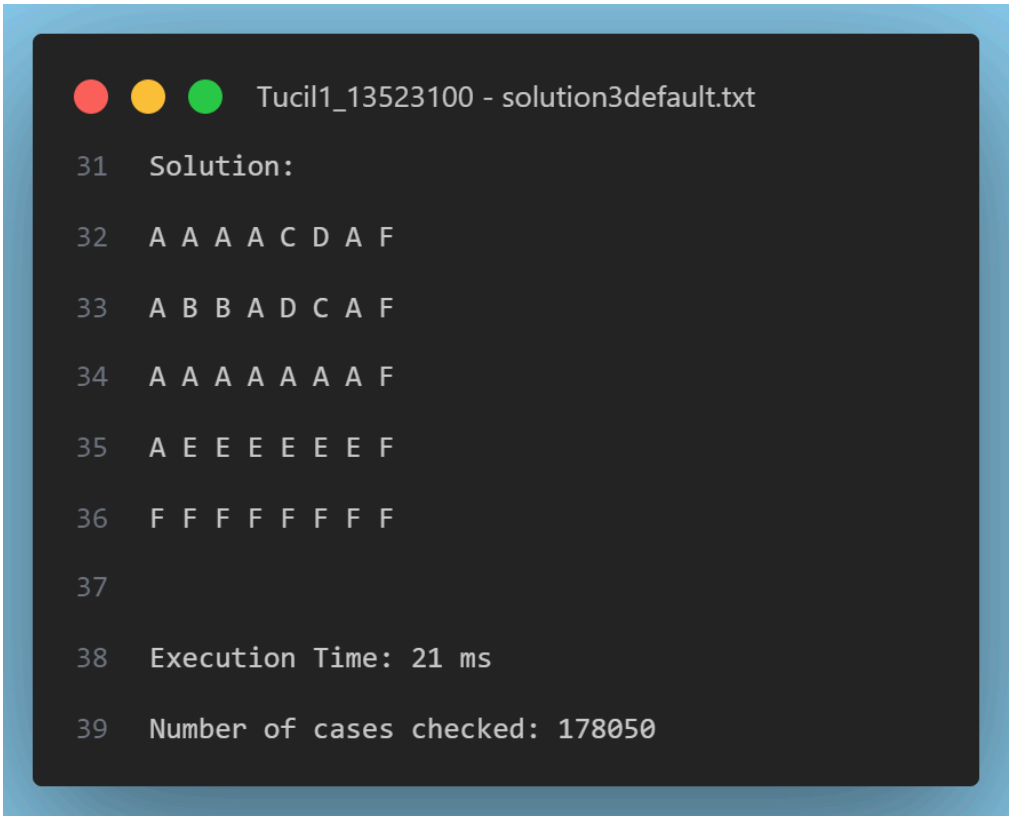
Test Case 3 (default)



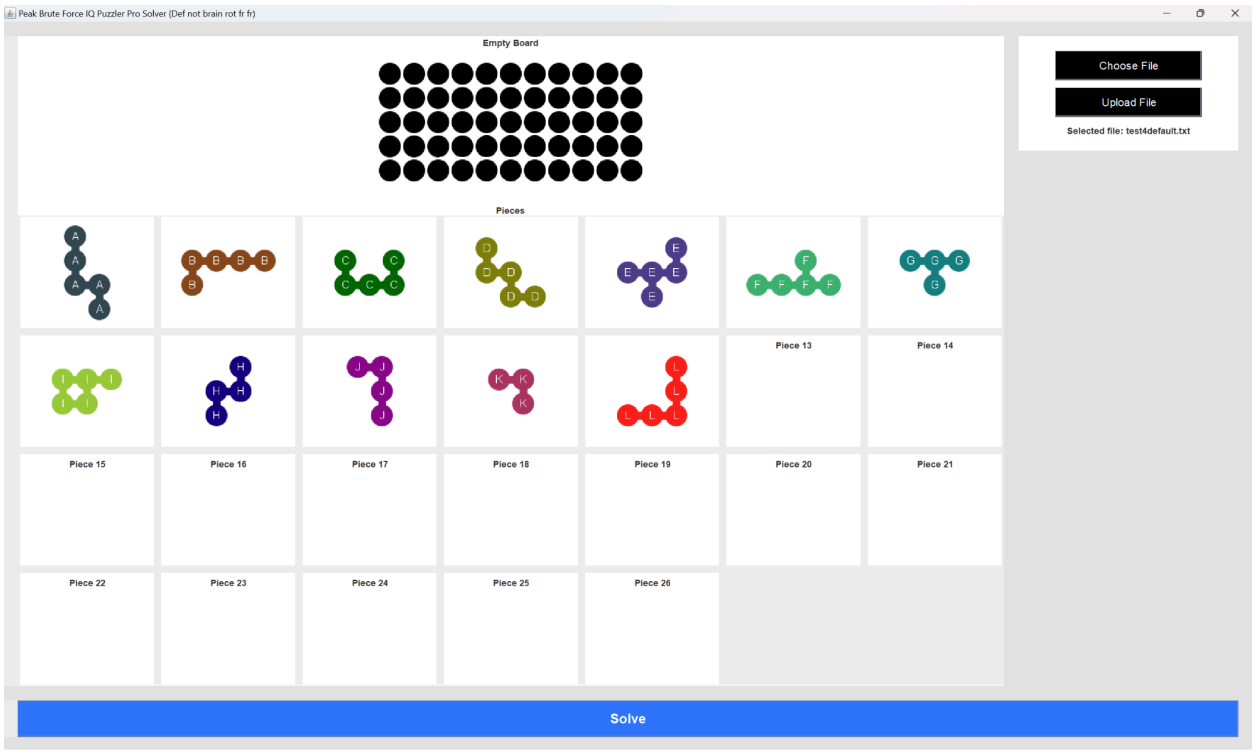

Peak Brute Force IQ Puzzler Pro Solver (Def not brain rot fr fr)
Empty Board
Choose File
Upload File
Selected file: test3default.txt
Pieces
Piece 7
Piece 8
Piece 9
Piece 10
Piece 11
Piece 12
Piece 13
Piece 14
Piece 15
Piece 16
Piece 17
Piece 18
Piece 19
Piece 20
Piece 21
Piece 22
Piece 23
Piece 24
Piece 25
Piece 26
Solve

Output


Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
Execution Time = 21 ms
Number of Cases Explored = 178050
Save Solution Text
Save Solution Image


Tucil1_13523100 - solution3default.txt
31 Solution:
32 A A A A C D A F
33 A B B A D C A F
34 A A A A A A A F
35 A E E E E E E F
36 F F F F F F F F
37
38 Execution Time: 21 ms
39 Number of cases checked: 178050

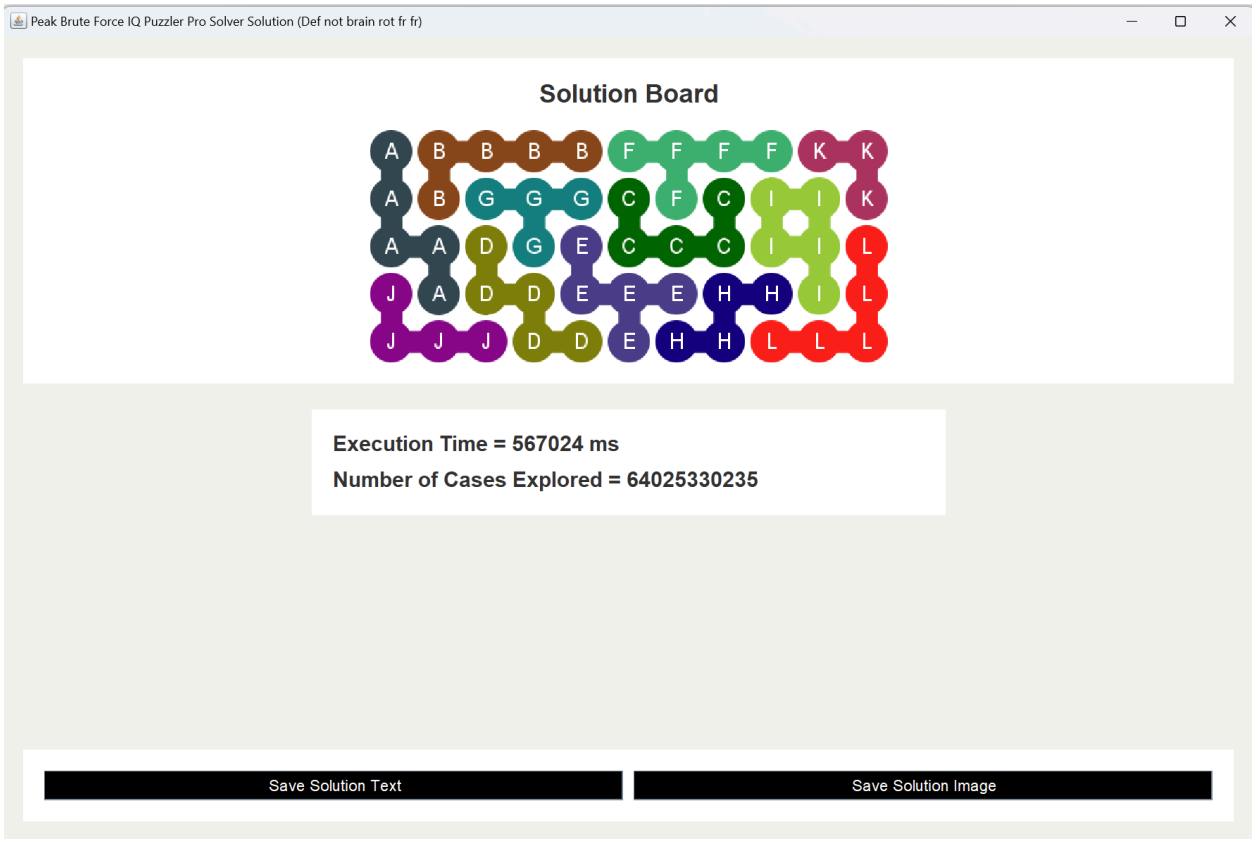
Test Case 4 (default)

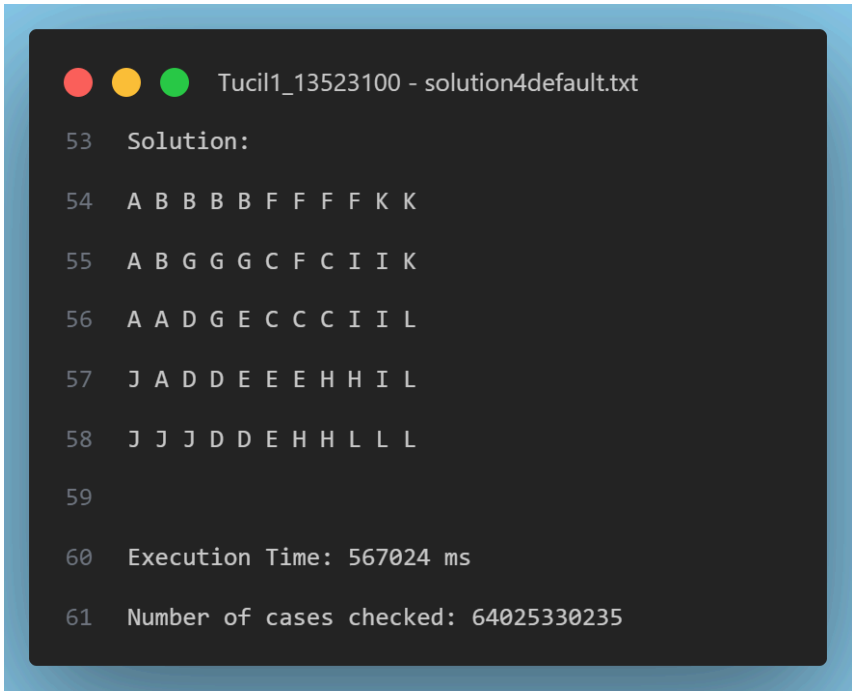


test > test4default.txt

```
5 11 12
DEFAULT
A
A
AA
 A
BBBB
B
C C
CCC
D
DD
 DD
  E
EEE
 E
  F
FFFF
GGG
 G
III
II
 H
HH
H
JJ
 J
 J
KK
 K
  L
  L
LLL
```

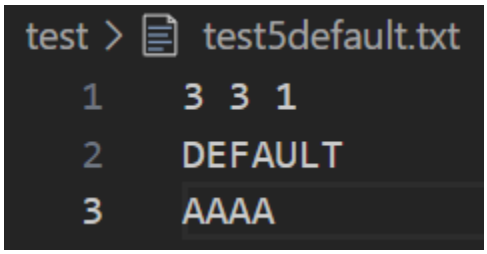
Output

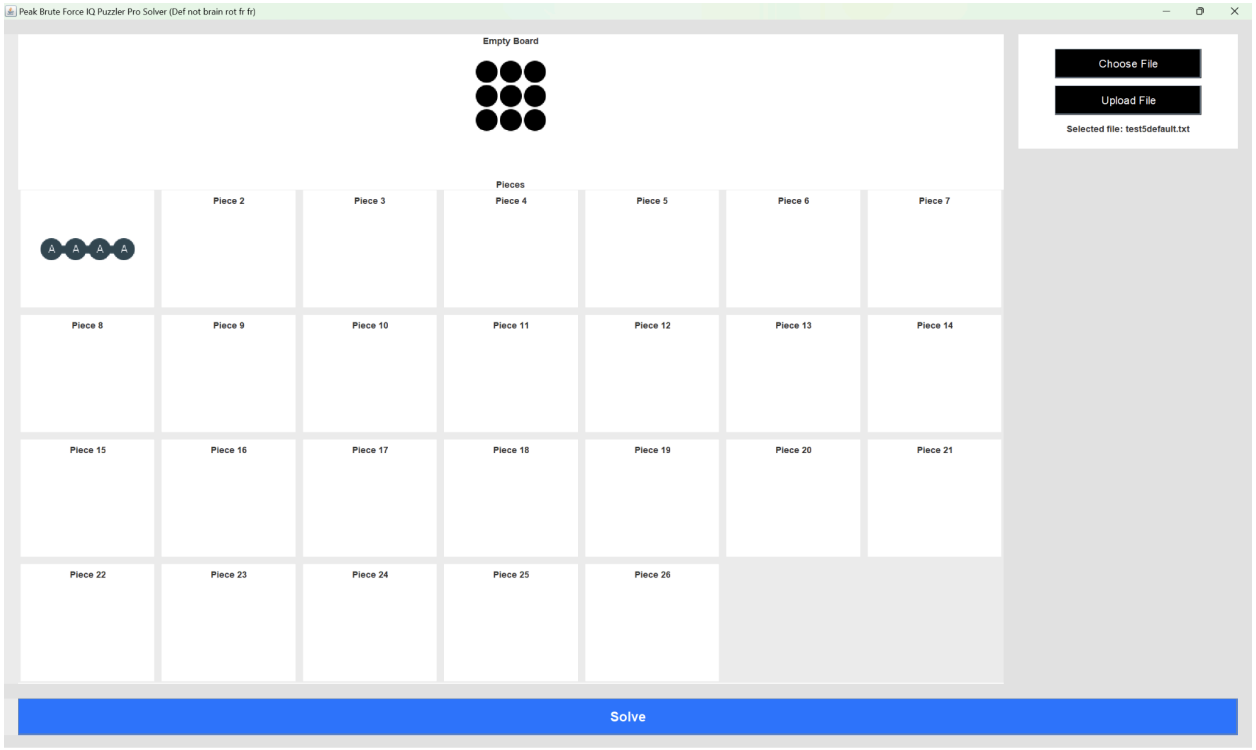

Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
A B B B B F F F F K K
A B G G G C F C I I K
A A D G E C C C I I L
J A D D E E E H H I L
J J J D D E H H L L L
Execution Time = 567024 ms
Number of Cases Explored = 64025330235
Save Solution Text
Save Solution Image


Tucil1_13523100 - solution4default.txt
53 Solution:
54 A B B B B F F F F K K
55 A B G G G C F C I I K
56 A A D G E C C C I I L
57 J A D D E E E H H I L
58 J J J D D E H H L L L
59
60 Execution Time: 567024 ms
61 Number of cases checked: 64025330235

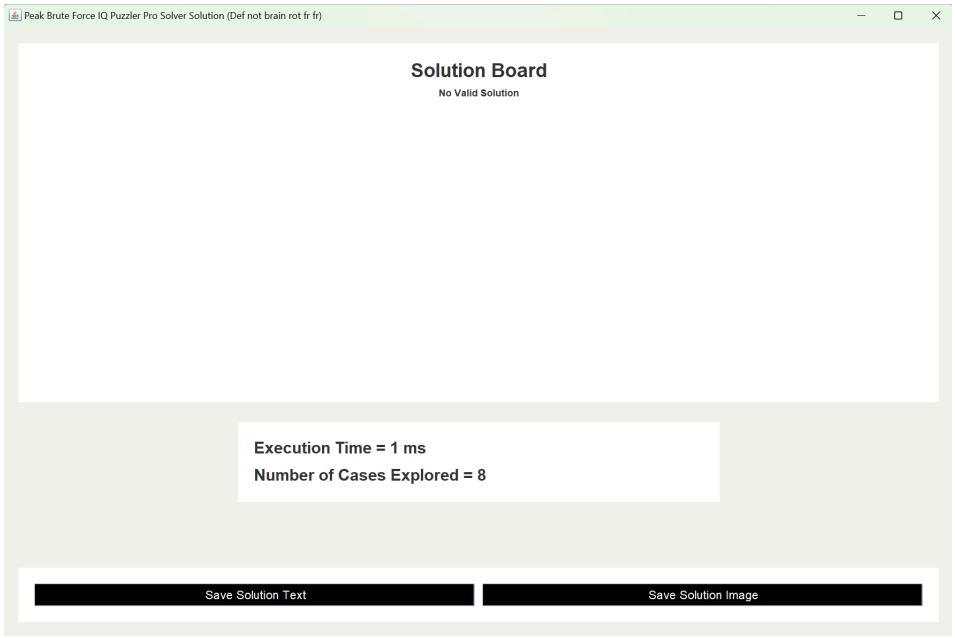
Test Case 5 (default)

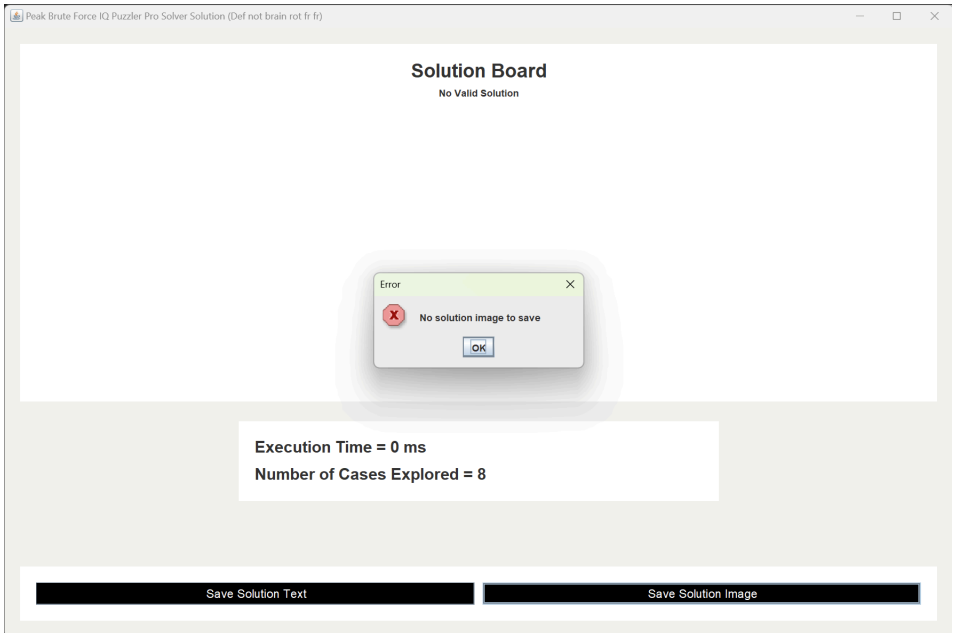
Input

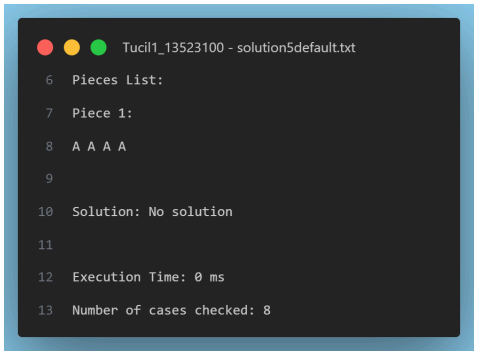




Output

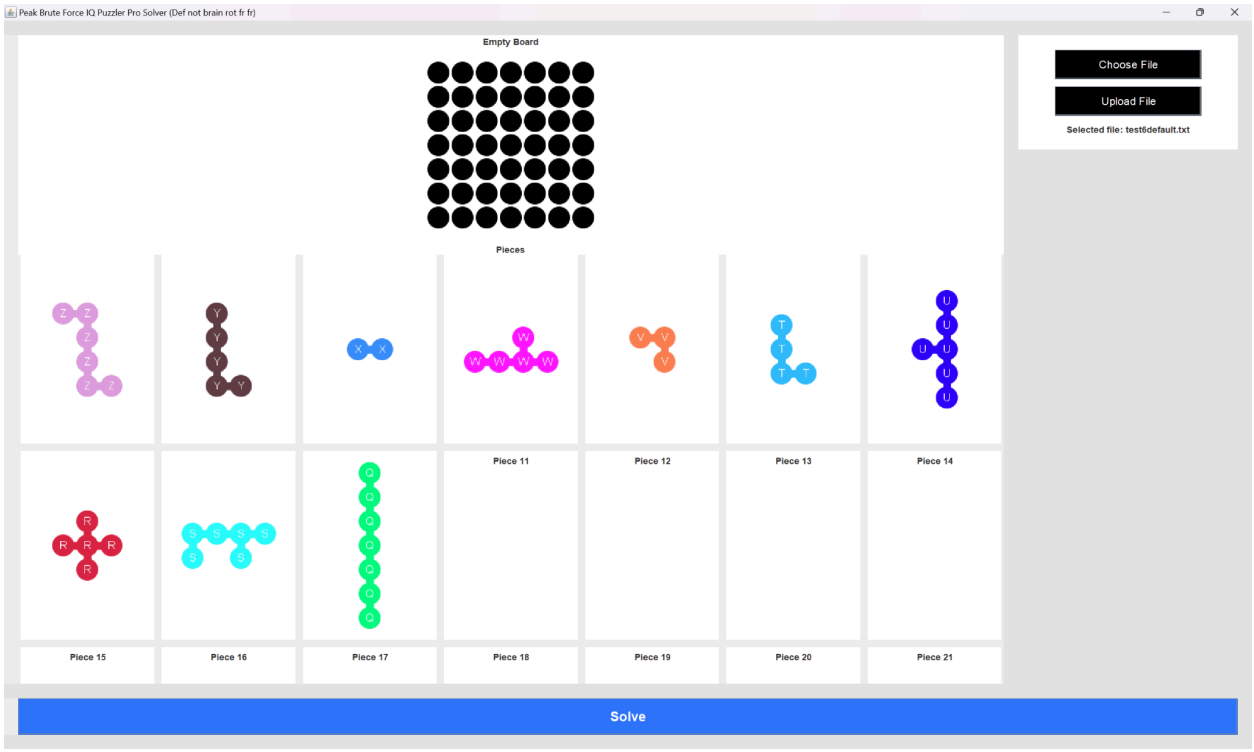

Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
No Valid Solution
Execution Time = 1 ms
Number of Cases Explored = 8
Save Solution Text
Save Solution Image


Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
No Valid Solution
Error
No solution image to save
OK
Execution Time = 0 ms
Number of Cases Explored = 8
Save Solution Text
Save Solution Image


Tucil1_13523100 - solution5default.txt
Pieces List:
Piece 1:
A A A A
Solution: No solution
Execution Time: 0 ms
Number of cases checked: 8

Test Case 6 (default)

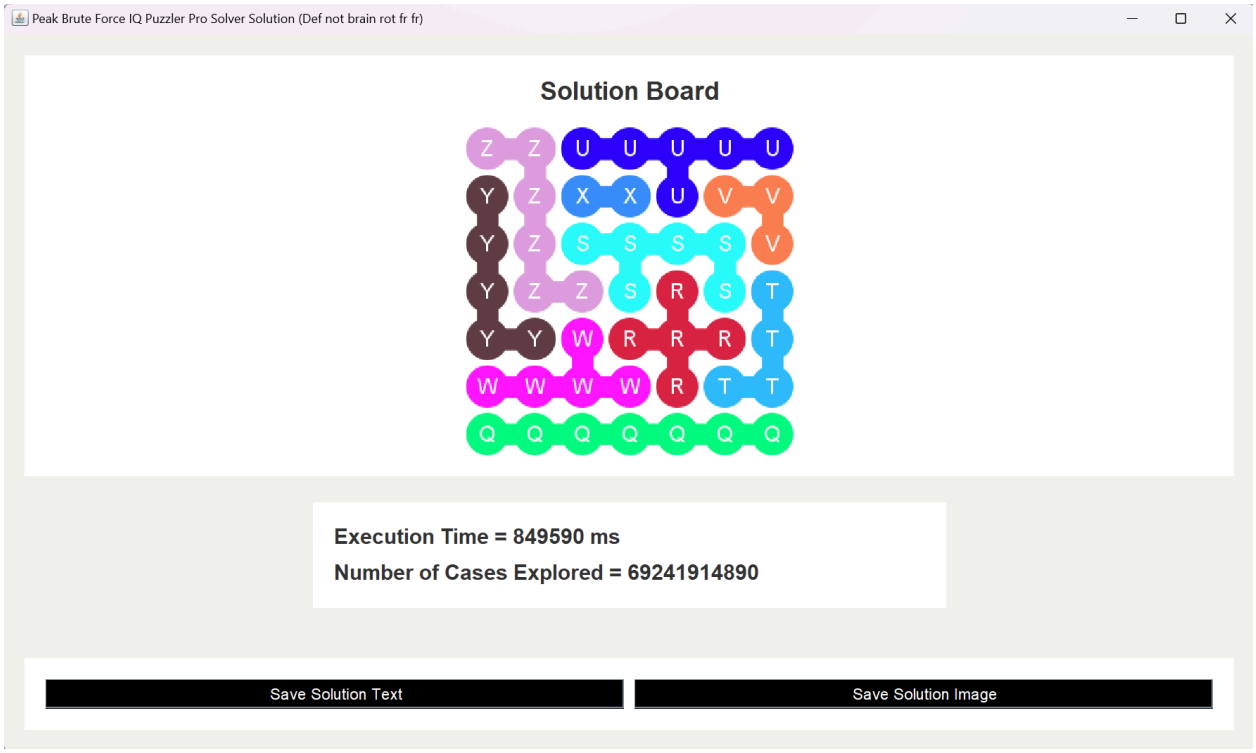
Input

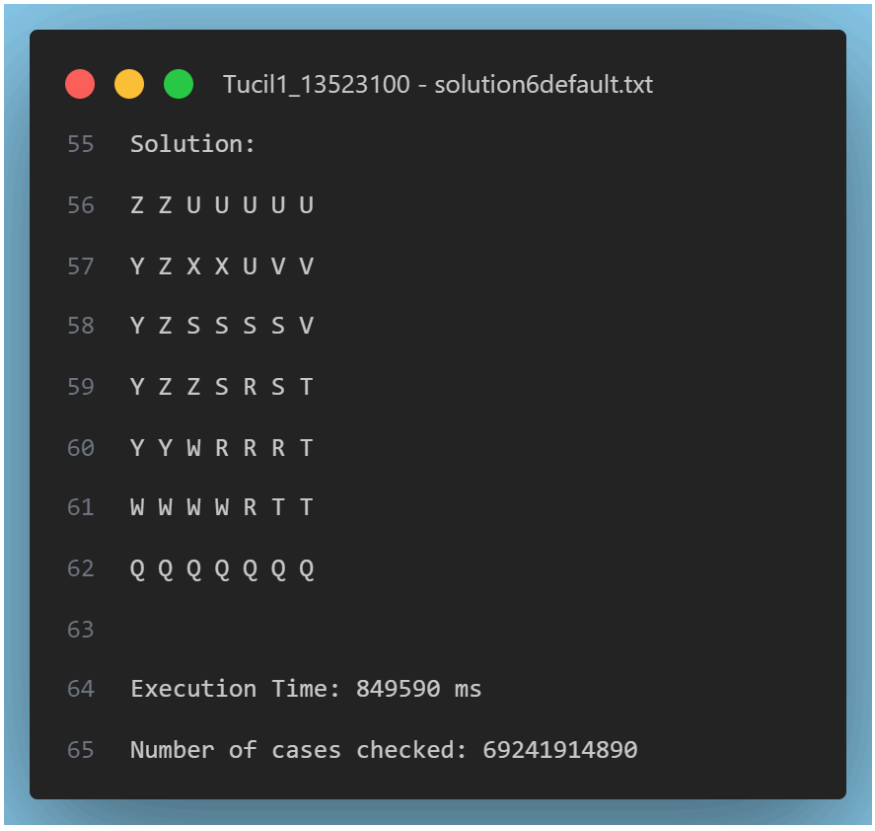


test > test6default.txt

```
7 7 10
DEFAULT
ZZ
 Z
 Z
 ZZ
Y
Y
Y
YY
XX
  W
WWWW
VV
 V
T
T
TT
 U
 U
UU
 U
 U
 R
RRR
 R
SSSS
S S
Q
Q
Q
Q
Q
Q
Q
```

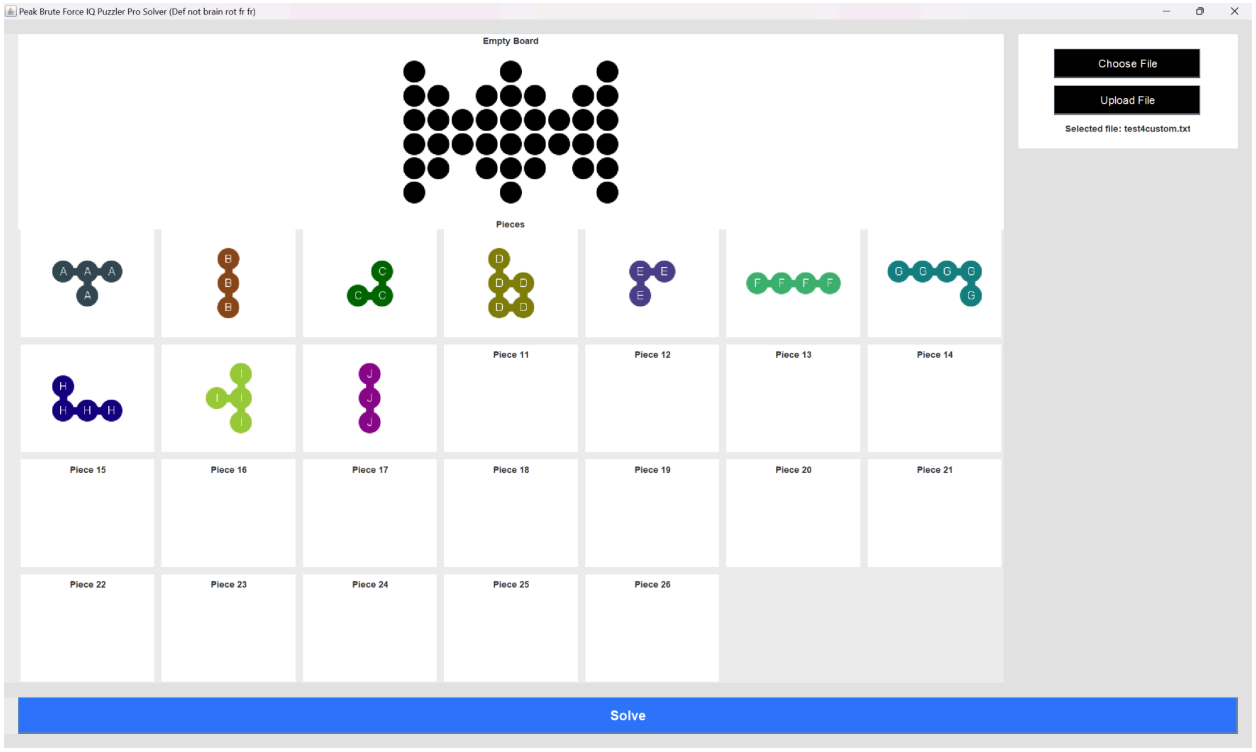
Output


Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
Z Z U U U U U
Y Z X X U V V
Y Z S S S S V
Y Z Z S R S T
Y Y W R R R T
W W W W R T T
Q Q Q Q Q Q Q
Execution Time = 849590 ms
Number of Cases Explored = 69241914890
Save Solution Text
Save Solution Image


Tucil1_13523100 - solution6default.txt
55 Solution:
56 Z Z U U U U U
57 Y Z X X U V V
58 Y Z S S S S V
59 Y Z Z S R S T
60 Y Y W R R R T
61 W W W W R T T
62 Q Q Q Q Q Q Q
63
64 Execution Time: 849590 ms
65 Number of cases checked: 69241914890

Test Case 7 (Custom)

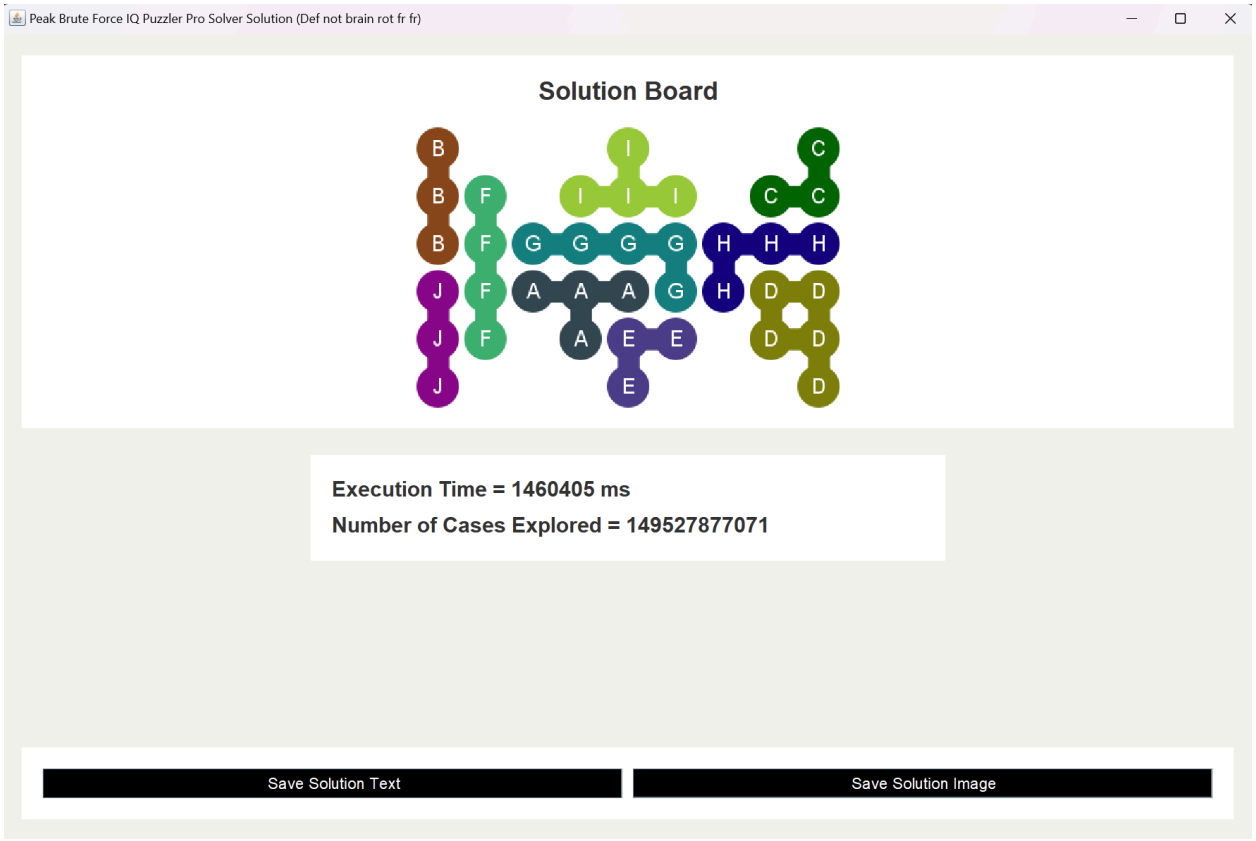
Input

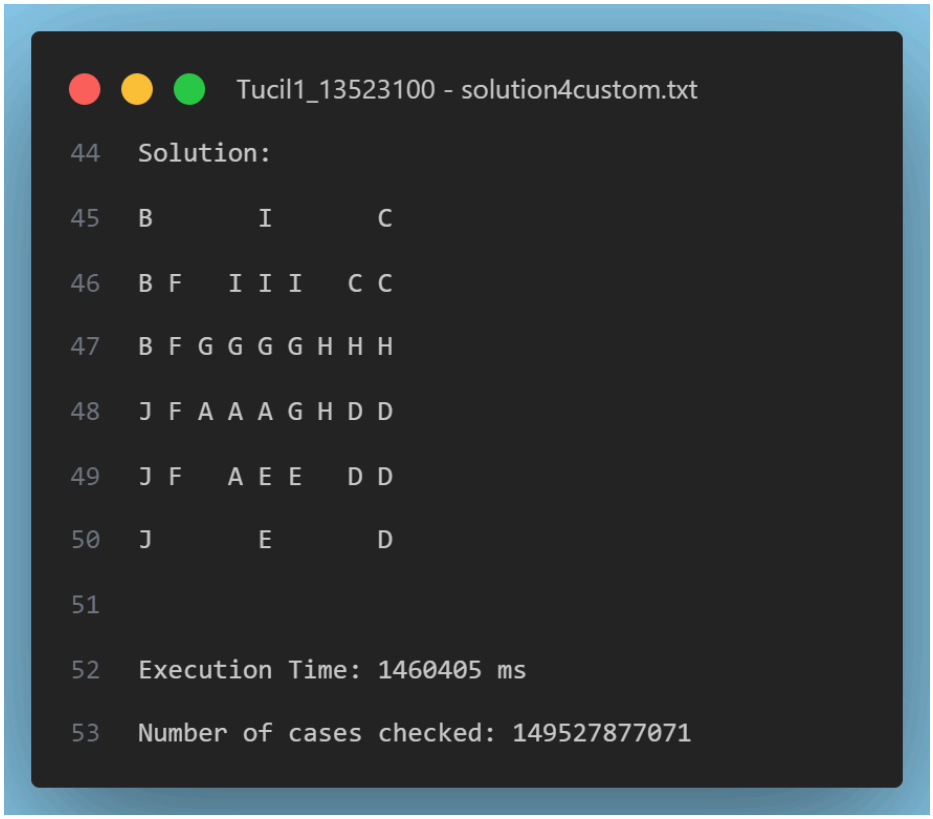


test > test4custom.txt

```
6 9 10
CUSTOM
X...X...X
XX.XXX.XX
XXXXXXXXX
XXXXXXXXX
XX.XXX.XX
X...X...X
AAA
 A
B
B
B
 C
CC
D
DD
DD
EE
E
FFFF
GGGG
   G
H
HHH
 I
II
 I
J
J
J
```

Output


Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
Execution Time = 1460405 ms
Number of Cases Explored = 149527877071
Save Solution Text
Save Solution Image

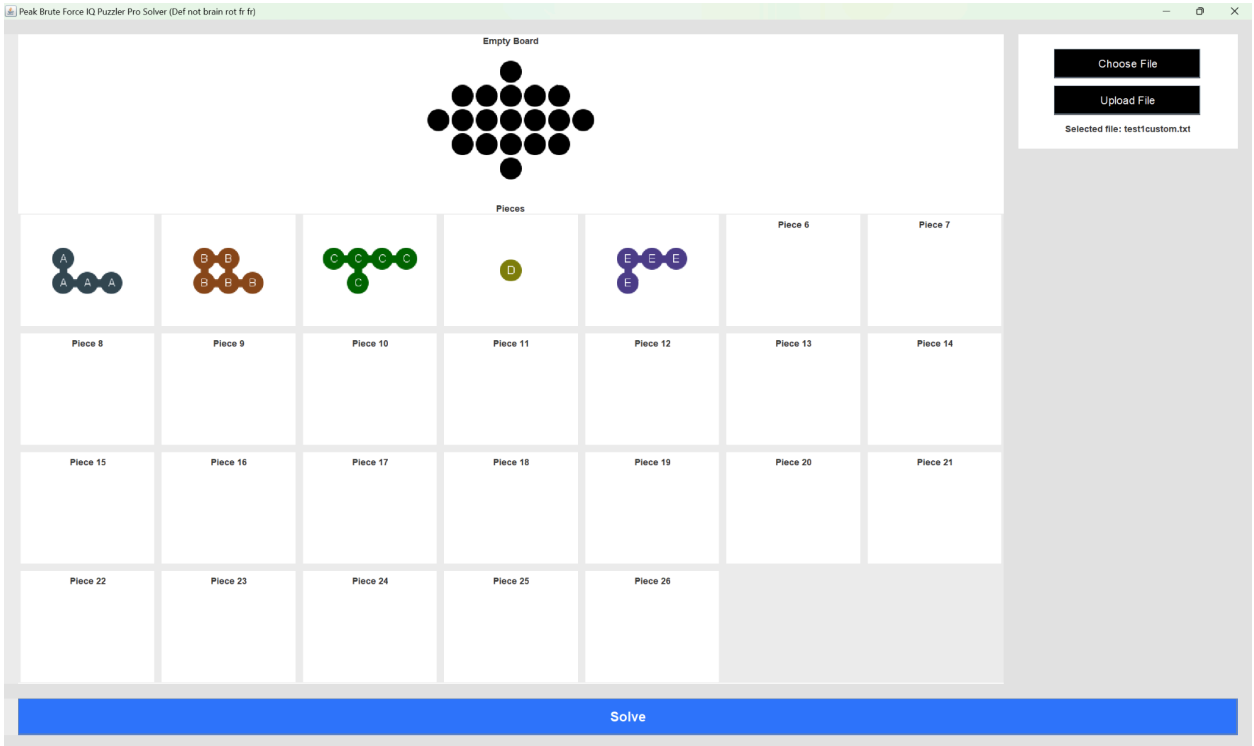

Tucil1_13523100 - solution4custom.txt
44 Solution:
45 B I C
46 B F I I I C C
47 B F G G G G H H H
48 J F A A A G H D D
49 J F A E E D D
50 J E D
51
52 Execution Time: 1460405 ms
53 Number of cases checked: 149527877071

Test Case 8 (custom)

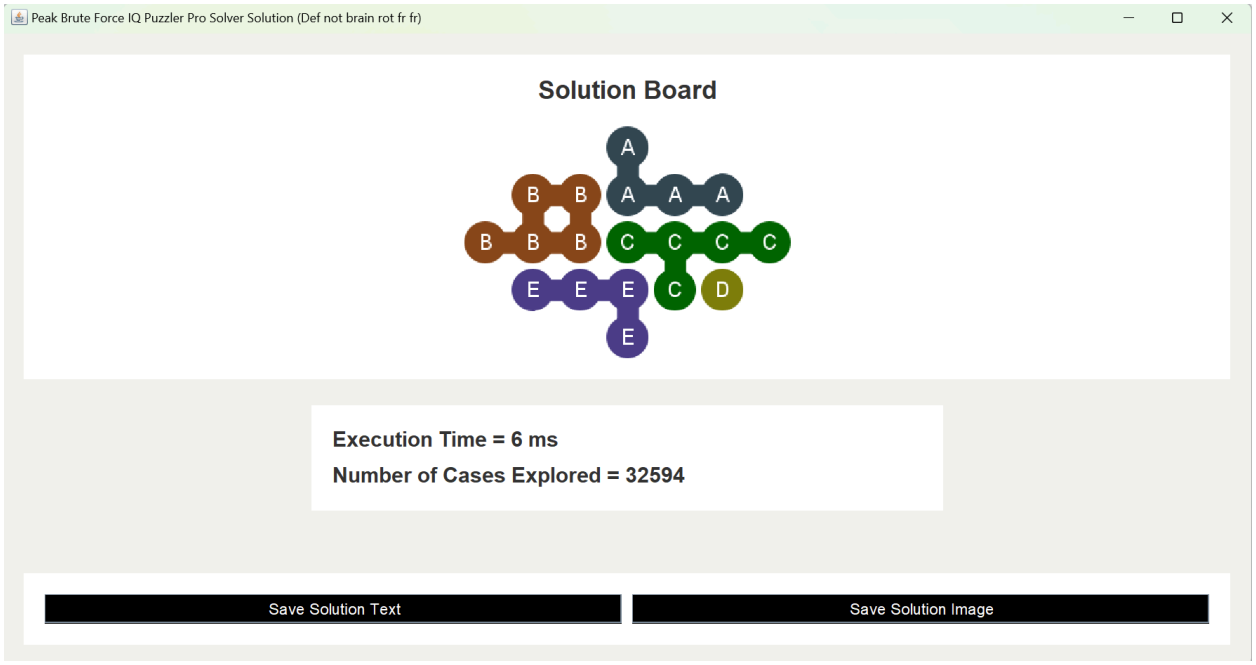
Input

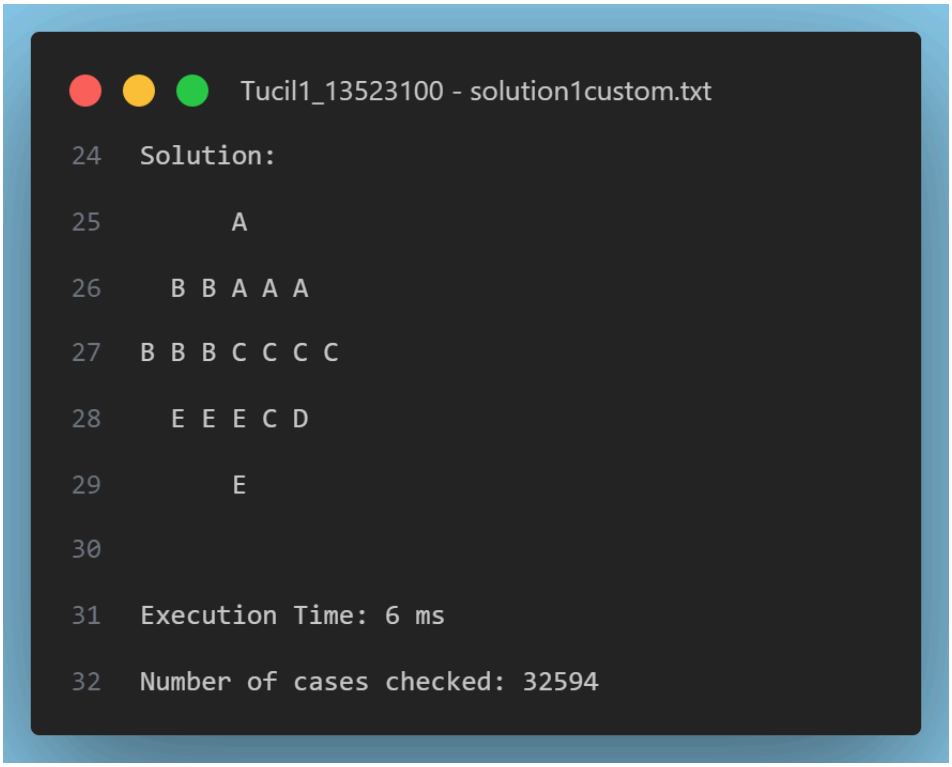
test > test1custom.txt

```
5 7 5
CUSTOM
...X...
.XXXXX.
XXXXXXX
.XXXXX.
...X...
A
AAA
BB
BBB
CCCC
 C
D
EEE
E
```



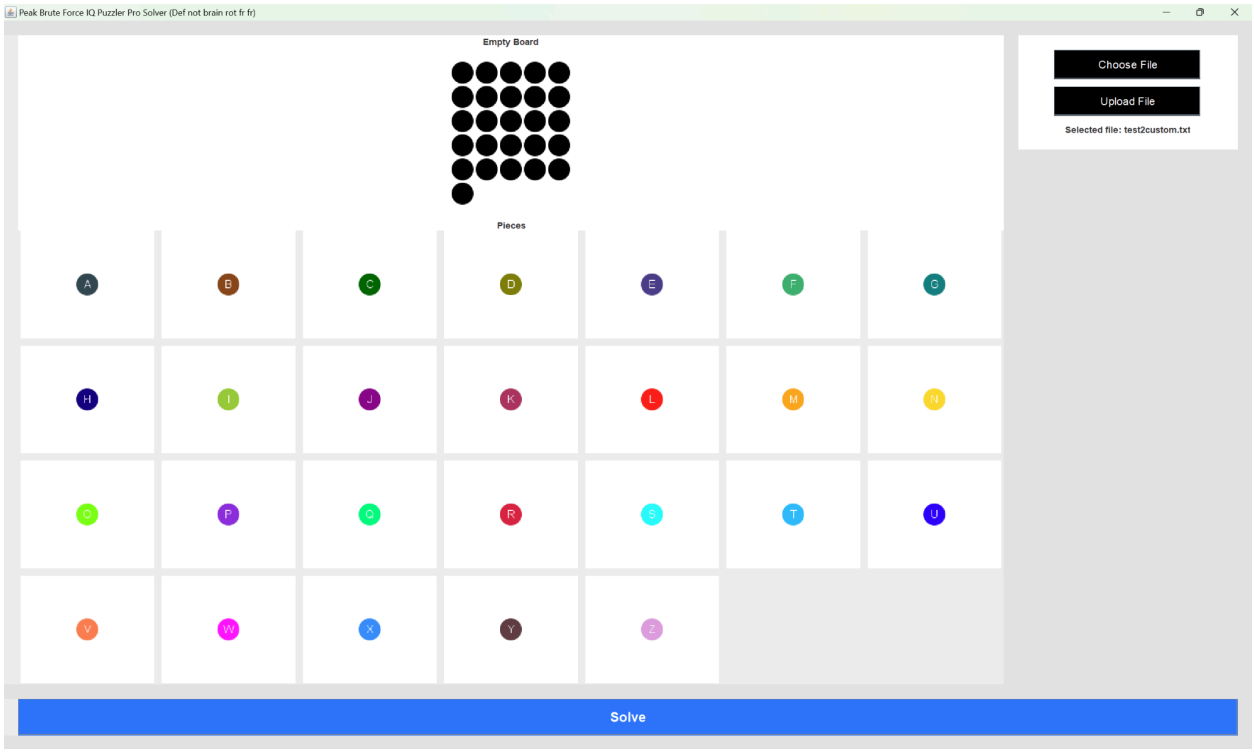
Output


Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
Execution Time = 6 ms
Number of Cases Explored = 32594
Save Solution Text
Save Solution Image


Tucil1_13523100 - solution1custom.txt
24 Solution:
25 A
26 B B A A A
27 B B B C C C C
28 E E E C D
29 E
30
31 Execution Time: 6 ms
32 Number of cases checked: 32594

Test Case 9 (custom)

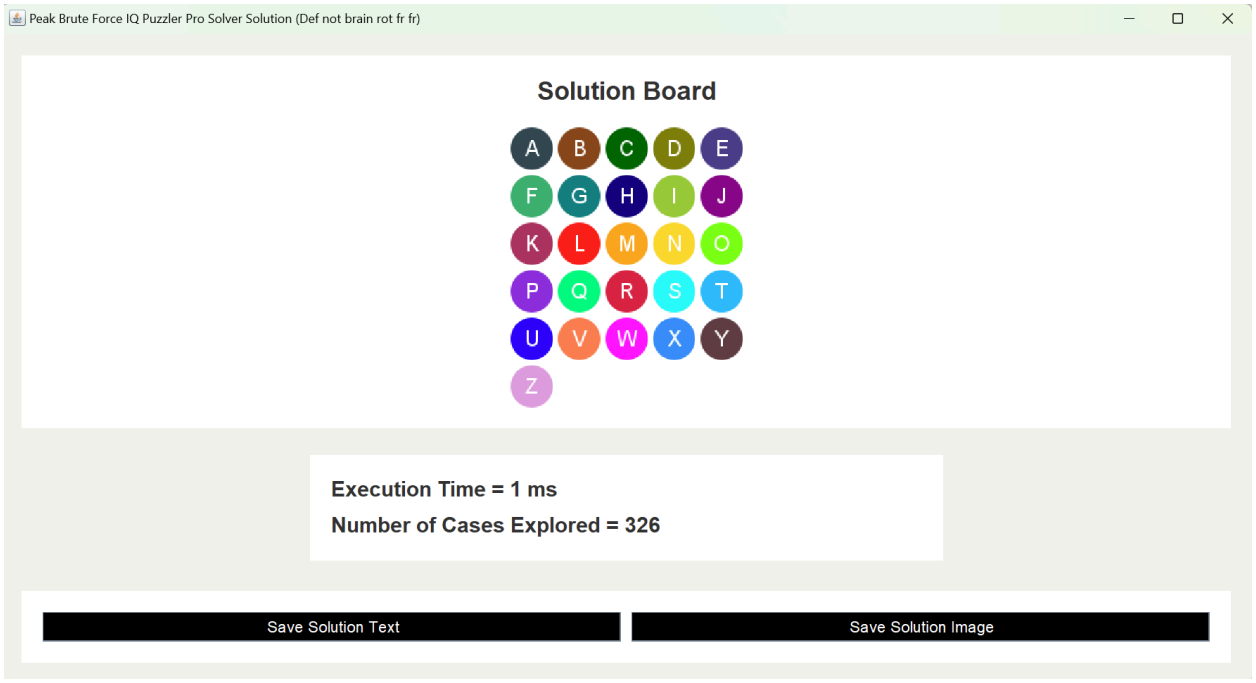
Input

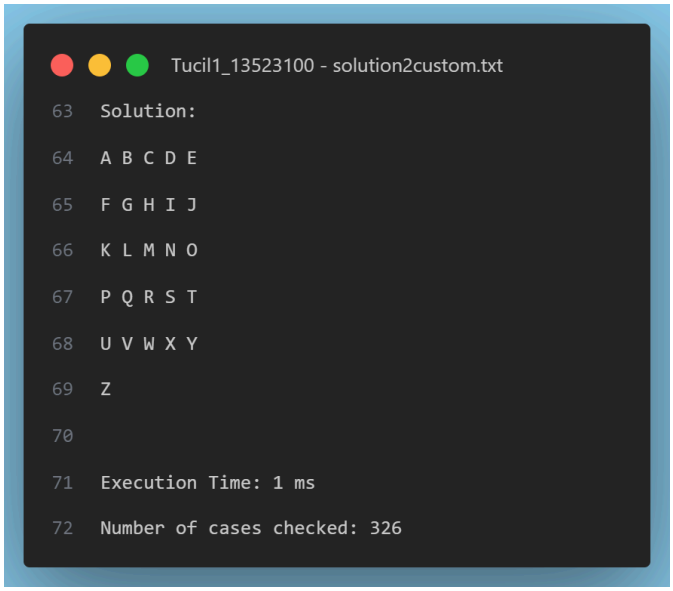


test > test2custom.txt

```
6 5 26
CUSTOM
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
X....
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
```

Output


Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
A B C D E
F G H I J
K L M N O
P Q R S T
U V W X Y
Z
Execution Time = 1 ms
Number of Cases Explored = 326
Save Solution Text
Save Solution Image

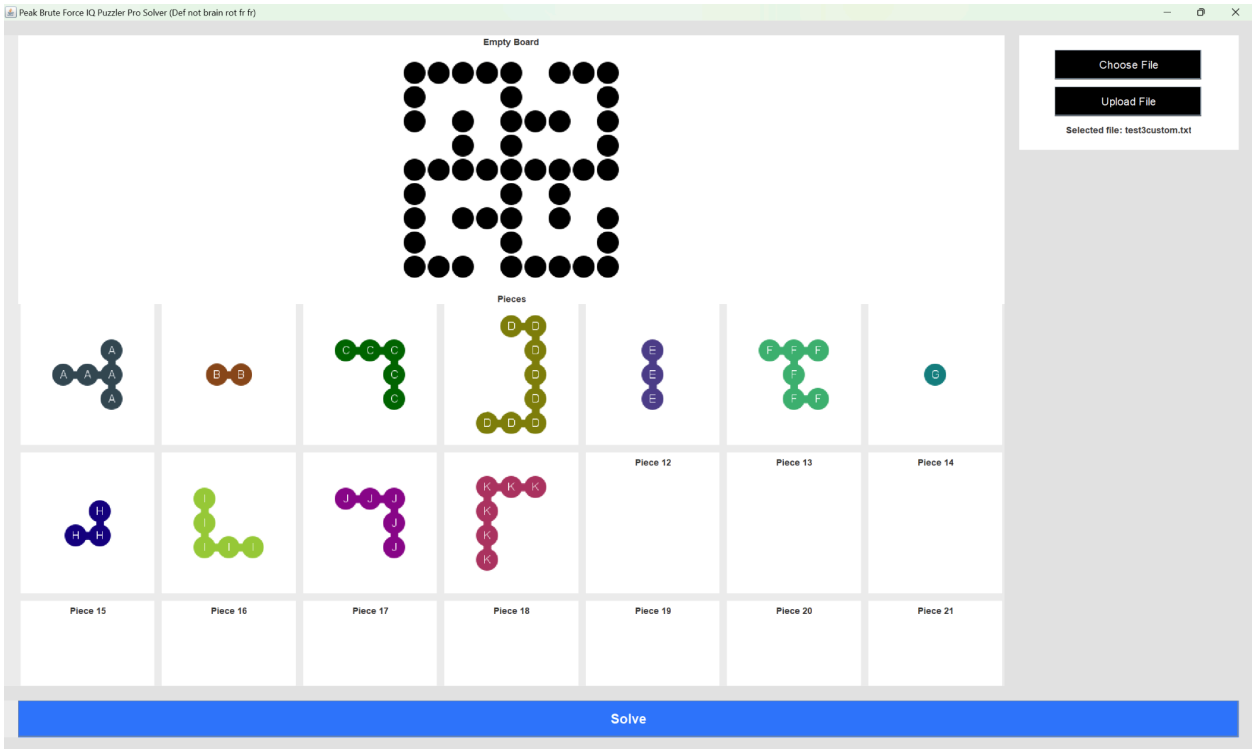

Tucil1_13523100 - solution2custom.txt
63 Solution:
64 A B C D E
65 F G H I J
66 K L M N O
67 P Q R S T
68 U V W X Y
69 Z
70
71 Execution Time: 1 ms
72 Number of cases checked: 326

Test Case 10 (custom)

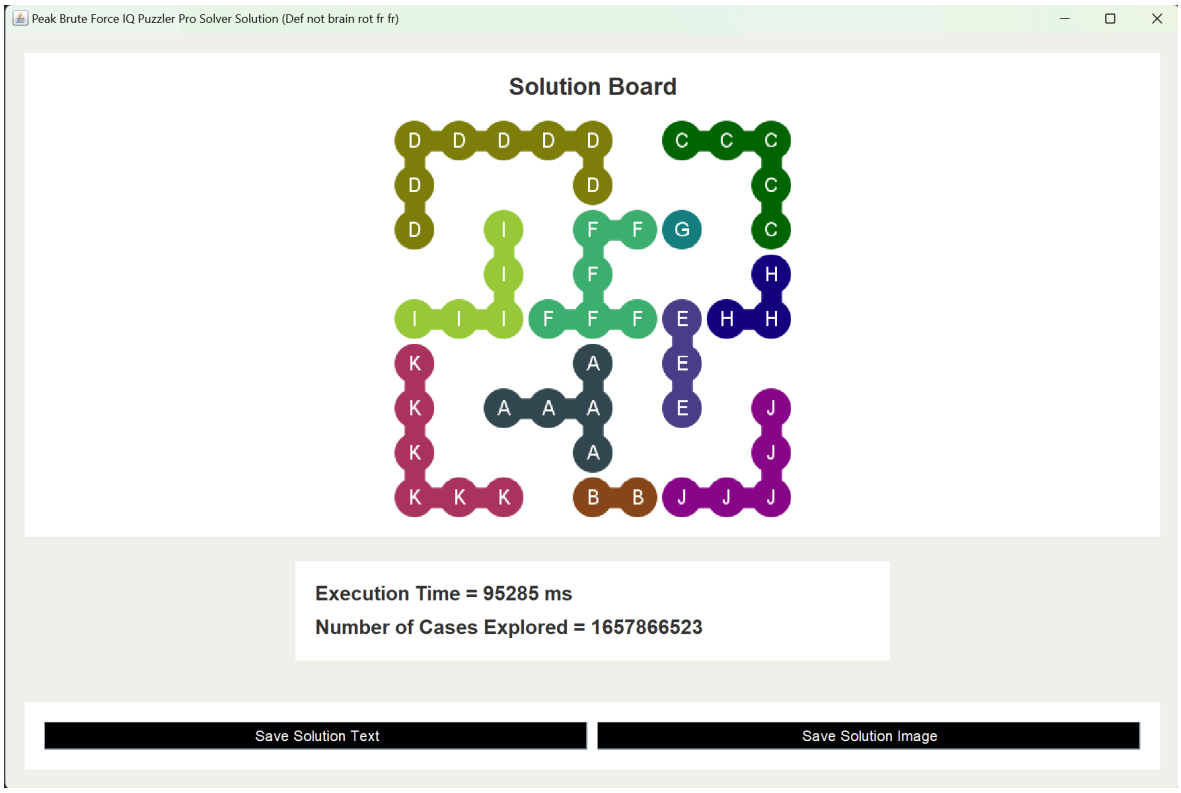
Input

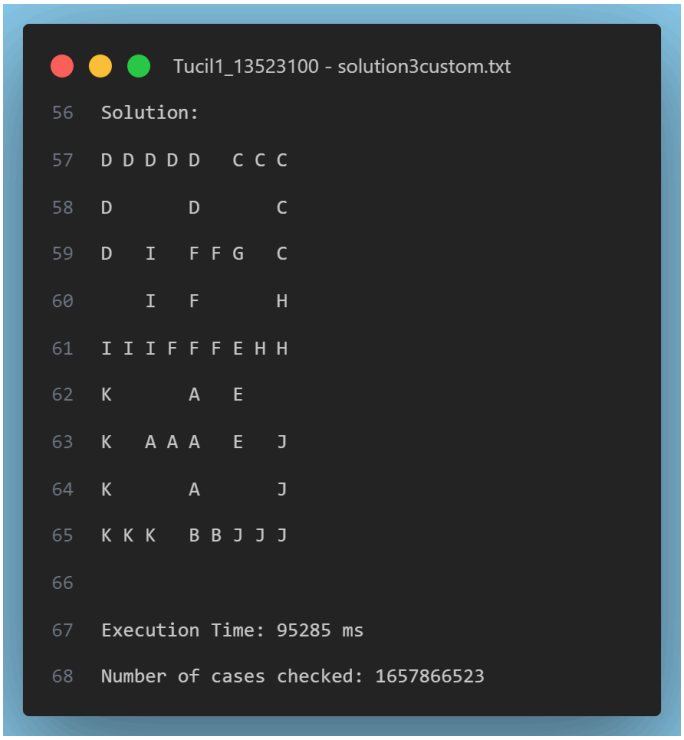
test > test3custom.txt

```
9 9 11
CUSTOM
XXXXX.XXX
X...X...X
X.X.XXX.X
..X.X...X
XXXXXXXXX
X...X.X..
X.XXX.X.X
X...X...X
XXX.XXXXX
 A
AAA
 A
BB
CCC
 C
 C
DD
 D
 D
 D
DDD
E
E
E
FFF
F
FF
G
H
HH
I
I
III
JJJ
 J
 J
KKK
K
K
K
```


Peak Brute Force IQ Puzzler Pro Solver (Def not brain rot fr fr)
Empty Board
Choose File
Upload File
Selected file: test3custom.txt
Pieces
Piece 12
Piece 13
Piece 14
Piece 15
Piece 16
Piece 17
Piece 18
Piece 19
Piece 20
Piece 21
Solve

Output


Peak Brute Force IQ Puzzler Pro Solver Solution (Def not brain rot fr fr)
Solution Board
Execution Time = 95285 ms
Number of Cases Explored = 1657866523
Save Solution Text
Save Solution Image


Tucil1_13523100 - solution3custom.txt
56 Solution:
57 D D D D D C C C
58 D D C
59 D I F F G C
60 I F H
61 I I I F F F E H H
62 K A E
63 K A A A E J
64 K A J
65 K K K B B J J J
66
67 Execution Time: 95285 ms
68 Number of cases checked: 1657866523

4. Lampiran

Link Repositori: https://github.com/Staryo40/Tucil1_13523100

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 *Graphical User Interface* (GUI)	✓	
6	Program dapat menyimpan solusi dalam bentuk file gambar	✓	
7	Program dapat menyelesaikan kasus konfigurasi *custom*	✓	
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		✓
9	Program dibuat oleh saya sendiri	✓	