Laporan Tugas Kecil 1 IF-2211 Strategi Algoritma

Semester II Tahun Akademik 2024/2025

Penyelesaian Permainan IQ Puzzler Pro Dengan Menggunakan Algoritma Brute-Force

Disusun Oleh:

Nayaka Ghana Subrata - 13523090

K-02

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2025

DAFTAR ISI

DAFTAR GAMBAR

BAB I: PENDAHULUAN

1.1 Deskripsi Tugas

IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games. Tujuan dari permainan ini adalah pemain harus dapat mengisi seluruh papan dengan *piece* (blok puzzle) yang telah tersedia.

Gambar 1. Papan Permainan IQ Puzzler Pro
(Sumber: https://www.smartgamesusa.com/sites/default/files/SmartGames-IQ-puzzlerPro-Step3_0.jpg)

Komponen penting dari permainan IQ Puzzler Pro terdiri dari:

1. **Board (Papan)** – *Board* merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area papan menggunakan blok-blok yang telah disediakan.

2. **Blok/Piece** – Blok adalah komponen yang digunakan pemain untuk mengisi papan kosong hingga terisi penuh. Setiap blok memiliki bentuk yang unik dan semua blok harus digunakan untuk menyelesaikan puzzle.

Permainan dimulai dengan **papan yang kosong**. Pemain dapat meletakkan blok puzzle sedemikian sehingga **tidak ada blok yang bertumpang tindih** (kecuali dalam kasus 3D). Setiap blok puzzle dapat **dirotasikan** maupun **dicerminkan**. Puzzle dinyatakan **selesai** jika dan hanya jika papan **terisi penuh** dan **seluruh blok puzzle berhasil diletakkan**.

Tujuan dari tugas kecil ini adalah untuk menemukan **cukup satu solusi** dari permainan **IQ Puzzler Pro** dengan menggunakan **algoritma *Brute-Force***, atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari puzzle tanpa menggunakan pendekatan **heuristik**.

BAB II: PENYELESAIAN DAN ALGORITMA

2.1 Algoritma Brute Force

Sebelum melakukan proses pencarian solusi dari permainan, program pertama-tama akan melakukan parsing terhadap bentuk *piece*, jika bentuk *piece*-nya sudah persegi, maka tidak akan dilakukan proses apa-apa, tetapi jika *piece*-nya berbentuk seperti L, U, dan bentuk unik yang lain, maka program akan melakukan penyesuaian ukuran agar representasi *piece* berbentuk persegi. Contoh jika ada bentuk *piece* unik dengan dimensi NxM, dengan M > N, maka *piece* akan disesuaikan agar representasinya berbentuk persegi dengan ukuran MxM, dengan *piece* ditandai dengan simbol "#" dan area kosong ditandai dengan ".". Untuk board, akan diinisialisasi dengan nilai -1 sebesar NxM.

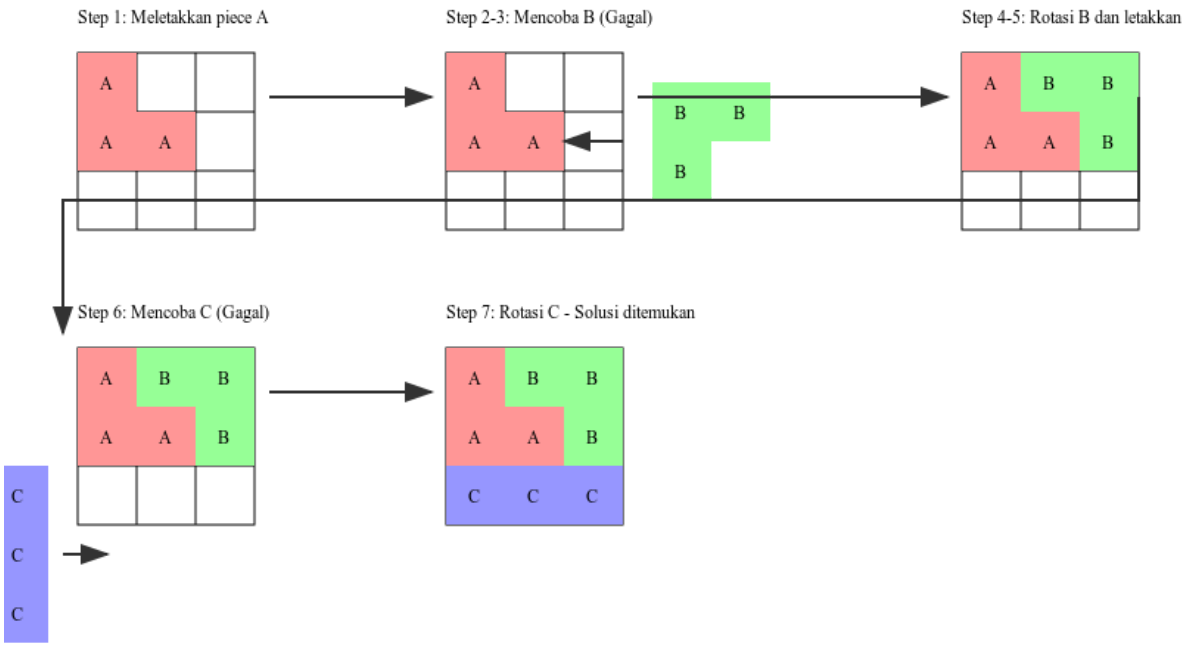
Setelah melakukan inisialisasi *piece* dan *board*, selanjutnya program akan masuk ke tahap pencarian solusi. Cara untuk penyelesaian permainan ini mirip seperti permainan sudoku, dengan langkah sebagai berikut:

1. Program akan inisialisasi larik dengan tipe boolean untuk tiap *piece*, tujuannya adalah untuk mengecek apakah *piece* sedang terpakai atau tidak.
2. Program memulai *brute-force searching* untuk penemuan solusi, dengan basis jika solusi sudah ditemukan atau semua sel dalam *board* sudah terisi, maka pencarian akan dihentikan.
3. Pada saat melakukan pencarian, program akan mengecek posisi *piece* yang dicoba saat ini di *board*.
4. Jika di *board* ada posisi yang kosong, maka program akan mencoba untuk memasukkan potongan yang belum dicoba, dan jika tidak bisa menempatkan bentuk awal dari *piece*, program akan mencoba untuk melakukan refleksi terhadap *piece* dan juga rotasi dengan kelipatan sembilan puluh derajat (0-270).
5. Saat program mencoba untuk menempatkan *piece*, program akan mengecek apakah *piece* bisa dimasukkan atau tidak, jika bisa maka program akan menuju ke sel pada *board* berikutnya dan menandai bahwa sel tersebut terisi, jika tidak bisa diisi dan sudah dicoba semua permutasi yang mungkin dari *piece*, maka potongan akan dihapus dari larik dengan tipe boolean tadi dan juga menghapus tanda terisi pada *board*. Setelahnya, program akan mencoba untuk melakukan proses *backtracking*.
6. Jika terjadi proses *backtracking*, maka yang dilakukan oleh program adalah melakukan *reverse-step* dan mencoba-coba tiap *piece* dengan semua permutasinya,

sehingga muncul permutasi baru pada *board*. Langkah ini dilakukan jika *piece* tidak bisa ditempatkan pada *board*.

7. Jika program mencapai kasus terburuk (*worst case*) dengan mencoba semua permutasi pada *piece* dan *board*, maka program akan memberikan kesimpulan bahwa tidak ada solusi yang tersedia, selain itu program akan menampilkan solusi yang didapatkan.

2.2 Visualisasi Sederhana



Gambar 2. Visualisasi sederhana dari algoritma *brute-force* yang digunakan pada *board* berukuran 3x3 (Sumber: arsip penulis)

2.3 Pseudocode

```
procedure findSolution(input pos: integer, input used: array of boolean, output findSolution: boolean)
        if foundSolution then
           → { keluar dari prosedur }

        if filledCells = totalCells then
           foundSolution ← true
           → { keluar dari prosedur }

        { Hitung posisi i,j dari pos }
        i ← (pos - 1) div block.m + 1
```

```
j ← (pos - 1) mod block.m + 1

if i > block.n then
   → { keluar dari prosedur }

{ Jika sel sudah terisi atau tidak valid dalam custom config }
if block.get(i,j) ≠ -1 or
  (block.configType = "CUSTOM" and not block.customConfig[i-1,j-1]) then
   findSolution(pos + 1, used)
   → { keluar dari prosedur }

{ Coba setiap piece yang belum digunakan }
cur_piece traversal [1..block.pieceCount]
   if not foundSolution then
      if not used[cur_piece] then
         piece ← block.pieces[cur_piece]
         used[cur_piece] ← true

         { Coba setiap kemungkinan rotasi }
         rot traversal [0..3]
            if not foundSolution then
               { Coba setiap kemungkinan pencerminan }
               flip traversal [0..1]
                  if not foundSolution then
                     if block.placePiece(piece, i, j) then
                        pieceCells ← countPieceCells(piece)
                        filledCells ← filledCells + pieceCells

                        findSolution(pos + 1, used)

                        if not foundSolution then
                           block.removePiece(piece, i, j)
                           filledCells ← filledCells - pieceCells

                     if flip ≠ 1 then
                        piece ← piece.reflect()

               if rot ≠ 3 then
                  piece ← piece.rotate()

         if not foundSolution then
            used[cur_piece] ← false
```

BAB III: STRUKTUR REPOSITORY DAN SOURCE CODE

3.1 Struktur Repository

Berikut adalah struktur *repository* dari program ini. Jika ingin menjalankan program dengan Command Line Interface (CLI), *input file* harus berada di …/Tucil1_13523090/tucil1/test/input dan berupa .txt.

```
Tucil1_13523090/
├── .vscode/
│   └── settings.json
├── tucil1/
│   ├── bin/
│   │   ├── tucil1/src/
│   │   │   ├── Board$PieceOperation.class
│   │   │   ├── Board.class
│   │   │   ├── InputReader.class
│   │   │   ├── Main.class
│   │   │   ├── Pair.class
│   │   │   ├── Piece.class
│   │   │   └── Solve.class
│   │   ├── gradle-wrapper.jar
│   │   └── tucil1.jar
│   ├── doc/
│   │   └── Tucil1_K2_13523090_Nayaka Ghana Subrata.pdf
│   ├── etc/
│   │   ├── cli.gif
│   │   ├── guiload.gif
│   │   └── guimanual.gif
│   ├── gradle/
│   │   └── wrapper/
│   │       ├── gradle-wrapper.jar
│   │       └── gradle-wrapper.properties
│   ├── src/
│   │   ├── Board.java
│   │   ├── InputReader.java
│   │   ├── Main.java
│   │   ├── Pair.java
│   │   ├── Piece.java
│   │   ├── PuzzleSolverGUI.java
```

```
│   │   └── Solve.java
│   ├── test/
│   │   ├── input/
│   │   │   ├── 1.txt
│   │   │   ├── 2.txt
│   │   │   ├── 3.txt
│   │   │   ├── 4.txt
│   │   │   ├── 5.txt
│   │   │   ├── 6.txt
│   │   │   ├── 7.txt
│   │   │   └── 8.txt
│   │   └── solutions/
│   │       ├── solution1.png
│   │       ├── solution1.txt
│   │       ├── solution2.png
│   │       ├──solution 2.txt
│   │       ├── solution3.png
│   │       ├── solution 3.txt
│   │       ├── solution4.png
│   │       ├── solution4.txt
│   │       ├── solution5.png
│   │       ├── solution5.txt
│   │       ├── solution6.png
│   │       ├── solution6.txt
│   │       ├── solution7.png
│   │       ├── solution 7.txt
│   │       ├── solution8.png
│   │       └── solution8.txt
│   ├── Makefile
│   ├── build.gradle
│   ├── gradlew
│   ├── gradlew.bat
│   ├── manifest.txt
│   └── settings.gradle
└── README.md
```

Berikut adalah penjelasan dari tiap folder:

1. Folder **src** berisi *source code* program.
2. Folder **bin** berisi *executable file*
3. Folder **test** berisi input beserta solusi dari data uji yang digunakan.

4. Folder **doc** berisi laporan.
5. Folder **gradle** berisi wrapper untuk Graphical User Interface (GUI)
6. Folder **etc** berisi file-file lain.

3.2 Source Code

Berikut adalah *source code* dari program ini yang berupa tangkapan layar, untuk lebih lengkapnya, *source code* dapat diakses pada [tautan ini](). (Beberapa dari gambar yang ditampilkan hanyalah *overview* atau fungsi utama).

3.2.1 Board.java

```java
// package
package tucil1.src;

// module
import java.awt.Color;
import java.io.File;
import java.util.Random;



public class Board {
    int n, m, pieceCount;
    int[][] arr;
    boolean[][] customConfig;
    String[] color;
    Color[] actualColors;
    Piece[] pieces;
    boolean foundSolution = false;
    public final String OUTPUT_DIRECTORY = System.getProperty("user.dir") + File.separator + "test" + File.separator + "output" + File.separator;
    public String configType;

    public Board(Piece[] pieces, int n, int m, int pieceCount, String configType, boolean[][] customConfig) {
      this.n = n;
      this.m = m;
      this.pieceCount = pieceCount;
      this.configType = configType;
      this.pieces = new Piece[pieceCount+2];
      this.customConfig = customConfig;

      this.arr = new int[n+2][m+2];
      for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
          arr[i][j] = -1;
        }
      }

      for (int i = 1; i <= pieceCount; i++) {
        this.pieces[i] = pieces[i];
      }

      this.color = new String[30];
      this.actualColors = new Color[30];

      Random random = new Random();
      for (int i = 1; i <= 26; i++) {
          float hue = (float) i / 26 + random.nextFloat() * 0.038f;
          float saturation = 0.7f + random.nextFloat() * 0.3f;
          float brightness = 0.8f + random.nextFloat() * 0.2f;

          Color color = Color.getHSBColor(hue, saturation, brightness);
          this.actualColors[i] = color;
          this.color[i] = "\u001B[38;2;" + color.getRed() + ";" + color.getGreen() + ";" + color.getBlue() + "m";
      }

      File outputDir = new File(OUTPUT_DIRECTORY);
      if (!outputDir.exists()) {
        outputDir.mkdirs();
      }
    }
```

Gambar 3. Konstuktor papan permainan

(Sumber: arsip penulis)

```java
    public int get(int x, int y) {
      return arr[x][y];
    }

    public void set(int x, int y, int value) {
      arr[x][y] = value;
    }


    public boolean isValidPosition(int x, int y) {
      if (configType.equals("CUSTOM")) {
        return x >= 1 && x <= n && y >= 1 && y <= m &&
              arr[x][y] == -1 &&
              customConfig[x-1][y-1];
      } else {
        return x >= 1 && x <= n && y >= 1 && y <= m && arr[x][y] == -1;
      }
    }

    private boolean iteratePiece(Piece piece, int x, int y, PieceOperation action) {
      for (int i = 1; i <= piece.n; i++) {
          for (int j = 1; j <= piece.m; j++) {
              if (piece.get(i, j) && !action.apply(i, j)) {
                  return false;
              }
          }
      }
      return true;
  }

    private interface PieceOperation {
        boolean apply(int i, int j);
    }

    private boolean canPlacePiece(Piece piece, int x, int y) {
      return iteratePiece(piece, x, y, (i, j) -> isValidPosition(x + i - 1, y + j - 1));
  }

    public boolean placePiece(Piece piece, int x, int y) {
        if (!canPlacePiece(piece, x, y)) {
            return false;
        }

        iteratePiece(piece, x, y, (i, j) -> {
            set(x + i - 1, y + j - 1, piece.id);
            return true;
        });

        return true;
    }

    public void removePiece(Piece piece, int x, int y) {
      iteratePiece(piece, x, y, (i, j) -> {
          set(x + i - 1, y + j - 1, -1);
          return true;
      });
  }
}
```

Gambar 4. Method dari papan permainan

(Sumber: arsip penulis)

3.2.2 InputReader.java

```
private Piece processSinglePiece(List<String> piece) {
    int maxLength = 0;
    for (String line : piece) {
        maxLength = Math.max(maxLength, line.length());
    }

    int height = piece.size();
    int width = maxLength;
    boolean[][] arr = new boolean[height][width];
    int rootn = -1, rootm = -1;

    for (int i = 0; i < height; i++) {
        String line = piece.get(i);
        char[] chars = line.toCharArray();

        for (int j = 0; j < chars.length; j++) {
            if (isCapitalLetter(chars[j])) {
                arr[i][j] = true;
                if (rootn == -1) {
                    rootn = i;
                    rootm = j;
                }
            }
        }
    }

    if (rootn == -1 || rootm == -1) {
        throw new IllegalArgumentException("Invalid Piece: Found empty piece");
    }

    // Remove empty space
    boolean[][] trimmedArr = trimArray(arr);

    isConnected(trimmedArr, trimmedArr.length, trimmedArr[0].length,
                findRoot(trimmedArr).getKey(), findRoot(trimmedArr).getValue());

    return new Piece(trimmedArr.length, trimmedArr[0].length, trimmedArr);
}

private Pair<Integer, Integer> findRoot(boolean[][] arr) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr[0].length; j++) {
            if (arr[i][j]) {
                return new Pair<>(i, j);
            }
        }
    }
    throw new IllegalArgumentException("Invalid Piece: No root found");
}

private boolean[][] trimArray(boolean[][] original) {
    int minRow = original.length, maxRow = -1;
    int minCol = original[0].length, maxCol = -1;

    // cari edge case
    for (int i = 0; i < original.length; i++) {
        for (int j = 0; j < original[0].length; j++) {
            if (original[i][j]) {
                minRow = Math.min(minRow, i);
                maxRow = Math.max(maxRow, i);
                minCol = Math.min(minCol, j);
                maxCol = Math.max(maxCol, j);
            }
        }
    }

    // Piece kosong
    if (maxRow == -1) {
        throw new IllegalArgumentException("Invalid Piece: Empty piece");
    }

    int newHeight = maxRow - minRow + 1;
    int newWidth = maxCol - minCol + 1;
    boolean[][] trimmed = new boolean[newHeight][newWidth];

    for (int i = 0; i < newHeight; i++) {
        for (int j = 0; j < newWidth; j++) {
            trimmed[i][j] = original[i + minRow][j + minCol];
        }
    }

    return trimmed;
}

private void isConnected(boolean[][] arr, int n, int m, int rootn, int rootm) {
    boolean[][] visited = new boolean[n][m];
    Queue<Pair<Integer, Integer>> q = new LinkedList<>();
    q.add(new Pair<>(rootn, rootm));
    visited[rootn][rootm] = true;

    // Kemungkinan letak piece
    int[][] directions = {
        {-1, 0},  // atas
        {1, 0},   // bawah
        {0, -1},  // kiri
        {0, 1},   // kanan
        {-1, -1}, // atas kiri
        {-1, 1},  // atas kanan
        {1, -1},  // bawah kiri
        {1, 1}    // bawah kanan
    };

    while (!q.isEmpty()) {
        Pair<Integer, Integer> p = q.poll();
        int x = p.getKey();
        int y = p.getValue();

        for (int[] dir : directions) {
            int newX = x + dir[0];
            int newY = y + dir[1];

            if (newX >= 0 && newX < n && newY >= 0 && newY < m &&
                !visited[newX][newY] && arr[newX][newY]) {
                q.add(new Pair<>(newX, newY));
                visited[newX][newY] = true;
            }
        }
    }

    // cek konektivitas
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (arr[i][j] && !visited[i][j]) {
                throw new IllegalArgumentException("Invalid Piece: Piece is not connected");
            }
        }
    }
}
```

Gambar 5. Input Parser

(Sumber: arsip penulis)

3.2.3 Main.java

```java
// package
package tucil1.src;

// module
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Main {
    private static final String INPUT_FOLDER = System.getProperty("user.dir") + File.separator + "test" + File.separator + "input";
    private static Scanner scanner = new Scanner(System.in);


    private static void displayMainMenu() {
        System.out.println("\n=== Main Menu ===");
        System.out.println("1. Solve puzzle");
        System.out.println("2. Exit");
        System.out.print("Enter your choice (1-2): ");
    }

    private static void solvePuzzle() {
        System.out.print("Enter filename (without .txt extension): ");
        String filename = scanner.nextLine();
        String fullPath = INPUT_FOLDER + File.separator + filename + ".txt";

        try {
            File file = new File(fullPath);
            if (!file.exists()) {
                throw new FileNotFoundException("File '" + filename + ".txt' not found in the input directory.");
            }

            InputReader reader = new InputReader(fullPath);
            reader.readInput();

            Piece[] pieces = reader.processPieces();
            System.out.println("Processed Pieces: ");
            for (int i = 1; i <= reader.getPieceCount(); i++) {
                System.out.println("Piece " + pieces[i].id + " : ");
                pieces[i].printPiece();
            }

            if (reader.getConfigType().equals("CUSTOM")) {
                if (!validateCustomConfig(pieces, reader)) {
                    throw new IllegalArgumentException(
                        "Invalid Configuration: Total area of pieces does not match with the custom configuration area"
                        );
                }
            } else if (!validatePieceArea(pieces, reader)) {
                throw new IllegalArgumentException(
                    "Invalid Piece: Total area of pieces does not match with the default configuration area"
                    );
                }


                Board board = new Board(pieces, reader.getBoardHeight(), reader.getBoardWidth(),
                reader.getPieceCount(), reader.getConfigType(), reader.getCustomConfig());
                Solve solver = new Solve(board);
                solver.solve();

        } catch (FileNotFoundException e) {
            System.err.println("Error: " + e.getMessage());
            System.err.println("Please make sure the file exists in: " + INPUT_FOLDER);
        } catch (Exception e) {
            System.err.println("Error occurred: " + e.getMessage());
        }
    }

    private static boolean validateCustomConfig(Piece[] pieces, InputReader reader) {
        int totalPieceArea = calculateTotalArea(pieces, reader.getPieceCount());
        int customConfigArea = calculateCustomConfigArea(reader.getCustomConfig());

        System.out.println("Total piece area: " + totalPieceArea);
        System.out.println("Custom configuration area: " + customConfigArea);

        return totalPieceArea == customConfigArea;
    }

    private static int calculateCustomConfigArea(boolean[][] customConfig) {
        int area = 0;
        for (int i = 0; i < customConfig.length; i++) {
            for (int j = 0; j < customConfig[0].length; j++) {
                if (customConfig[i][j]) {
                    area++;
                }
            }
        }
        return area;
    }

    private static boolean validatePieceArea(Piece[] pieces, InputReader reader) {
        int totalArea = calculateTotalArea(pieces, reader.getPieceCount());
        int boardArea = reader.getBoardHeight() * reader.getBoardWidth();

        System.out.println("Total piece area: " + totalArea);
        System.out.println("Board area: " + boardArea);

        return totalArea == boardArea;
    }

    private static int calculateTotalArea(Piece[] pieces, int pieceCount) {
        int totalArea = 0;
        for (int i = 1; i <= pieceCount; i++) {
            for (int j = 1; j <= pieces[i].m; j++) {
                for (int k = 1; k <= pieces[i].n; k++) {
                    if (pieces[i].get(j, k)) {
                        totalArea++;
                    }
                }
            }
        }
        return totalArea;
    }

    public static void main(String[] args) {
        while (true) {
            displayMainMenu();
            String choice = scanner.nextLine().trim();

            switch (choice) {
                case "1":
                    Solve.reset();
                    Piece.count = 1;
                    solvePuzzle();
                    break;
                case "2":
                    System.out.println("Exiting program...");
                    scanner.close();
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid input. Please enter 1 or 2.");
            }
        }
    }
}
```

Gambar 6. *Main driver* dari program

(Sumber: arsip penulis)

3.2.4 Pair.java

```java
// package
package tucil1.src;



public class Pair<K, V> {
    private final K key;
    private final V value;

    public Pair(K key, V value) {
        this.key = key;
        this.value = value;
    }

    public K getKey() {
        return key;
    }

    public V getValue() {
        return value;
    }
}
```

Gambar 7. Pair Class

(Sumber: arsip penulis)

3.2.5 Piece.java

```java
// package
package tucil1.src;



public class Piece {
  int n, m;
  boolean[][] arr;
  int id;
  static int count = 1;

  public Piece(int n, int m, boolean arr[][]) {
    this.n = n;
    this.m = m;
    this.arr = new boolean[n+2][m+2];
    this.id = count++;

    for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= m; j++) {
        this.arr[i][j] = arr[i-1][j-1];
      }
    }
  }

  public Piece(int n, int m, boolean arr[][], int id) {
    this.n = n;
    this.m = m;
    this.arr = new boolean[n+2][m+2];
    this.id = id;

    for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= m; j++) {
        this.arr[i][j] = arr[i-1][j-1];
      }
    }
  }

  public void set(int x, int y, boolean value) {
    this.arr[x][y] = value;
  }

  public boolean get(int x, int y) {
    return this.arr[x][y];
  }

  public boolean isValidPosition(int x, int y) {
    return x >= 1 && x <= this.n && y >= 1 && y <= this.m;
  }

  public Piece rotate() {
    boolean[][] newArr = new boolean[this.m][this.n];
    for (int i = 0; i < this.m; i++) {
      for (int j = 0; j < this.n; j++) {
        newArr[i][j] = this.arr[this.n - j][i + 1];
      }
    }
    return new Piece(this.m, this.n, newArr, this.id);
  }

  public Piece reflect() {
    boolean[][] newArr = new boolean[this.n][this.m];
    for (int i = 0; i < this.n; i++) {
      for (int j = 0; j < this.m; j++) {
        newArr[i][j] = this.arr[i + 1][this.m - j];
      }
    }
    return new Piece(this.n, this.m, newArr, this.id);
  }

  public void printPiece() {
    for (int i = 1; i <= this.n; i++) {
      for (int j = 1; j <= this.m; j++) {
        System.out.print(this.arr[i][j] ? "#" : ".");
      }
      System.out.println();
    }
  }
}
```

Gambar 8. Method dan Class untuk p*iece*

(Sumber: arsip penulis)

3.2.6 Solve.java

```
private int calculateTotalCells() {
    int count = 0;
    if (block.configType.equals("CUSTOM")) {
        for (int i = 0; i < block.n; i++) {
            for (int j = 0; j < block.m; j++) {
                if (block.customConfig[i][j]) count++;
            }
        }
    } else {
        count = block.n * block.m;
    }
    return count;
}

public static void reset() {
    isSolving = false;
}

public void printBlock() {
    for (int i = 1; i <= block.n; i++) {
        for (int j = 1; j <= block.m; j++) {
            if (block.get(i, j) == -1) {
                System.out.print("  ");
            } else {
                System.out.print(block.color[block.get(i, j)] + (char)('A' + block.get(i, j) - 1) + "\u001B[0m");
            }
        }
        System.out.println();
    }
}

private boolean isBoardComplete() {
    for (int i = 1; i <= block.n; i++) {
        for (int j = 1; j <= block.m; j++) {
            if (block.configType.equals("CUSTOM")) {
                if (block.customConfig[i-1][j-1] && block.get(i, j) == -1) {
                    return false;
                }
            } else {
                if (block.get(i, j) == -1) {
                    return false;
                }
            }
        }
    }
    return true;
}

private void saveToText(String filename) {
    try (FileWriter writer = new FileWriter(OUTPUT_DIRECTORY + filename)) {
        writer.write("Solution:\n");
        for (int i = 1; i <= block.n; i++) {
            for (int j = 1; j <= block.m; j++) {
                if (block.get(i, j) == -1) {
                    writer.write("  ");
                } else {
                    writer.write(String.valueOf((char)('A' + block.get(i, j) - 1)));
                }
            }
            writer.write("\n");
        }

        writer.write("\nTime taken: " + solutionTime + " ms\n");
        writer.write("Iterations: " + bruteForceCount + " times\n");

        System.out.println("Solution saved to " + OUTPUT_DIRECTORY + filename);
    } catch (IOException e) {
        System.out.println("Error saving text file: " + e.getMessage());
    }
}

public void saveToImage(String filename) {
    final int CELL_SIZE = 50;
    final int MARGIN = 20;
    final int TEXT_MARGIN = 40;

    int width = block.m * CELL_SIZE + 2 * MARGIN;
    int height = block.n * CELL_SIZE + 2 * MARGIN + TEXT_MARGIN;

    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics2D g2d = image.createGraphics();

    g2d.setColor(Color.WHITE);
    g2d.fillRect(0, 0, width, height);

    for (int i = 1; i <= block.n; i++) {
        for (int j = 1; j <= block.m; j++) {
            int x = MARGIN + (j-1) * CELL_SIZE;
            int y = MARGIN + (i-1) * CELL_SIZE;

            if (block.get(i, j) != -1) {
                g2d.setColor(block.actualColors[block.get(i, j)]);
                g2d.fillRect(x, y, CELL_SIZE, CELL_SIZE);

                g2d.setColor(Color.BLACK);
                g2d.drawString(String.valueOf((char)('A' + block.get(i, j) - 1)),
                               x + CELL_SIZE/2 - 5, y + CELL_SIZE/2 + 5);
            }

            g2d.setColor(Color.BLACK);
            g2d.setStroke(new BasicStroke(1));
            g2d.drawRect(x, y, CELL_SIZE, CELL_SIZE);
        }
    }

    g2d.setColor(Color.BLACK);
    g2d.setStroke(new BasicStroke(2));
    g2d.drawRect(MARGIN, MARGIN, block.m * CELL_SIZE, block.n * CELL_SIZE);

    g2d.drawString("Time taken: " + solutionTime + " ms",
                   MARGIN, height - TEXT_MARGIN/2);
    g2d.drawString("Iterations: " + bruteForceCount,
                   MARGIN + width/2, height - TEXT_MARGIN/2);

    g2d.dispose();

    try {
        File outputFile = new File(OUTPUT_DIRECTORY + filename);
        ImageIO.write(image, "png", outputFile);
        System.out.println("Solution saved to " + outputFile.getAbsolutePath());
    } catch (IOException e) {
        System.out.println("Error saving image: " + e.getMessage());
    }
}

private void handleSaveOptions() {
    Scanner scanner = new Scanner(System.in);
    while (true) {
        System.out.println("\nSave options:");
        System.out.println("1. Save as text file (.txt)");
        System.out.println("2. Save as image file (.png)");
        System.out.print("Enter your choice (1-2): ");

        String choice = scanner.nextLine().trim();

        switch (choice) {
            case "1":
                String txtFilename = "solution_" + block.n + "x" + block.m + ".txt";
                saveToText(txtFilename);
                return;
            case "2":
                String pngFilename = "solution_" + block.n + "x" + block.m + ".png";
                saveToImage(pngFilename);
                return;
            default:
                System.out.println("Invalid input. Please enter 1 or 2.");
        }
    }
}

private int countPieceCells(Piece piece) {
    int count = 0;
    for (int i = 1; i <= piece.n; i++) {
        for (int j = 1; j <= piece.m; j++) {
            if (piece.get(i, j)) count++;
        }
    }
    return count;
}
```

Gambar 9. Helper untuk *brute-force* dan juga *saving algorithm*

(Sumber: arsip penulis)

```java
public void findSolution(int pos, boolean[] used) {
    this.bruteForceCount++;

    if (foundSolution) return;

    if (filledCells == totalCells) {
        foundSolution = true;
        return;
    }

    int i = (pos - 1) / block.m + 1;
    int j = (pos - 1) % block.m + 1;

    if (i > block.n) return;

    if (block.get(i, j) != -1 ||
        (block.configType.equals("CUSTOM") && !block.customConfig[i-1][j-1])) {
        findSolution(pos + 1, used);
        return;
    }

    for (int cur_piece = 1; cur_piece <= block.pieceCount && !foundSolution; cur_piece++) {
        if (used[cur_piece]) continue;

        Piece piece = block.pieces[cur_piece];
        used[cur_piece] = true;

        for (int rot = 0; rot < 4 && !foundSolution; rot++) {
            for (int flip = 0; flip < 2 && !foundSolution; flip++) {
                if (block.placePiece(piece, i, j)) {
                    int pieceCells = countPieceCells(piece);
                    filledCells += pieceCells;

                    findSolution(pos + 1, used);

                    if (!foundSolution) {
                        block.removePiece(piece, i, j);
                        filledCells -= pieceCells;
                    }
                }
                if (flip != 1) piece = piece.reflect();
            }
            if (rot != 3) piece = piece.rotate();
        }

        if (!foundSolution) used[cur_piece] = false;
    }
}
```

Gambar 10. Algoritma utama dari proses *brute-force*

(Sumber: arsip penulis)

3.2.7 PuzzleSolverGUI.java

```java
// package
package tucil1.src;

// javafx
import javafx.application.Application;
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Text;
import javafx.util.Duration;
import javafx.animation.PauseTransition;
import javafx.scene.Node;

// modules
import java.io.File;
import java.util.ArrayList;
import java.util.List;



public class PuzzleSolverGUI extends Application {
    private int boardHeight;
    private int boardWidth;
    private int pieceCount;
    private String configType;
    private boolean[][] customConfig;
    private Piece[] pieces;
    private List<boolean[][]> pieceDesigns = new ArrayList<>();
    private int currentPieceIndex = 0;
    private Board board;
    private GridPane solutionGrid;
    private Label statusLabel;
    private long solveStartTime;
    private long solutionTimes;
    private int iterationCount;

    @Override
    public void start(Stage primaryStage) {
        showInputMethodDialog(primaryStage);
    }
    private void showInputMethodDialog(Stage primaryStage) {
        VBox layout = new VBox(15);
        layout.setPadding(new Insets(20));
        layout.setAlignment(Pos.CENTER);

        Label titleLabel = new Label("Puzzle Solver");
        titleLabel.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");

        Label subtitleLabel = new Label("Choose input method:");
        subtitleLabel.setStyle("-fx-font-size: 16px;");

        Button manualButton = new Button("Manual Input");
        Button fileButton = new Button("Load from File");

        manualButton.setStyle("-fx-min-width: 150px;");
        fileButton.setStyle("-fx-min-width: 150px;");

        manualButton.setOnAction(e -> showInitialDialog(primaryStage));
        fileButton.setOnAction(e -> handleFileInput(primaryStage));

        layout.getChildren().addAll(titleLabel, subtitleLabel, manualButton, fileButton);

        Scene scene = new Scene(layout, 400, 300);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Puzzle Solver - Input Method");
        primaryStage.show();
    }

    private void handleFileInput(Stage primaryStage) {
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Open Puzzle File");
        fileChooser.getExtensionFilters().add(
            new FileChooser.ExtensionFilter("Text Files", "*.txt")
        );

        File file = fileChooser.showOpenDialog(primaryStage);
        if (file != null) {
            try {
                InputReader reader = new InputReader(file.getAbsolutePath());
                reader.readInput();

                boardHeight = reader.getBoardHeight();
                boardWidth = reader.getBoardWidth();
                pieceCount = reader.getPieceCount();
                configType = reader.getConfigType();
                customConfig = reader.getCustomConfig();

                pieces = reader.processPieces();
                pieceDesigns = new ArrayList<>();
                for (int i = 1; i <= pieceCount; i++) {
                    boolean[][] layout = new boolean[pieces[i].n][pieces[i].m];
                    for (int row = 0; row < pieces[i].n; row++) {
                        for (int col = 0; col < pieces[i].m; col++) {
                            layout[row][col] = pieces[i].arr[row + 1][col + 1];
                        }
                    }
                    pieceDesigns.add(layout);
                }

                showSolvingView(primaryStage);

            } catch (Exception e) {
                showAlert("File Error", "Error reading file: " + e.getMessage());
                showInputMethodDialog(primaryStage);
            }
        }
    }
```

Gambar 11. Graphical User Interface (GUI) *overview*

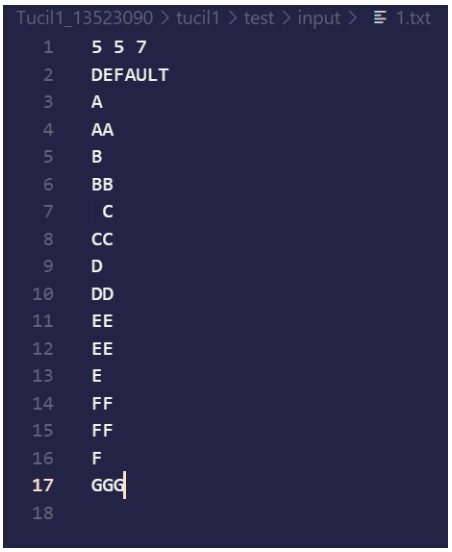
(Sumber: arsip penulis)

BAB IV: TESTING

Bagian ini merupakan bagian untuk melakukan tes pada program, input untuk tes akan disimpan pada folder …/Tucil1_13523090/tucil1/test/input dalam bentuk .txt. Hasil dari setiap test case akan disimpan pada …/Tucil1_13523090/tucil1/test/solutions. Anda juga bisa mencoba sendiri dengan tata cara dan petunjuk sudah terlampir pada file README.md. Catatan terakhir, jika Anda ingin menyimpan file, maka file akan tersimpan pada …/Tucil1_13523090/tucil1/test/output.

4.1 Test case 1

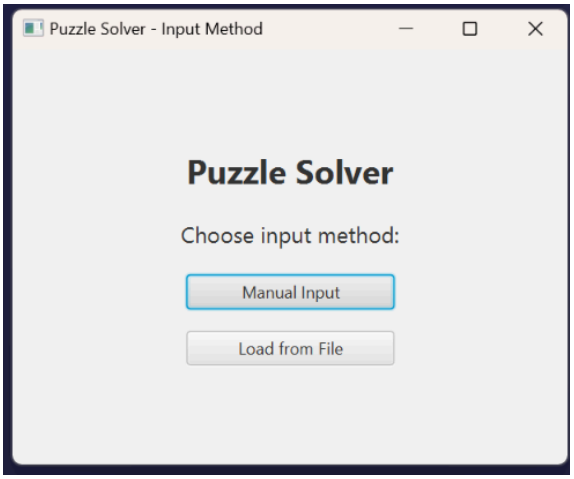
- Input

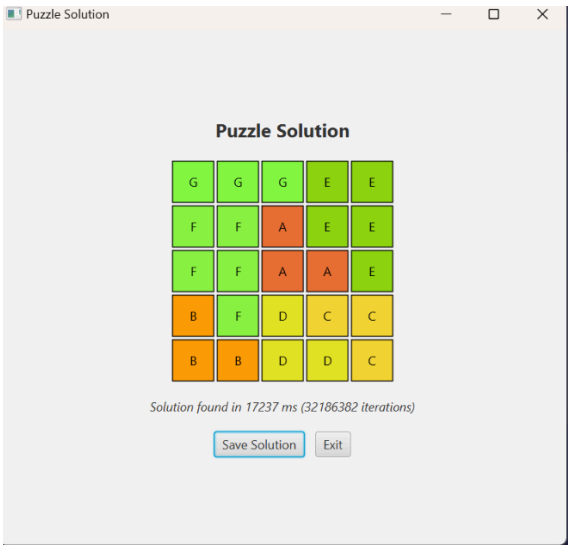


- Output
    a. Command Line Interface (CLI):

```
Processed Pieces:
Piece 1 :
#.
##
Piece 2 :
#.
##
Piece 3 :
.#
##
Piece 4 :
#.
##
Piece 5 :
##
##
#.
Piece 6 :
##
##
#.
Piece 7 :
###
Total piece area: 25
Board area: 25
Time taken: 16917 ms
Iterations: 32186382
GGGEE
FFAEE
FFAAE
BFDCC
BBDDC
Save solution? (yes/no):
```

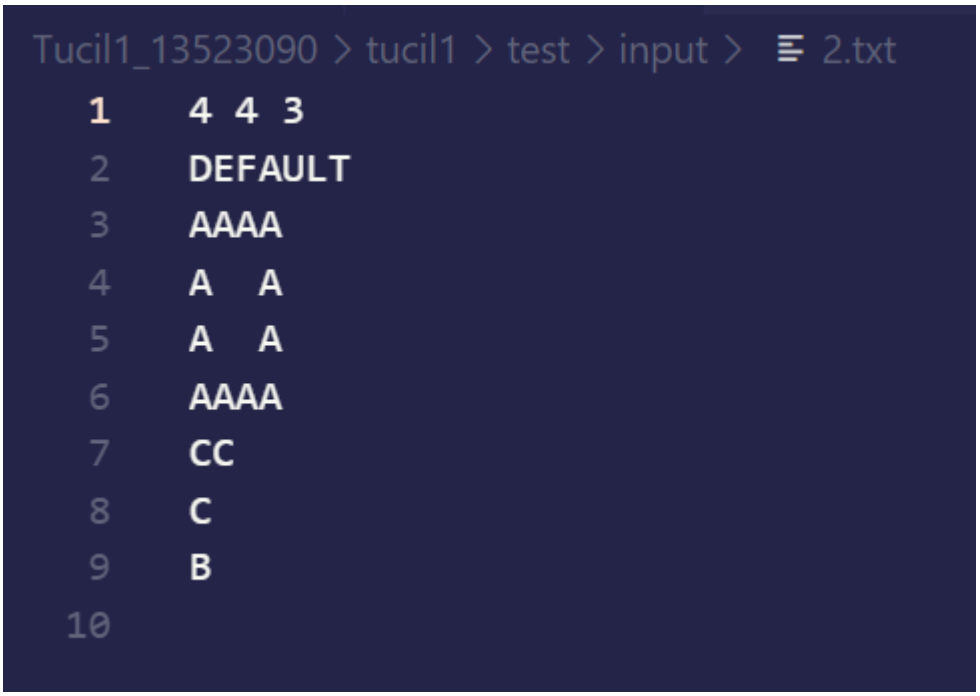
b. Graphical User Interface (GUI - Load from File)





4.2 Test case 2

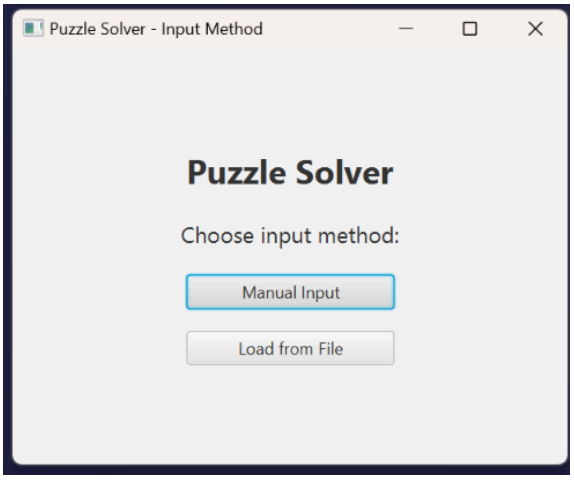
- Input

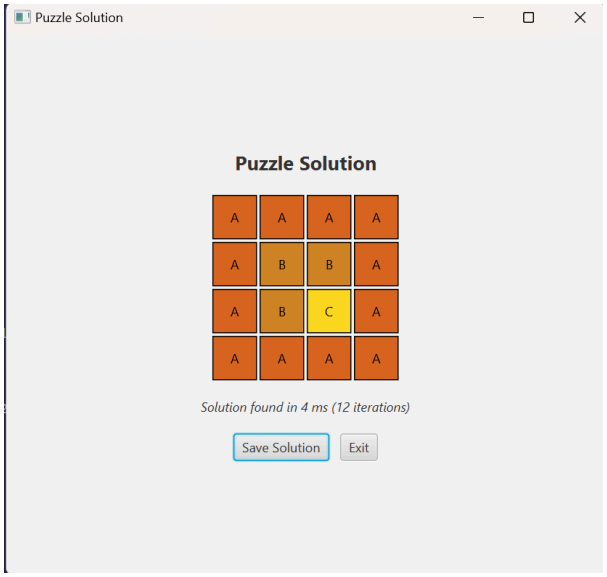


- Output
    a. Command Line Interface (CLI):

```
Processed Pieces:
Piece 1 :
####
#..#
#..#
####
Piece 2 :
##
#.
Piece 3 :
#
Total piece area: 16
Board area: 16
Time taken: 2 ms
Iterations: 12
AAAA
ABBA
ABCA
AAAA
Save solution? (yes/no):
```

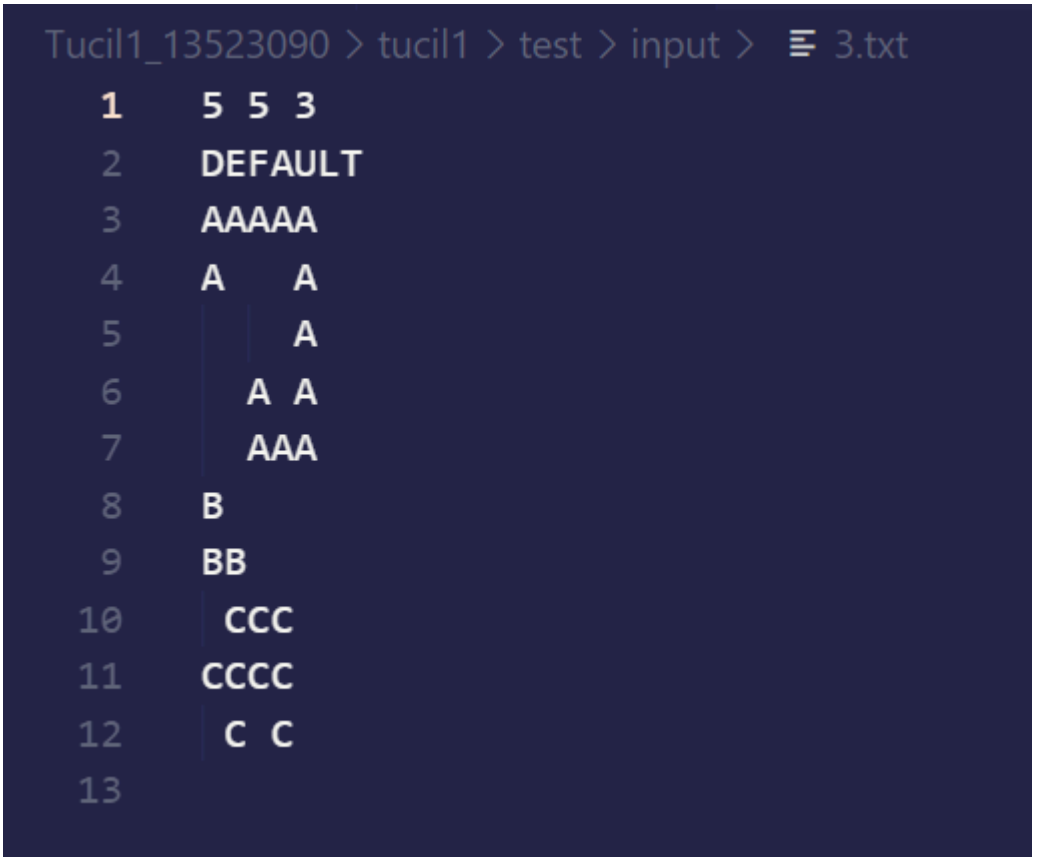
b. Graphical User Interface (GUI - Load from File)





4.3 Test case 3

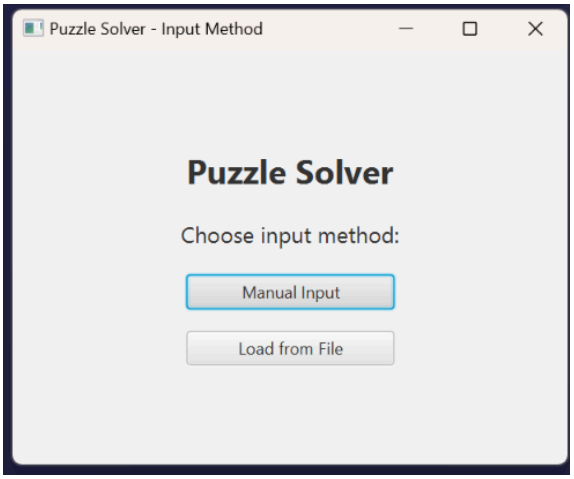
- Input

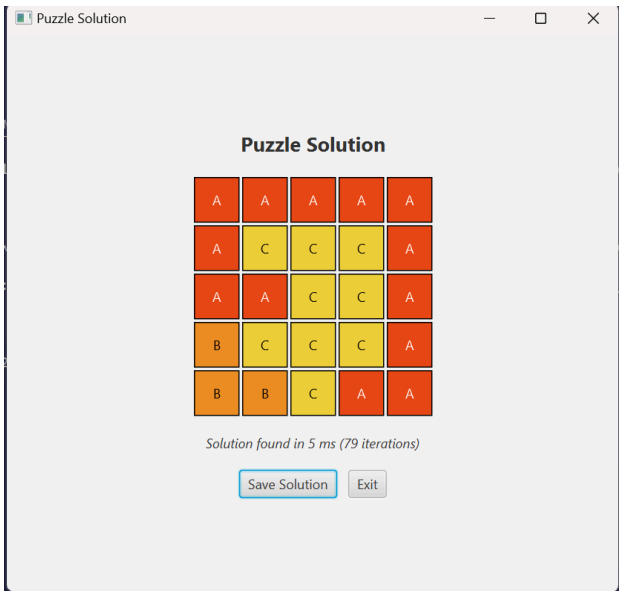


- Output
    a. Command Line Interface (CLI):

```
Processed Pieces:
Piece 1 :
#####
#...#
....#
..#.#
..###
Piece 2 :
#.
##
Piece 3 :
.###
####
.#.#
Total piece area: 25
Board area: 25
Time taken: 4 ms
Iterations: 79
AAAAA
ACCCA
AACCA
BCCCA
BBCAA
Save solution? (yes/no):
```

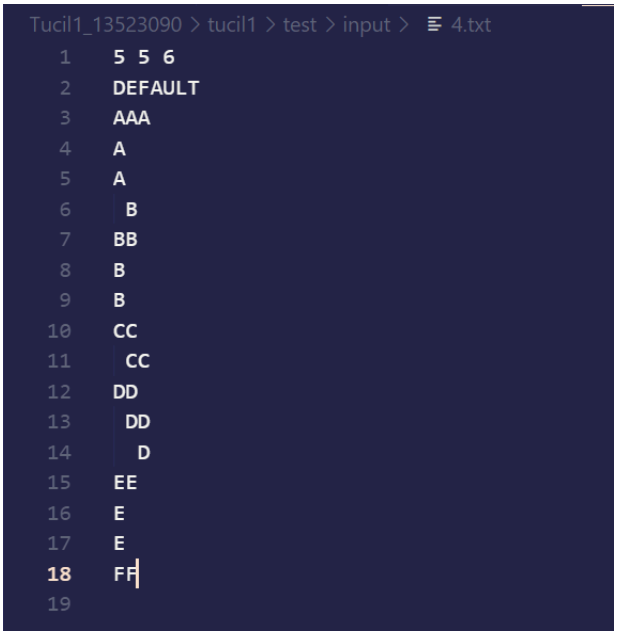
b. Graphical User Interface (GUI - Load from File)





4.4 Test case 4

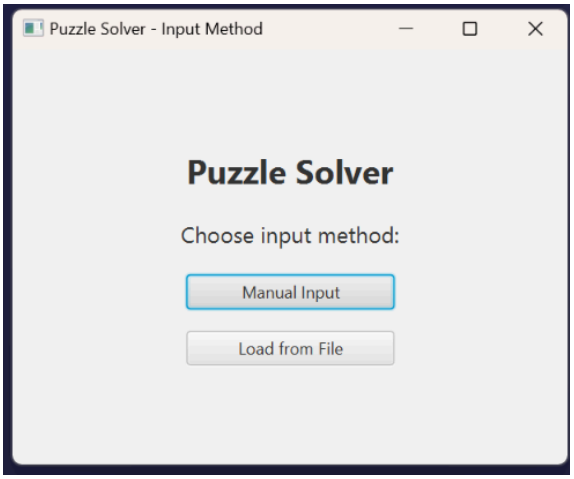
- Input

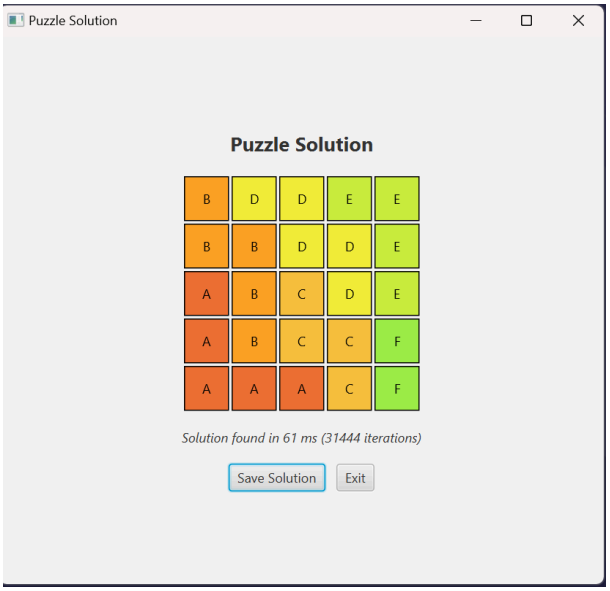


- Output
    a. Command Line Interface (CLI):

```
Processed Pieces:
Piece 1 :
###
#..
#..
Piece 2 :
.#
##
#.
#.
Piece 3 :
##.
.##
Piece 4 :
##.
.##
..#
Piece 5 :
##
#.
#.
Piece 6 :
##
Total piece area: 25
Board area: 25
Time taken: 58 ms
Iterations: 31444
BDDEE
BBDDE
ABCDE
ABCCF
AAACF
Save solution? (yes/no):
```

b. Graphical User Interface (GUI - Load from File)



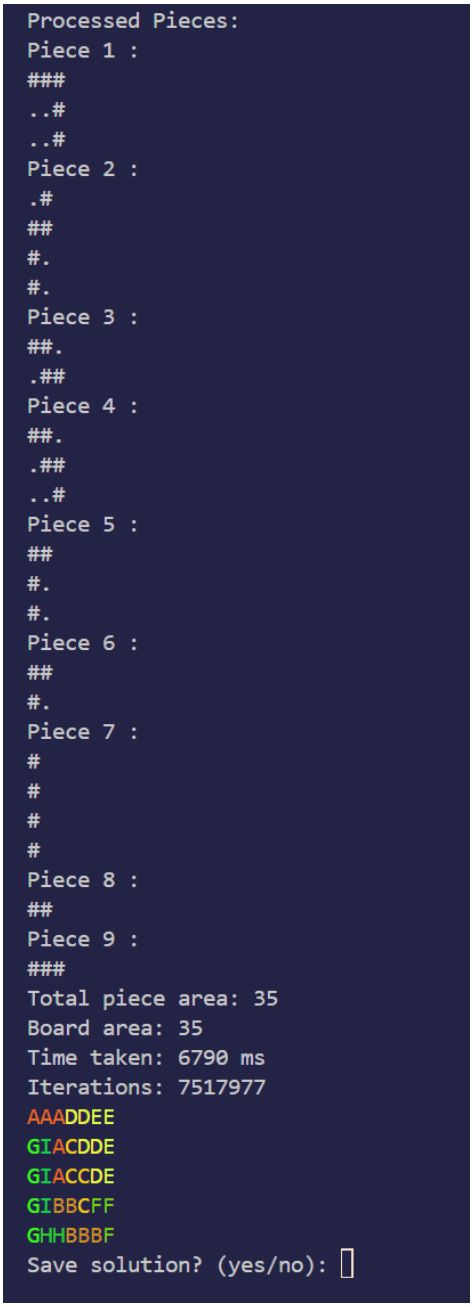


4.5 Test case 5

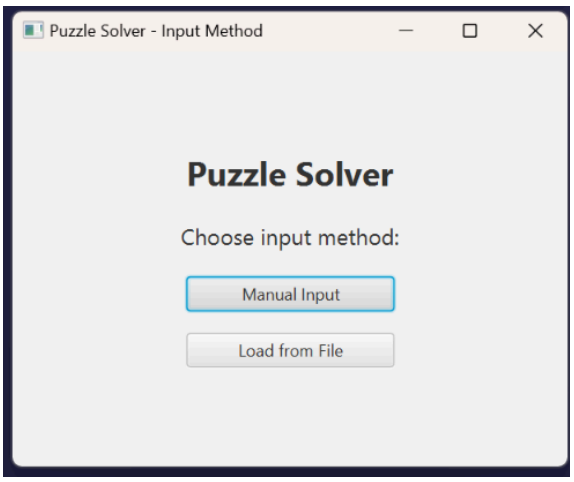
- Input

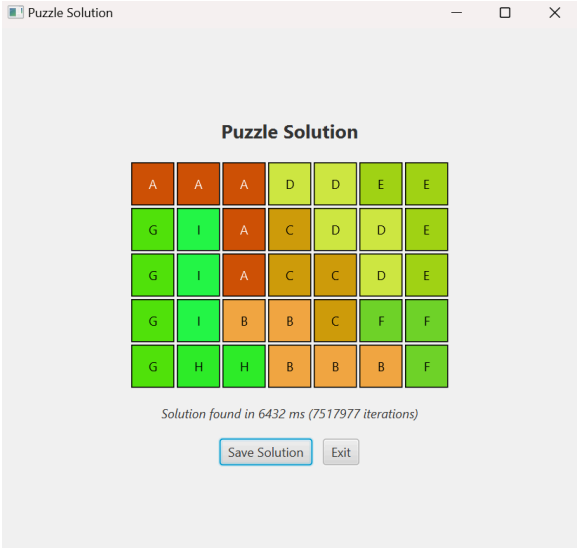
```
Tucil1_13523090 > tucil1 > test > input > 5.txt
1   5 7 9
2   DEFAULT
3   AAA
4      A
5      A
6    B
7   BB
8   B
9   B
10  CC
11   CC
12  DD
13   DD
14     D
15  EE
16  E
17  E
18  FF
19  F
20  G
21  G
22  G
23  G
24  HH
25  III
26
```

- Output
    a. Command Line Interface (CLI):



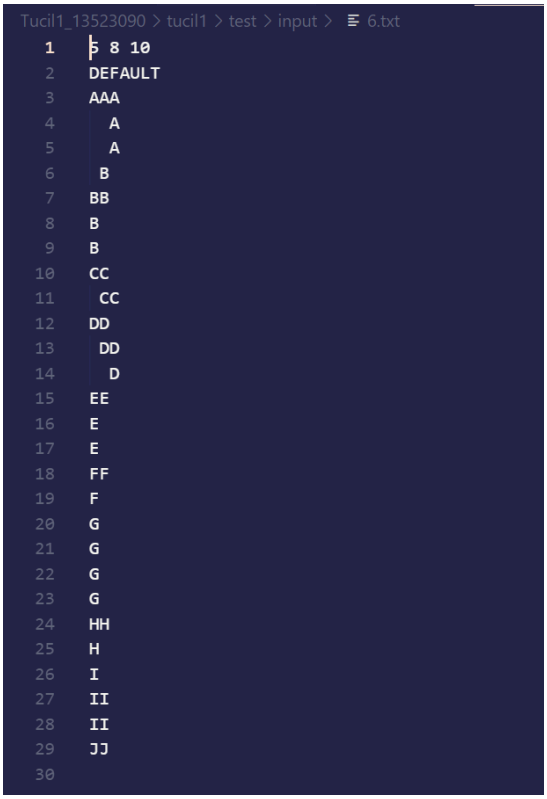
b. Graphical User Interface (GUI - Load from File)


Puzzle Solver - Input Method
Puzzle Solver
Choose input method:
Manual Input
Load from File


Puzzle Solution
Puzzle Solution
A A A D D E E
G I A C D D E
G I A C C D E
G I B B C F F
G H H B B B F
Solution found in 6432 ms (7517977 iterations)
Save Solution
Exit

4.6 Test case 6

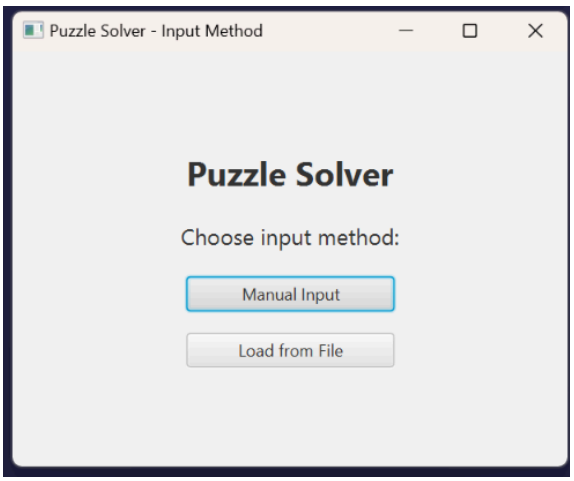
- Input

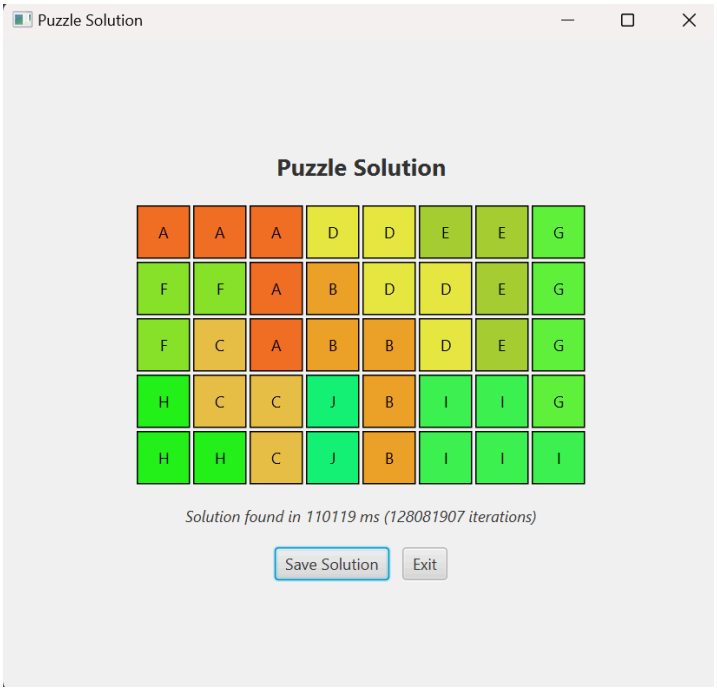


- Output
    a. Command Line Interface (CLI):

```
Processed Pieces:
Piece 1 :
###
..#
..#
Piece 2 :
.#
##
#.
#.
Piece 3 :
##.
.##
Piece 4 :
##.
.##
..#
Piece 5 :
##
#.
#.
Piece 6 :
##
#.
Piece 7 :
#
#
#
#
Piece 8 :
##
#.
Piece 9 :
#.
##
##
Piece 10 :
##
Total piece area: 40
Board area: 40
Time taken: 126248 ms
Iterations: 128081907
AAADDEEG
FFABDDEG
FCABBDEG
HCCJBIIG
HHCJBIII
Save solution? (yes/no):
```

b. Graphical User Interface (GUI - Load from File)





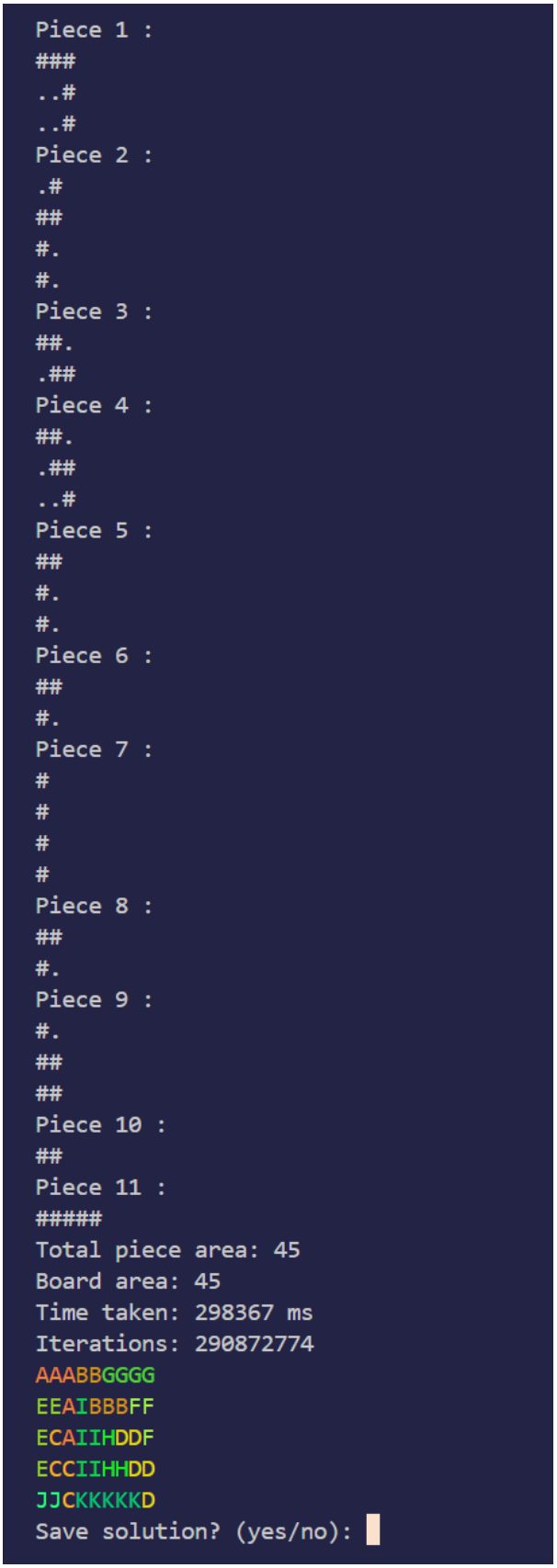
4.7 Test case 7

- Input

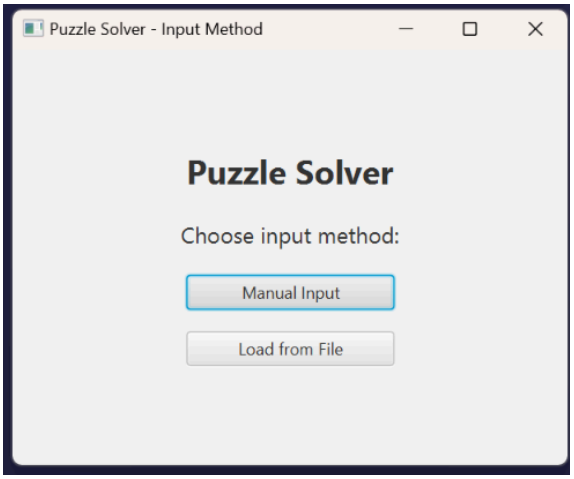
```
Tucil1_13523090 > tucil1 > test > input > 7.txt
1   5 9 11
2   DEFAULT
3   AAA
4     A
5     A
6    B
7   BB
8   B
9   B
10  CC
11   CC
12  DD
13   DD
14     D
15  EE
16  E
17  E
18  FF
19  F
20  G
21  G
22  G
23  G
24  HH
25  H
26  I
27  II
28  II
29  JJ
30  KKKKK
31
```

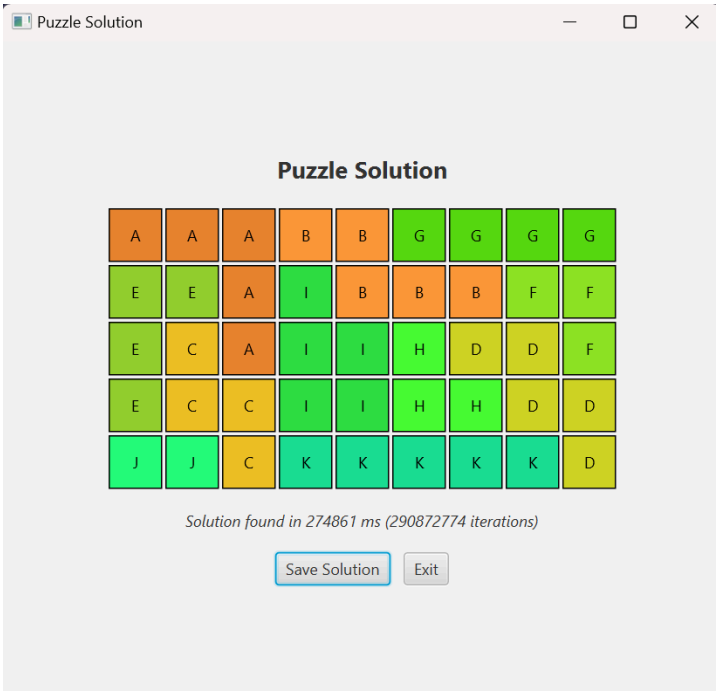
- Output

a. Command Line Interface (CLI):



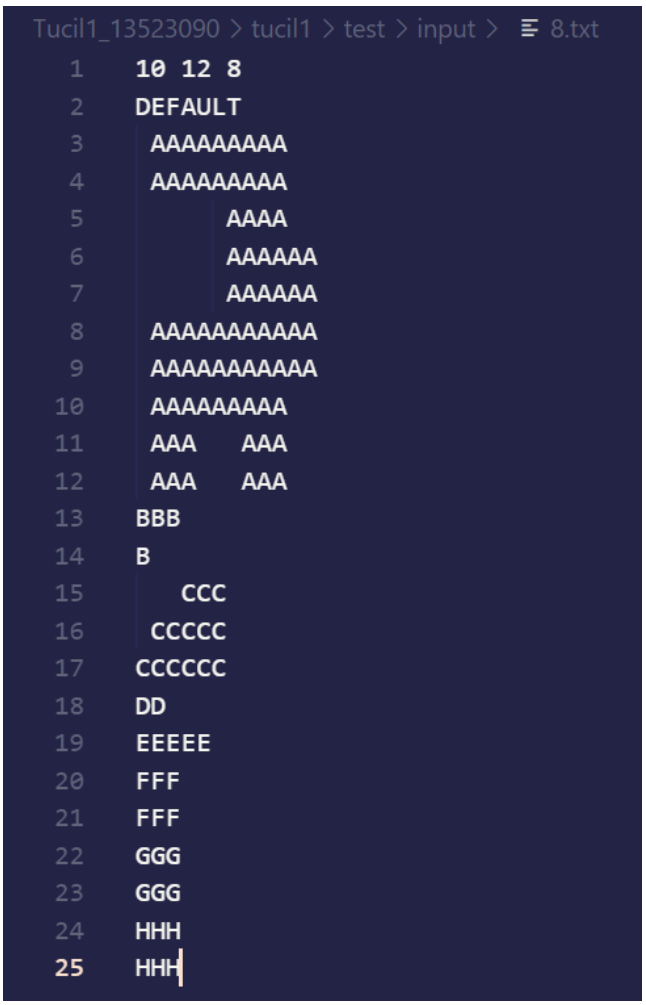
b. Graphical User Interface (GUI - Load from File)



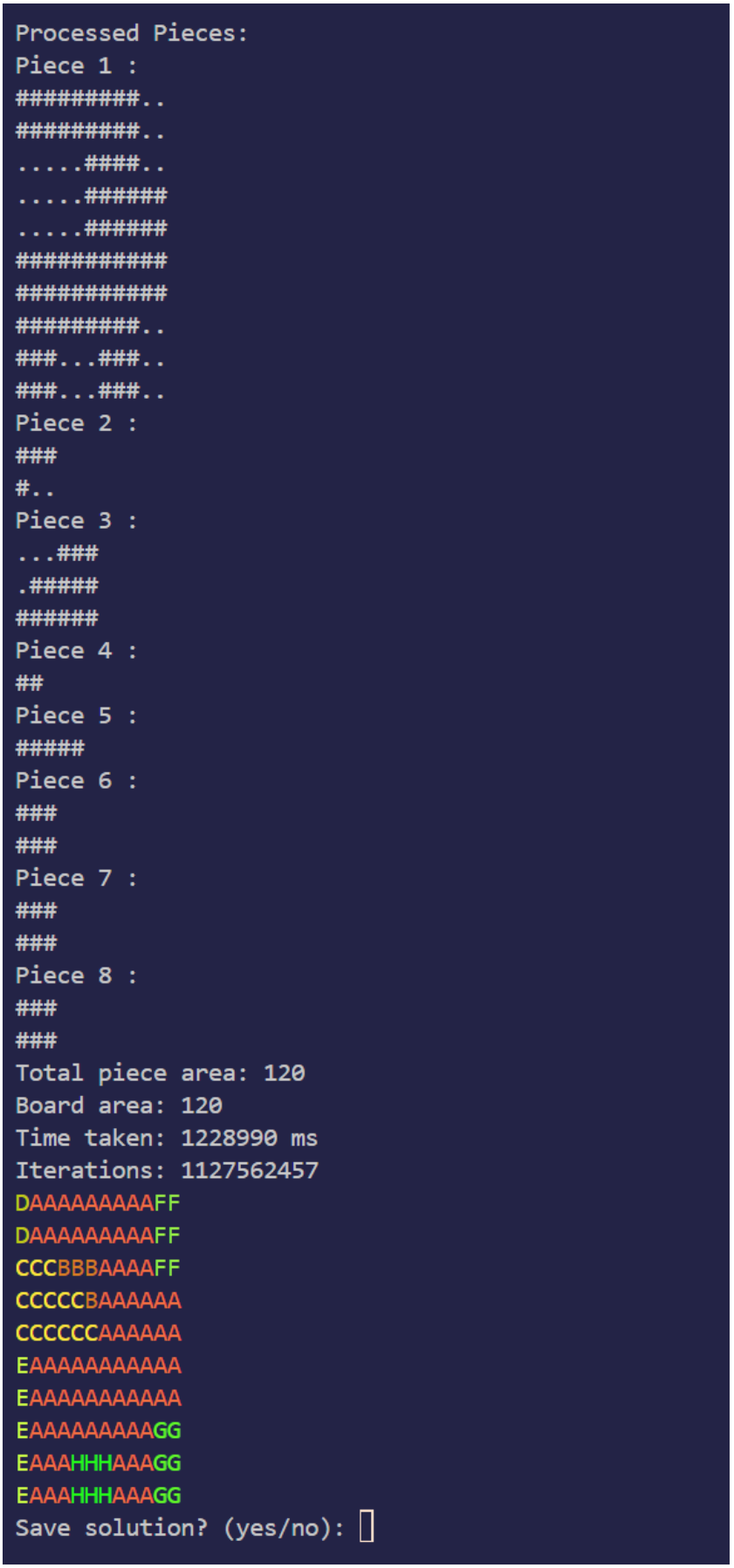


4.8 Test case 8

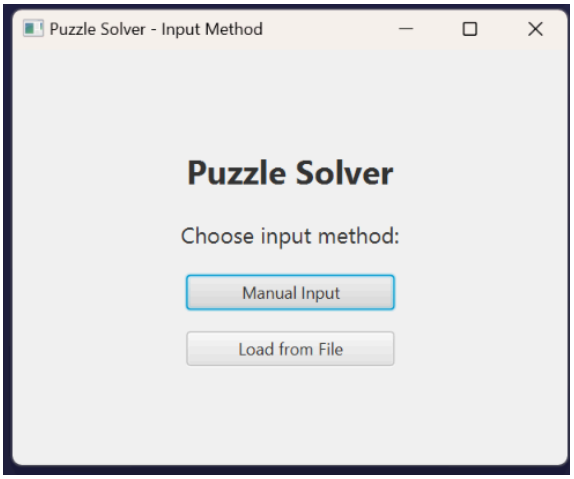
- Input



- Output
    a. Command Line Interface (CLI):

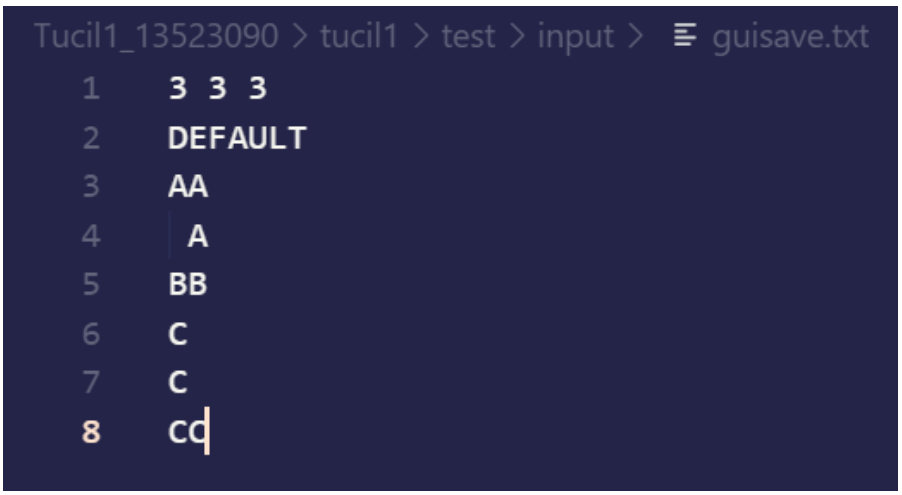

Processed Pieces:
Piece 1 :
#########..
#########..
.....####..
.....######
.....######
##########
##########
#########..
###...###..
###...###..
Piece 2 :
###
#..
Piece 3 :
...###
.#####
######
Piece 4 :
##
Piece 5 :
#####
Piece 6 :
###
###
Piece 7 :
###
###
Piece 8 :
###
###
Total piece area: 120
Board area: 120
Time taken: 1228990 ms
Iterations: 1127562457
DAAAAAAAAAFF
DAAAAAAAAAFF
CCCBBBAAAAFF
CCCCCBAAAAAA
CCCCCCAAAAAA
EAAAAAAAAAAA
EAAAAAAAAAAA
EAAAAAAAAAGG
EAAAHHHAAAGG
EAAAHHHAAAGG
Save solution? (yes/no):

b. Graphical User Interface (GUI - Load from File)

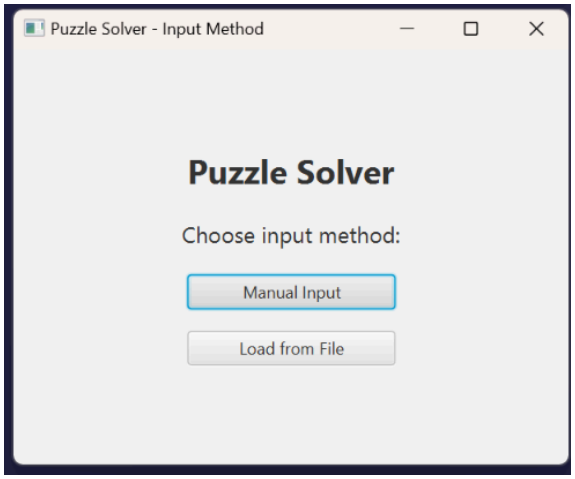


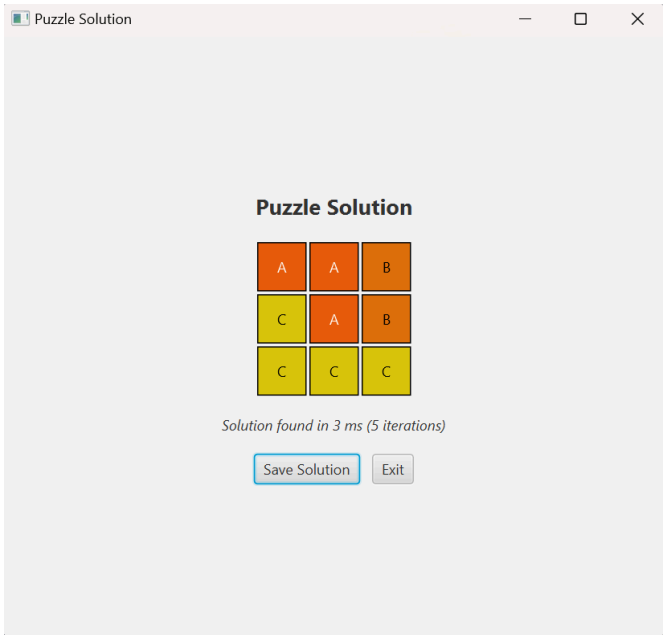
4.9 Test case: GUI menyimpan ke .txt dan .png

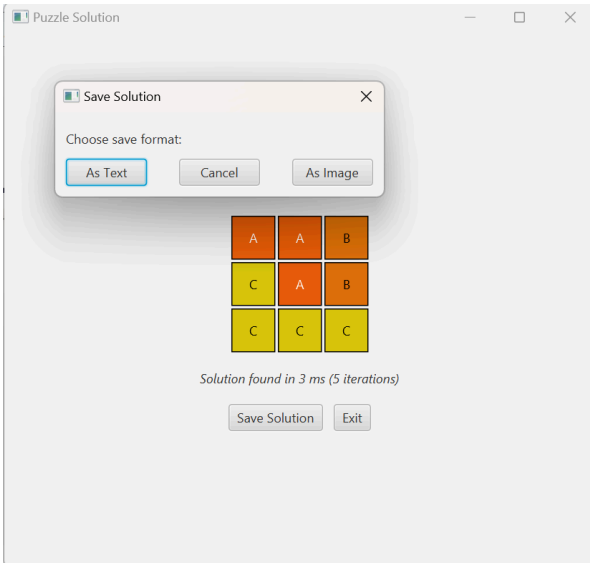
- Input

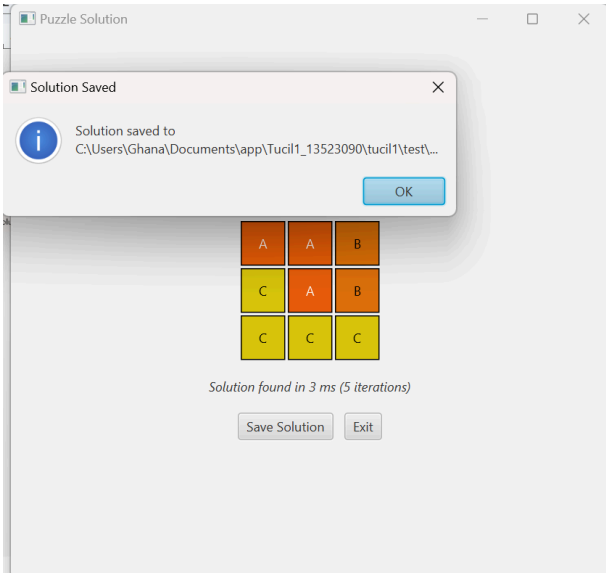


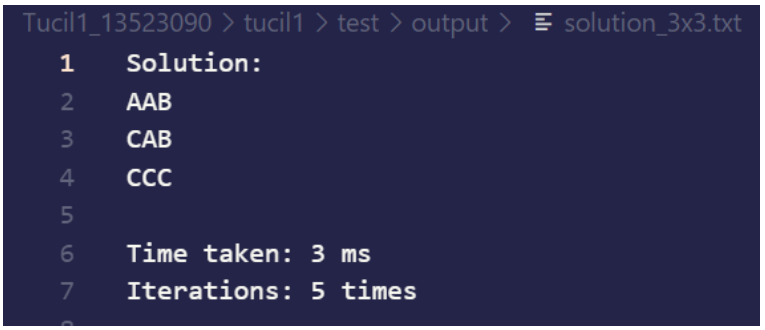
- Output
    a. Text file (.txt)



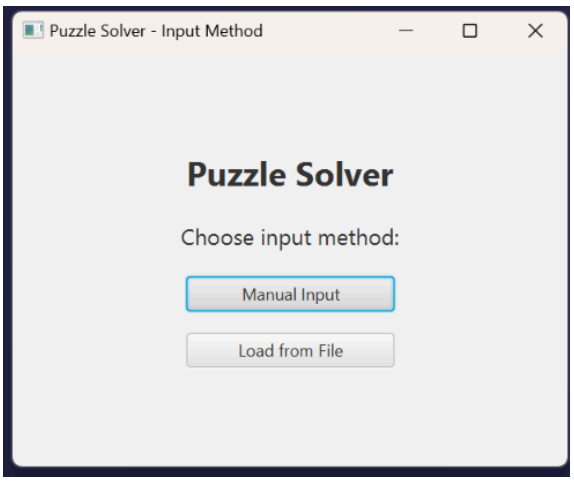


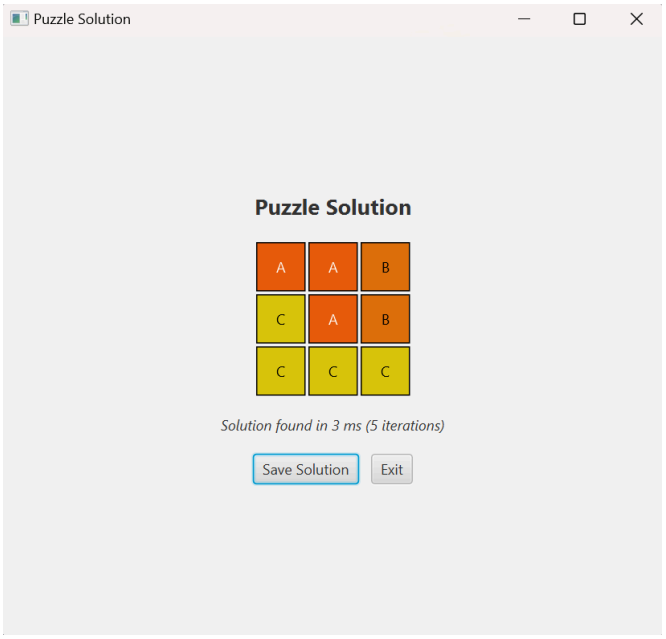


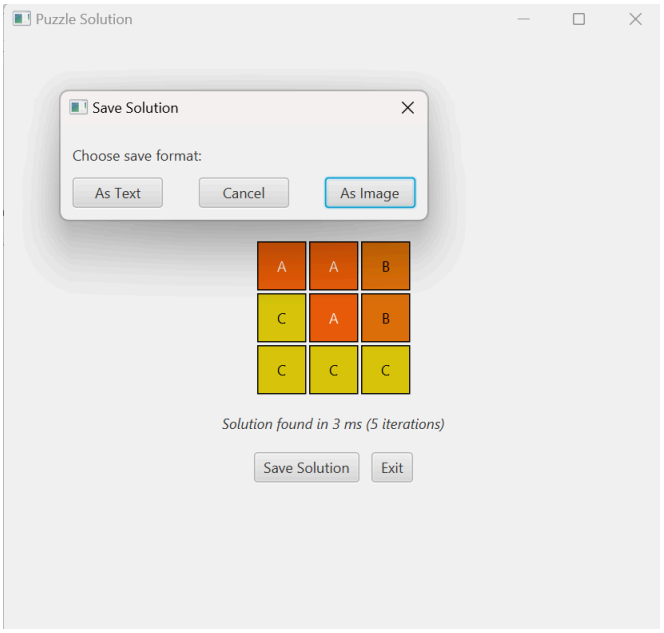


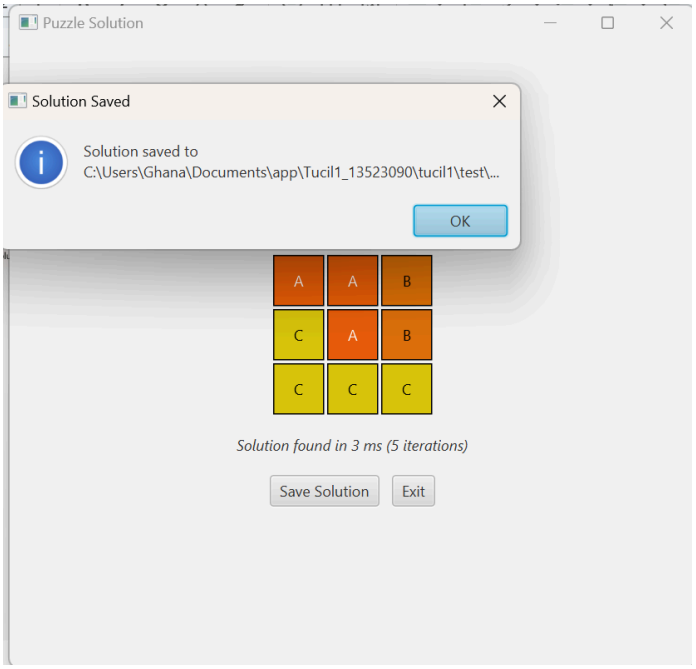


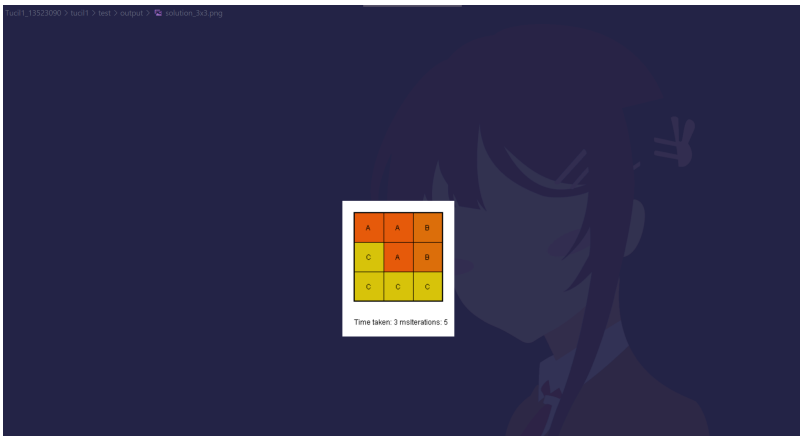
b. Image file (.png) - Load from File


Puzzle Solver - Input Method
Puzzle Solver
Choose input method:
Manual Input
Load from File


Puzzle Solution
Puzzle Solution
A A B
C A B
C C C
Solution found in 3 ms (5 iterations)
Save Solution
Exit

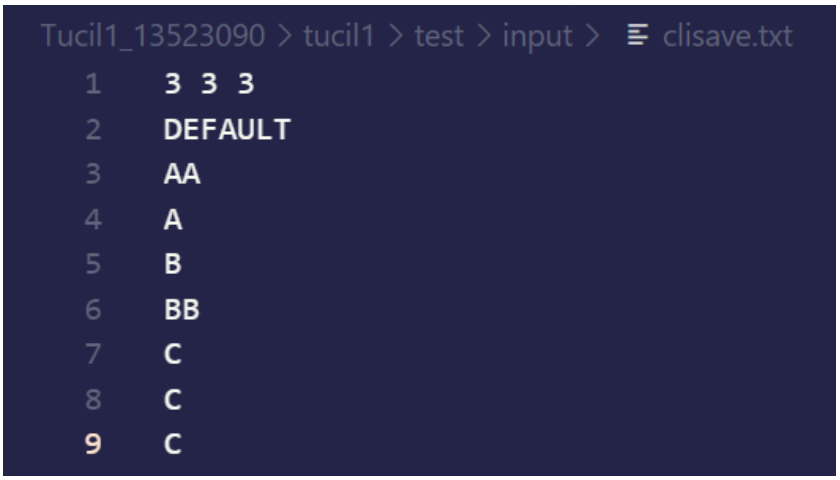

Puzzle Solution
Save Solution
Choose save format:
As Text
Cancel
As Image
A A B
C A B
C C C
Solution found in 3 ms (5 iterations)
Save Solution
Exit





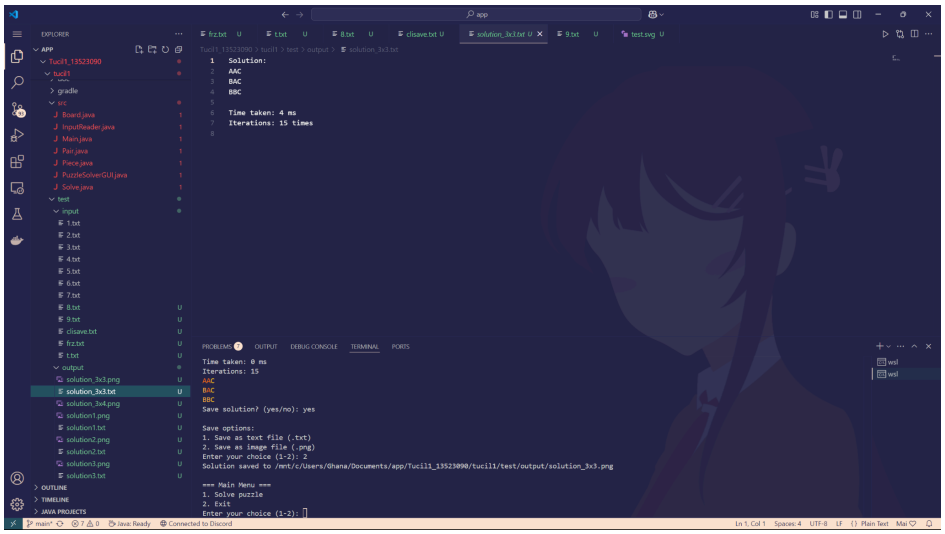
4.10 Test case: CLI menyimpan ke .txt dan .png

- Input



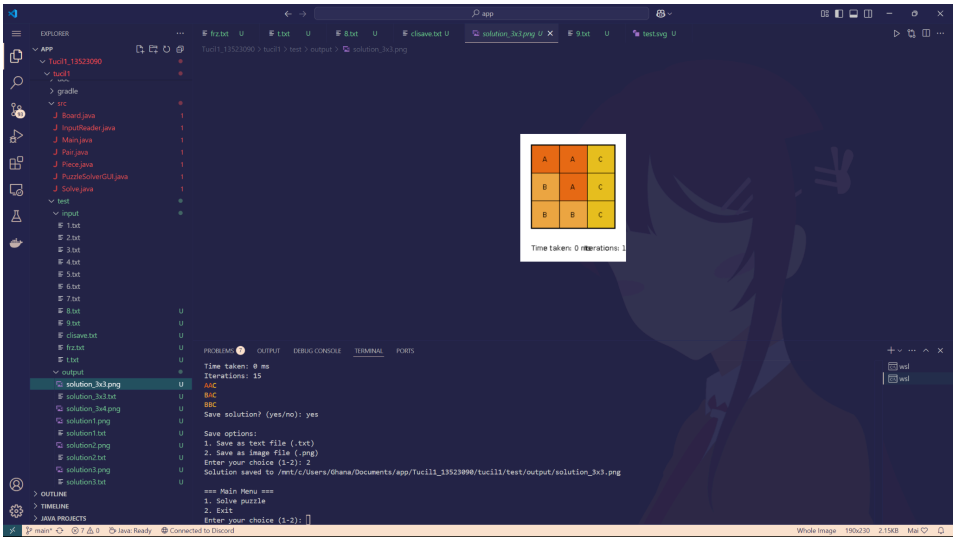
- Output
    a. Text file (.txt)

```
Processed Pieces:
Piece 1 :
##
#.
Piece 2 :
#.
##
Piece 3 :
#
#
#
Total piece area: 9
Board area: 9
Time taken: 4 ms
Iterations: 15
AAC
BAC
BBC
Save solution? (yes/no): yes

Save options:
1. Save as text file (.txt)
2. Save as image file (.png)
Enter your choice (1-2): 1
Solution saved to /mnt/c/Users/Ghana/Documents/app/Tucil1_13523090/tucil1/test/output/solution_3x3.txt
```



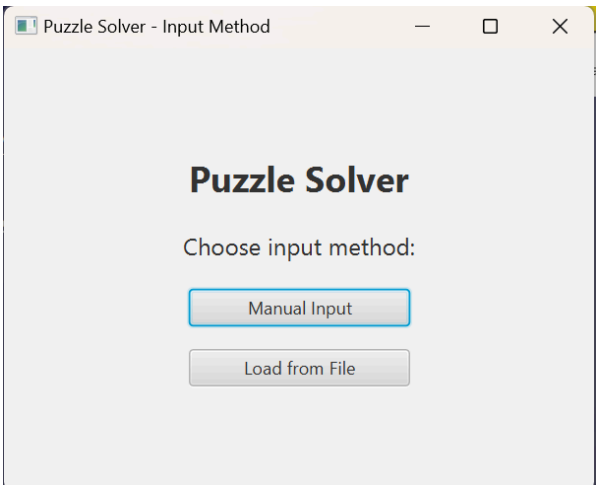
    b. Image file (.png)

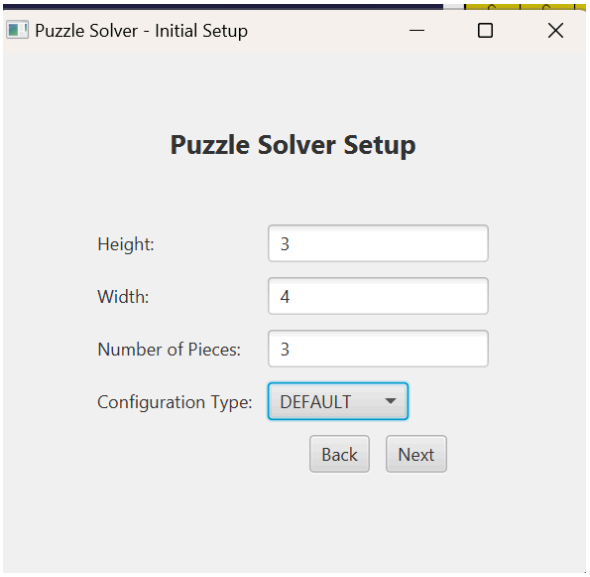
```
Processed Pieces:
Piece 1 :
##
#.
Piece 2 :
#.
##
Piece 3 :
#
#
#
Total piece area: 9
Board area: 9
Time taken: 0 ms
Iterations: 15
AAC
BAC
BBC
Save solution? (yes/no): yes

Save options:
1. Save as text file (.txt)
2. Save as image file (.png)
Enter your choice (1-2): 2
Solution saved to /mnt/c/Users/Ghana/Documents/app/Tucil1_13523090/tucil1/test/output/solution_3x3.png
```

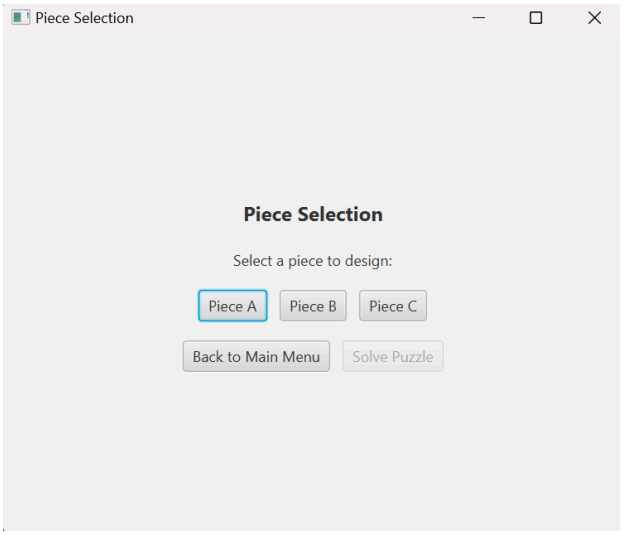


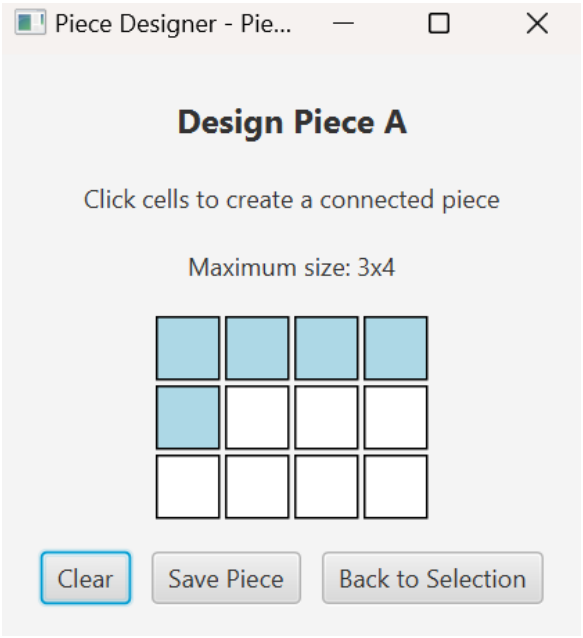
4.11 Test case: GUI Manual Input

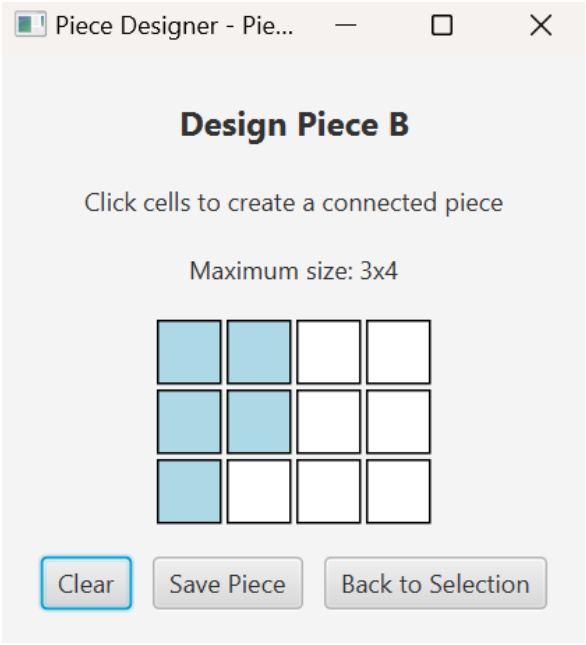
- Input

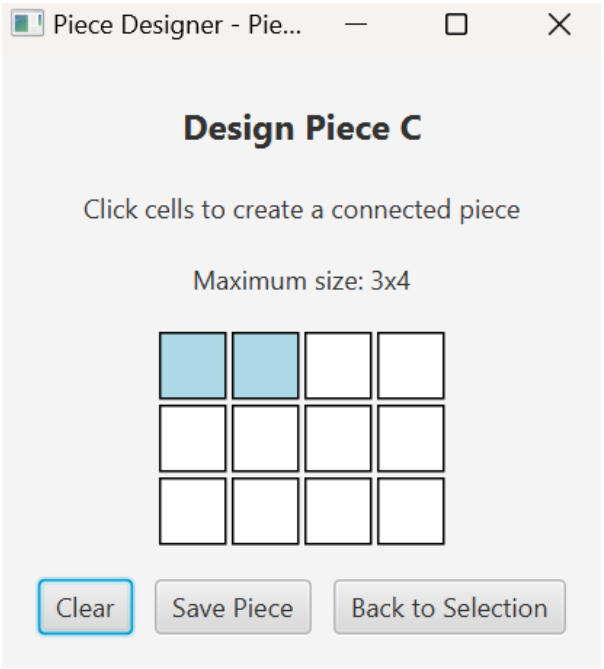

Puzzle Solver - Input Method
Puzzle Solver
Choose input method:
Manual Input
Load from File

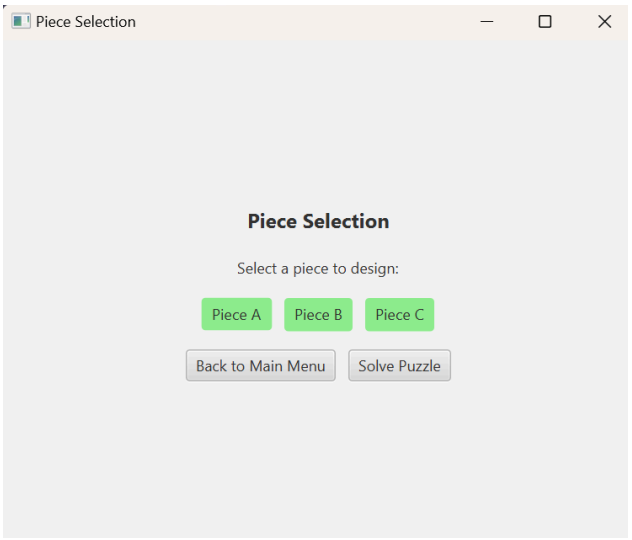

Puzzle Solver - Initial Setup
Puzzle Solver Setup
Height:
3
Width:
4
Number of Pieces:
3
Configuration Type:
DEFAULT
Back
Next

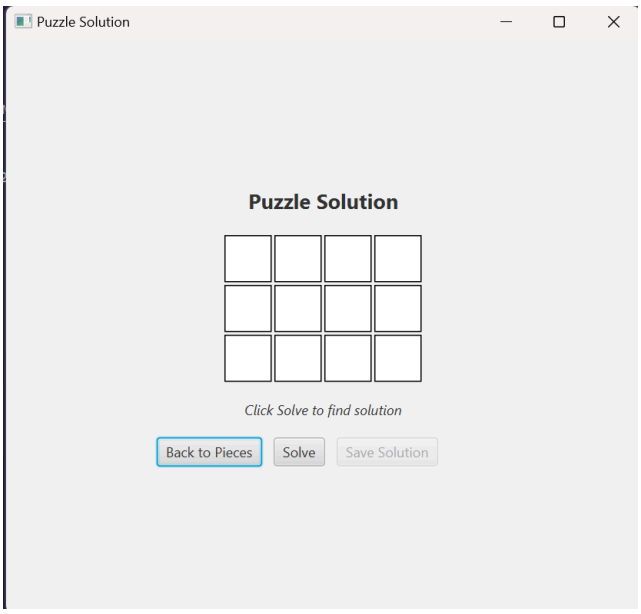

Piece Selection
Piece Selection
Select a piece to design:
Piece A
Piece B
Piece C
Back to Main Menu
Solve Puzzle


Piece Designer - Pie...
Design Piece A
Click cells to create a connected piece
Maximum size: 3x4
Clear
Save Piece
Back to Selection

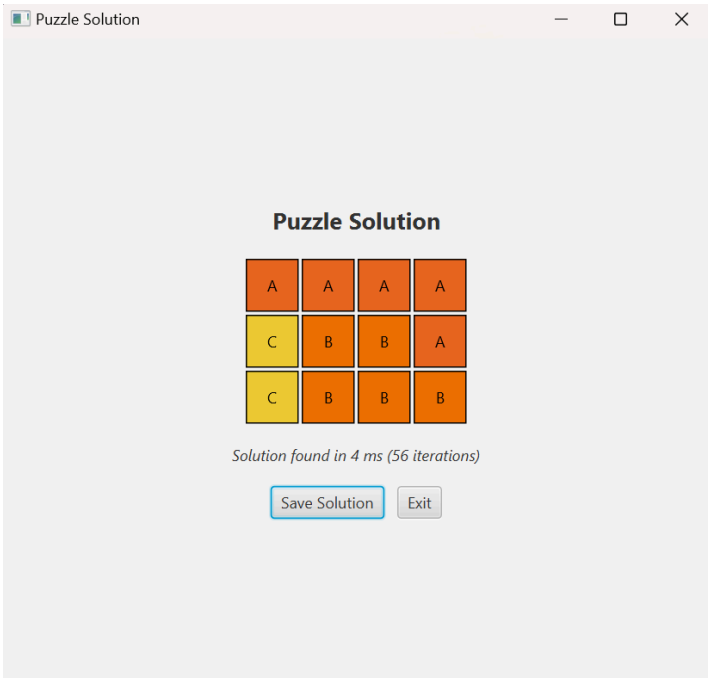

Piece Designer - Pie...
Design Piece B
Click cells to create a connected piece
Maximum size: 3x4
Clear
Save Piece
Back to Selection


Piece Designer - Pie...
Design Piece C
Click cells to create a connected piece
Maximum size: 3x4
Clear
Save Piece
Back to Selection


Piece Selection
Piece Selection
Select a piece to design:
Piece A
Piece B
Piece C
Back to Main Menu
Solve Puzzle

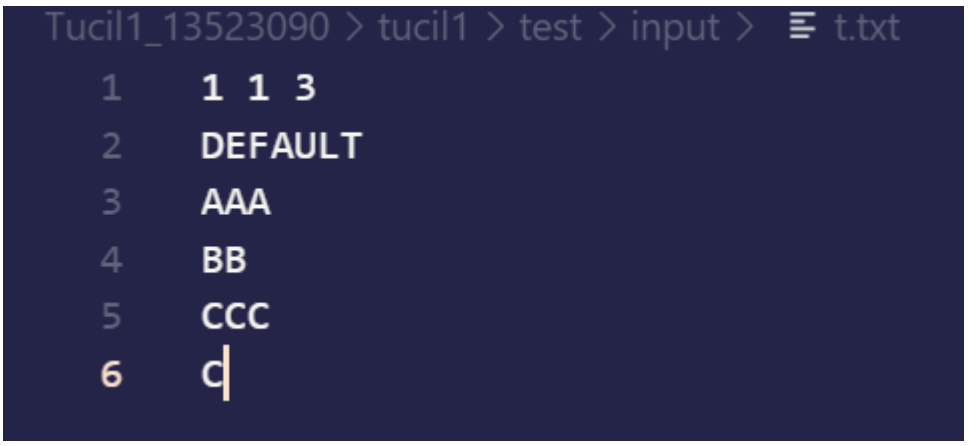


- Output

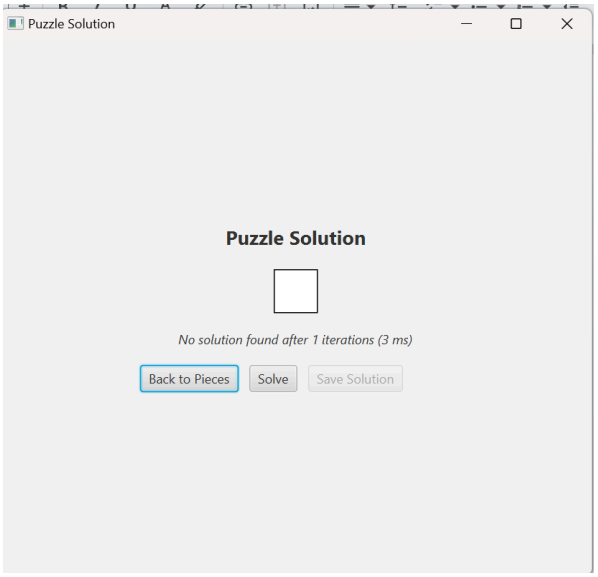


4.12 Test case: Tidak ada solusi

- Input (GUI : load from file)

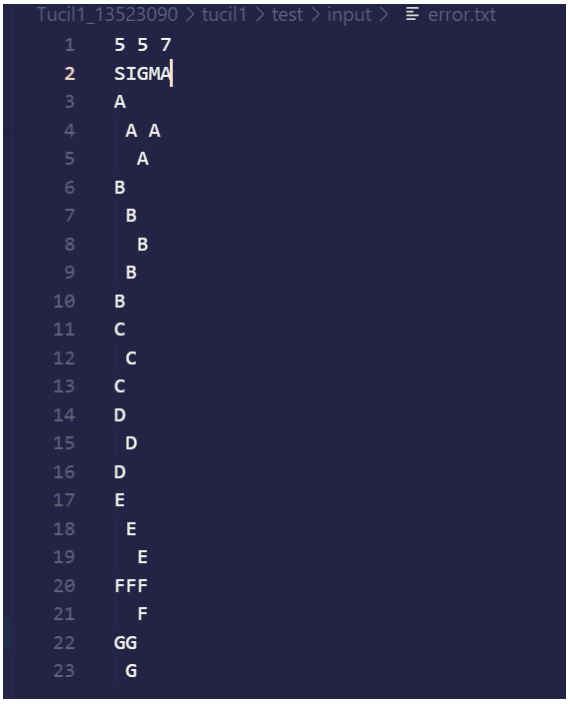


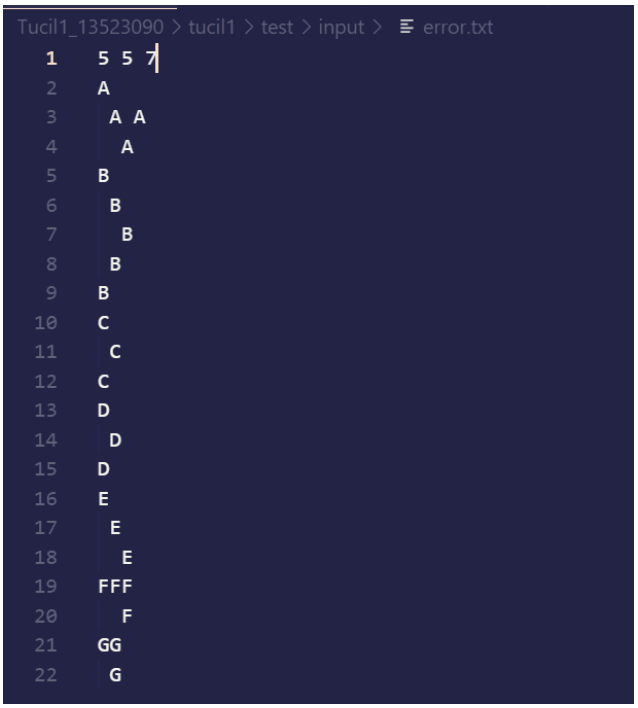
- Output



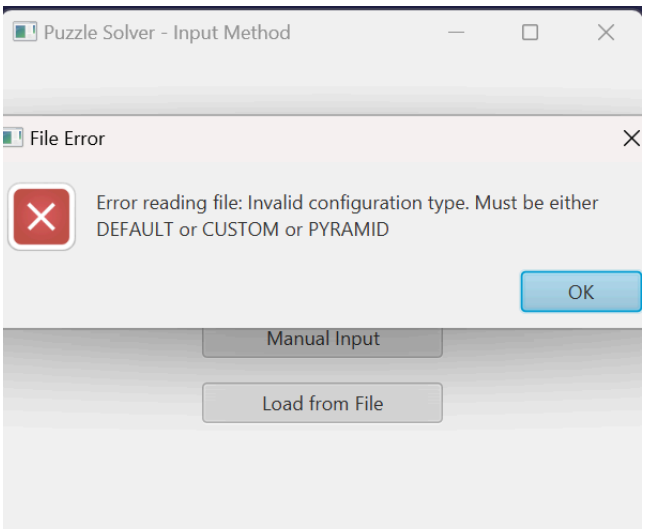
4.13 Test case: Error Handling

- Input

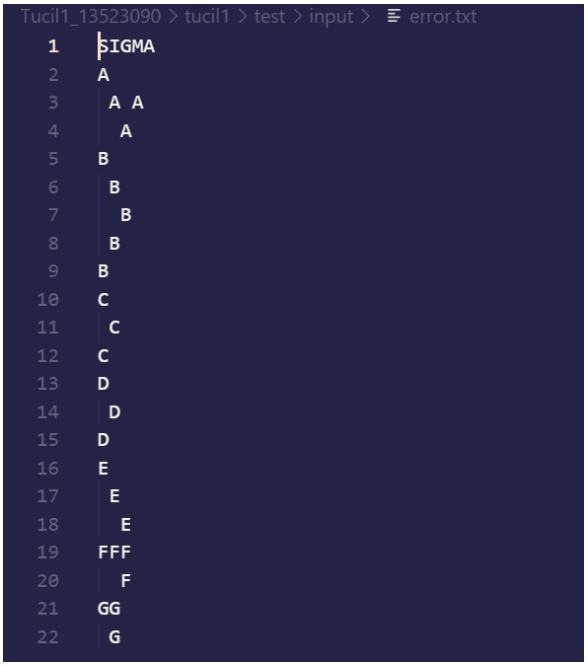

Tucil1_13523090 > tucil1 > test > input > error.txt
5 5 7
SIGMA
A
A A
A
B
B
B
B
B
C
C
C
D
D
D
E
E
E
FFF
F
GG
G


Tucil1_13523090 > tucil1 > test > input > error.txt
5 5 7
A
A A
A
B
B
B
B
B
C
C
C
D
D
D
E
E
E
FFF
F
GG
G

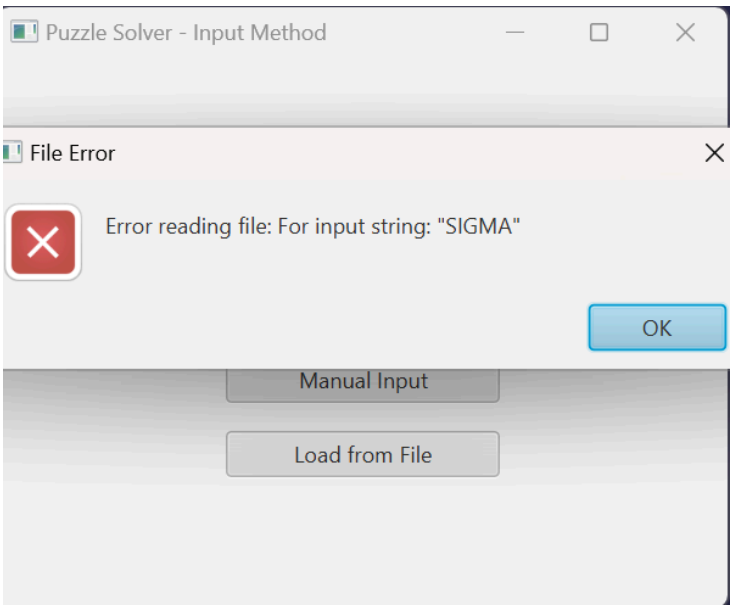
- Output



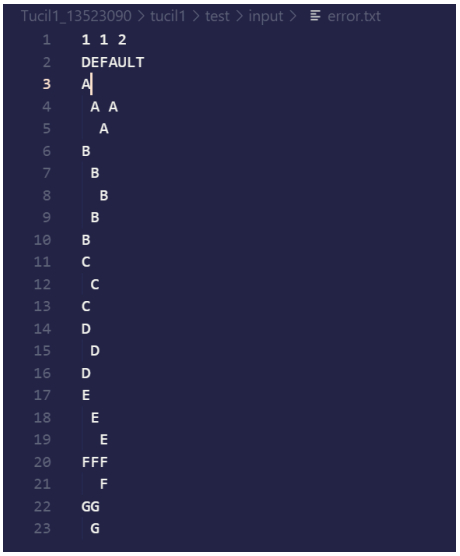
- Input



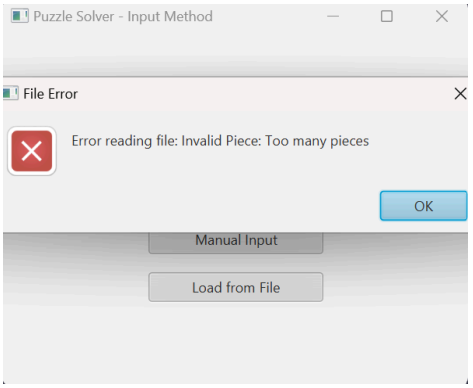
- Output



- Input



- Output



LAMPIRAN

Github Repository

Program dapat diakses pada https://github.com/Nayekah/Tucil1_13523090

Miscellaneous (Tabel Poin)

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	✓	