Tugas Kecil 1 Strategi Algoritma

Penyelesaian Games IQ Puzzle Pro

Disusun oleh:

13523137 – Muhammad Aulia Azka

Institut Teknologi Bandung

2025

Daftar Isi

Halaman Sampul....1

Daftar Isi....2

Bab I....4

Deskripsi Masalah dan Algoritma....4

1.1 Algoritma Brute Force....4
1.2 IQ Puzzle Pro....4
1.3 Algoritma penyelesaian IQ Puzzle Pro....4

Bab II....6

Implementasi Algoritma dalam Bahasa Java....6

2.1 File Main.java....6
2.2 File DataType.java....6
2.3 File Point.java....6
2.4 File Piece.java....6
2.5 File TestReader.java....7
2.6 File Solver.java....7

Bab III....9

SOURCE CODE PROGRAM....9

3.1 Repositori....9
3.2 Main.java....9
3.3 DataType.java....11
3.4 Piece.java....13
3.5 Point.java....14
3.6 TestReader.java....15
3.7 Solver.java....19

Bab IV....32

Masukan dan Luaran Program....32

4.1 Test_1....32
4.2 Test_2....33

4.3 Test_3.....34
4.4 Test_4.....35
4.5 Test_5.....36
4.6 Test_6.....37
4.7 Test_7.....39
Bab V.....41
Lampiran.....41

BAB I

DESKRIPSI MASALAH DAN ALGORITMA

1.1 Algoritma Brute Force

Algoritma brute force adalah algoritma yang memiliki pendekatan yang lampang. Biasanya algoritma brute force didasarkan pada pernyataan pada persoalan dan Definisi/konsep yang terlibat. Algoritma brute force menyelesaikan persoalan secara sangat sederhana, langsung, dan jelas caranya.

Algoritma brute force umumnya tidak “cerdas” dan tidak mangkus, karena ia membutuhkan volume komputasi yang besar dan waktu yang lama dalam penyelesaiannya. Algoritma brute force juga kadang disebut algoritma naif. Algoritma ini cocok untuk persoalan yang masukannya kecil. Meskipun bukan metode problem solving yang mangkus, hampir semua persoalan dapat diselesaikan dengan algoritma brute force. Ini adalah kelebihan brute force Sukar menunjukkan persoalan yang tidak dapat diselesaikan dengan metode brute force.

1.2 IQ Puzzle Pro

IQ Puzzle 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. 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.

1.3 Algoritma penyelesaian IQ Puzzle Pro

Dalam penyelesaian games IQ Puzzle Pro, penulis menggunakan pendekatan

Brute force. Berikut adalah langkah – langkah penyelesaian:

1. Pengguna pertama – tama memasukan ukuran papan, jumlah blok, dan blok. (masukkan dari file).
2. Blok akan diletakan secara berurutan (Blok ke-0, blok ke-1, dan seterusnya).
3. Cek apakah semua blok telah diletakan. Caranya adalah indeks blok sama dengan jumlah bloknya. Awalnya indeks blok adalah 0.
4. Transformasikan blok agar dapat dimasukkan ke papan. Blok dapat dirotasikan sebesar 90°, 180°, dan 270°, dan blok dapat dicerminkan secara horizontal dan vertikal.

5. Coba letakan blok ke setiap posisi papan. Cek apakah blok (dengan transformasi tertentu) dapat ditempatkan. Jika bisa, letakan bloknya. Metode yang digunakan untuk mencari semua kemungkinan adalah metode *backtracking*. Jadi misal satu blok telah berhasil ditempatkan dan blok selanjutnya sedang dicoba menempatkan. Jika gagal ditempatkan maka blok satu akan dihapus dan coba posisi/orientasi lain.
6. Jika akhirnya tidak ada yang berhasil maka dapat disimpulkan puzzle tidal punya solusi.
7. Jika mendapat solusi program akan menampilkan konfigurasi blok puzzle, waktu eksekusi, banyaknya kasus, dan menanyakan apakah ingin menyimpan solusi ke file berekstensi .txt

BAB II

Implementasi Algoritma dalam Bahasa Java

2.1 File Main.java

File ini merupakan program utama sehingga hanya terdapat class Main untuk memulai programnya

2.2 File DataType.java

Objek	Deskripsi
jumlahPiece (int)	Menyimpan informasi jumlah blok/Piece
panjangPapan (int)	Menyimpan informasi panjang papan
lebarPapan (int)	Menyimpan informasi lebar papan
caseType (String)	Menyimpan informasi jenis kasus
Board (Matriks of char)	Menyimpan informasi board/papan
Pieces (List of Piece)	Menyimpan informasi Piece (Bentuk Piece dan IDnya).
DataType	Method konstruktor
getJumlahPiece	Method mengambil jumlah Piece/blok
getPanjangPapan	Method mengambil panjang papan
getLebarPapan	Method mengambil lebar papan
getCaseType	Method mengambil jenis kasus
getBoard	Method mengambil board/papan
addPiece	Method menambah Piece
getPieces	Method mengambil Piece
setElmtBoard	Method menginisialisasi board dengan ‘ ‘

2.3 File Point.java

Objek	Deskripsi
X (int)	Menyimpan informasi absis
Y (int)	Menyimpan informasi ordinat
Point	Method konstruktor
getX	Method mengambil absis
getY	Method mengambil ordinat
setX	Method inisialisasi absis
SetY	Method inisialisasi ordinat

2.4 File Piece.java

Objek	Deksripsi
Id (char)	Menyimpan informasi ID dari blok/Piece
Coords (List of Point)	Menyimpan informasi bentuk dari blok/Piece dalam bentuk koordinat
Piece	Method konstruktor
getID	Method mengambil ID
getCoords	Method mengambil koordinat
addCoords	Method menambah koordinat
setcoord	Method set koordinat

2.5 File TestReader.java

Objek	Deksripsi
readDataFile	Method untuk membaca file input, memproses seluruh isinya, dan mengonversi menjadi objek DataType
PieceResult	Kelas pembantu yang digunakan sebagai “wadah” untuk mengembalikan piece dan lineCount
readOnePiece	Method untuk membaca sekelompk baris dari list mulai dari indeks awal yang mendefinisikan sebuah Piece
convertLineToCoords	Method mengonversi satu baris teks dari file menjadi koordinat untuk sebuah Piece

2.6 File Solver.java

Objek	Deskripsi
Data (DataType)	Menyimpan informasi objek DataType
Iteration (long)	Menyimpan informasi banyaknya kasus
solFound (boolean)	Menyimpan status solusi
Solver	Method konstruktor
getIteration	Method mengambil Iteration
getsolFound	Method mengambil solFound
printSol	Method untuk menampilkan solusi
getColorForPiece	Method untuk mewarnai blok/Piece
saveSol	Method untuk menyimpan solusi
solve	Method utama algoritma brute force
isBoardFull	Method untuk mengecek apakah papan/board penuh
rotate90	Method untuk rotasi 90 derajat
rotate180	Method untuk rotasi 180 derajat

rotate270	Method untuk rotasi 270 derajat
mirrorHorizontal	Method untuk pencerminan horizontal
mirrorVertical	Method untuk pencerminan vertikal
normalize	Method untuk normalisasi
canPlace	Method untuk mengecek apakah Piece/blok dapat ditempatkan
placePiece	Method untuk menempatkan Piece/blok
removePiece	Method untuk menghapus Piece/blok dari papan
transAll	Method untuk menjalankan semua transformasi
copyPiece	Method untuk mengcopy Piece/blok

BAB III

SOURCE CODE PROGRAM

3.1 Repositori

https://github.com/Azzkaaaa/Tucil_13523137

3.2 Main.java

```
import java.io.File;
import java.util.Scanner;


public class Main {
   public static void main(String[] args) {
      Boolean valid = false;
      TestReader test = new TestReader();
      DataType dt = null;
      Scanner scanFile = new Scanner(System.in);


      while(!valid){
         System.out.print("Masukkan nama file: ");
         String testName = scanFile.nextLine();


         File file = new File("test/" + testName);


         if (!file.exists()){
            System.out.println("File tidak ditemukan, COba lagi!..");
         }else {
            try {
               dt = test.readDataFile("test/" + testName);
```

```
            valid = true;
        } catch (Exception e) {
            System.out.println("Terjadi kesalahan: " + e.getMessage());
            System.out.println("Silakan coba lagi.\n");
        }
    }
}



Solver solver = new Solver(dt);
long startTime = System.currentTimeMillis();
boolean solved = solver.solve(0);
long endTime = System.currentTimeMillis();
long elapsed = endTime - startTime;

if (solved) {
    System.out.println("Solusi Ditemukan!");
    solver.printSol();
    System.out.println("Waktu: " + elapsed + " ms");
    System.out.println("Banyak iterasi: " + solver.getIteration());

    System.out.print("Simpan solusi ke file (ya/tidak)? ");
    Scanner sc = new Scanner(System.in);
    String ans = sc.nextLine();
    if (ans.equalsIgnoreCase("ya")) {
        System.out.print("Mau save di mana? ");
        String filename = sc.nextLine();
```

```
            solver.saveSol(filename, elapsed);
            System.out.println("Solusi di simpan di: " + "save/" + filename);
        }
    } else {
        System.out.println("Tidak ditemukan solusi.");
    }
  }
}
```

3.3 DataType.java

```
import java.util.ArrayList;
import java.util.List;

public class DataType {
  private int jumlahPiece;
  private int panjangPapan;
  private int lebarPapan;
  private String caseType;
  private char[][] board;
  private List<Piece> pieces;

  /* ========== CONSTRUCTOR ========= */
  public DataType(int jumlahPiece, int panjangPapan, int lebarPapan, String caseType) {
    this.jumlahPiece = jumlahPiece;
    this.panjangPapan = panjangPapan;
    this.lebarPapan = lebarPapan;
    this.caseType = caseType;
```

```
    this.board = new char[lebarPapan][panjangPapan];
    this.pieces = new ArrayList<>();
    setElmtBoard();
}

/* ========== Methods ========= */
public int getJumlahPiece() {
    return jumlahPiece;
}
public int getPanjangPapan() {
    return panjangPapan;
}
public int getLebarPapan() {
    return lebarPapan;
}
public String getCaseType() {
    return caseType;
}
public char[][] getBoard() {
    return board;
}

public void addPiece(Piece p) {
    this.pieces.add(p);
}

public List<Piece> getPieces() {
```

```
    return this.pieces;
  }

  public void setElmtBoard(){
    for (int i = 0; i < lebarPapan; i++){
      for (int j = 0; j < panjangPapan; j++){
        this.board[i][j] = ' ';
      }
    }
  }
}
```

3.4 Piece.java

```
import java.util.ArrayList;
import java.util.List;

public class Piece {
  private char id;
  private List<Point> coords;
  /* ========== CONSTRUCTOR ========= */
  public Piece(char id) {
    this.id = id;
    this.coords = new ArrayList<>();
  }

  /* ========== Getter ========= */
  public char getId() {
```

```
    return this.id;
  }

  public List<Point> getCoords() {
    return this.coords;
  }

  /* ========== Setter ========= */
  public void addCoord(int x, int y) {
    this.coords.add(new Point(x, y));
  }

  public void setcoord(List<Point> newCoord){
    this.coords = newCoord;
  }
}
```

3.5 Point.java

```
public class Point {

  /* ========== ATTRIBUTES ========== */
  private int X;
  private int Y;

  /* ============ GETTER ============ */
  public int getX() {
    return this.X;
```

```
    }

    public int getY() {
        return this.Y;
    }

    /* ========== CONSTRUCTOR ========= */
    public Point(int x, int y) {
        this.X = x;
        this.Y = y;
    }

    /* ============ Setter ============ */
    public void setX(int newX){
        this.X = newX;
    }

    public void setY(int newY){
        this.Y = newY;
    }
}
```

3.6 TestReader.java

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Scanner;

public class TestReader {
  /* ========== Methods ========= */
  public DataType readDataFile(String filePath) throws FileNotFoundException {
    Scanner sc = new Scanner(new File(filePath));
    List<String> lines = new ArrayList<>();
    while (sc.hasNextLine()) {
      lines.add(sc.nextLine());
    }
    sc.close();
    String[] firstLine = lines.get(0).split(" ");
    int lebarPapan = Integer.parseInt(firstLine[0]);
    int panjangPapan = Integer.parseInt(firstLine[1]);
    int jumlahPiece = Integer.parseInt(firstLine[2]);

    String caseType = lines.get(1);

    DataType dt = new DataType(jumlahPiece, panjangPapan, lebarPapan, caseType);

    int index = 2;
    for (int i = 0; i < jumlahPiece; i++) {
      PieceResult pr = readOnePiece(lines, index);
      dt.addPiece(pr.piece);
      index += pr.lineCount;
    }
```

```
        // System.out.println("M = " + lebarPapan + ", N = " + panjangPapan + ", P = " + jumlahPiece);
    // System.out.println("Case = " + caseType);
    // for (Piece pc : dt.getPieces()) {
            //         System.out.println("Piece " + pc.getId() + " coords = " + formatCoords(pc.getCoords()));
    // }

    return dt;
  }

  /* ========== CONSTRUCTOR Healper ========= */
  private static class PieceResult {
    Piece piece;
    int lineCount;

    public PieceResult(Piece piece, int lineCount) {
      this.piece = piece;
      this.lineCount = lineCount;
    }
  }

  private PieceResult readOnePiece(List<String> lines, int startIndex) {
    String firstLine = lines.get(startIndex);
    int fnsi = 0;
    while (fnsi < firstLine.length() && firstLine.charAt(fnsi) == ' ') {
      fnsi++;
    }
```

```
if (fnsi >= firstLine.length()) {
   return new PieceResult(new Piece(' '), 1);
}
char pieceId = firstLine.charAt(fnsi);

Piece piece = new Piece(pieceId);
int lineCount = 0, y = 0, idx = startIndex;

while (idx < lines.size()) {
   String line = lines.get(idx);
   if (line.trim().isEmpty()) {
      break;
   }
   int pos = 0;
   while (pos < line.length() && line.charAt(pos) == ' ') {
      pos++;
   }
   if (pos < line.length() && line.charAt(pos) == pieceId) {
      convertLineToCoords(line, piece, y);
      y++;
      idx++;
      lineCount++;
   } else {
      break;
   }
}
```

```
    return new PieceResult(piece, lineCount);
  }



  /* ========== Healper ========= */
  private void convertLineToCoords(String line, Piece piece, int y) {
    for (int x = 0; x < line.length(); x++) {
      char c = line.charAt(x);
      if (c != ' ') {
        piece.addCoord(x, y);
      }
    }
  }

  // private String formatCoords(List<Point> coords) {
  //    StringBuilder sb = new StringBuilder();
  //    sb.append("[");
  //    for (Point p : coords) {
  //       sb.append("(").append(p.getX()).append(",").append(p.getY()).append(") ");
  //    }
  //    sb.append("]");
  //    return sb.toString();
  // }


}
```

3.7 Solver.java

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

public class Solver {
   private DataType data;
   private long iteration = 0;
   private boolean solFound = false;

   /* ========== Constructor ========= */
   public Solver(DataType data){
      this.data = data;
   }

   /* ========== Getter ========= */
   public long getIteration(){
      return iteration;
   }

   public boolean  getsolFound(){
      return solFound;
   }
   /* ========== Methods ========= */
   /* ========== Print ========= */
```

```
public void printSol(){
   char[][] board = data.getBoard();
   int row = data.getLebarPapan();
   int col = data.getPanjangPapan();

   for (int i = 0; i < row; i++){
      for (int j = 0; j < col; j++){
         char pieceID = board[i][j];
         System.out.print(getColorForPiece(pieceID) + pieceID + "\u001B[0m ");
      }
      System.out.println();
   }
}
/* ========== Color ========= */
private String getColorForPiece(char pieceId) {
   switch (pieceId) {
      case 'A': return "\u001B[31m";
      case 'B': return "\u001B[32m";
      case 'C': return "\u001B[33m";
      case 'D': return "\u001B[34m";
      case 'E': return "\u001B[35m";
      case 'F': return "\u001B[36m";
      case 'G': return "\u001B[91m";
      case 'H': return "\u001B[92m";
      case 'I': return "\u001B[93m";
      case 'J': return "\u001B[94m";
      case 'K': return "\u001B[95m";
```

```
      case 'L': return "\u001B[96m";
      case 'M': return "\u001B[37m";
      case 'N': return "\u001B[90m";
      case 'O': return "\u001B[1;31m";
      case 'P': return "\u001B[1;32m";
      case 'Q': return "\u001B[1;33m";
      case 'R': return "\u001B[1;34m";
      case 'S': return "\u001B[1;35m";
      case 'T': return "\u001B[1;36m";
      case 'U': return "\u001B[1;91m";
      case 'V': return "\u001B[1;92m";
      case 'W': return "\u001B[1;93m";
      case 'X': return "\u001B[1;94m";
      case 'Y': return "\u001B[1;95m";
      case 'Z': return "\u001B[1;96m";
      default:  return "\u001B[0m";
    }
}
/* ========== Save ========= */
public void saveSol(String filename, long elapsed){
    if (!filename.endsWith(".txt")) {
      filename += ".txt";
    }
    String path = "save/" + filename;

    try (PrintWriter pw = new PrintWriter(new FileWriter(path))){
      char[][] board = data.getBoard();
```

```
      int row = data.getLebarPapan();
      int col = data.getPanjangPapan();

      for (int i = 0; i < row; i++){
         for (int j = 0; j < col; j++){
            pw.print(board[i][j]);
         }
         pw.println();
      }
   } catch(IOException e){
      e.printStackTrace();
   }

}
/* ========== Solve ========= */
public boolean solve(int pieceIndex){
   if (pieceIndex == data.getPieces().size()){
      if (isBoardFull()){
         solFound = true;
         return true;
      }
      return false;
   }

   Piece original = data.getPieces().get(pieceIndex);

   List<Piece> transform = transAll(original);
```

```
    for (Piece trans : transform){
        for (int i = 0; i < data.getLebarPapan(); i++){
            for (int j = 0; j < data.getPanjangPapan(); j++){
                if (canPlace(trans, i, j)){
                    iteration++;
                    placePiece(trans, i, j);
                    if (solve(pieceIndex + 1)){
                        return true;
                    }
                    removePiece(trans, i, j);
                }
            }
        }
    }
    return false;
}

private boolean isBoardFull(){
    char[][] board = data.getBoard();

    for (int i = 0; i < data.getLebarPapan(); i++){
        for (int j = 0; j < data.getPanjangPapan(); j++){
            if (board[i][j] == ' '){
                return false;
            }
        }
```

```
    }
    return true;
}

/* ========== Rotate ========= */
public void rotate90(Piece piece){
    List<Point> oldCoords = piece.getCoords();
    List<Point> newCoords = new ArrayList<>();

    for (Point p : oldCoords){
        int oldX = p.getX();
        int oldY = p.getY();

        int newX = oldY;
        int newY = -oldX;
        newCoords.add(new Point(newX, newY));
    }

    piece.setcoord(newCoords);
    normalize(piece);
}

public void rotate180(Piece piece){
    rotate90(piece);
    rotate90(piece);
}
```

```
public void rotate270(Piece piece){
   rotate180(piece);
   rotate90(piece);
}

/* ========== Mirror ========= */
public void mirrorHorizontal(Piece piece){
   List<Point> oldCoords = piece.getCoords();
   List<Point> newCoords = new ArrayList<>();

   for (Point p : oldCoords){
      int oldX = p.getX();
      int oldY = p.getY();

      int newX = -oldX;
      int newY = oldY;
      newCoords.add(new Point(newX, newY));
   }

   piece.setcoord(newCoords);
   normalize(piece);
}

public void mirrorVertical(Piece piece){
   List<Point> oldCoords = piece.getCoords();
   List<Point> newCoords = new ArrayList<>();
```

```
    for (Point p : oldCoords){
        int oldX = p.getX();
        int oldY = p.getY();

        int newX = oldX;
        int newY = -oldY;
        newCoords.add(new Point(newX, newY));
    }

    piece.setcoord(newCoords);
    normalize(piece);
}

/* ========== Normalize ========= */

private void normalize(Piece piece){
    List<Point> Coords = piece.getCoords();

    int minX = Integer.MAX_VALUE;
    int minY = Integer.MAX_VALUE;

    for (Point p : Coords){
        if (p.getX() < minX){
            minX = p.getX();
        }
        if (p.getY() < minY){
            minY = p.getY();
```

```
        }
    }

    for (Point p : Coords){
        p.setX(p.getX() - minX);
        p.setY(p.getY() - minY);
    }
}

/* ========== Placing ========= */
public boolean canPlace(Piece piece, int i, int j){
    int row = data.getLebarPapan();
    int col = data.getPanjangPapan();
    char[][] board = data.getBoard();

    for (Point coord : piece.getCoords()){
        int coordX = coord.getX();
        int coordY = coord.getY();

        int boardX = coordX + i;
        int boardY = coordY + j;

        if (boardX < 0 || boardX >= row || boardY < 0 || boardY >= col){
            return false;
        }

        if (board[boardX][boardY] != '\u0000' && board[boardX][boardY] != ' '){
```

```
            return false;
        }
    }

    return true;
}

public void placePiece(Piece piece, int i, int j){
    char charID = piece.getId();
    char[][] board = data.getBoard();

    for(Point coord : piece.getCoords()){
        int coordX = coord.getX();
        int coordY = coord.getY();

        int boardX = coordX + i;
        int boardY = coordY + j;

        board[boardX][boardY] = charID;
    }
}

public void removePiece(Piece piece, int i, int j){
    char[][] board = data.getBoard();

    for (Point coord : piece.getCoords()){
        int coordX = coord.getX();
```

```
        int coordY = coord.getY();

        int boardX = coordX + i;
        int boardY = coordY + j;
        board[boardX][boardY] = ' ';
    }
}

/* ========== Healper ========= */
public List<Piece> transAll(Piece original){
    List<Piece> result = new ArrayList<>();

    Piece rot0 = copyPiece(original);
    normalize(rot0);
    result.add(rot0);

    Piece rot90 = copyPiece(original);
    rotate90(rot90);
    result.add(rot90);

    Piece rot180 = copyPiece(original);
    rotate180(rot180);
    result.add(rot180);

    Piece rot270 = copyPiece(original);
    rotate270(rot270);
    result.add(rot270);
```

```
    Piece mirrHorizontal = copyPiece(original);
    mirrorHorizontal(mirrHorizontal);
    result.add(mirrHorizontal);

    Piece mirrVertical = copyPiece(original);
    mirrorVertical(mirrVertical);
    result.add(mirrVertical);

    return result;
  }

  public Piece copyPiece(Piece original){
    Piece copy = new Piece(original.getId());

    for (Point p : original.getCoords()){
      copy.addCoord(p.getX(), p.getY());
    }

    return copy;
  }
}
```

BAB IV

MASUKAN DAN LUARAN PROGRAM

4.1 Test_1

Masukkan:

5 5 7

DEFAULT

A

AA

B

BB

C

CC

D

DD

EE

EE

E

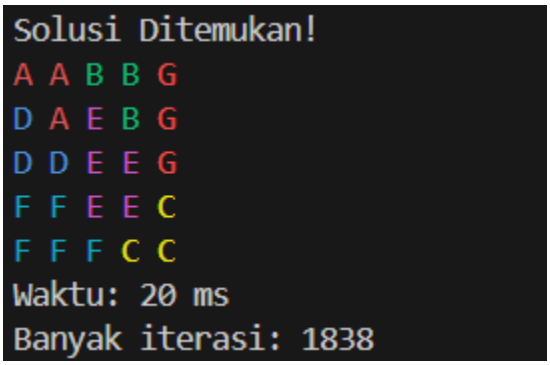
FF

FF

F

GGG

Keluaran:



4.2 Test_2

Masukkan:

5 5 8

DEFAULT

A

AA

B

BB

C

CC

D

DD

EE

EE

E

FF

FF

F

GGG

HHH

Keluaran:

```
Masukkan nama file: test_2.txt
Tidak ditemukan solusi.
```

4.3 test_3

Masukkan:

5 11 12

DEFAULT

CC

C

CC

W

WW

WW

L

LLLL

Z

ZZ

Z

Z

C

C

CCC

TTT

T

M

MMMM

G

GG

GG

QQ

QQ

BB

B

PP

PPP

N

NNN

Keluaran:

4.4 test_4

Masukkan:

3 10 8

DEFAULT

A

A

AA

BB

BB

B

B

B

B

C

DD

D

E

E

E

FF

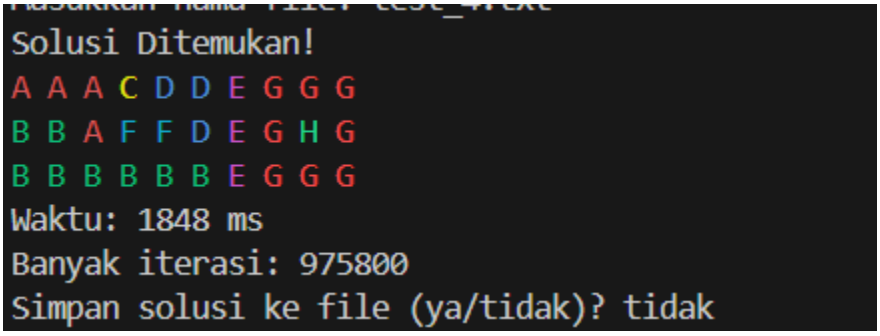
GGG

G G

GGG

H

Keluaran:



4.5 test_5

Masukan:

3 10 8

DEFAULT

A

A

AA

BB

BB

B

B

B

B

C

DD

D

E

E

E

FF

GGG

G G

GGG

H

Keluaran:

```
Masukkan nama file: test_5.txt
Solusi Ditemukan!
A A A C D D E G G G
B B A F F D E G H G
B B B B B B E G G G
Waktu: 1883 ms
Banyak iterasi: 975800
Simpan solusi ke file (ya/tidak)? tidak
```

4.6 test_6

Masukkan:

5 11 12

DEFAULT

A

A

AA

A

B B

BBB

CCCC

C

D

DD

DD

E

EEE

E

F

FFFF

GGG

G

HHH

HH

I

II

I

JJ

J

J

KK

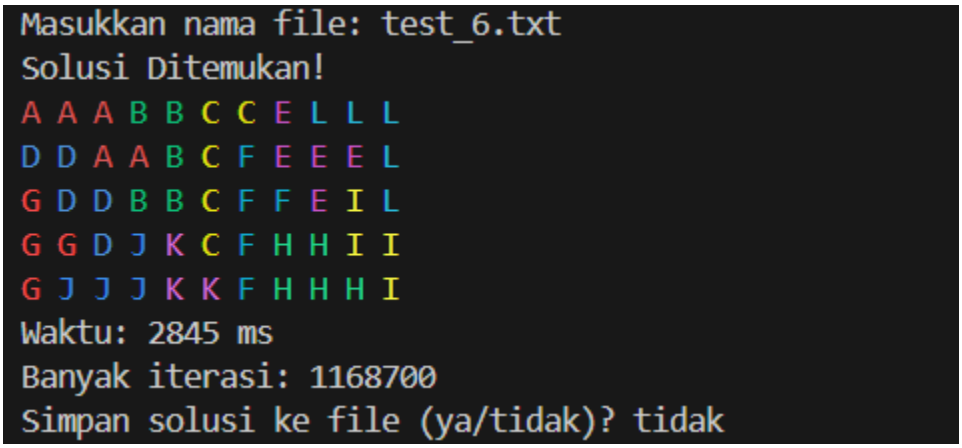
K

L

L

LLL

Keluaran:



4.7 test_7

Masukkan:

3 5 7

DEFAULT

ZZ

Z

Z

F

FF

I

L

LL

L

A

A

B

JJ

Keluaran:

```
Masukkan nama file: test_7.txt
Tidak ditemukan solusi.
```

BAB V

LAMPIRAN

No	**Poin**	**Ya**	**Tidak**
1	Program berhasil dikompilasi tanpa kesalahan	✓	
2	Program berhasil dijalankan	✓	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	✓	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	✓	
5	Program memiliki *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		✓