LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Penyelesaian game iq puzzle dengan algoritma *bruteforce*

Disusun Oleh:

Aryo Bama Wiratama (13523088)

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Tahun 2025

A. Algoritma Bruteforce

Program ini menggunakan algoritma *bruteforce* untuk menemukan solusi dalam gim iq puzzler. Algoritma *bruteforce* adalah algoritma yang memiliki pendekatan yang *straighforward* dalam memecahkan masalah. Oleh karena itu, algoritma ini cenderung simpel dan mudah dimengerti. Akan tetapi, algoritma ini tidak efisien dan cenderung membutuhkan waktu komputasi yang lama.

Berikut penjelasan algoritma bruteforce dalam mencari solusi pada gim iq puzzler:

1. Misalkan terdapat P buah piece dan papan berukuran n x m
2. Susunlah piece tersebut. Penyusunan piece tidak berdasarkan syarat apapun.
3. Ambil satu piece kemudian letakkan pada papan mulai dari titik O (0,0).
4. Kemudian ambil piece selanjutnya. Coba letakkan piece dalam papan dimulai dari titik O. Apabila suatu titik telah ditempati oleh piece lain, maju ke titik selanjutnya.
5. Lakukan hal no.4 hingga mencapai piece terakhir.
6. Apabila dalam proses no 4 dan 5 terdapat piece yang tidak menemukan tempat kosong pada papan, coba lakukan semua gerakan rotasi dan refleksi.
7. Apabila langkah no 6 juga tidak berhasil, lakukan backtracking dengan cara melakukan rotasi dan refleksi pada piece sebelumnya. Kemudian letakkan kembali piece yang gagal ditempatkan.
8. Ulangi langkah 1-7 hingga semua kemungkinan telah dicoba.

Jika terdapat satu solusi yang ditemukan, pencarian tidak usah dilanjutkan.

Dalam kasus terburuk, algoritma ini memiliki kompleksitas waktu T((n x m x 8)^P) atau O((n x m)^P). Piece ke-I memiliki kemungkinan penempatan sebanyak m x n dan di setiap penempatan terdapat 8 gerakan sehingga setiap piece memilik 8 x n x m kemungkinan. Apabila terdapat P piece, berdasakan aturan perkalian, total kemungkinan dapat dhitung dari perkalian (m x n x 8) sebanyak P kali atau dapat ditulis ((n x m x 8)^P.

Berikut adalah pseudocode dari algoritma bruteforce yang digunakan.

```
FUNCTION allSolution(solutions, grid, allPieces, start, end)
    IF start > end THEN
        IF grid.isSolve() AND NOT existSolution(solutions, grid.grid) THEN
            CREATE copy as a 2D character array of grid dimensions
            found ← TRUE
            Utility.copyMatrix(grid.grid, copy)
            ADD copy to solutions
        END IF
    ELSE
        FOR i FROM 0 TO grid.n - 1 DO
            FOR j FROM 0 TO grid.m - 1 DO
                IF found THEN
                    BREAK

                FOR k FROM 0 TO 3 DO  // Try 4 rotations
                    IF grid.canPlace(allPieces[start], j, i) THEN
                        grid.placePiece(allPieces[start], j, i)
                        counter ← counter + 1
                        alphabet ← allPieces[start].alphabet
                        allSolution(solutions, grid, allPieces, start + 1, end)
                        grid.unplacePiece(alphabet)
                    END IF
                    allPieces[start].quarterRotate()
                END FOR

                allPieces[start].reflection()

                FOR k FROM 0 TO 3 DO  // Try 4 more rotations after reflection
                    IF grid.canPlace(allPieces[start], j, i) THEN
                        grid.placePiece(allPieces[start], j, i)
                        counter ← counter + 1
                        alphabet ← allPieces[start].alphabet
                        allSolution(solutions, grid, allPieces, start + 1, end)
                        grid.unplacePiece(alphabet)
                    END IF
                    allPieces[start].quarterRotate()
                END FOR

                allPieces[start].reflection()  // Reset piece to original orientation
            END FOR
        END FOR
    END IF
END FUNCTION
```

B. Source Code

Projek ini ditulis dalam bahasa java menggunakan library sebagai berikut:

1. java.util
2. java.io

Projek ini terbagi menjadi beberapa file, yaitu:

1. Main.java
2. Point.java
3. Piece.java
4. Grid.java
5. Algorithm.java
6. ReadFile.java
7. WriteFile.java
8. Mapping.java

9. Utility.java

   Berikut source codenya:

   1. Main.java

```java
package src;

import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println(x:"Selamat datang di IQ puzzler solver");
        boolean done = false;
        while (true){
            String fileName;

            do {
                System.out.print(s:"Masukkan nama file: ");
                fileName = scanner.nextLine();
                if(ReadFile.isExistFile(fileName)){
                    System.out.println(x:"Mencari solusi...");
                    break;
                }
                System.out.println(x:"File tidak ditemukan. Letakkan file pada folder IO");

            } while (true);

            ArrayList<Piece> allPieces = new ArrayList<>();
            Grid board = ReadFile.readCase(fileName, allPieces);
            int counter;
            ArrayList<char[][]> solutions = new ArrayList<>();

            long startTime = System.currentTimeMillis();

            Algorithm.allSolution(solutions, board, allPieces, start:0, allPieces.size()-1);

            long endTime = System.currentTimeMillis();
```

```java
            counter = Algorithm.counter;

            if (solutions.isEmpty()){
                System.out.println(x:"No solution found");
            }else{
                char[][] solution = solutions.get(index:0);
                System.out.println(x:"Solusi ditemukan !");
                for (int i = 0; i < solution.length; i++){
                    for (int j = 0; j < solution[0].length; j++){
                        System.out.print(Mapping.findColor(solution[i][j]) + solution[i][j] + Mapping.RESET);
                    }
                    System.out.println(x:"");
                }
            }
            System.out.println(x:"");

            long elapsedTime = endTime - startTime;
            System.out.println("Waktu eksekusi: " + elapsedTime + " ms" + "\n");
            System.out.println("Jumlah iterasi: " + counter + "\n");

            String choice;

            while (true){
                System.out.print(s:"Apakah anda ingin menyimpan solusi? (Ya/Tidak): ");
                choice = scanner.nextLine();
                String ya = "YA";
                String tidak = "TIDAK";
                if(ya.equals(choice.toUpperCase()) ){
                    System.out.print(s:"Masukkan nama file: ");
                    fileName = scanner.nextLine();
                    WriteFile.WriteSolution(fileName, solutions, elapsedTime, counter);
                    done = true;
                    break;
                }else if(tidak.equals(choice.toUpperCase())){
```


System.out.println(x:"Terimakasih sudah menggunakan solver ini");
done = true;
break;
}else{
System.out.println(x:"Pilihan tidak valid");
}
}
if (done){
break;
}
}
scanner.close();
}
}

2. Point.java

```java
package src;

public class Point{
    private int x;
    private int y;

    public Point(int x, int y){
        this.x = x;
        this.y = y;
    }
    public int getX(){
        return this.x;
    }

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

    public void setX(int newX){
        this.x = newX;
    }

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


    public void translation(int dx, int dy){
        this.x += dx;
        this.y += dy;
    }

    public boolean isNeighborPoint(Point targetPoint){
        return Math.abs(this.x - targetPoint.getX()) == 1
        || Math.abs(this.x - targetPoint.getY()) == 1
        || Math.abs(this.y - targetPoint.getX()) == 1
        || Math.abs(this.y - targetPoint.getY()) == 1;
    }
```

3. Piece.java

```java
package src;

import java.util.ArrayList;

public class Piece {
    ArrayList<Point> piece = new ArrayList<>();
    char alphabet;

    public Piece(char alphabet){
        this.alphabet = alphabet;
    }

    public char getAlphabet(){
        return this.alphabet;
    }

    public void addPoint(int x, int y){
        this.piece.add(new Point (x, y));
    }

    public void quarterRotate(){
        if (this.piece.size() > 1){
            int centerX = this.piece.get(index:0).getX();
            int centerY = this.piece.get(index:0).getY();

            for (Point p : piece){
                int temp = p.getX();
                p.setX( -(p.getY() - centerY) + centerX);
                p.setY( (temp - centerX) + centerY);
            }
        }
    }
```

```
public void reflection(){
    if (this.piece.size() > 1){
        int centerY = this.piece.get(index:0).getY();

        for (Point p : piece){
            p.setY(2*centerY - p.getY());
        }
    }
}

public void display(){
    if (this.piece.size() >= 1){

        int xLeft = this.piece.get(index:0).getX();
        int yLeft = this.piece.get(index:0).getY();

        for (Point p : this.piece) {
            if (p.getX() < xLeft) {
                xLeft = p.getX();
            }
            if (p.getY() < yLeft) {
                yLeft = p.getY();
            }
        }
        int offsetX =  -xLeft;
        int offsetY =  -yLeft;



        int maxX = this.piece.get(index:0).getX() + offsetX;
        int maxY = this.piece.get(index:0).getY() + offsetY;

        for (int i = 1; i < this.piece.size(); i++){
            if (maxX < this.piece.get(i).getX()  + offsetX){
                maxX = this.piece.get(i).getX()  + offsetX;
            }

            if (maxY < this.piece.get(i).getY() + offsetY){
                maxY = this.piece.get(i).getY() + offsetY;
            }
        }


        boolean[][] grid = new boolean[maxX+1][maxY+1];

        for (Point point: piece){
            grid[point.getX() + offsetX][point.getY() + offsetY] = true;
        }

        for (int i = 0; i < maxX+1; i++){
            for (int j = 0; j < maxY+1; j++){
                if (grid[i][j] == true){
                    System.out.print(Mapping.findColor(this.alphabet) + this.alphabet + Mapping.RESET);
                }else{
                    System.out.print(s:"");
                }
            }
            System.out.println(x:"");
        }
    }
    else{
        System.out.println(x:"Piece undefined");
    }
}
```

```java
public void centralization(){
    boolean point0 = this.piece.get(index:0).getX() == 0 && this.piece.get(index:0).getY() == 0;

    if (this.piece.size() >= 1 && !point0){
        int offsetX, offsetY;
        offsetX = this.piece.get(index:0).getX();
        offsetY = this.piece.get(index:0).getY();

        for (Point point: piece){
            point.setX(point.getX() - offsetX);
            point.setY(point.getY() - offsetY);
        }
    }
}

public boolean isValidPiece(){
    for (Point point: this.piece){
        boolean hasNeighbor = false;

        for (Point point2: this.piece){
            if (point.isNeighborPoint(point2)){
                hasNeighbor = true;
                break;
            }
        }
        if (!hasNeighbor){
            return false;
        }
    }
    return true;
}
}
```

4. Grid.java

```java
package src;

import java.util.ArrayList;

public class Grid {
    public int n;
    public int m;
    public char[][] grid = null;
    ArrayList<Piece> onPieces = new ArrayList<>();
    ArrayList<Piece> offPieces = new ArrayList<>();
    String type;

    public Grid (int n, int m){
        this.n = n;
        this.m = m;
        this.grid = new char[n][m];
        for (int i = 0; i < n; i++){
            for (int j = 0; j < m; j++){
                this.grid[i][j] = '.';
            }
        }
    }

    public void addPiece(Piece piece){
        if (piece.piece.size() < 1){
            System.out.println(x:"Sorry can't add empty piece");
            return;
        }
        if (isPieceExist(piece)){
            System.out.printf(format:"Sorry piece %s already added in this grid. Failed to add piece", Mapping.findColor(piece.alphabet) + piece.alphabet + M
            return;
        }
        this.offPieces.add(piece);
    }
```

```java
public Piece onPieces(int idx){
    return this.onPieces.get(idx);
}

public Piece offPieces(int idx){
    return this.offPieces.get(idx);
}

public boolean canPlace(Piece p, int offsetX, int offsetY){
    for (Point point: p.piece){
        int x = point.getX()+offsetX;
        int y = point.getY()+offsetY;
        if (x < 0 || x >= this.m || y < 0 || y >= this.n || grid[y][x] != '.'){
            return false;
        }
    }
    return true;
}

public void placePiece(Piece p, int offsetX, int offsetY){

    if (!canPlace(p, offsetX, offsetY)){
        System.out.println(x:"Sorry u can't place the piece in this condition");
        return;
    }
    for (Point point: p.piece){
        point.setX(point.getX()+offsetX);
        point.setY(point.getY()+offsetY);
    }
```

```java
    for (Point point: p.piece){
        int x = point.getX();
        int y = point.getY();
        this.grid[y][x] = p.getAlphabet();
    }
    int idx = -1;
    for (Piece piece: offPieces){
        idx++;
        if (piece.alphabet == p.alphabet){
            break;
        }
    }
    if (idx != -1){
        this.offPieces.remove(idx);
    }
    this.onPieces.add(p);
}

public void placePiece(Piece p){ // default condition
    placePiece(p,offsetX:0,offsetY:0);
}
```

```java
public void unplacePiece(char alphabet){
    int idx = -1;
    for (Piece p: onPieces){
        idx++;

        if (p.alphabet == alphabet){
            int centerX = p.piece.get(index:0).getX();
            int centerY = p.piece.get(index:0).getY();
            for (Point point: p.piece){
                grid[point.getY()][point.getX()] = '.';
                point.setY(point.getY()-centerY);
                point.setX(point.getX()-centerX);
            }
            this.offPieces.add(p);
            break;
        }
    }
    if (idx != -1) {
        this.onPieces.remove(idx);
    }
}

public void quarterRotate(char alphabet){
    for (Piece piece: offPieces){
        if (piece.alphabet == alphabet){
            for(Point point: piece.piece){
                grid[point.getX()][point.getY()] = '.';
            }
            piece.quarterRotate();
            return;
        }
    }
    System.out.printf(format:"Piece '%c' not in off pieces list\n", alphabet);
}
```

```java
public void reflection(char alphabet){
    for (Piece piece: offPieces){
        if (piece.alphabet == alphabet){
            for(Point point: piece.piece){
                grid[point.getX()][point.getY()] = '.';
            }
            piece.reflection();
            return;
        }
    }
    System.out.printf(format:"Piece '%c' is not in off pieces list\n", alphabet);
}

public boolean isSolve(){
    if (!this.offPieces.isEmpty()){
        return false;
    }

    for (int i = 0; i < this.n; i++){
        for (int j = 0; j < this.m; j++){
            if (grid[i][j] == '.'){
                return false;
            }
        }
    }
    return true;
}

public boolean isPieceExist(Piece piece){
    for (Piece p: offPieces){
        if (p.alphabet == piece.alphabet){
            return true;
        }
    }
```

```java
        }
        return false;
    }

    public void displayGrid(){
        for (int i = 0; i < n; i++){
            for (int j = 0; j < m; j++){
                System.out.print(Mapping.findColor(grid[i][j]) + grid[i][j] + Mapping.RESET);
                System.out.print(s:" ");
            }
            System.out.println(x:"");
        }
    }
}
```

5. Algorithm.java

```java
package src;

import java.util.ArrayList;

public class Algorithm {
    public static int counter;
    public static boolean found = false;

    public static boolean existSolution(ArrayList<char[][]> solutions, char[][] solution){
        for (char[][] s: solutions){
            if(Utility.isEqualMatrix(s, solution)){
                return true;
            }
        }

        return false;
    }

    public static void allSolution(ArrayList<char[][]> solutions, Grid grid, ArrayList<Piece> allPieces, int start, int end){

        if (start > end){

            if (grid.isSolve() && !existSolution(solutions, grid.grid)){
                char[][] copy = new char[grid.grid.length][grid.grid[0].length];
                found = true;
                Utility.copyMatrix(grid.grid, copy);
                solutions.add(copy);
            }

        }else{
            for (int i = 0; i < grid.n; i ++){
                for (int j = 0; j < grid.m; j++)
                {
                    if (found){
                        break;
                    }
                    for (int k = 0; k < 4; k++){
                        if(grid.canPlace(allPieces.get(start),j,i)){
                            grid.placePiece(allPieces.get(start),j,i);
                            counter++;
                            char alphabet = allPieces.get(start).alphabet;
                            allSolution(solutions, grid, allPieces, start + 1,  end);
                            grid.unplacePiece(alphabet);
                        }
                        allPieces.get(start).quarterRotate();
                    }

                    allPieces.get(start).reflection();

                    for (int k = 0; k < 4; k++){
                        if(grid.canPlace(allPieces.get(start),j,i)){
                            grid.placePiece(allPieces.get(start),j,i);
                            counter++;
                            char alphabet = allPieces.get(start).alphabet;
                            allSolution(solutions, grid, allPieces, start + 1,  end);
                            grid.unplacePiece(alphabet);
                        }
                        allPieces.get(start).quarterRotate();
                    }
                    allPieces.get(start).reflection();

                }
            }
        }
    }
}
```

6. ReadFile.java

```java
package src;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class ReadFile {

    public static boolean isAllSpace(String line){
        for (int i = 0; i < line.length(); i++){
            if (line.charAt(i) != ' '){
                return false;
            }
        }
        return true;
    }

    public static boolean isValidRowPiece(String line){
        if (!line.isEmpty()){
            char alphabet='a';

            for (int i = 0; i < line.length(); i++){
                boolean isAlphabet = (Character.toUpperCase(line.charAt(i)) >= 'A' &&  Character.toUpperCase(line.charAt(i)) <= 'Z');
                if(isAlphabet){
                    alphabet = line.charAt(i);
                }
            }

            for (int i = 0; i < line.length(); i++){
                boolean isAlphabet = (Character.toUpperCase(line.charAt(i)) >= 'A' &&  Character.toUpperCase(line.charAt(i)) <= 'Z');
                if (isAlphabet){
                    if(alphabet != line.charAt(i)){
                        return false;
                    }
                }
            }
            return true;
        }
        return false;
    }

    public static Grid readCase(String fileName, ArrayList<Piece> allPieces){
        fileName = "input/" + fileName;
        File file = new File(fileName);
        Grid grid = null;

        try(Scanner scanner = new Scanner(file)) {
            int nPiece = 0;
            if (scanner.hasNextLine()){
                int n = scanner.nextInt();
                int m = scanner.nextInt();
                grid = new Grid(n,m);
                nPiece = scanner.nextInt();
            }
            if (scanner.hasNextLine()){
                grid.type = scanner.nextLine();
                grid.type = scanner.nextLine();

            }

            boolean readPiece = false;
            Piece newPiece = null;
            char currentAlphabet = '!';// inisalisasi ajah
            int y = 0;
            String line = "";

            while (nPiece != 0){
                if (scanner.hasNextLine()){
                    line = scanner.nextLine();
                }
```

```
Loading...
        if(!isValidRowPiece(line) || isAllSpace(line) || line.isEmpty()){ // kalau dalam satu baris ga valid
            System.out.println();
            System.out.println(x:"found invalid line. Failed add line");

            readPiece = false;
            continue;

        }

        for(int i = 0; i < line.length(); i++){
            if((line.charAt(i) != ' ' && currentAlphabet != line.charAt(i)) || !scanner.hasNextLine()){
                currentAlphabet = line.charAt(i);
                if (scanner.hasNextLine() || currentAlphabet != newPiece.alphabet){
                    readPiece = false;
                }

                if (newPiece != null){

                    if (!grid.isPieceExist(newPiece)){


                        newPiece.centralization();
                        allPieces.add(newPiece);
                        grid.addPiece(newPiece);
                        nPiece--;
                        break;

                    }else{
                        System.out.println(x:"Piece already in grid");
                    }
                }

            }
```

```
            }
            if (nPiece == 0 && !readPiece){
                break;
            }


            if (!readPiece){ // kalau masih belum masukin piece atau transisi ke piece yang beda
                y = 0;
                newPiece = new Piece(Character.toUpperCase(currentAlphabet));

                if (scanner.hasNextLine()){
                    readPiece = true;
                }
                for(int i = 0; i < line.length(); i++){
                    if(line.charAt(i) != ' '){
                        newPiece.addPoint(y, i);
                    }
                }
            }else{
                for(int i = 0; i < line.length(); i++){
                    if(line.charAt(i) != ' '){
                        newPiece.addPoint(y, i);
                    }
                }
            }
            y++;
        }
    } catch (FileNotFoundException e) {
        System.out.println(x:"File not found");
    }
    return grid;

}
```

7. WriteFile.java

```java
package src;

import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

public class WriteFile {
    public static void WriteSolution(String fileName, ArrayList<char[][]> solution,long time, int iteration){

        try {
            fileName = "input/" + fileName;
            FileWriter writer = new FileWriter(fileName);
            if(solution.isEmpty()){
                writer.write(str:"No Solution Found!\n");
            }else{
                for (int i = 0; i < solution.get(index:0).length; i++){
                    for (int j = 0; j < solution.get(index:0)[0].length; j++){
                        writer.write(solution.get(index:0)[i][j]);
                    }
                    writer.write(str:"\n");
                }
            }
            writer.write("Waktu eksekusi: " + Long.toString(time) + "\n");
            writer.write("Banyak iterasi: " + Integer.toString(iteration) + "\n");
            writer.close();
            System.out.println(x:"Berhasil menulis solusi.");

        } catch (IOException e) {
            System.out.println(x:"Terjadi kesalahan.");
            e.printStackTrace();
        }
    }
}
```

import java.util.Scanner;

8. Mapping.java

```java
package src;

import java.util.List;

public class Mapping {
    public static List<String> colors = List.of(
        ...elements:"\u001B[31m", "\u001B[32m", "\u001B[33m", "\u001B[34m",
        "\u001B[35m", "\u001B[36m", "\u001B[91m", "\u001B[92m",
        "\u001B[93m", "\u001B[94m", "\u001B[95m", "\u001B[96m",
        "\u001B[41m", "\u001B[42m", "\u001B[43m", "\u001B[44m",
        "\u001B[45m", "\u001B[46m", "\u001B[101m", "\u001B[102m",
        "\u001B[103m", "\u001B[104m", "\u001B[105m", "\u001B[106m",
        "\u001B[41;93m", "\u001B[44;96m"
    );
    public static String RESET = "\u001B[0m";

    public static char[] alphabets = {
        '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'
    };

    public static String findColor(char alphabet){
        for (int i = 0; i < alphabets.length; i++ ){
            if (Character.toUpperCase(alphabet)== alphabets[i]){
                return colors.get(i);
            }
        }
        return RESET;
    }
}
```

C. Testing

1. Input:

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

Output:

```
Mencari solusi...
Solusi ditemukan !
AABBG
CADBG
CCDDG
EEEFF
EEFFF

Waktu eksekusi: 2 ms

Jumlah iterasi: 512

Apakah anda ingin menyimpan solusi? (Ya/Tidak):
```

2. Input:

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

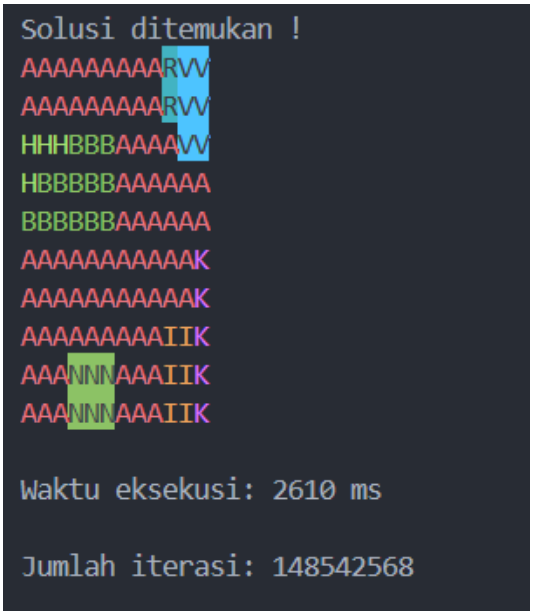
Output:

```
Mencari solusi...
No solution found

Waktu eksekusi: 663553 ms

Jumlah iterasi: 24591510600
```

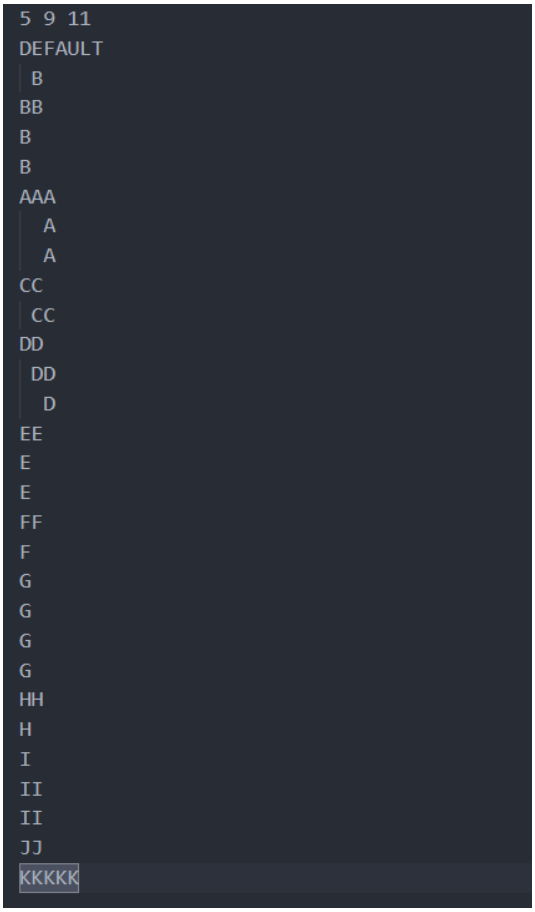
3. Input:

```
10 12 8
DEFAULT
AAAAAAAAA
AAAAAAAAA
      AAAA
      AAAAAA
      AAAAAA
AAAAAAAAAAA
AAAAAAAAAAA
AAAAAAAAA
AAA   AAA
AAA   AAA
HHH
H
   BBB
 BBBBB
BBBBBB
RR
KKKKK
VVV
VVV
III
III
NNN
NNN
```

Output:



4. Input:



Output:

```
Masukkan nama file: input.txt
Mencari solusi...
Solusi ditemukan !
BAAACFFHH
BBEACCFHG
JBEADCIIG
JBEEDDIIG
KKKKKDDIG

Waktu eksekusi: 46152 ms

Jumlah iterasi: 3132247432
```

5. Input:

```
5 5 7
DEFAULT
 B
BB
AA
 AA
 C
CCC
DD
DD
E
E
F
F
FF
I
I
II
```

Output:

```
Solusi ditemukan !
FBEEA
FBBAA
FFIAC
DDICC
DDIIC

Waktu eksekusi: 16293 ms

Jumlah iterasi: 684670072
```

6. Input:

```
5 5 3
DEFAULT
AAAAA
A   A
    A
  A A
  AAA
B
BB
 CCC
CCCC
 C C
```

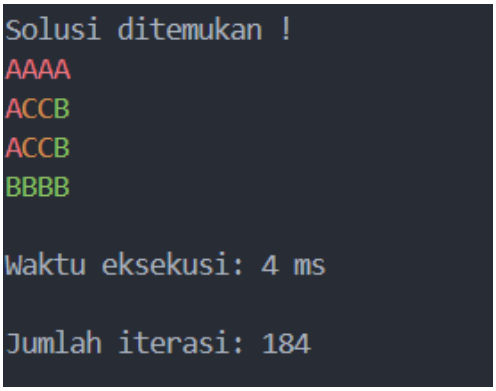
Output:

```
Solusi ditemukan !
AACBB
ACCCB
ACCAA
ACCCA
AAAAA

Waktu eksekusi: 4 ms

Jumlah iterasi: 696
```

7. Input:

```
4 4 3
DEFAULT
AAAA
A
A
BBBB
B
B
CC
CC
```

Output:



D. Link Repository

https://github.com/AryoBama/Tucil1_13523088

E. Checklist

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 menyelesaikankasus konfigurasi custom		✓
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		✓
9	Program dibuat oleh saya sendiri	✓	