Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II tahun 2024/2025
Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Mayla Yaffa Ludmilla - 13523050

I. Deskripsi program

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.

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.

Program 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.

II. Algoritma Brute Force

```
public boolean solve() {
        if (pieces.isEmpty()) {
```

```
            return boardFull();
        }

        Block currentPiece = pieces.remove(0);

        for (int i = 0; i < boardRow; i++) {
            for (int j = 0; j < boardCol; j++) {
                for (int k = 0; k < 4; k++) {
                    if (canPutPiece(currentPiece, i, j)) {
                        count++;
                        putPiece(currentPiece, i, j,
getPieceLetter(currentPiece));
                        if (solve()) {
                            return true;
                        }
                        removePiece(currentPiece, i, j);
                    }
                    currentPiece.rotate(currentPiece);
                }

                currentPiece.mirrorX(currentPiece);
                for (int k = 0; k < 4; k++) {
                    if (canPutPiece(currentPiece, i, j)) {
                        count++;
                        putPiece(currentPiece, i, j,
getPieceLetter(currentPiece));
                        if (solve()) {
                            return true;
                        }
                        removePiece(currentPiece, i, j);
                    }
                    currentPiece.rotate(currentPiece);
                }
            }
        }

        pieces.add(0, currentPiece);
        return false;
    }
```

Algoritma brute force yang saya gunakan dalam program ini menggunakan pendekatan rekursif. Fungsi dibuat sebagai boolean untuk menentukan apakah solusi untuk input ditemukan atau tidak (jika solusi ditemukan, fungsi mengembalikan true, dan sebaliknya).

```
         if (pieces.isEmpty()) {
            return boardFull();
        }
```

Basis rekursi mengecek apakah masih ada blok/piece yang dapat diletakkan ke board/papan. Jika sudah tidak ada lagi piece yang tersedia, maka fungsi boardFull() dipanggil untuk mengecek apakah board sudah penuh. Jika board penuh, maka ada solusi dan mengembalikan true, dan sebaliknya.

```
        Block currentPiece = pieces.remove(0);
```

Baris ini mengambil 1 piece dari list piece yang ada dan menyimpannya di currentPiece. currentPiece ini akan menjadi piece yang dicoba untuk diletakkan ke atas board.

```
        for (int i = 0; i < boardRow; i++) {
            for (int j = 0; j < boardCol; j++) {
        }
```

Bagian ini mengiterasi papan sesuai banyaknya row/baris dan column/kolom dari board. Setiap petak di board akan dicek.

```
                for (int k = 0; k < 4; k++) {
                    if (canPutPiece(currentPiece, i, j)) {
                        count++;
                        putPiece(currentPiece, i, j,
getPieceLetter(currentPiece));
                        if (solve()) {
                            return true;
                        }
                        removePiece(currentPiece, i, j);
                    }
                    currentPiece.rotate(currentPiece);
```

```
                }
```

Bagian ini adalah bagian rekursi untuk piece yang diputar/rotate. Piece akan diputar sebanyak 4 kali, masing masing sebesar 90 derajat. Setiap posisi dicek apakah memungkinkan untuk meletakkan piece dengan posisi tersebut di papan menggunakan fungsi canPutPiece(). Apabila canPutPiece() mengembalikan true maka piece dapat diletakkan di board menggunakan fungsi putPiece(). Variabel count sebagai penghitung berapa kasus yang ditinjau ditambah setiap kali block dapat diletakkan. Fungsi solve() dipanggil lagi untuk melanjutkan ke piece berikutnya (rekursi). Jika rekursi menghasilkan solusi, maka fungsi mengembalikan true. Jika tidak, piece diambil lagi dari posisinya menggunakan fungsi removePiece(). Jika piece tidak dapat diletakkan ke board, maka piece diputar dan dicoba lagi untuk diletakkan ke atas board.

```
                currentPiece.mirrorX(currentPiece);
                for (int k = 0; k < 4; k++) {
                    if (canPutPiece(currentPiece, i, j)) {
                        count++;
                        putPiece(currentPiece, i, j,
getPieceLetter(currentPiece));
                        if (solve()) {
                            return true;
                        }
                        removePiece(currentPiece, i, j);
                    }
                    currentPiece.rotate(currentPiece);
                }
            }
        }
```

Bagian ini sama dengan bagian sebelumnya, tetapi sebelum dilakukan proses iterasi, currentPiece dicerminkan terlebih dahulu untuk mencari semua kemungkinan posisi dari piece.

```
pieces.add(0, currentPiece);
        return false;
    }
```

Jika tidak ada posisi dari piece yang dapat diletakkan ke atas board, maka piece dimasukkan lagi ke dalam list piece lalu fungsi mengembalikan nilai false.

III. Source Code

Program Main.java

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        System.err.println("-+-+-+-+-+-+-+-+-+-+-+-+-+-+-");
        System.err.println("  IQ Puzzler Pro Solver  ");
        System.err.println("-+-+-+-+-+-+-+-+-+-+-+-+-+-+-");

        ArrayList<String> fileContent = IO.readFile();

        if (fileContent.isEmpty()) {
            System.out.println("File kosong atau tidak ditemukan.");
        } else {
            String[] firstLine = fileContent.get(0).split(" ");
            int row, col, pieces;
            try {
                row = Integer.parseInt(firstLine[0]);
                col = Integer.parseInt(firstLine[1]);
                pieces = Integer.parseInt(firstLine[2]);

                if (row <= 0 || col <= 0) {
                    System.out.println("Jumlah baris dan kolom harus
bilangan bulat positif.");
                    return;
                }

                if (pieces <= 0 || pieces > 26) {
                    System.out.println("Jumlah potongan block harus di
antara 1 dan 26.");
                    return;
                }

            } catch (NumberFormatException e) {
                System.out.println("Input format salah, pastikan jumlah
baris, kolom, dan potongan block adalah bilangan bulat.");
                return;
            }
```

```
            try {
                String mode = fileContent.get(1);
                if (!mode.equals("DEFAULT")){
                    System.err.println("Mode harus DEFAULT.");
                    return;
                }
            } catch (Exception e) {

            }

            fileContent.remove(0);
            fileContent.remove(0);

            ArrayList<Block> blockPieces =
IO.proccessBlock(fileContent);

            long startTime = System.nanoTime();
            Solver solver = new Solver(blockPieces, row, col);
            boolean found = solver.solve();
            long endTime = System.nanoTime();
            long time = (endTime - startTime)/1000000;

            if (found) {
                System.out.println("Solusi ditemukan!");
                System.out.println("Waktu pencarian: " + time + " ms.");
                int count = solver.getCount();
                System.out.println("Banyak kasus yang ditinjau: " +
count);
                solver.printBoard();
                Scanner scanner = new Scanner(System.in);
                System.out.print("Apakah Anda ingin menyimpan solusi?
(ya/tidak)");
                String save = scanner.nextLine().trim().toLowerCase();

                if (save.equals("ya")) {
                    String[][] board = solver.getBoard();
                    IO.writeFile(board);
                }
```

```
            } else {
                System.out.println("Tidak ada solusi.");
                System.out.println("Waktu pencarian: " + time + " ms.");
                int count = solver.getCount();
                System.out.println("Banyak kasus yang ditinjau: " +
count);
            }

        }
    }
}
```

Program Block.java

```
public class Block {
    public String block[][];
    public int size;

    public void createBlock(int size){
        String a[][];
        a = new String[size][size];

        int i, j;
        for(i = 0; i < size; i++){
            for(j = 0; j < size; j++){
                a[i][j] = "-";
            }
        }

        this.block = a;
        this.size = size;
    }

    public int getBlockSize(){
        return this.size;
    }

    public void rotate(Block b){
        int l = 0;
```

```
        int r = b.size - 1;
        int i;

        while (l < r){
            for(i = 0; i < r-l; i++){
                int top = l;
                int bottom = r;

                String topLeft = b.block[top][l+i];

                b.block[top][l+i] = b.block[bottom-i][l];
                b.block[bottom-i][l] = b.block[bottom][r-i];
                b.block[bottom][r-i] = b.block[top+i][r];
                b.block[top+i][r] = topLeft;
            }
            r -= 1;
            l += 1;
        }
     }

    public Block mirrorX(Block b){
     Block hasil = new Block();
        hasil.createBlock(b.size);
        int i, j;

        for(i = 0; i < b.size; i++){
            for(j = 0; j < b.size; j++){
                hasil.block[b.size-1-i][j] = b.block[i][j];
            }
        }
        return hasil;
    }


     public void printBlock() {
        int i, j;
        for (i = 0; i < size; i++) {
            for (j = 0; j < size; j++) {
                System.out.print(block[i][j] + " ");
```

```
            }
            System.out.println();
        }
    }

}
```

Program IO.java

```
import java.io.*;
import java.util.*;


public class IO {
    public static Scanner scanner = new Scanner(System.in);
    public static BufferedReader inputFile = new BufferedReader(new
InputStreamReader(System.in));

    public static ArrayList<String> readFile() {
        System.out.print("Masukkan nama file (contoh: 1.txt) ");
        String fileName = scanner.nextLine();

        String path = "../test" + File.separator + "Input" +
File.separator + fileName;

        File file = new File(path);
        // System.out.println("Trying path: " + file.getAbsolutePath());

        if (file.exists()) {
            try (BufferedReader br = new BufferedReader(new
FileReader(file))) {
                ArrayList<String> input = new ArrayList<>();
                String line;

                while ((line = br.readLine()) != null) {
                    input.add(line);
                }

                if (input.isEmpty()) {
```

```
                    System.out.println("File kosong");
                    return new ArrayList<>();
                }

                return input;
            } catch (IOException e) {
                System.out.println("Error reading file: " +
e.getMessage());
            }
        }

        System.out.println("File tidak ditemukan.");
        return new ArrayList<>();
    }

    public static ArrayList<Block> proccessBlock(ArrayList<String>
fileContent) {
        ArrayList<ArrayList<String>> blocksArray = new ArrayList<>();
        ArrayList<String> currentBlock = new ArrayList<>();

        char currentLetter = ' ';

        for (String line : fileContent) {
            char firstChar = ' ';
            int i;
            for (i = 0; i < line.length(); i++){
                if (line.charAt(i) != ' '){
                    firstChar = line.charAt(i);
                }
            }

            if (firstChar != currentLetter) {
                if (!currentBlock.isEmpty()) {
                    blocksArray.add(new ArrayList<>(currentBlock));
                    currentBlock.clear();
                }
                currentLetter = firstChar;
            }
            currentBlock.add(line);
```

```
        }

        if (!currentBlock.isEmpty()) {
            blocksArray.add(new ArrayList<>(currentBlock));
        }

        //System.out.println(blocksArray);

        ArrayList<Block> blocks = new ArrayList<>();
        for (ArrayList<String> blockLines : blocksArray) {
            if (!blockLines.isEmpty()) {
                int maxSize = 0;
                for (String line : blockLines) {
                    maxSize = Math.max(maxSize, line.length());
                }
                maxSize = Math.max(maxSize, blockLines.size());

                Block block = new Block();
                block.createBlock(maxSize);
                int i,  j;
                for (i = 0; i < blockLines.size(); i++) {
                    String line = blockLines.get(i);
                    for (j = 0; j < line.length(); j++) {
                        if (line.charAt(j) == ' '){
                            block.block[i][j] = "-";
                        } else {
                            block.block[i][j] =
String.valueOf(line.charAt(j));
                        }
                    }
                }

                // block.fillBlock(blockLines);

                blocks.add(block);
                // System.out.println("Created piece:");
                // block.printBlock();
                // System.out.println();
            }
```

```
        }

        return blocks;
    }

    public static void writeFile(String[][] board) {
            String nameFile = "";
            //System.out.println("Current working directory: " +
System.getProperty("user.dir"));

            System.out.println("Masukkan nama file (contoh: solusi.txt)
");
            try {
                nameFile = inputFile.readLine();
                String path = "../test/Output/" + nameFile;

                File file = new File(path);
                if (file.exists()) {
                    System.out.println("File sudah ada. Apakah Anda
ingin menimpanya? (y/n)");
                    char choice = scanner.next().charAt(0);
                    if (choice != 'y' && choice != 'Y') {
                        System.out.println("Output dibatalkan.");
                        scanner.close();
                        return;
                    }
                }

            } catch (IOException err) {
                err.printStackTrace();
            }

            try {
                FileWriter file = new FileWriter("../test/Output/" +
nameFile);
                int i, j;
                for(i = 0; i < board.length; i++){
                    for(j = 0; j < board[0].length; j++){
                        file.write(board[i][j] + ' ');
```

```
                    }
                    file.write("\n");
                }
                file.close();
                System.out.println("File berhasil dibuat!");
            } catch (IOException err) {
                err.printStackTrace();
            }

    }
}
```

Program Solver.java

```
import java.util.ArrayList;
public class Solver {
    private int boardRow;
    private int boardCol;
    private String[][] board;
    private ArrayList<Block> pieces;
    private int count;

    public Solver(ArrayList<Block> blockPieces, int row, int col) {
        boardRow = row;
        boardCol = col;
        board = new String[boardRow][boardCol];
        pieces = new ArrayList<>(blockPieces);
        count = 0;
        makeBoard();
    }

    private void makeBoard() {
        int i,j;
        for (i = 0; i < boardRow; i++) {
            for (j = 0; j < boardCol; j++) {
                board[i][j] = "-";
            }
        }
    }
```

```
    public boolean solve() {
        if (pieces.isEmpty()) {
            return boardFull();
        }

        Block currentPiece = pieces.remove(0);

        for (int i = 0; i < boardRow; i++) {
            for (int j = 0; j < boardCol; j++) {
                for (int k = 0; k < 4; k++) {
                    if (canPutPiece(currentPiece, i, j)) {
                        count++;
                        putPiece(currentPiece, i, j,
getPieceLetter(currentPiece));
                        if (solve()) {
                            return true;
                        }
                        removePiece(currentPiece, i, j);
                    }
                    currentPiece.rotate(currentPiece);
                }

                currentPiece.mirrorX(currentPiece);
                for (int k = 0; k < 4; k++) {
                    if (canPutPiece(currentPiece, i, j)) {
                        count++;
                        putPiece(currentPiece, i, j,
getPieceLetter(currentPiece));
                        if (solve()) {
                            return true;
                        }
                        removePiece(currentPiece, i, j);
                    }
                    currentPiece.rotate(currentPiece);
                }
            }
        }
```

```
        pieces.add(0, currentPiece);
        return false;
    }

    private String getPieceLetter(Block piece) {
        int i, j;
        for (i = 0; i < piece.size; i++) {
            for (j = 0; j < piece.size; j++) {
                if (!piece.block[i][j].equals("-")) {
                    return piece.block[i][j];
                }
            }
        }
        return "-";
    }

    private boolean canPutPiece(Block piece, int row, int col) {
        int i, j;
        for (i = 0; i < piece.size; i++) {
            for (j = 0; j < piece.size; j++) {
                if (!piece.block[i][j].equals("-")) {
                    if (row + i >= boardRow || col + j >= boardCol) {
                        return false;
                    }
                    if (!board[row + i][col + j].equals("-")) {
                        return false;
                    }
                }
            }
        }
        return true;
    }

    private void putPiece(Block piece, int row, int col, String id) {
        int i, j;
        for (i = 0; i < piece.size; i++) {
            for (j = 0; j < piece.size; j++) {
                if (!piece.block[i][j].equals("-")) {
                    board[row + i][col + j] = id;
```

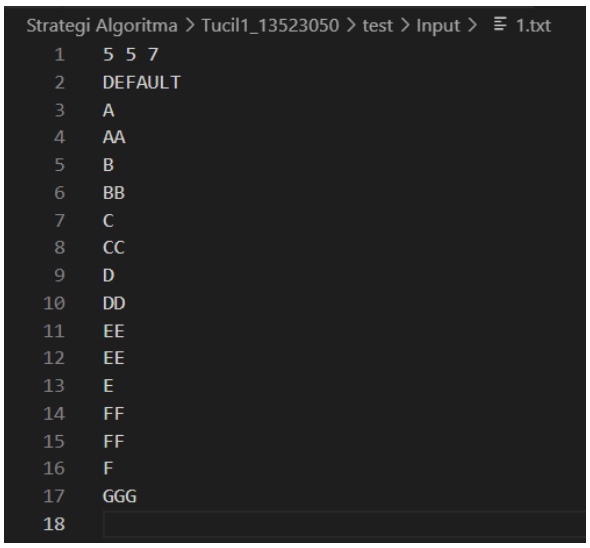
```
                }
            }
        }
    }

    private void removePiece(Block piece, int row, int col) {
        int i, j;
        for (i = 0; i < piece.size; i++) {
            for (j = 0; j < piece.size; j++) {
                if (!piece.block[i][j].equals("-")) {
                    board[row + i][col + j] = "-";
                }
            }
        }
    }

    private boolean boardFull() {
        int i, j;
        for (i = 0; i < boardRow; i++) {
            for (j = 0; j < boardCol; j++) {
                if (board[i][j].equals("-")) {
                    return false;
                }
            }
        }
        return true;
    }

    public void printBoard() {
        int i, j;
        for (i = 0; i < boardRow; i++) {
            for (j = 0; j < boardCol; j++) {
                String pieceLetter = board[i][j];
                System.out.print(getColorForLetter(pieceLetter) +
pieceLetter + " \033[0m");
            }
            System.out.println();
        }
    }
```

```
public String[][] getBoard() {
    return board;
}

public int getCount() {
    return count;
}

private String getColorForLetter(String pieceLetter) {
    switch (pieceLetter) {
        case "A": return "\033[31m";  // red
        case "B": return "\033[32m";  // green
        case "C": return "\033[33m";  // yellow
        case "D": return "\033[34m";  // blue
        case "E": return "\033[35m";  // magenta
        case "F": return "\033[36m";  // cyan
        case "G": return "\033[90m";  // gray
        case "H": return "\033[38;5;214m";  // LIGHT red
        case "I": return "\033[92m";  // bright green
        case "J": return "\033[38;5;117m";  // sky blue
        case "K": return "\033[38;5;32m";  // medium Blue
        case "L": return "\033[38;5;201m";  // fuschia
        case "M": return "\033[96m";  // bright cyan
        case "N": return "\033[97m";  // bright white
        case "O": return "\033[38;5;208m"; // orange
        case "P": return "\033[38;5;125m"; // pink
        case "Q": return "\033[38;5;36m";  // sea green
        case "R": return "\033[38;5;226m"; // gold
        case "S": return "\033[38;5;82m";  // lime green
        case "T": return "\033[38;5;51m";  // teal
        case "U": return "\033[38;5;220m"; // peach
        case "V": return "\033[38;5;213m"; // orchid
        case "W": return "\033[38;5;56m";  // dark blue
        case "X": return "\033[38;5;220m"; // amber
        case "Y": return "\033[38;5;58m"; // light yellow
        case "Z": return "\033[38;5;160m"; // brick red
        default: return "\033[0m";  // Default color (reset)
    }
```

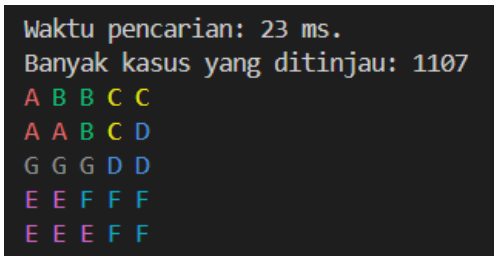
```
        }
    }
```

IV. Testcase

- Testcase 1

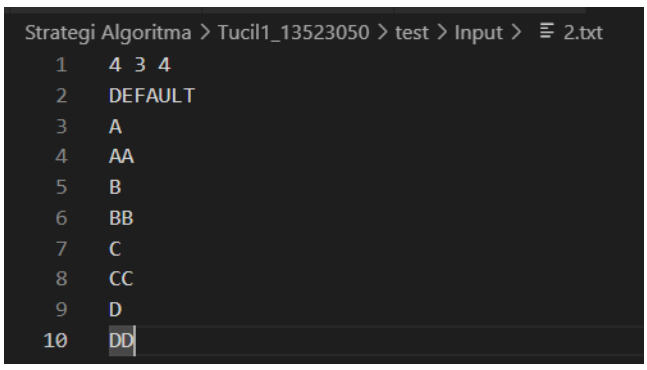


Gambar 4.1 Testcase 1

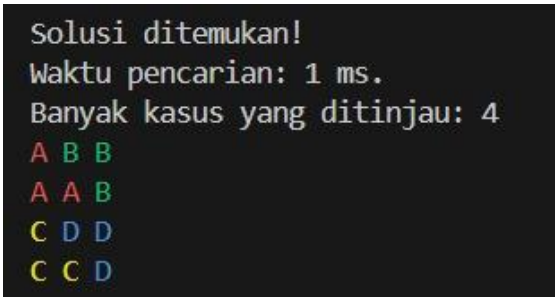


Gambar 4.2 Hasil Testcase 1

- Testcase 2

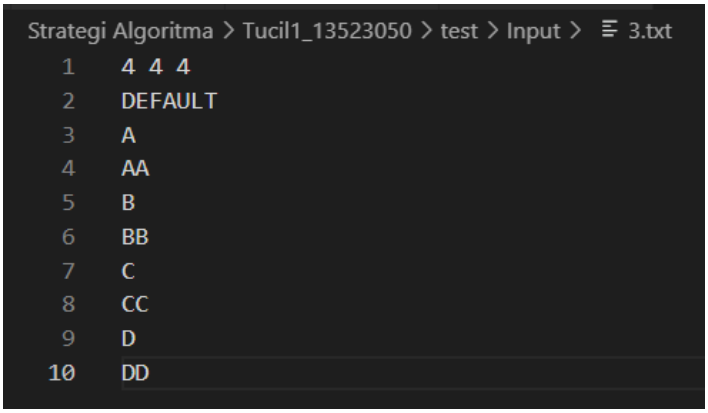


Gambar 4.3 Testcase 2

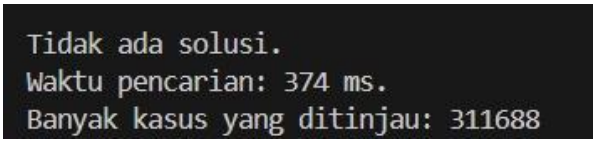


Gambar 4.6 Hasil Testcase 2

- Testcase 3



Gambar 4.5 Testcase 3



Gambar 4.6 Hasil Testcase 3

- Testcase 4

Strategi Algoritma > Tucil1_13523050 > test > Input > 4.txt

```
4 5 6
DEFAULT
A
A
A
A
BB
BB
CC
C
DD
E
E
F
FFFF
```

Gambar 4.7 Testcase 4

```
Solusi ditemukan!
Waktu pencarian: 4 ms.
Banyak kasus yang ditinjau: 64
A B B C C
A B B C E
A F D D E
A F F F F
```

Gambar 4.8 Hasil Testcase 4

- Testcase 5

Strategi Algoritma > Tucil1_13523050 > test > Input > 5.txt

```
6 6 7
DEFAULT
AAAA
AAA
BBB
BB
B
CCCC
 CC
DDD
DD
E
EEE
FFF
GG
GG
```

Gambar 4.9 Testcase 5

```
Tidak ada solusi.
Waktu pencarian: 79370 ms.
Banyak kasus yang ditinjau: 5441328
```

Gambar 4.10 Hasil Testcase 5

- Testcase 6

Strategi Algoritma > Tucil1_13523050 > test > Input > 6.txt

```
5 5 7
DEFAULT
MM
MM
M
AA
AA
Y
Y
Y
Y
ZZ
 ZZ
LLL
L
 B
SSS
```

Gambar 4.11 Testcase 6

```
Solusi ditemukan!
Waktu pencarian: 18 ms.
Banyak kasus yang ditinjau: 389
M M A A Y
M M A A Y
M Z S L Y
Z Z S L Y
Z B S L L
```

Gambar 4.12 Hasil Testcase 6

- Testcase 7

Strategi Algoritma > Tucil1_13523050 > test > Input > 7.txt

```
3 6 6
DEFAULT
A
A
BBB
B
CCC
D
DDD
 EE
EE
F
```

Gambar 4.13 Testcase 7

```
Solusi ditemukan!
Waktu pencarian: 17 ms.
Banyak kasus yang ditinjau: 408
A B B B D C
A B E E D C
F E E D D C
```

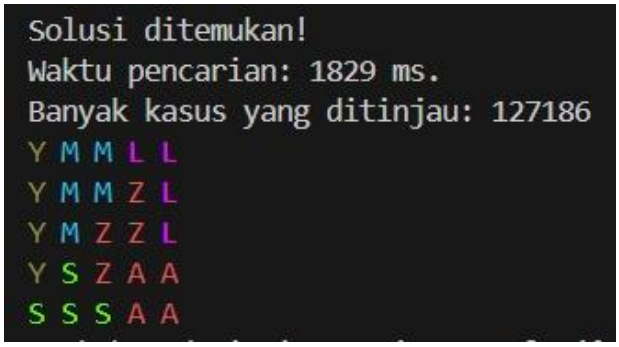
Gambar 4.14 Hasil Testcase 7

- Testcase 8

Strategi Algoritma > Tucil1_13523050 > test > Input > 8.txt

```
5 5 6
DEFAULT
MM
MM
M
AA
AA
Y
Y
Y
Y
ZZ
 ZZ
LLL
L
 S
SSS
```

Gambar 4.15 Testcase 8



Gambar 4.16 Hasil Testcase 8

V. Pranala GitHub

https://github.com/maymilla/Tucil1_13523050