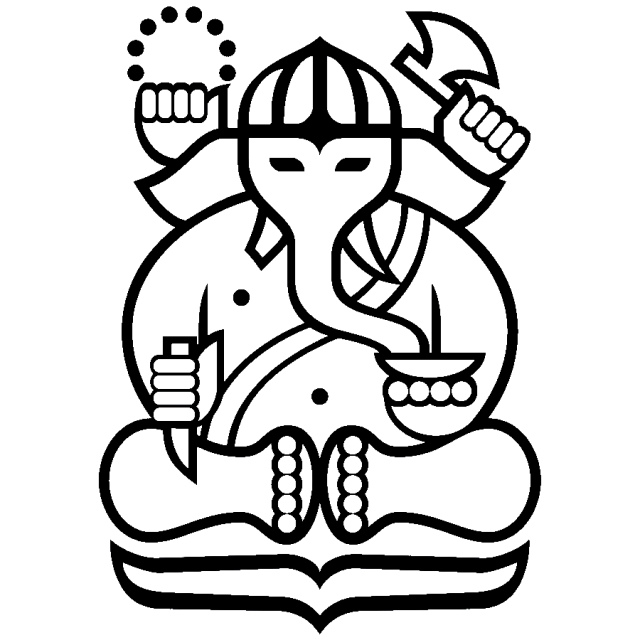
LAPORAN TUGAS KECIL 1

IF2211 Strategi Algoritma

Penyelesaian IQ Puzzle Pro dengan Algoritma *Brute Force*

Disusun oleh :

Dzubyan Ilman Ramadhan (10122010)



**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**JL. GANESA 10, BANDUNG 40132**

**2024**

# Algoritma

Algoritma yang digunakan adalah brute force dengan rincian berikut

1. Taruh blok puzzle di pojok atas
2. Cek apakah blok tersebut dapat ditaruh di posisi tersebut.
3. A. Apabila bisa, lanjutkan pengecekan blok puzzle dengan blok puzzle selanjutnya yang masih tersedia ke step 1

B. Apabila tidak bisa, geser blok puzzle ke kanan 1 unit, geser ke bawah dan mulai dari kiri apabila sudah mencapai pinggir kanan papan, lalu laksanakan step 2.

C. Apabila piece tidak bisa ditaruh untuk semua kemungkinan di papan, lanjutkan ke langkah 4

1. Putar konfigurasi blok tersebut, lalu ulangi dari step 1. Apabila blok telah terputar sebanyak 4 kali (kembali ke konfigurasi awal), lanjutkan ke langkah 5
2. Refleksikan konfigurasi blok tersebut, lalu ulangi dari step 1. Apabila blok masih tidak bisa ditaruh di papan, Lanjutkan ke langkah 6
3. Kembali ke blok sebelumnya, pindahkan blok tersebut ke kanan, lalu kembali ke step 2.
4. Jika blok pertama sudah diputar dan direfleksikan masih tidak bisa ada solusi, berarti tidak ada solusi.
5. Bila papan penuh dan semua blok dapat dipasang di papan, maka susunan blok di papan itu adalah jawaban yang kita cari

# 2. Source Code

Kelas Piece

import java.util.ArrayList;

import java.util.Arrays;

public class Piece {

private char symbol;

private int w = 0;

private int h = 0;

private int row\_coord = 0;

private int col\_coord = 0;

public int rotation\_index = 0;

public int reflection\_index = 0;

//private ArrayList<ArrayList<String>> blockShape = new ArrayList<>();

private ArrayList<ArrayList<String>> blockShape = new ArrayList<ArrayList<String>>();

private int[][] blockShapeMatrix;

public Piece(char x) {

symbol = x;

}

public void updateRowShape(String blockRow) {

ArrayList<String> blockRowArray = new ArrayList<>(Arrays.asList(blockRow.split("")));

blockShape.add(blockRowArray);

checkResize(blockRowArray);

makeBlockShapeMatrix(blockShape);

}

public void makeBlockShapeMatrix(ArrayList<ArrayList<String>> blockShape) {

blockShapeMatrix = new int[h][w];

for (int i=0; i < h; i++) {

for(int j = 0; j < w; j++) {

if (j >= blockShape.get(i).size()) {

blockShapeMatrix[i][j] = 0;

} else if(blockShape.get(i).get(j).equals(" ")) {

blockShapeMatrix[i][j] = 0;

} else{

blockShapeMatrix[i][j] = 1;

}

}

}

}

public void checkResize(ArrayList<String> blockRowArray) {

if (blockRowArray.size() > w) {

w = blockRowArray.size();

}

h++;

}

public void rotatePiece90Deg() {

rotation\_index++;

int[][] newBlockShapeMatrix = new int[w][h];

for (int i = 0; i < w; i++) {

for (int j=0; j< h; j++) {

newBlockShapeMatrix[i][j] = blockShapeMatrix[h-1-j][i];

}

}

blockShapeMatrix = newBlockShapeMatrix;

w = h+w;

h = w-h;

w = w-h;

}

public void reflectPiece() {

reflection\_index++;

int[][] newBlockShapeMatrix = new int[h][w];

for (int i = 0; i < h; i++) {

for (int j=0; j< w; j++) {

newBlockShapeMatrix[i][j] = blockShapeMatrix[i][w-1-j];

}

}

blockShapeMatrix = newBlockShapeMatrix;

}

public char getSymbol() {

return symbol;

}

public int[][] getBlockShapeMatrix() {

return blockShapeMatrix;

}

public int getWidth() {

return w;

}

public int getHeight() {

return h;

}

public int getRowCoord() {

return row\_coord;

}

public int getColCoord() {

return col\_coord;

}

public void setRowCoord(int x) {

row\_coord = x;

}

public void setColCoord(int y) {

col\_coord = y;

}

public void printShape() {

for(int i = 0; i < blockShape.size(); i++){

System.out.println(blockShape.get(i));

}

}

}

Kelas Board

import java.util.Arrays;

import java.io.\*;

public class Board {

private int boardMatrix[][];

private char boardConfig[][];

private int height;

private int width;

public long cases = 0;

public Board(int h, int w) {

boardMatrix = new int[h][w];

boardConfig = new char[h][w];

width = w;

height = h;

}

public void addPiece(Piece piece, int row, int col) { //Assumes position is valid

int[][] piece\_matrix = piece.getBlockShapeMatrix();

int h\_piece = piece\_matrix.length;

int w\_piece = piece\_matrix[0].length;

for(int i = 0; i < h\_piece; i++) {

for(int j = 0; j < w\_piece; j++) {

boardMatrix[row + i][col + j] += piece\_matrix[i][j];

}

}

piece.setRowCoord(row);

piece.setColCoord(col);

}

public void removePiece(Piece piece) {

int[][] piece\_matrix = piece.getBlockShapeMatrix();

int row = piece.getRowCoord();

int col = piece.getColCoord();

int h\_piece = piece\_matrix.length;

int w\_piece = piece\_matrix[0].length;

for(int i = 0; i < h\_piece; i++) {

for(int j = 0; j < w\_piece; j++) {

boardMatrix[row + i][col + j] -= piece\_matrix[i][j];

}

}

}

public boolean checkFitPieceOnSize(Piece piece) {

if (piece.getHeight() > height || piece.getWidth() > width) {

return false;

} else {

return true;

}

}

public boolean checkFitPieceAtPlace(Piece piece, int row, int col) {

int[][] shape = piece.getBlockShapeMatrix();

for(int i = 0; i < shape.length; i++) {

for(int j = 0; j < shape[0].length; j++) {

if(shape[i][j] == 1) {

if(row + i >= width || col + j >= height || boardMatrix[row + i][col + j] != 0) {

return false;

}

}

}

}

return true;

}

public boolean checkBoardFull() {

boolean isFull = true;

for(int i = 0; i < boardMatrix.length; i++) {

for(int j = 0; j < boardMatrix[0].length; j++) {

if(boardMatrix[i][j] != 1) {

isFull = false;

}

}

}

return isFull;

}

public int getWidth() {

return width;

}

public int getHeight() {

return height;

}

public void showBoardMatrix() {

System.out.println(Arrays.deepToString(boardMatrix));

}

public void editBoardConfig(Piece[] piece\_arr) {

for (int k = 0; k < piece\_arr.length; k++) {

int[][] matrix = piece\_arr[k].getBlockShapeMatrix();

int row = piece\_arr[k].getRowCoord();

int col = piece\_arr[k].getColCoord();

for(int i = 0; i < matrix.length; i++) {

for(int j = 0; j < matrix[0].length; j++) {

if(matrix[i][j] == 1) {

boardConfig[row + i][col + j] = piece\_arr[k].getSymbol();

}

}

}

}

}

// public void showBoardConfig(FileWriter file\_out) throws IOException{

// for(int i = 0; i < boardConfig.length; i++) {

// for(int j = 0; j < boardConfig[0].length; j++) {

// file\_out.write(boardConfig[i][j]);

// }

// file\_out.write("\n");

// }

// file\_out.close();

// }

public String showBoardConfig() {

String out = "";

for(int i = 0; i < boardConfig.length; i++) {

for(int j = 0; j < boardConfig[0].length; j++) {

out += boardConfig[i][j];

}

out += "\n";

}

return out;

}

}

Kelas Main

import java.util.Scanner;

import java.io.\*;

public class IQPuzzleProSolver {

public static void main (String[] args) throws IOException {

File file = new File("./input.txt");

File outFile = new File("output.txt");

Scanner input = new Scanner(file);

int M = Integer.parseInt(input.next());

int N = Integer.parseInt(input.next());

int P = Integer.parseInt(input.next());

input.nextLine();

Board board = new Board(M, N);

Piece[] piece\_arr = new Piece[P];

String keyword = input.nextLine();

if (keyword.equals("DEFAULT")) {

char letter = '0';

int numOfPieces = 0;

int i = 0;

String block\_row = input.nextLine();

do {

if (letter != getLetter(block\_row)) {

numOfPieces++;

letter = getLetter(block\_row);

piece\_arr[i] = new Piece(letter);

i++;

}

piece\_arr[numOfPieces-1].updateRowShape(block\_row);

block\_row = input.nextLine();

} while (numOfPieces <= P && !block\_row.equals(""));

}

input.close();

//Algoritma Brute Force

long start = System.currentTimeMillis();

bruteForceIt(piece\_arr, 0, board);

long time = System.currentTimeMillis() - start;

FileWriter fWriter = new FileWriter(outFile);

PrintWriter pWriter = new PrintWriter(fWriter);

if(board.checkBoardFull()) {

board.showBoardMatrix();

board.editBoardConfig(piece\_arr);

String output = board.showBoardConfig();

System.out.println(output);

System.out.println("Waktu yang dibutuhkan : ");

System.out.println(time + " ms");

System.out.println("Banyak kasus yang ditinjau :");

System.out.println(board.cases);

System.out.println("Apakah Anda ingin menyimpan solusi?");

Scanner input2 = new Scanner(System.in);

String toSave = input2.nextLine();

if (toSave.equals("Y")) {

System.out.println("Solusi tersimpan!");

pWriter.println(output);

pWriter.close();

fWriter.close();

input2.close();

}

// System.out.println(Arrays.deepToString(piece\_arr[0].getBlockShapeMatrix()));

// System.out.println(piece\_arr[0].getRowCoord());

// System.out.println(piece\_arr[0].getColCoord());

// System.out.println(Arrays.deepToString(piece\_arr[1].getBlockShapeMatrix()));

// System.out.println(piece\_arr[1].getRowCoord());

// System.out.println(piece\_arr[1].getColCoord());

// System.out.println(Arrays.deepToString(piece\_arr[2].getBlockShapeMatrix()));

// System.out.println(piece\_arr[2].getRowCoord());

// System.out.println(piece\_arr[2].getColCoord());

} else {

// System.out.println(Arrays.deepToString(piece\_arr[0].getBlockShapeMatrix()));

// System.out.println(piece\_arr[0].getRowCoord());

// System.out.println(piece\_arr[0].getColCoord());

// System.out.println(Arrays.deepToString(piece\_arr[1].getBlockShapeMatrix()));

// System.out.println(piece\_arr[1].getRowCoord());

// System.out.println(piece\_arr[1].getColCoord());

// System.out.println(Arrays.deepToString(piece\_arr[2].getBlockShapeMatrix()));

// System.out.println(piece\_arr[2].getRowCoord());

// System.out.println(piece\_arr[2].getColCoord());

System.out.println("Tidak ada solusi yg mungkin");

}

}

public static boolean bruteForceIt(Piece[] piece\_arr, int k, Board board) {

if(k == piece\_arr.length) {

return true;

}

Piece piece = piece\_arr[k];

int h = piece.getHeight();

int w = piece.getWidth();

for(int row = 0; row < board.getHeight(); row++) {

for(int col = 0; col < board.getWidth(); col++) {

for(int r = 0; r < 4; r++) {

board.cases++;

piece.rotatePiece90Deg();

if(board.checkFitPieceAtPlace(piece, row, col)) {

board.addPiece(piece, row, col);

if(bruteForceIt(piece\_arr, k+1, board)){

return true;

}

board.removePiece(piece);

}

}

piece.reflectPiece();

for(int r = 0; r < 4; r++) {

board.cases++;

piece.rotatePiece90Deg();

if(board.checkFitPieceAtPlace(piece, row, col)) {

board.addPiece(piece, row, col);

if(bruteForceIt(piece\_arr, k+1, board)){

return true;

}

board.removePiece(piece);

}

}

}

}

return false;

}

public static char getLetter(String block\_row) {

for (int i = 0; i < block\_row.length(); i++) {

char ch = block\_row.charAt(i);

if (Character.isLetter(ch)) {

return ch; // Return the first letter found

}

}

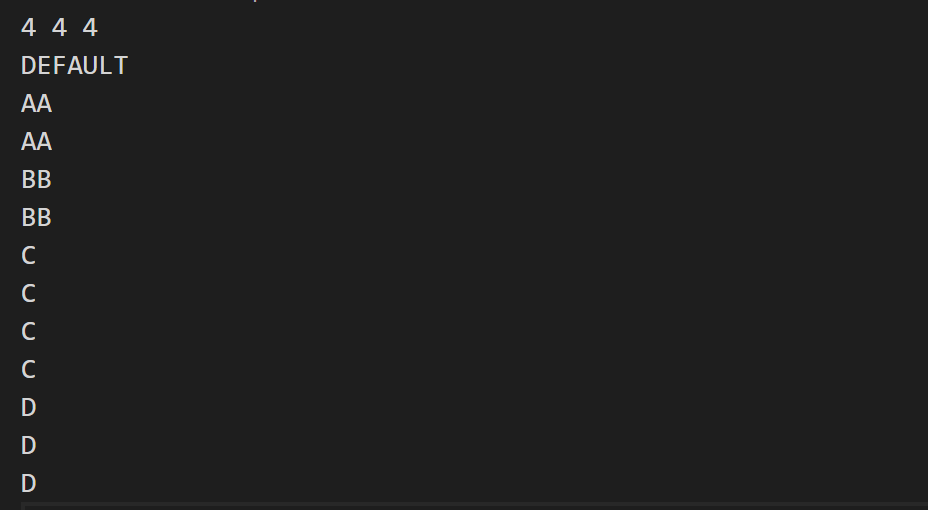
return '\0'; // Return null character if no letter is found

}

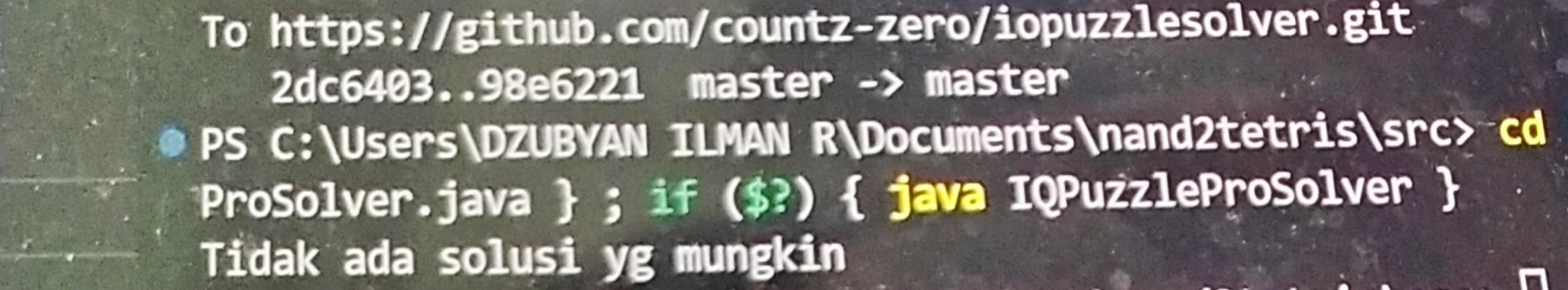
}

# 3. Tangkapan Layar

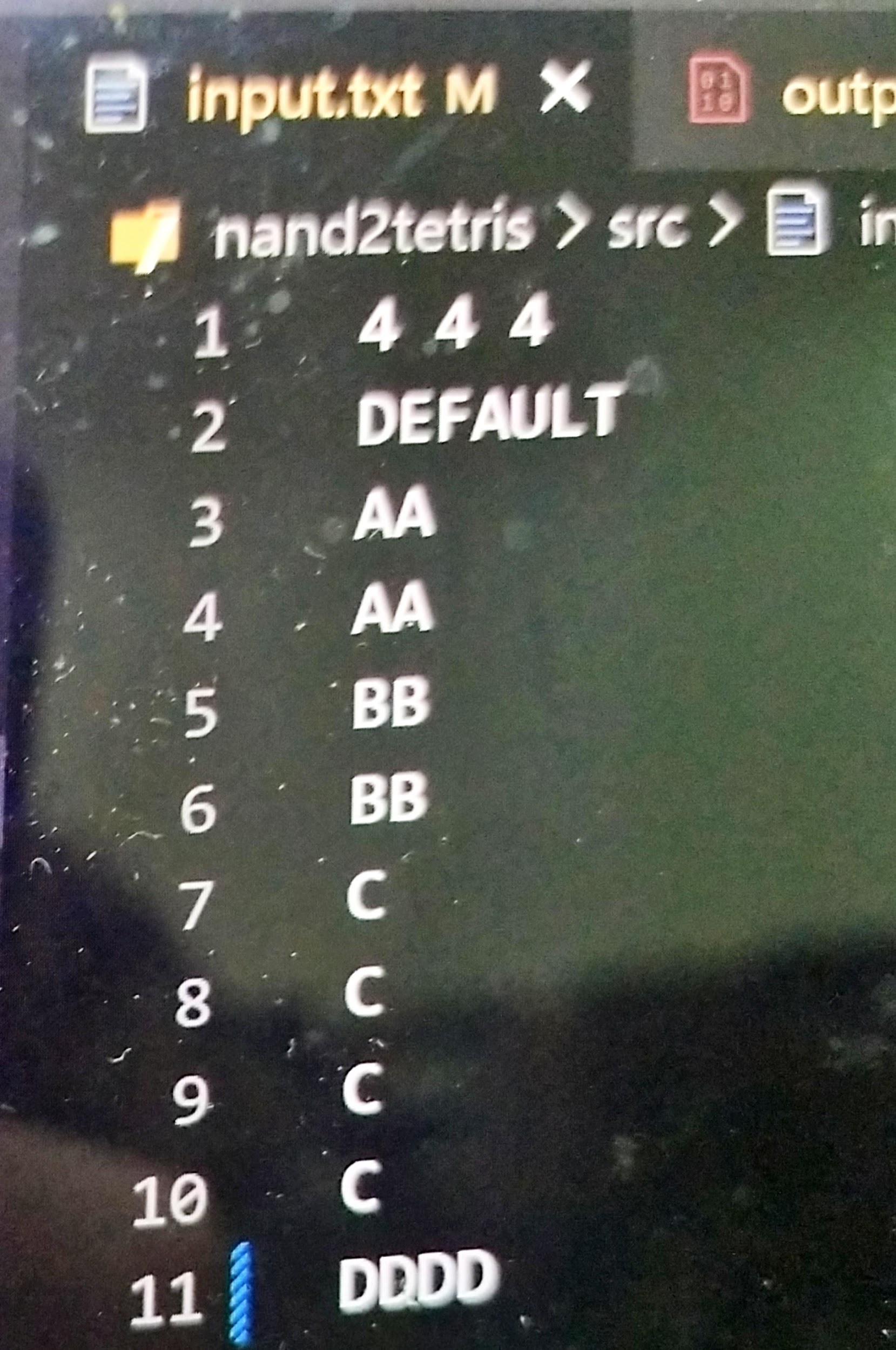
File .txt Input : Kasus tidak ada solusi



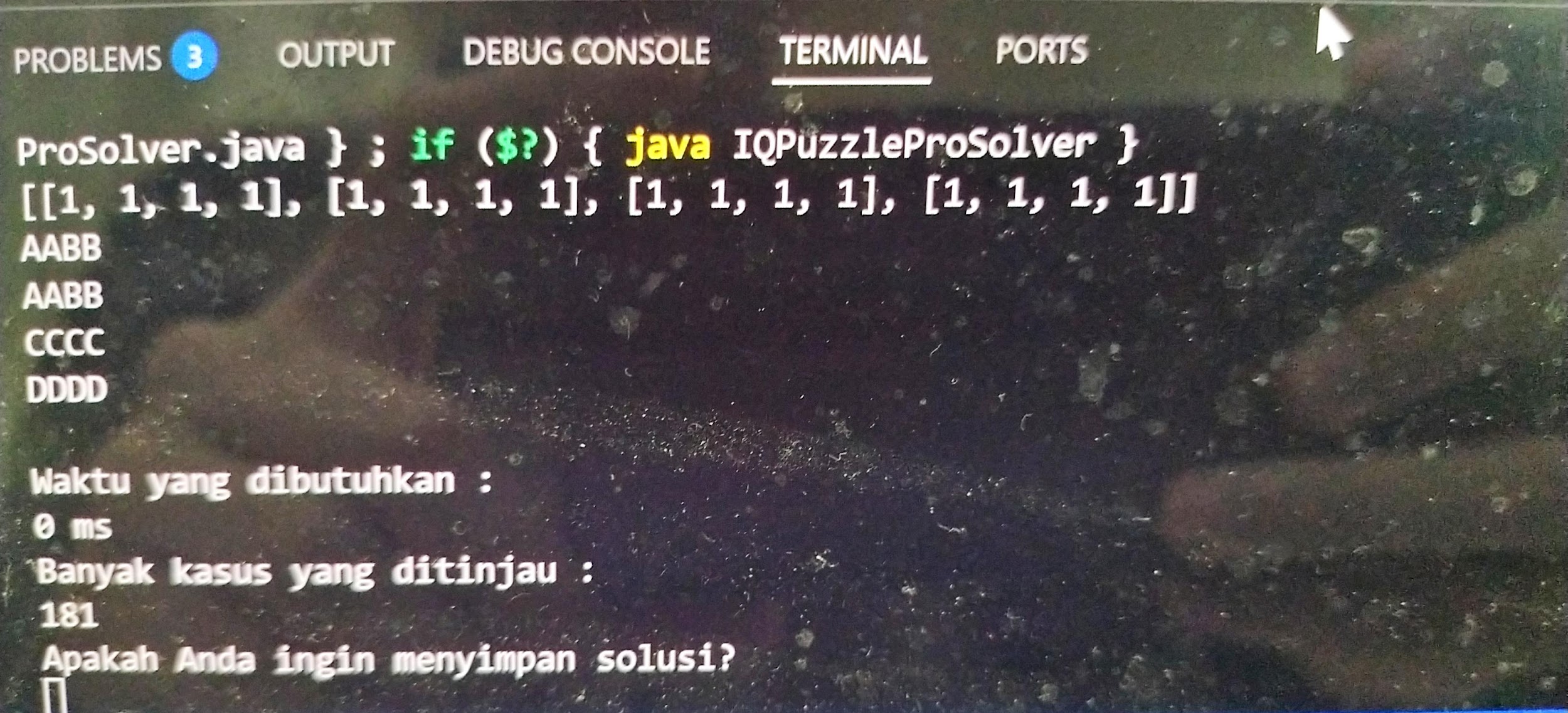
Keluaran terminal



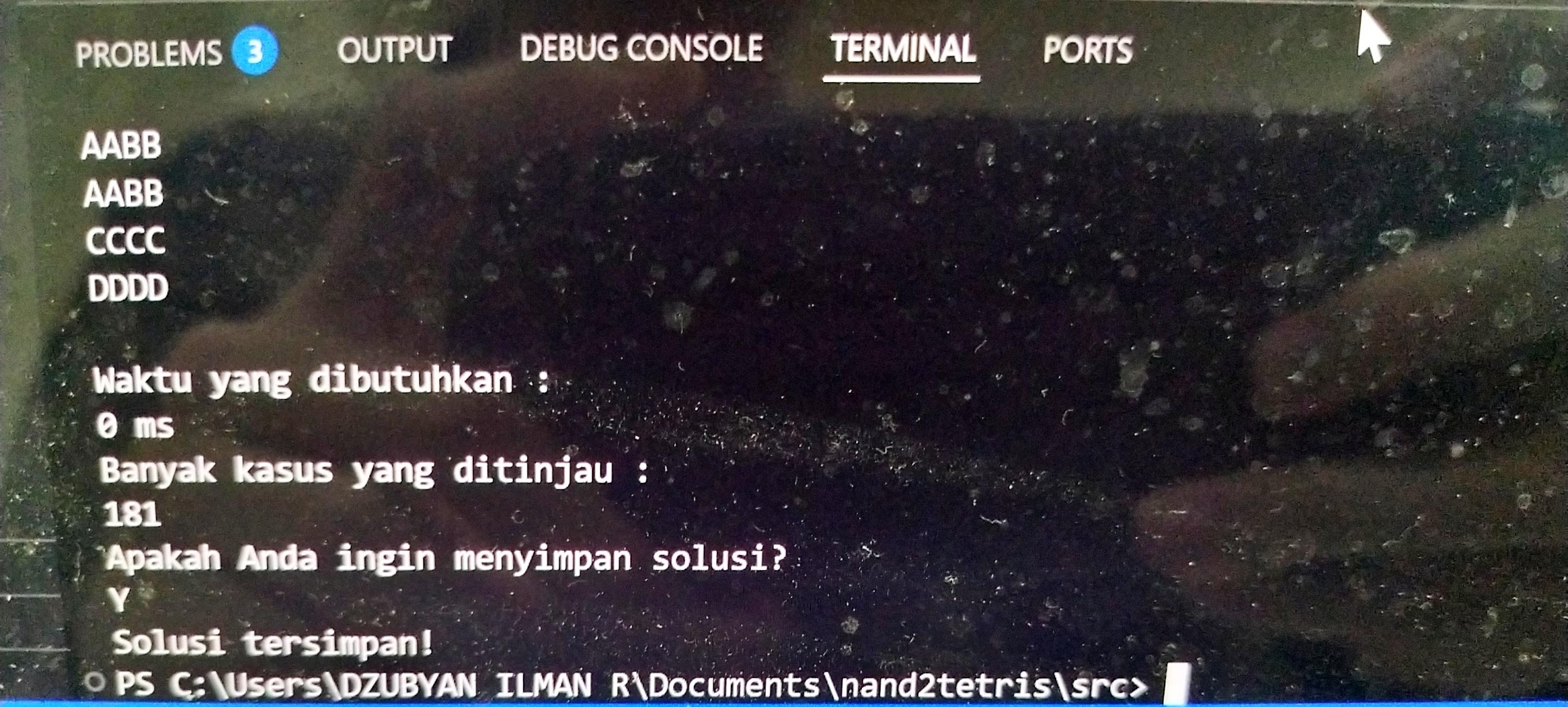
File .txt Input : Kasus terdapat solusi



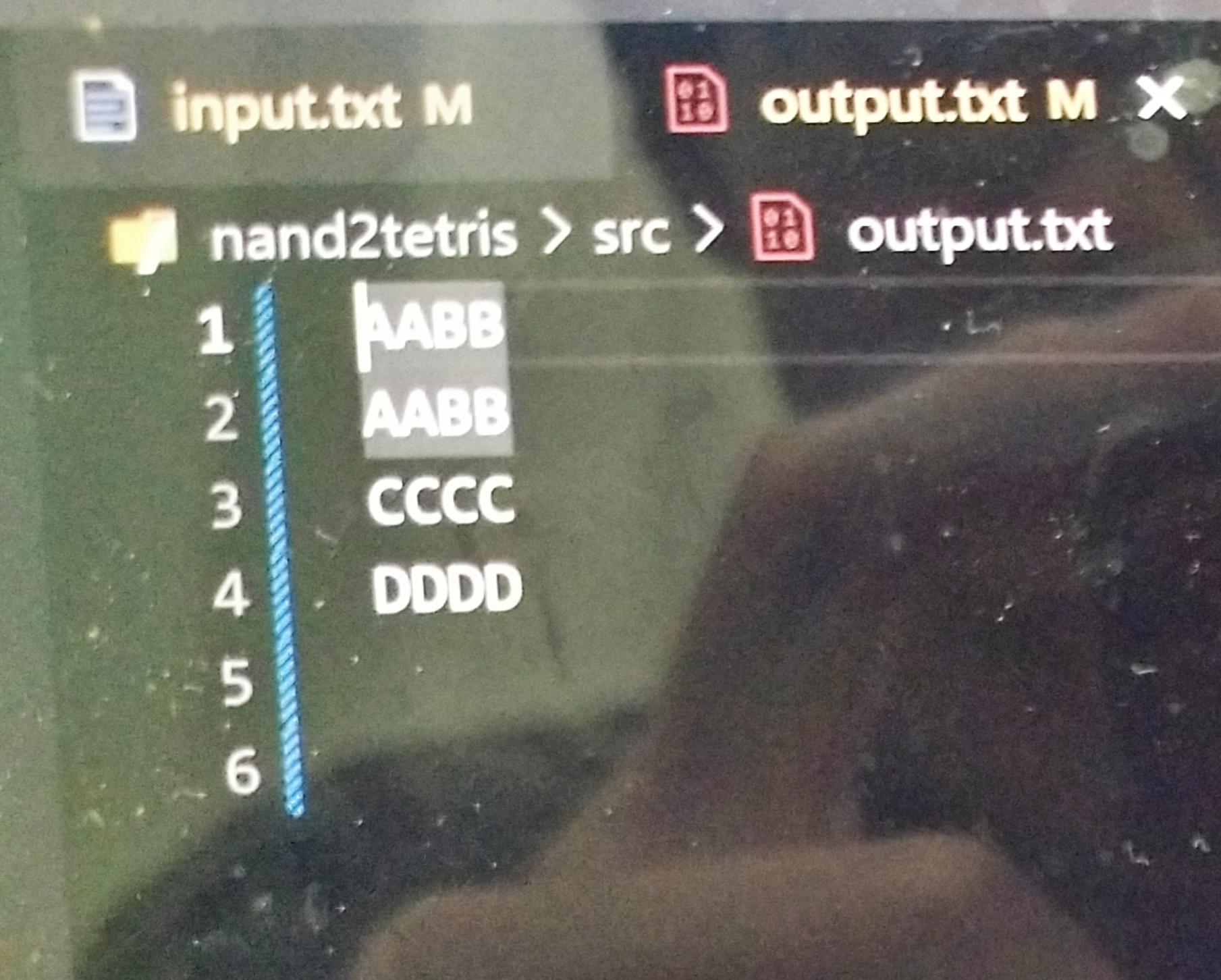
Keluaran terminal :



Apabila ingin menyimpan solusi :



Keluaran solusi yang tersimpan :



# 4. Pranala

Link untuk mengakses folder di remote repository dapat diakses di link berikut

<https://github.com/countz-zero/iopuzzlesolver>