Laporan Tugas Kecil 1 Strategi Algoritma 2024/2025

Muhammad Izzat Jundy - 13523092 - K-2

Deskripsi Masalah

IQ Puzzler Pro adalah sebuah permainan puzzle yang mengharuskan pemain untuk memasang seluruh piece yang ada ke dalam wadah yang disediakan, sedemikian sehingga wadah dipenuhi oleh piece-piece tersebut.

Algoritma *Brute Force*

Setiap *piece* yang diterima akan disimpan setiap skenario peletakannya. Untuk setiap kondisi *piece*, yaitu seberapa banyak terotasi dan tercerminkan atau tidak, serta untuk setiap posisi *piece* pada wadah. Akan diperoleh jumlah skenario yang disimpan sebanyak $n * m * 8$. Skenario-skenario tersebut kemudian dicocokkan antar-*piece*, hingga terdapat kumpulan skenario *piece* yang cocok satu sama lain. Pencocokan dilakukan dengan menggunakan algoritma kombinasi sederhana. Sebelum pencocokan dilakukan, *piece-piece* yang ada diperiksa apakah luas gabungannya sama dengan luas wadah. Jika iya, maka pencocokan dilakukan, sementara jika tidak, dianggap tidak ada solusi. Kompleksitas algoritma yang digunakan adalah $O((n * m)^p)$.

Source Code

Main.java

```java
import java.io.File;
import java.util.Scanner;
import ADT.*;
import Function.*;

public class Main {

    Run | Debug
    public static void main(String[] args){

        String cmd, s, temp, solutionString, fileName;
        int n, m, p;
        long timeStart, timeFinish, duration;
        Scanner input = new Scanner(System.in), readFile;

        System.out.println(x:"Selamat datang di Tugas Kecil 1 Strategi Algoritma oleh Muhammad Izzat Jundy");

        while(true){

            System.out.println();
            System.out.println(x:"Program ini dapat membantumu menemukan solusi dari IQ Puzzler Pro. Ingin mencoba?");
            System.out.println(x:"1. Input File");
            System.out.println(x:"0. Keluar");
            System.out.print(s:"> ");
            cmd = input.nextLine();

            if(cmd.equals(anObject:"1")){

                while(true){
                    System.out.println();
                    System.out.println(x:"Silakan masukkan nama file, diakhiri dengan \".txt\"");
                    System.out.print(s:"> ");
                    fileName = input.nextLine();

                    if(fileName.endsWith(suffix:".txt")){

                        try{
                            temp = System.getProperty(key:"user.dir");

```

```
38
39                      while(!temp.endsWith(suffix:"Tucil1_13523092")){
40
41                          temp = temp.substring(beginIndex:0, temp.length() - 1);
42
43                      }
44
45                      File file = new File(temp, "\\test\\input\\" + fileName).getAbsoluteFile();
46                      readFile = new Scanner(file);
47
48                      timeStart = System.currentTimeMillis();
49
50                      n = readFile.nextInt();
51                      m = readFile.nextInt();
52                      p = readFile.nextInt();
53                      s = readFile.nextLine();
54                      s = readFile.nextLine();
55
56                      Piece solution = new Piece();
57                      solution = Solve.solve(n, m, p, s, temp + "\\test\\input\\" + fileName);
58
59                      timeFinish = System.currentTimeMillis();
60                      readFile.close();
61
62                      System.out.println();
63                      if(Solve.langkah != 0){
64
65                          if(solution.isEmpty()){
66                              System.out.println(x:"Tidak ditemukan solusi untuk mengisi board!");
67                          }else{
68                              System.out.println(x:"Diperoleh solusi sebagai berikut:");
69                              solution.printPiece();
70                          }
71
72                          duration = timeFinish - timeStart;
73                          System.out.println();
```

```
74                          System.out.println("Waktu yang diperlukan: " + duration + " ms");
75
76                          System.out.println();
77                          System.out.println("Banyak kasus yang ditinjau: " + Solve.langkah);
78
79                      }else{
80
81                          System.out.println(x:"Piece yang ada tidak sesuai dengan board!");
82
83                          break;
84                      }
85
86                      while(true){
87                          System.out.println();
88                          System.out.println(x:"Apakah Anda ingin menyimpan solusi? (ya/tidak)");
89                          System.out.print(s:"> ");
90                          cmd = input.nextLine();
91
92                          if(cmd.toLowerCase().equals(anObject:"ya")){
93                              solutionString = Piece.pieceToString(solution);
94                              Solve.writeStringToFile(solutionString, temp + "\\test\\output\\" + fileName);
95                              break;
96                          }else if(cmd.toLowerCase().equals(anObject:"tidak")){
97                              break;
98                          }else{
99                              System.out.println();
100                             System.out.println(x:"Input tidak valid!");
101                         }
102                     }
103
104                     break;
105                 }catch (Exception e){
106                     System.out.println();
107                     System.out.println("Error message: " + e);
108                 }
109
```

```
                    }else{

                        System.out.println();
                        System.out.println(x:"Input tidak valid!");

                    }
                }

            }else if(cmd.equals(anObject:"0")){
                System.out.println();
                System.out.println(x:"Terima kasih telah menjajal program ini!");
                break;
            }else{
                System.out.println();
                System.out.println(x:"Silakan masukkan input berupa angka yang sesuai!");
            }
        }

        input.close();

    }

}
```

ADT/Piece.java

```
package ADT;

import java.util.Arrays;

import Function.*;

public class Piece{

    public char matrix[][];
    public int row;
    public int col;

    public char MARK = '.';

    public void toPiece(char a[][], int row, int col){
        this.matrix = a;
        this.row = row;
        this.col = col;
    }

    public boolean isEmpty(){
        int i, j;

        i = 0;
        while(i<this.row){
            j = 0;
            while(j<this.col){
                if(this.matrix[i][j] != '.') return false;
                j+=1;
            }
            i+=1;
        }

        return true;
    }

    public static Piece copyOf(Piece p){

        int i, j;
```

```
        Piece res = new Piece();

        res.row = p.row;
        res.col = p.col;
        res.matrix = new char[res.row][res.col];

        i = 0;
        while(i<res.row){
            j = 0;
            while(j<res.col){
                res.matrix[i][j] = p.matrix[i][j];
                j+=1;
            }
            i+=1;
        }

        return res;
    }

    public static Piece emptyBoard(int n, int m){

        int i, j;
        Piece res = new Piece();

        res.row = n;
        res.col = m;
        res.matrix = new char[n][m];

        i = 0;
        while(i<res.row){

            j = 0;
            while(j<res.col){

                res.matrix[i][j] = '.';

                j+=1;
            }

            i+=1;
        }

        return res;
    }

    public void shiftRight(){
        boolean isOnRight, isOnBottom;
        int i, j, k, mostLeft;

        isOnRight = false;
        i = 0;
        while(i<this.row){

            if(this.matrix[i][this.col-1] != MARK){

                isOnRight = true;

                break;
            }

            i+=1;
        }

        isOnBottom = false;
        i = 0;
        while(i<this.col){

            if(this.matrix[this.row-1][i] != MARK){

                isOnBottom = true;

                break;
```

```
            }

            i+=1;
        }

        if(isOnRight){

            if(!isOnBottom){

                // mendapatkan posisi part dari piece yang paling kiri
                mostLeft = -1;
                i = 0;
                while(i<this.col){

                    j = 0;
                    while(j<this.row){

                        if(this.matrix[j][i] != '.'){
                            mostLeft = i;
                            break;
                        }

                        j+=1;
                    }

                    if(mostLeft!=-1) break;

                    i+=1;
                }

                // mindahin piece ke paling kiri dan turun 1
                i = this.row-1;
                while(i>0){

                    j = mostLeft;
                    k = 0;
                    while(j<this.col){

                        this.matrix[i][k] = this.matrix[i-1][j];

                        j+=1;
                        k+=1;
                    }

                    while(k<this.col){

                        this.matrix[i][k] = MARK;

                        k+=1;
                    }

                    i-=1;
                }

                // ngosongin atasnya
                i = 0;
                while(i<this.col){
                    this.matrix[0][i] = '.';
                    i+=1;
                }

            }

        }else{

            i = this.row-1;
            while(i>=0){

                j = this.col-1;
                while(j>0){

                    this.matrix[i][j] = this.matrix[i][j-1];
```

```
                j-=1;
            }

            i-=1;
        }

        i = 0;
        while(i<this.row){
            this.matrix[i][0] = '.';
            i+=1;
        }

    }
}

// NOTE: fungsi rotateRight hanya bisa dipakai saat piece berada di pojok kiri atas (awal)
public void rotateClockwise(){

    int i, j, mostRight, mostBottom;
    Piece temp = new Piece();
    temp.row = this.row;
    temp.col = this.col;
    temp.matrix = new char[temp.row][temp.col];
    i = 0;
    while(i<this.row){
        j = 0;
        while(j<this.col){
            temp.matrix[i][j] = MARK;
            j+=1;
        }
        i+=1;
    }

    // mencari panjang x piece
    mostRight = -1;
```

```
    i = this.col-1;
    while(i>=0){

        j = 0;
        while(j<this.row){

            if(this.matrix[j][i] != MARK){
                mostRight = i;
                break;
            }

            j+=1;
        }

        if(mostRight!=-1) break;

        i-=1;
    }

    // mencari panjang y piece
    mostBottom = -1;
    i = this.row-1;
    while(i>=0){

        j = 0;
        while(j<this.col){

            if(this.matrix[i][j] != MARK){
                mostBottom = i;
                break;
            }

            j+=1;
        }

        if(mostBottom!=-1) break;
```

```
            i-=1;
        }

        if(mostRight < this.row && mostBottom < this.col){

            i = 0;
            while(i<=mostBottom){
                j = 0;
                while(j<=mostRight){
                    temp.matrix[j][i] = this.matrix[mostBottom-i][j];
                    j+=1;
                }
                i+=1;
            }

            i = 0;
            while(i<this.row){
                this.matrix[i] = Arrays.copyOf(temp.matrix[i], this.col);
                i+=1;
            }

        }

    }

    // NOTE: fungsi mirrorX hanya bisa digunakan kalo piece-nya ada di pojok kiri atas (awal)
    public void mirrorX(){

        int i, j, mostRight;
        char temp;

        mostRight = -1;

        i = this.col-1;
        while(i>=0){

```

```
            j = 0;
            while(j<this.row){

                if(this.matrix[j][i] != MARK){
                    mostRight = i;
                    break;
                }

                j+=1;
            }

            if(mostRight!=-1) break;

            i-=1;
        }

        i = 0;
        while(i<mostRight-i){

            j = 0;
            while(j<this.row){
                temp = this.matrix[j][i];
                this.matrix[j][i] = this.matrix[j][mostRight-i];
                this.matrix[j][mostRight-i] = temp;
                j+=1;
            }

            i+=1;
        }

    }

    public void printPiece(){

        int i, j;
        char temp;

        i = 0;
        while(i<this.row){

            j = 0;
            while(j<this.col){

                temp = this.matrix[i][j];
```

```
                if(temp == 'A' || temp == 'H' || temp == 'O' || temp == 'V') System.err.print("\u001B[31m" + temp + "\u001B[0m");
                else if(temp == 'B' || temp == 'I' || temp == 'P' || temp == 'W') System.err.print("\u001B[32m" + temp + "\u001B[0m");
                else if(temp == 'C' || temp == 'J' || temp == 'Q' || temp == 'X') System.err.print("\u001B[33m" + temp + "\u001B[0m");
                else if(temp == 'D' || temp == 'K' || temp == 'R' || temp == 'Y') System.err.print("\u001B[34m" + temp + "\u001B[0m");
                else if(temp == 'E' || temp == 'L' || temp == 'S' || temp == 'Z') System.err.print("\u001B[35m" + temp + "\u001B[0m");
                else if(temp == 'F' || temp == 'M' || temp == 'T') System.err.print("\u001B[36m" + temp + "\u001B[0m");
                else if(temp == 'G' || temp == 'N' || temp == 'U') System.err.print("\u001B[35m" + temp + "\u001B[0m");
                else System.out.print(temp);

                j+=1;
            }

            System.out.println();

            i+=1;
        }

    }

    public void fillPieceWithMark(int startAt){
        int i;
        char[] temp = new char[0];

        i = startAt;
        while(i<this.row){
            this.matrix[i] = Arrays.copyOf(Solve.fillRowWithMark(temp, this.col), this.col);
            i+=1;
        }
    }

    public static String pieceToString(Piece piece){
        int i, j;
        String res;

        res = "";

        i = 0;
        while(i<piece.row){
```

```
            j = 0;
            while(j<piece.col){

                res+=piece.matrix[i][j];

                j+=1;
            }

            res+="\n";

            i+=1;
        }

        return res;
    }

    public static boolean areEqual(Piece a, Piece b){

        if(a.row != b.row || a.col != b.col) return false;

        int i, j;

        i = 0;
        while(i<a.row){

            j = 0;
            while(j<a.col){

                if(a.matrix[i][j] != b.matrix[i][j]) return false;

                j+=1;
            }

            i+=1;
        }

        return true;
    }

    public int pieceSpace(){
        int i, j, count;

        count = 0;
        i = 0;
        while(i<this.row){
```

```
            j = 0;
            while(j<this.col){
                if(this.matrix[i][j] != '.') count += 1;
                j+=1;
            }
            i+=1;
        }

        return count;
    }
}
```

Function/Solve.java

```
package Function;

import java.io.File;
import java.io.FileOutputStream;
import java.util.Arrays;
import java.util.Scanner;

import ADT.*;

public class Solve {

    public static int langkah;

    private static Piece[][] getPieces(int n, int m, int p, String s, String path){
        Piece[][] res = new Piece[p][n*m*8];
        Piece walkPiece = new Piece();
        char[] signature, temp;
        int i, j;
        Scanner readFile;
        String shifter;

        File file = new File(path).getAbsoluteFile();
        try{
            readFile = new Scanner(file);
        }catch (Exception e){
            System.out.println("Error: " + e);
            return res;
        }


        i = 0;
        while(i<p){

            j = 0;
            while(j<n*m*8){

                res[i][j] = new Piece();
                res[i][j].matrix = new char[n][m];
                res[i][j].row = n;
                res[i][j].col = m;

                j+=1;
            }

            i+=1;
        }

```

```
        shifter = readFile.nextLine();
        shifter = readFile.nextLine();

        signature = readFile.nextLine().toCharArray();
        temp = signature;

        temp = Solve.fillRowWithMark(temp, m);

        while(p>0){

            res[p-1][0].matrix[0] = Arrays.copyOf(temp, m);

            i = 1;
            if(readFile.hasNextLine()){

                temp = readFile.nextLine().toCharArray();
                temp = Solve.fillRowWithMark(temp, m);

                while(temp[0] == getSignature(signature)){

                    res[p-1][0].matrix[i] = Arrays.copyOf(temp, m);

                    if(readFile.hasNextLine()){
                        temp = readFile.nextLine().toCharArray();
                        temp = Solve.fillRowWithMark(temp, m);
                    }else{
                        i+=1;
                        break;
                    }

                    i+=1;
                }

            }

            res[p-1][0].fillPieceWithMark(i);

            // piece ditaruh di berbagai posisi

            i = 0;
            // normal - awal
            walkPiece = Piece.copyOf(res[p-1][0]);
            while(i<n*m*1){

                res[p-1][i] = Piece.copyOf(walkPiece);
```

```

                walkPiece.shiftRight();
                i+=1;
            }

            // normal - rotate 1
            walkPiece = Piece.copyOf(res[p-1][0]);
            walkPiece.rotateClockwise();
            while(i<n*m*2){

                res[p-1][i] = Piece.copyOf(walkPiece);

                walkPiece.shiftRight();
                i+=1;
            }

            // normal - rotate 2
            walkPiece = Piece.copyOf(res[p-1][0]);
            walkPiece.rotateClockwise();
            walkPiece.rotateClockwise();
            while(i<n*m*3){

                res[p-1][i] = Piece.copyOf(walkPiece);

                walkPiece.shiftRight();
                i+=1;
            }

            // normal - rotate 3
            walkPiece = Piece.copyOf(res[p-1][0]);
            walkPiece.rotateClockwise();
            walkPiece.rotateClockwise();
            walkPiece.rotateClockwise();
            while(i<n*m*4){

                res[p-1][i] = Piece.copyOf(walkPiece);

                walkPiece.shiftRight();
                i+=1;
            }

            // mirror - awal
            walkPiece = Piece.copyOf(res[p-1][0]);
            walkPiece.mirrorX();
            while(i<n*m*5){
```

```
                res[p-1][i] = Piece.copyOf(walkPiece);

                walkPiece.shiftRight();
                i+=1;
            }

            // mirror - rotate 1
            walkPiece = Piece.copyOf(res[p-1][0]);
            walkPiece.mirrorX();
            walkPiece.rotateClockwise();
            while(i<n*m*6){

                res[p-1][i] = Piece.copyOf(walkPiece);

                walkPiece.shiftRight();
                i+=1;
            }

            // mirror - rotate 2
            walkPiece = Piece.copyOf(res[p-1][0]);
            walkPiece.mirrorX();
            walkPiece.rotateClockwise();
            walkPiece.rotateClockwise();
            while(i<n*m*7){

                res[p-1][i] = Piece.copyOf(walkPiece);

                walkPiece.shiftRight();
                i+=1;
            }

            // mirror - rotate 3
            walkPiece = Piece.copyOf(res[p-1][0]);
            walkPiece.mirrorX();
            walkPiece.rotateClockwise();
            walkPiece.rotateClockwise();
            walkPiece.rotateClockwise();
            while(i<n*m*8){

                res[p-1][i] = Piece.copyOf(walkPiece);

                walkPiece.shiftRight();
                i+=1;
            }
```

```
            signature = Arrays.copyOf(temp, temp.length);
            p-=1;
        }

        readFile.close();

        return res;
    }

    private static boolean isFit(Piece a, Piece b){
        int i, j;

        i = 0;
        while(i<a.row){

            j = 0;
            while(j<a.col){
                if(a.matrix[i][j] != '.' && b.matrix[i][j] != '.'){
                    return false;
                }
                j+=1;
            }

            i+=1;
        }

        return true;
    }

    private static boolean isFull(Piece piece){
        int i, j;

        i = 0;
        while(i<piece.row){
            j = 0;
            while(j<piece.col){
                if(piece.matrix[i][j] == '.') return false;
                j+=1;
            }
            i+=1;
        }

        return true;
    }
```

```java

    private static Piece placePiece(Piece board, Piece piece){

        // Prekondisi: board dan piece sudah pasti fit

        int i, j;

        i = 0;
        while(i<board.row){

            j = 0;
            while(j<board.col){

                if(piece.matrix[i][j] != '.') board.matrix[i][j] = piece.matrix[i][j];

                j+=1;
            }

            i+=1;
        }

        return board;
    }

    public static char[] fillRowWithMark(char[] s, int m){

        char[] hasil = new char[m];
        int i;

        i = 0;
        while(i<s.length){
            if(s[i] == ' '){
                hasil[i] = '.';
            }else{
                hasil[i] = s[i];
            }
            i+=1;
        }

        while(i<m){
            hasil[i] = '.';
            i+=1;
        }

        return hasil;
    }
```

```
274
275     private static boolean isProbablySolvable(Piece[][] pieces, int p){
276         // menentukan apakah luas piece-piece-nya sama atau engga dengan wadah
277
278         int i, count;
279
280         count = 0;
281         i = 0;
282         while(i<p){
283             count += pieces[i][0].pieceSpace();
284             i+=1;
285         }
286
287         if(count == pieces[0][0].row * pieces[0][0].col) return true;
288         else return false;
289     }
290
291     private static Piece permutation(Piece board, Piece[][] pieces, int pieceIndex){
292
293         if(isFull(board)){
294             if(pieceIndex < 0){
295                 return board;
296             }else{
297                 return Piece.emptyBoard(board.row, board.col);
298             }
299         }else if(pieceIndex < 0){
300             return Piece.emptyBoard(board.row, board.col);
301         }else{
302
303             Piece tempBoard, res;
304             int i;
305
306             i = 0;
307             while(i<board.row*board.col*8){
308                 langkah+=1;
309                 if(i>0){
310                     if(i % (board.row * board.col) == 0){
311                         if(Piece.areEqual(pieces[pieceIndex][i], pieces[pieceIndex][0])){
312                             break;
313                         }
314                     }else if(Piece.areEqual(pieces[pieceIndex][i], pieces[pieceIndex][i-1])){
315                         i = ((i / (board.row * board.col)) + 1) * board.row * board.col;
316                         if(i==board.row*board.col*8) break;
317                     }
318                 }
319
320                 if(isFit(board, pieces[pieceIndex][i])){
321                     tempBoard = placePiece(Piece.copyOf(board), pieces[pieceIndex][i]);
322
323                     res = permutation(tempBoard, pieces, pieceIndex-1);
324
325                     if(isFull(res)) return res;
326                 }
327                 i+=1;
328             }
329
330             return Piece.emptyBoard(board.row, board.col);
331         }
332
333     }
334
335     public static Piece solve(int n, int m, int p, String s, String path){
336
337         Piece res = new Piece();
338         Piece[][] pieces = getPieces(n, m, p, s, path);
339
340         langkah = 0;
341         if(!isProbablySolvable(pieces, p)) return Piece.emptyBoard(n, m);
342
343         langkah+=1;
344         res = permutation(Piece.emptyBoard(n, m), pieces, p-1);
345
346         return res;
347     }
348
349     public static void writeStringToFile(String s, String path){
350
351         try{
352             FileOutputStream output = new FileOutputStream(path);
353
354             output.write(s.getBytes());
355
356             output.close();
357
358             System.out.println();
359             System.out.println("Solusi berhasil tersimpan di " + path);
360         }catch (Exception e){
361             System.out.println();
362             System.out.println("Error: " + e);
363         }
364     }
```

```java

    private static char getSignature(char[] signature){
        int i;

        i = 0;
        while(i<signature.length){
            if(signature[i] >= 65 && signature[i] <= 90) return signature[i];
            i+=1;
        }

        return ' ';
    }
}

```

Contoh *Input* dan *Output*

Test Case 0 (dari Spesifikasi)

Input

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

```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 0.txt

Diperoleh solusi sebagai berikut:
AGGGD
AABDD
CCBBE
CFFEE
FFFEE

Waktu yang diperlukan: 148 ms

Banyak kasus yang ditinjau: 2731226

Apakah Anda ingin menyimpan solusi? (ya/tidak)
>
```

Test Case 1 - Semua Huruf, Berhasil

Input

```
3 9 26
DEFAULT
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
ZZ
```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 1.txt

Diperoleh solusi sebagai berikut:
ABCDEFGHI
JKLMNOPQR
STUVWXYZZ

Waktu yang diperlukan: 9 ms

Banyak kasus yang ditinjau: 350

Apakah Anda ingin menyimpan solusi? (ya/tidak)
> tidak
```

Test Case 2 - Semua Huruf, Piece Kurang

Input

```
3 9 26
DEFAULT
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

```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 2.txt

Piece yang ada tidak sesuai dengan board!

Program ini dapat membantumu menemukan solusi dari IQ Puzzler Pro. Ingin mencoba?
1. Input File
0. Keluar
>
```

Test Case 3 - Piece dengan Rongga, Berhasil

Input

```
3 3 3
DEFAULT
  A
AAA
A
BB
CC

```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 3.txt

Diperoleh solusi sebagai berikut:
BBA
AAA
ACC

Waktu yang diperlukan: 43 ms

Banyak kasus yang ditinjau: 9

Apakah Anda ingin menyimpan solusi? (ya/tidak)
>
```

Test Case 4 - Piece dengan Rongga, Tidak Ada Solusi

Input

```
3 3 3
DEFAULT
  A
AAA
A
B
BB
C

```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 4.txt

Tidak ditemukan solusi untuk mengisi board!

Waktu yang diperlukan: 4 ms

Banyak kasus yang ditinjau: 274

Apakah Anda ingin menyimpan solusi? (ya/tidak)
>
```

Test Case 5 - Perlu Rotasi

Input

```
3 3 2
DEFAULT
A
AA
A
BB
B
BB

```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 5.txt

Diperoleh solusi sebagai berikut:
ABB
AAB
ABB

Waktu yang diperlukan: 4 ms

Banyak kasus yang ditinjau: 8

Apakah Anda ingin menyimpan solusi? (ya/tidak)
>
```

Test Case 6 - Perlu Mirror

Input

```
5 5 6
DEFAULT
AAA
 B
BBBBB
CCCCC
DDDD
F
   G
GGGGG

```

Output

```
Silakan masukkan nama file, diakhiri dengan ".txt"
> 6.txt

Diperoleh solusi sebagai berikut:
AAABF
BBBBB
CCCCC
DDDDG
GGGGG

Waktu yang diperlukan: 48 ms

Banyak kasus yang ditinjau: 111531

Apakah Anda ingin menyimpan solusi? (ya/tidak)
>
```

Pranala *Repository*

https://github.com/izzatjundy/Tucil1_13523092

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	✓	