Laporan Hasil Tugas Kecil 1 Strategi Algoritma

Aramazaya
13523082

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

DATFTAR ISI

1. Penjelasan Program

Program memiliki dua kelas utama, yaitu kelas *map* atau peta dan kelas *blocks* atau balok. Program berusaha untuk mencari solusi dimana setiap balok dapat disimpan di dalam peta dengan orientasi seperti apapun. Peta merupakan objek dengan atribut *grid* yang merupakan *matrix of char*. Sementara *blocks* merupakan objek dengan atribut *block, character,* dan *lastPos*. *Block* merupakan matriks angka bentuk dari balok, *character* merupakan representasi huruf balok, dna *lastPos* merupakan variable posisi matriks pada matriks peta yang digunakan dalam algoritma.

2. Algoritma dan Penjelasan

Algoritma Bruteforce yang digunakan adalah sebagai berikut. Pertama algoritma akan meletakan objek *blocks* pertama di sebuah list of *blocks*. Cara meletakan objek adalah dengan mengecek matriks angka pada objek bloks dengan matriks peta yang digunakan oleh program. Jika setiap '1' di matriks angka objek berkorelasi dengan elemen 'x' pada matriks peta, maka *block* dapat disimpan pada posisi tersebut.

```
public boolean placeBlock(blocks block){
        if (block.lastPos[0]+block.block.length > this.grid.length || block.lastPos[1]+block.block[0].length > 
this.grid[0].length){
            return false;
        }
        for (int i = 0; i < block.block.length; i++){
            for (int j = 0; j < block.block[0].length; j++){
                if (this.grid[block.lastPos[0]+i][block.lastPos[1]+j] != 'x'){
                    if (block.block[i][j] == 1){
                        return false;
                    }
                }
            }
        }

        for (int i = 0; i < block.block.length; i++){
            for (int j = 0; j < block.block[0].length; j++){
                if (block.block[i][j] == 1){
                    this.grid[block.lastPos[0]+i][block.lastPos[1]+j] = block.character;
                }
            }
        }
        return true;
    }
```

Selanjutnya digunakan stack untuk menyimpan *state* peta, rotasi, dan *mirroring block* yang telah disimpan. Setelah *block* berhasil disimpan pada sebuah peta, hasil peta baru dan kondisi balok (rotasi, mirroring) di-*push* ke dalam stack.

```
if (map.placeBlock(currentBlock)){
    mapStack.push(new map(map.grid.length, map.grid[0].length));
    mapStack.peek().grid = copyGrid(map.grid);
    state.push(new int[]{currentBlockIDX, i, j, k, l});
    succeed = true;
    startRow = 0;
    startCol = 0;
    currentBlockIDX++;
```

```
    break;
}
```

Lalu pencarian lokasi untuk meletakan *block* selanjutnya dimulai. Program mencari dengan cara mencoba meletakan *block* pada tiap titik di peta yang memungkinkan. Setelah itu, program kembali ke titik 0,0 dan mencoba dengan orientasi *block* yang baru. Jika tidak ditemukan lokasi, maka program melakukan *backtracking* ke *state* sebelum dilakukan pencarian *block* tersebut dengan menggunakan *stack*.

```
map last = mapStack.pop();
last.printMap();
System.out.println("CHARACTER: " + currentBlock.character);
int[] lastState = state.pop();
map.grid = copyGrid(mapStack.peek().grid);
currentBlockIDX = lastState[0];
currentBlock = bList.get(currentBlockIDX);
int rot = lastState[1];
for (int i = 0; i < rot; i++){
    currentBlock.rotateBlock();
}
int mir = lastState[2];
for (int i = 0; i < mir; i++){
    currentBlock.mirrorBlock();
}
int[] location = new int[]{lastState[3], lastState[4]};
startRow = location[0];
startCol = location[1] + 1;
if (startCol >= map.grid[0].length) {
    startCol = 0;
    startRow++;
}
```

Jika hasil ditemukan maka program berhenti. Jika hasil tidak ditemukan maka program mengeluarkan luaran “The Puzzle has no Solution”.

3. Testing Program

Test1 :
3 2 2
DEFAULT
AA
A
B
BB

Hasil :

```
PS C:\algeo\Tucil1_13523082> java -cp bin main
Masukkan nama file:
test.txt
The Puzzle has been solved
AA
AB
BB
```

Test2:
5 5 3
DEFAULT
AAA
A A
BBBBB
CC
CC

Hasil :

```
PS C:\algeo\Tucil1_13523082> java -cp bin main
Masukkan nama file:
test2.txt
The Puzzle has been solved
AAACC
AxACC
BBBBB
xxxxx
xxxxx
```

Test 3 :
3 3 3
DEFAULT
AAA
B B
C

Hasil:

```
PS C:\algeo\Tucil1_13523082> java -cp bin main
Masukkan nama file:
test3.txt
The Puzzle has been solved
AAA
BCB
xxx
```

Test 4:
3 3 3
DEFAULT
AAA

A
B
CCC
C

Hasil:

```
PS C:\algeo\Tucil1_13523082> java -cp bin main
Masukkan nama file:
test4.txt
The Puzzle has been solved
ABx
CCC
Cxx
```

Test 5 :
3 3 1
DEFAULT
AAAAA

Hasil:

```
PS C:\algeo\Tucil1_13523082> java -cp bin main
Masukkan nama file:
test5.txt
The Puzzle has No Solution
Time Elapsed: 0ms
Iteration: 72
```

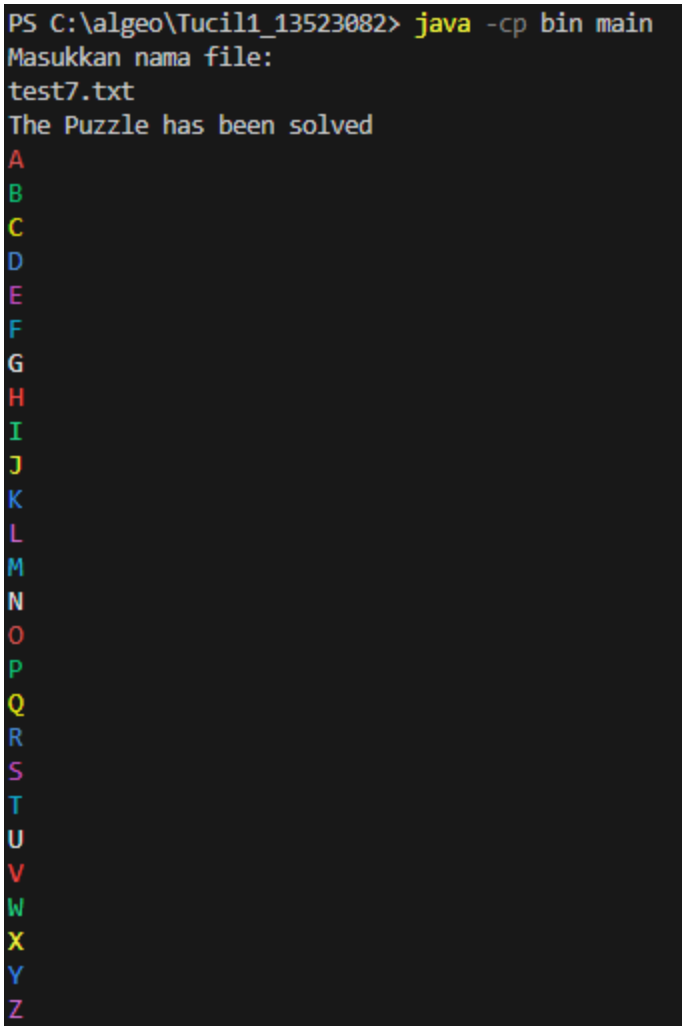
Test 6:
4 4 5
DEFAULT
AAAA
AA
BB
CC
DDDD
DD

Hasil:

```
PS C:\algeo\Tucil1_13523082> java -cp bin main
Masukkan nama file:
test6.txt
An error occurred.
java.util.NoSuchElementException
        at java.base/java.util.ArrayList$Itr.next(ArrayList.java:1053)
        at java.base/java.util.Collections.max(Collections.java:698)
        at scan.read_line(scan.java:58)
        at main.main(main.java:35)
The Puzzle has been solved
AAAA
AABB
CCDD
DDDD
```

Test 7:
26 1 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

Haisl:



4. Source Code Program

Source code dapat dilihat pada lampiran I atau pada github pada tautan https://github.com/Aramazaya/Tucil1_13523082.

LAMPIRAN I

File main.java:

```
import java.util.*;

class IntWrapper {
    public int value;
    public IntWrapper(int value){
        this.value = value;
    }
}

class CharWrapper {
    public char value;
    public CharWrapper(char value){
        this.value = value;
    }
}

public class main {
    public static char[][] copyGrid(char[][] original) {
        char[][] copy = new char[original.length][original[0].length];
        for (int i = 0; i < original.length; i++) {
            System.arraycopy(original[i], 0, copy[i], 0, original[i].length);
        }
        return copy;
    }
    public static void main(String[] args) {
        Scanner scans = new Scanner(System.in);
        System.out.println("Masukkan nama file: ");
        String filename = scans.nextLine();
        IntWrapper M = new IntWrapper(0);
        IntWrapper N = new IntWrapper(0);
        IntWrapper P = new IntWrapper(0);
        String S = "";
        scans.close();
        List<blocks> bList = new ArrayList<>();
        scan.read_line(filename, M, N, P, S, bList);
        map map = new map(M.value, N.value);
        Stack<map> mapStack = new Stack<>();
        Stack<int[]> state = new Stack<>();
        int currentBlockIDX = 0;
        mapStack.push(new map(M.value, N.value));
        mapStack.peek().grid = copyGrid(map.grid);
        state.push(new int[]{currentBlockIDX, 0, 0, 0, 0});
```

```
        boolean solveFlag = false;
        int iter = 0;
        int startRow = 0;
        int startCol = 0;
        long start = System.nanoTime();
        while (currentBlockIDX < bList.size()){
            blocks currentBlock = bList.get(currentBlockIDX);
            boolean succeed = false;
            for (int i = 0; i < 4; i++){
                for (int j = 0; j < 2; j++){
                    for (int k = startRow; k < map.grid.length; k++){
                        for (int l = startCol; l < map.grid[0].length; l++){
                            iter++;
                            currentBlock.lastPos = new int[]{k, l};
                            if (map.placeBlock(currentBlock)){
                                mapStack.push(new map(map.grid.length,
map.grid[0].length));
                                mapStack.peek().grid = copyGrid(map.grid);
                                state.push(new int[]{currentBlockIDX, i, j, k, l});
                                succeed = true;
                                startRow = 0;
                                startCol = 0;
                                currentBlockIDX++;
                                break;
                            }
                        } if (succeed){break;}
                    }if (succeed){break;}
                    currentBlock.mirrorBlock();
                }if (succeed){break;}
                currentBlock.mirrorBlock();
                currentBlock.rotateBlock();
            }
            currentBlock.rotateBlock();
            if (!succeed){
                if (mapStack.size() == 1){
                    System.out.println("The Puzzle has No Solution");
                    long end = System.nanoTime();
                    System.out.println("Time Elapsed: " + (end-start)/1000000 +
"ms");
                    System.out.println("Iteration: " + iter);
                    break;
                } else {
                    map last = mapStack.pop();
                    last.printMap();
                    System.out.println("CHARACTER: " + currentBlock.character);
                    int[] lastState = state.pop();
                    map.grid = copyGrid(mapStack.peek().grid);
```

```
                    currentBlockIDX = lastState[0];
                    currentBlock = bList.get(currentBlockIDX);
                    int rot = lastState[1];
                    for (int i = 0; i < rot; i++){
                        currentBlock.rotateBlock();
                    }
                    int mir = lastState[2];
                    for (int i = 0; i < mir; i++){
                        currentBlock.mirrorBlock();
                    }
                    int[] location = new int[]{lastState[3], lastState[4]};
                    startRow = location[0];
                    startCol = location[1] + 1;
                    if (startCol >= map.grid[0].length) {
                        startCol = 0;
                        startRow++;
                    }
                }
            }
        }
        if (currentBlockIDX == bList.size()){
            solveFlag = true;
        }
        if (solveFlag){
            long end = System.nanoTime();
            System.out.println("The Puzzle has been solved");
            map = mapStack.pop();
            map.printMap();
            System.out.println();
            System.out.println("Time Elapsed: " + (end-start)/1000000 + "ms");
            System.out.println("Iteration: " + iter);
            System.out.print("Save To File(Y/N): ");
            Scanner scanner = new Scanner(System.in);
            String save = scanner.nextLine();
            if (save.equals("Y")){
                map.printResult((end-start)/1000000, iter);
            }
        }
    }
}
```

File blocks.java:

```
public class blocks {
    public int[][] block;
```

```
    public char character;
    public int[] lastPos = new int[2];
    public int rotation;
    public blocks(int[][] block, char character) {
        this.block = block;
        this.character = character;
        this.lastPos[0] = 0;
        this.lastPos[1] = 0;
        this.rotation = 0;
    }

    public void rotateBlock(){
        int[][] matrix = new int[this.block[0].length][this.block.length];
        for (int i = 0; i < this.block.length; i++){
            for (int j = 0; j < this.block[0].length; j++){
                matrix[j][i] = this.block[i][j];
            }
        }
        this.block = matrix;
        this.rotation++;
    }

    public void mirrorBlock(){
        int[][] matrix = new int[this.block.length][this.block[0].length];
        for (int i = 0; i < this.block.length; i++){
            for (int j = 0; j < this.block[0].length; j++){
                matrix[i][this.block[0].length-1-j] = this.block[i][j];
            }
        }
        this.block = matrix;
    }
}
```

File map.java:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Map;

public class map {
    char[][] grid;
    char[][] nextGrid;
    public map(int row, int col){
```

```
        this.grid = new char[row][col];
        this.nextGrid = new char[row][col];
        for (int i = 0; i < row; i++){
            for (int j = 0; j < col; j++){
                this.grid[i][j] = 'x';
                this.nextGrid[i][j] = 'x';
            }
        }
    }
    public void resetGrid(){
        for (int i = 0; i < this.grid.length; i++){
            for (int j = 0; j < this.grid[0].length; j++){
                this.grid[i][j] = 'x';
            }
        }
    }
    public boolean placeBlock(blocks block){
        if (block.lastPos[0]+block.block.length > this.grid.length ||
block.lastPos[1]+block.block[0].length > this.grid[0].length){
            return false;
        }
        for (int i = 0; i < block.block.length; i++){
            for (int j = 0; j < block.block[0].length; j++){
                if (this.grid[block.lastPos[0]+i][block.lastPos[1]+j] != 'x'){
                    if (block.block[i][j] == 1){
                        return false;
                    }
                }
            }
        }

        for (int i = 0; i < block.block.length; i++){
            for (int j = 0; j < block.block[0].length; j++){
                if (block.block[i][j] == 1){
                    this.grid[block.lastPos[0]+i][block.lastPos[1]+j] =
block.character;
                }
            }
        }
        return true;
    }

    private static final Map<Character, String> colorMap = new HashMap<>();
    private static final String RESET = "\u001B[0m";

    static {
        String[] colors ={
```

```
            "\u001B[31m",
            "\u001B[32m",
            "\u001B[33m",
            "\u001B[34m",
            "\u001B[35m",
            "\u001B[36m",
            "\u001B[37m",
            "\u001B[91m",
            "\u001B[92m",
            "\u001B[93m",
            "\u001B[94m",
            "\u001B[95m",
            "\u001B[96m",
            "\u001B[97m"
        };
        for (char letter = 'A'; letter <= 'Z'; letter++) {
            colorMap.put(letter, colors[(letter - 'A') % colors.length]);
        }
    }
    public void printMap(){
        for (int i = 0; i < this.grid.length; i++){
            for (int j = 0; j < this.grid[0].length; j++){
                if (this.grid[i][j] == 'x'){
                    System.out.print("x");
                } else {
                    String color = colorMap.getOrDefault(this.grid[i][j], RESET);
                    System.out.print(color + this.grid[i][j] + RESET);
                }
            }
            System.out.println();
        }
    }
    public void printResult(long time, int iter){
        try {
            PrintWriter writer = new PrintWriter(new File("test", "output.txt"));
            writer.println("Waktu: " + time + " ms");
            writer.println("Iterasi : " + iter);
            for (int i = 0; i < this.grid.length; i++){
                for (int j = 0; j < this.grid[0].length; j++){
                    if (this.grid[i][j] == 'x'){
                        writer.print("x");
                    } else {
                        String color = colorMap.getOrDefault(this.grid[i][j],
RESET);
                        writer.print(color + this.grid[i][j] + RESET);
                    }
                }
```

```
                writer.println();
            }
        } catch (FileNotFoundException e) {
            System.out.println("Error: Could not create file.");
            e.printStackTrace();
        }
    }
}
```

File scan.java:

```
public class scan {
    public static void read_line(String filename, IntWrapper M, IntWrapper N,
IntWrapper P, String S, List<blocks> bList){
        try {
            File file = new File("test", filename);
            Scanner scanner = new Scanner(file);
            String line = scanner.nextLine();
            String[] Arr = line.split(" ");
            M.value = Integer.parseInt(Arr[0]);
            N.value = Integer.parseInt(Arr[1]);
            P.value = Integer.parseInt(Arr[2]);
            line = scanner.nextLine();
            S = line;
            int[][] Matrix;
            char Current = 'A';
            List<String> lines = new ArrayList<>();
            while (scanner.hasNextLine()){
                line = scanner.nextLine();
                lines.add(line);
            }
            scanner.close();
            int curline = 0;
            int copyP = P.value;
            boolean flag = false;
            List<Integer> width = new ArrayList<>();
            List<String> block = new ArrayList<>();
            while (copyP > 0){
                if (copyP != 0 && curline == lines.size()){
                    System.out.println("Not Enough Blocks");
                    break;
                }
                block.clear();
                width.clear();
```

```
                while (true) {
                    line = lines.get(curline);
                    Arr = line.split("");
                    for (int i = 0; i < Arr.length; i++){
                        if (line.charAt(i) != Current && line.charAt(i) != ' '){
                            flag = true;
                            break;
                        }
                    }
                    if (flag){
                        flag = false;
                        break;
                    }
                    if (curline == lines.size()-1){
                        block.add(line);
                        width.add(Arr.length);
                        break;
                    }
                    block.add(line);
                    width.add(Arr.length);
                    curline++;
                }
                int maxW = Collections.max(width);
                Matrix = new int[block.size()][maxW];
                for (int i = 0; i < block.size(); i++){
                    Arr = block.get(i).split("");
                    for (int j = 0; j < maxW; j++){
                        if (j >= Arr.length){
                            Matrix[i][j] = 0;
                        } else if (Arr[j].equals(" ")){
                            Matrix[i][j] = 0;
                        } else {
                            Matrix[i][j] = 1;
                        }
                    }
                }
                bList.add(new blocks(Matrix, Current));
                Current++;
                copyP--;
            }
        }
        catch (Exception e){
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

LAMPIRAN II

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	✅	