Tugas Kecil #1 - IF2211 Strategi Algoritma

Muhammad Adli Arindra
18222089
K2

Daftar Isi

Rumusan Masalah

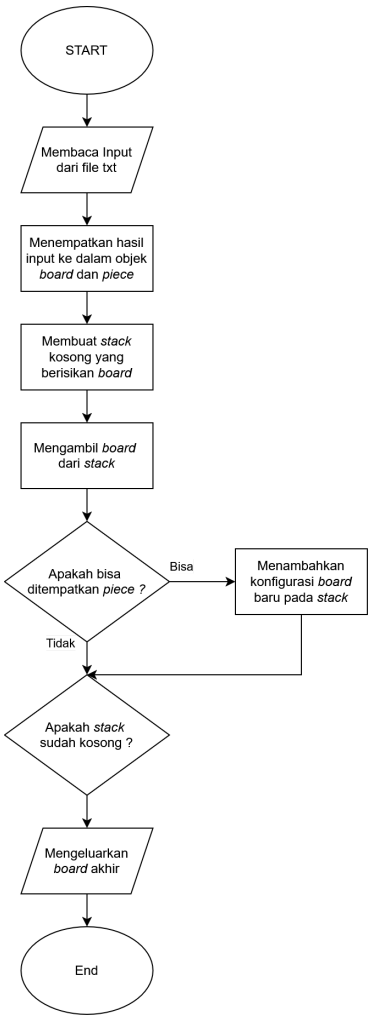
Diberikan sebuah *puzzle* bernama *IQ Puzzle Pro* yang bertujuan untuk **mengisi sebuah papan bermain hingga penuh** menggunakan potongan-potongan yang disediakan. Papan yang memiliki panjang N dan lebar M. Jumlah potongan yang tersedia berjumlah P. Terdapat banyak kombinasi cara menempatkan potongan-potongannya yang sangat banyak. Bagaimana cara mendapatkan kombinasi yang berhasil menyelesaikan permasalahan tersebut ?

Tujuan

Dalam pemrograman, terdapat sebuah jenis algoritma yang bernama *brute force*. *Brute force* adalah pendekatan langsung yang **mencoba semua kemungkinan solusi** tanpa optimasi. Meskipun tidak efisien untuk masalah besar, metode ini tetap digunakan dalam beberapa kasus seperti keamanan siber (brute force attack pada kata sandi) atau pemrograman kompetitif untuk masalah kecil.

Dalam tugas ini, algoritma *brute force* akan digunakan untuk memecahkan masalah *IQ Puzzle Pro*. Meskipun algoritma tidak akan berjalan secara efisien, namun karena sifat algoritmanya yang menjelajahi semua kemungkinan, solusi dari masalah pasti dapat ditemukan bila memang ada.

Penjelasan Algoritma



Pada gambar di samping, terdapat sebuah gambar yang menggambarkan alur program ini bekerja. Secara garis besar, program akan selalu menjalankan urutan algoritma tersebut untuk dapat menghasilkan konfigurasi yang dipilih.

Pertama, program akan membaca input dari sebuah *file* txt yang diberikan. Baris pertama dari *file* tersebut berisikan N dan M yang merupakan ukuran dari papan tersebut dan P yang merupakan banyak potongan yang harus dimasukan kepada papan. Baris kedua akan berisikan variabel *s* yang merupakan jenis dari solusi yang diberikan (untuk program ini, jenis solusi hanyalah **DEFAULT**). Lalu barisan-barisan berikutnya berisikan informasi potongan-potongan yang ada yang jumlahnya sama dengan P.

Kedua, informasi yang didapat dari *file* txt akan dimasukan ke dalam objek *Piece* dan *Board*. Informasi-informasi tersebut disimpan dalam sebuah objek dari *class* agar kelak pengelolaan informasi tersebut lebih mudah dan dikelompokkan dengan lebih jelas.

Pada *class Piece*, terdapat dua variabel yaitu karakter apa yang digunakan untuk potongan tersebut dan juga variasi dari bentuk potongan tersebut. Variasi yang dimaksud adalah segala jenis bentuk yang dihasilkan dari potongan yang dihasilkan dengan cara memutar dan/atau membalikkan potongan tersebut. Semua variasi ini disimpan agar kelak lebih mudah untuk mencari semua kemungkinan dari penempatan potongan pada papan.

Lalu pada *class Board*, informasi yang disimpan adalah *state* dari papan sekarang dan apakah papan sudah terisi hingga penuh atau belum.

Ketika sebuah potongan ditempatkan pada papan, *state* dari papan tersebut akan diperbarui sehingga program bisa mengetahui langkah apa yang selanjutnya harus diambil untuk memenuhi papan tersebut.

Setelah semua informasi disimpan, dibuatlah sebuah *Stack* yang berisikan semua kemungkinan papan-papan yang mungkin dari potongan-potongan yang tersedia. Pada awalnya, *stack* berisikan papan kosong dan seiring dengan berjalannya program *stack* akan berisi papan-papan yang sudah ditempati oleh potongan-potongan yang ada.

Lalu program akan menjalankan sebuah *while loop* yang akan jalan apabila solusi belum ditemukan dan *stack* belum kosong. Program akan menempatkan sebuah potongan pada papan di posisi yang belum terisi dan menambahkan pada papan apabila memungkinkan dan konfigurasi papan yang baru akan disimpan pada *stack*. Lalu program akan mengulang algoritma tersebut terus menerus hingga papan berhasil diisi dengan penuh.

Isi Program dan Pengetesan Program

Isi Program

Main.java

```
import java.util.List;
import java.util.Scanner;
import java.util.Stack;

import objects.Board;
import objects.Piece;
import utils.Input;
import utils.Timer;

public class Main {
    public static List<Piece> pieces;
    public static Stack<Board> boards;
    public static int iterations = 0;
    public static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {

        System.out.print("Masukan nama file txt: ");
        String path = scanner.nextLine();
        Input.read(path);
        pieces = Input.getPieces();

        boards = new Stack<>();
        Board emptyBoard = new Board(Input.n, Input.m);
        boards.push(emptyBoard);

        Timer.start();
        solve();
```

```
    }

    public static void solve() {
        while (!boards.isEmpty()) {
            Board currentBoard = boards.pop();
            if (currentBoard.isComplete()) {
                onComplete(currentBoard);
                return;
            }
            Piece currentPiece = pieces.get(currentBoard.piecesCount);
                                for   (List<List<Integer>>   variation   :
currentPiece.getVariations()) {
                int pieceWidth = variation.get(0).size();
                int pieceHeight = variation.size();

                for (int i = 0; i <= Input.n - pieceWidth; ++i) {
                    for (int j = 0; j <= Input.m - pieceHeight; ++j) {
                        Board newBoard = new Board(currentBoard);
                          boolean status = newBoard.addPiece(variation, i,
j, currentPiece.getC());
                        if (status) {
                            boards.push(newBoard);
                        }
                        iterations++;
                    }
                }
            }
        }
        System.out.println("Configuration not found!");
    }

    public static void onComplete(Board completedBoard) {
        System.out.println("Configuration found!\n");
        completedBoard.print();
```

```
        Timer.stop();
        System.out.println("Total iterations: " + iterations);
        System.out.println("Time spent: " + Timer.get() + " ms");
        System.out.print("Apakah anda ingin menyimpan solusi? (y/n): ");
        Character command = scanner.next().charAt(0);

        if (command == 'y') {
            completedBoard.writeToFile(iterations, Timer.get());
            System.out.println();
                    System.out.println("Berhasil  menyimpan  file  dengan
nama\u001B[34m output.txt \u001B[0m");
            System.out.println();
        }

        scanner.close();
    }
}
```

Board.java

```
package objects;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Board {
    public List<List<Character>> state = new ArrayList<>();
    public static final char EMPTY = '.';
```

```
    public int n, m;
    public int piecesCount = 0;

    private static final String RESET = "\u001B[0m";
    private static final String[] COLORS = {
        "\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[97m",   "\u001B[90m",   "\u001B[38;5;198m",
"\u001B[38;5;208m",
             "\u001B[38;5;214m",  "\u001B[38;5;220m",  "\u001B[38;5;226m",
"\u001B[38;5;82m",
               "\u001B[38;5;46m",  "\u001B[38;5;21m",  "\u001B[38;5;201m",
"\u001B[38;5;93m",
        "\u001B[38;5;129m", "\u001B[38;5;166m"
    };
    private final Map<Character, String> colorMap = new HashMap<>();

    public Board(int n, int m) {
        this.n = n;
        this.m = m;
        for (int i = 0; i < m; ++i) {
            List<Character> line = new ArrayList<>();
            for (int j = 0; j < n; ++j) {
                line.add(EMPTY);
            }
            this.state.add(line);
        }
    }

    public Board(Board b) {
        this.n = b.n;
        this.m = b.m;
        this.piecesCount = b.piecesCount;
```

```
        this.state = new ArrayList<>();

        for (List<Character> row : b.state) {
            this.state.add(new ArrayList<>(row));
        }
    }

    public boolean isComplete() {
        boolean ret = true;
        for (List<Character> line : this.state) {
            for (int i = 0; i < line.size(); ++i) {
                if (line.get(i) == EMPTY) ret = false;
            }
        }
        return ret;
    }

    public boolean addPiece(List<List<Integer>> piece, int x, int y, char
letter) {
        int lenX = piece.get(0).size();
        int lenY = piece.size();

        for (int i = 0; i < lenX; ++i) {
            for (int j = 0; j < lenY; ++j) {
                if (!insideBoard(i + x, j + y)) return false;
                if (piece.get(j).get(i) == 1) {
                      if (this.state.get(j + y).get(i + x) != '.') return
false;
                }
            }
        }

        for (int i = 0; i < lenX; ++i) {
            for (int j = 0; j < lenY; ++j) {
```

```
                if (piece.get(j).get(i) == 1) {
                    this.state.get(j + y).set(i + x, letter);
                }
            }
        }
        this.piecesCount ++;

        return true;
    }

    private boolean insideBoard(int x, int y) {
        if (x < 0 || x >= this.n || y < 0 || y >= this.m) return false;
        return true;
    }

    private void assignColors() {
        char[] alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".toCharArray();
        for (int i = 0; i < alphabet.length; i++) {
            colorMap.put(alphabet[i], COLORS[i]);
        }
    }

    public void print() {
        assignColors();
        for (List<Character> line : this.state) {
            for (Character c : line) {
                                                    String   color   =
colorMap.getOrDefault(Character.toUpperCase(c), RESET);
                System.out.print(color + c + RESET);
            }
            System.out.println();
        }
        System.out.println();
    }
```

```
    public void writeToFile(int iterations, long time) {
                try  (BufferedWriter  writer  =  new  BufferedWriter(new
FileWriter("output.txt"))) {
            for (List<Character> line : this.state) {
                for (Character ch : line) {
                    writer.write(ch);
                }
                writer.newLine();
            }
            writer.newLine();
            writer.write("Total iterations: " + iterations + '\n');
            writer.write("Time spent: " + time + " ms\n");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Piece.java

```
package objects;
import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

import utils.Transformer;

public class Piece {
    private char c;
    private List<List<List<Integer>>> variations;

    public Piece(List<String> piece) {
        this.c = '!';
```

```
        String firstline = piece.get(0);
        for (int i = 0; i < firstline.length(); ++i) {
            if (this.c == '!' && firstline.charAt(i) != ' ') {
                this.c = firstline.charAt(i);
            }
        }
        this.makeVariations(piece);
    }

    private void makeVariations(List<String> piece) {
        this.variations = new ArrayList<>();
        List<List<Integer>> shape = new ArrayList<>();
        int maxLen = 0;
        for (int i = 0; i < piece.size(); ++i) {
            maxLen = Math.max(maxLen, piece.get(i).length());
            List<Integer> currentLine = new ArrayList<>();
            for (int j = 0; j < piece.get(i).length(); ++j) {
                char currentChar = piece.get(i).charAt(j);
                if (currentChar != ' ') {
                    currentLine.add(1);
                }
            }
            shape.add(currentLine);
        }
        for (int i = 0; i < shape.size(); ++i) {
            while (shape.get(i).size() < maxLen) shape.get(i).add(0);
        }
        variations.add(shape);
        for (int i = 1; i <= 3; ++i) {
            shape = Transformer.rotateClockwise(variations.get(i - 1));
            variations.add(shape);
        }
        for (int i = 0; i < 4; ++i) {
            variations.add(Transformer.flipX(variations.get(i)));
```

```
            variations.add(Transformer.flipY(variations.get(i)));

variations.add(Transformer.flipX(Transformer.flipY(variations.get(i))));
        }

        removeDuplicates();
    }

    public void removeDuplicates() {
                        Set<List<List<Integer>>>   uniqueSet   =   new
LinkedHashSet<>(this.variations);
        this.variations =  new ArrayList<>(uniqueSet);
    }

    public char getC() {
        return this.c;
    }

    public List<List<List<Integer>>> getVariations() {
        return this.variations;
    }
}
```

Selain tiga program di atas, terdapat juga program-program pembantu lainnya yaitu **Input.java**, **Timer.java**, dan **Transformer.java**. Informasi lengkap dari ketiga program pembantu tersebut dapat dilihat pada tautan *github* yang diberikan di akhir laporan ini.

Pengetesan Program

Test Case#1

```
5 5 7
```

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

```
Enter the name of the txt file located in the test folder: 1.txt
Configuration found!

FFFDD
FFCDB
GCCBB
GEEEA
GEEAA

Total iterations: 70186
Time spent: 95 ms
Apakah anda ingin menyimpan solusi? (y/n):
```

Test Case #2

```
2 2 2
DEFAULT
AA
B
B
```

```
Enter the name of the txt file located in the test folder: 2.txt
Configuration found!

BA
BA

Total iterations: 8
Time spent: 14 ms
Apakah anda ingin menyimpan solusi? (y/n):
```

Test Case #3

```
5 5 3
DEFAULT
AAA
A A
AAA
B
CCCCC
C   C
C   C
C   C
CCCCC
```

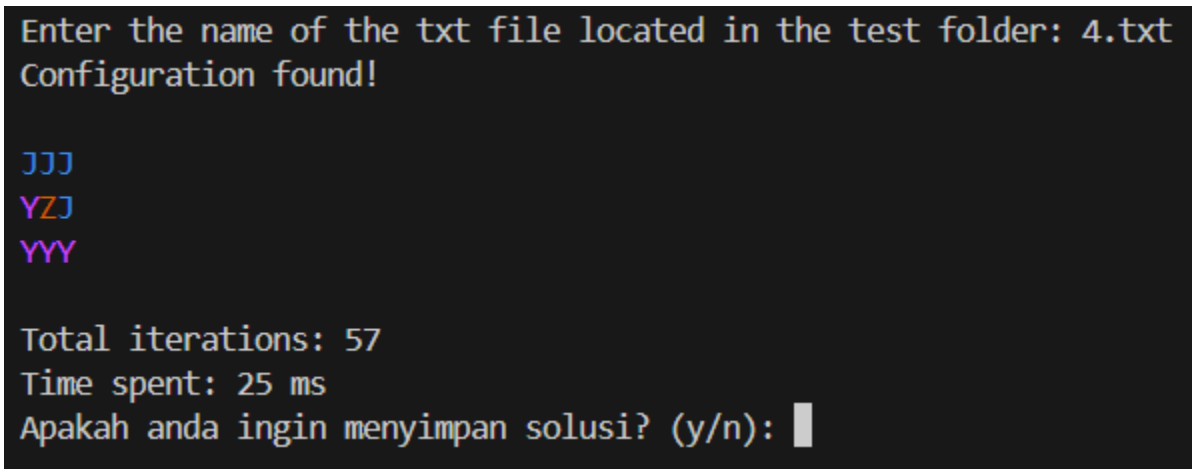
```
Enter the name of the txt file located in the test folder: 3.txt
Configuration found!

CCCCC
CAAAC
CABAC
CAAAC
CCCCC

Total iterations: 211
Time spent: 23 ms
Apakah anda ingin menyimpan solusi? (y/n):
```

Test Case #4

```
3 3 3
DEFAULT
J
J
JJ
YY
 Y
 Y
Z
```



Test Case #5

```
3 3 1
DEFAULT
AAA
AAA
AAA
```

```
Enter the name of the txt file located in the test folder: 5.txt
Configuration found!

AAA
AAA
AAA

Total iterations: 1
Time spent: 27 ms
Apakah anda ingin menyimpan solusi? (y/n):
```

Test Case #6

```
3 3 3
DEFAULT
A
```

```
Enter the name of the txt file located in the test folder: 6.txt
Make sure the input is correct and follows the required format!
```

Test Case #7

```
3 3 5
DEFAULT
AA
AAA
B
CCC
DDD
EEEEEEE
EEEEEEE
```

```
Enter the name of the txt file located in the test folder: 7.txt
Pieces that is given is bigger than the board!
```

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	✓	

Referensi

Tautan *GitHub*