Laporan Tugas Kecil 1 IF2211 Strategi Algoritma Semester II tahun 2024/2025

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Disusun oleh :

13523116 – Fityatul Haq Rosyidi

INSTITUT TEKNOLOGI BANDUNG

2025

DAFTAR ISI

BAB I DESKRIPSI TUGAS

1.1 IQ Puzzler Pro

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 block puzzle berhasil diletakkan.

Tugas anda adalah 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.

1.2 Algoritma Brute Force

Algoritma Brute Force adalah algoritma penyelesaian suatu masalah berbasis komputasi dengan pendekatan yang lempang(straightforward) dan sederhana. Algoritma Brute Force adalah algoritma natural yang biasanya pertama kali terpikirkan oleh problem solver karena implementasinya yang mudah. Algoritma ini bekerja dengan mencoba semua kemungkinan jalan penyelesaian yang mungkin terjadi tanpa pikir panjang.

Hampir semua persoalan dapat diselesaikan dengan algoritma brute force, namun algoritma ini kurang efisien dan umumnya lambat untuk masukan yang besar.

BAB II STRATEGI ALGORITMA

2.1 Rancangan Algoritma

Tugas ini mengharuskan kami menggunakan Algoritma Brute Force. Jadi saya melakukan penyelesaian dengan algoritma brute force ditambah backtracking pada beberapa bagian.

Berikut adalah alur penyelesaian Puzzle:

1. File input dibaca. jika isi input benar, setiap informasi akan disimpan
2. Akan ada p banyak piece, piece-piece akan di pasang ke board dimulai dari piece pertama.
3. Jika piece pertama dapat masuk, lanjut masukkan piece kedua.
4. Seterusnya begitu sehingga terdapat piece yang tidak bisa masuk ke board.
5. Piece yang tidak bisa masuk ke board akan digeser untuk disesuaikan koordinat posisinya. Misal ukuran piece adalah 2 x 3 satuan, piece akan digeser horizontal 3 kali dan vertikal 2 kali. Untuk seterusnya, ini saya sebut dengan **pergeseran posisi.** Jika ada piece yang berhasil masuk, lanjut ke piece berikutnya.
6. Jika pergeseran posisi belum berhasil, selanjutnya dilakukan **perubahan bentuk.** Piece dapat dirotasikan sebesar 90 derajat berlawan arah jarum jam dan dapat dicerminkan. Total semua kemungkinan perubahan bentuk ini ada 8. Pada masing2 perubahan bentuk, dilakukan juga pergeseran posisi. Jika ada piece yang berhasil masuk, lanjut ke piece berikutnya.
7. Jika perubahan bentuk masih belum berhasil, akan dilakukan **backtracking**. Mundur ke piece sebelumnya, lalu coba pergeseran posisi dan perubahan bentuk yang lain. Jika ada piece yang berhasil masuk, kembali ke piece selanjutnya. Jika tidak ada, mundur lagi ke piece sebelumnya, dan begitu seterusnya sampai kembali ke piece awal.
8. Pada setiap iterasi, selalu dicek apakah ada sel yang kosong di board. Jika sudah tidak ada sel kosong artinya puzzle terselesaikan, program berhenti dan board ditampilkan.

9. Jika semua kemungkinan sudah dilakukan dan board masih belum terisi penuh, program berhenti dan pesan gagal akan ditampilkan.

BAB III IMPLEMENTASI DALAM BAHASA JAVA

Disini penulis mengimplementasikan penyelesaian persoalan puzzle dengan bahasa pemrograman java. Penulis menerapkan paradigma Object Oriented Programming agar kode program lebih terstruktur dan meningkatkan readability.

3.1 Source Code

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

class Piece {
    String letter;
    int row, col;
    List<List<Integer>> shape;

    // constructor
    public Piece(String letter, int row, int col, List<List<Integer>> shape) {
    this.letter = letter;
    this.row = row;
    this.col = col;
    this.shape = shapePadding(shape);  // shape di padding dengan 0 pada bagian kosong agar berbentuk persegi /+ panjang
    }

    // method untuk padding shape
    private List<List<Integer>> shapePadding(List<List<Integer>> shape) {
        for (List<Integer> shapeRow : shape) {
            int delta = this.col - shapeRow.size();
            for (int i = 0; i < (delta); i++) {
                shapeRow.add(0);
            }
        }
        return shape;
    }
```

```java
// method untuk menampilkan piece (debugging)
    public void getPiece(){
        System.out.println("======================");
        System.out.println("Letter : " + this.letter);
        System.out.println("row : " + this.row);
        System.out.println("col : " + this.col);
        System.out.println("shape : ");
        for (int i = 0; i < this.shape.size(); i++){
            for (int j = 0; j < this.shape.get(i).size(); j++) {
                System.out.print(this.shape.get(i).get(j) + " ");
            }
            System.out.println();
        }
    }

    // method untuk merotasi piece
    private void rotate(){
        List<List<Integer>> rotated = new ArrayList<>();
        for (int j = this.col - 1; j > -1; j--){
            List<Integer> rotatedRow = new ArrayList<>();
            for (int i = 0; i < this.row; i++){
                rotatedRow.add(this.shape.get(i).get(j));
            }
            rotated.add(rotatedRow);
        }
        this.shape = rotated;
        this.row = rotated.size();
        this.col = rotated.get(0).size();
    }

    // method untuk mirroing piece
    private void mirror(){
        List<List<Integer>> mirrored = new ArrayList<>();
        for (int i = 0; i < this.row; i++){
            List<Integer> mirroredRow = new ArrayList<>();
            for (int j = this.col - 1; j > -1; j--){
                mirroredRow.add(this.shape.get(i).get(j));
            }
            mirrored.add(mirroredRow);
        }
        this.shape = mirrored;
        this.row = mirrored.size();
        this.col = mirrored.get(0).size();
    }
```

```java

    // method untuk menghasilkan 8 kemungkinan posisi piece setelah di rotasi dan mirroring
    public List<List<List<Integer>>> getPositions(){
        List<List<List<Integer>>> pos = new ArrayList<>();
        pos.add(this.shape);
        // rotasi
        for (int i = 0; i < 3; i++){
            this.rotate();
            pos.add(this.shape);
        }
        // mirror
        this.mirror();
        pos.add(this.shape);
        // rotasi mirror
        for (int i = 0; i < 3; i++){
            this.rotate();
            pos.add(this.shape);
        }
        return pos;
    }
}

class Board {
    int row, col;
    List<List<Integer>> flag; // board dalam bentuk 0 1
    List<List<String>> shape; // board dalam bentuk huruf

    // constructor
    public Board(int row, int col){
        this.row = row;
        this.col = col;

        List<List<Integer>> flag = new ArrayList<>();
        List<List<String>> shape = new ArrayList<>();
        for (int i = 0; i < row; i++){
            List<Integer> flagRow = new ArrayList<>();
            List<String> shapeRow = new ArrayList<>();
            for (int j = 0; j < col; j++){
                flagRow.add(0);
                shapeRow.add(" ");
            }
            flag.add(flagRow);
            shape.add(shapeRow);
        }
        this.flag = flagPadding(flag);  // di padding biar aman kalo indexOutOfBound
        this.shape = shape;
    }

    // method untuk padding
    private List<List<Integer>> flagPadding(List<List<Integer>> flag){
        int max = Math.max(this.row, this.col);
        for (int i = 0; i < this.row; i++){
            for (int j = 0; j < max; j++){
                flag.get(i).add(1);
            }
        }

        for (int i = 0; i < max; i++){
            List<Integer> padRow = new ArrayList<>();
            for (int j = 0; j < this.row + max; j++){
                padRow.add(1);
            }
            flag.add(padRow);
        }
        return flag;
    }
```

```java

    // method untuk nampilin shape
    public void showBoardShape(){
        for (List<String> row : this.shape){
            for (String c : row) {
                System.out.print(c + " ");
            }
            System.out.println();
        }
    }

    // method untuk nampilin flag
    public void showBoardFlag(){
        for (List<Integer> row : this.flag){
            for (Integer c : row) {
                System.out.print(c + " ");
            }
            System.out.println();
        }
    }

    // method untuk masukin piece ke board
    public void install(List<List<Integer>> shape, int rowOffset, int colOffset, String letter){
        for (int i = 0; i < shape.size(); i++){
            for (int j = 0; j < shape.get(0).size(); j++){
                if (shape.get(i).get(j) == 1){
                    this.flag.get(i + rowOffset).set(j + colOffset, 1);
                    this.shape.get(i + rowOffset).set(j + colOffset, letter);
                }
            }
        }
    }

    // method untuk ngeluarin piece dari board
    public void uninstall(List<List<Integer>> shape, int rowOffset, int colOffset){
        for (int i = 0; i < shape.size(); i++){
            for (int j = 0; j < shape.get(0).size(); j++){
                if (shape.get(i).get(j) == 1){
                    this.flag.get(i + rowOffset).set(j + colOffset, 0);
                    this.shape.get(i + rowOffset).set(j + colOffset, " ");
                }
            }
        }
    }
}

```

```java

public class Main {
    // fungsi untuk baca input
    public static Object[] readFromFile(String filePath) {
        BufferedReader br = null;
        try {
            br = new BufferedReader(new FileReader(filePath));

            // Baris pertama
            String[] firstLine = br.readLine().split(" ");
            int m = Integer.parseInt(firstLine[0]);
            int n = Integer.parseInt(firstLine[1]);
            int p = Integer.parseInt(firstLine[2]);

            // Baris kedua
            String mode = br.readLine();

            // List warna
            String[] colors = {
                "\u001b[31m", // Merah
                "\u001b[32m", // Hijau
                "\u001b[33m", // Kuning
                "\u001b[34m", // Biru
                "\u001b[35m", // Magenta
                "\u001b[36m", // Cyan
                "\u001b[37m", // Putih
                "\u001b[91m", // Merah Terang
                "\u001b[92m", // Hijau Terang
                "\u001b[93m", // Kuning Terang
                "\u001b[94m", // Biru Terang
                "\u001b[95m", // Magenta Terang
                "\u001b[96m", // Cyan Terang
                "\u001b[97m", // Putih Terang
                "\u001b[90m", // Hitam Terang
                "\u001b[30m"  // Hitam
            };
```

```
        // Baris ketiga
        List<Piece> pieces = new ArrayList<>();
        List<List<Integer>> currShape = new ArrayList<>();
        char currLetter = ' ';
        int row = 0;
        int col = 0;
        int colorIndex = 0;
        String line;
        while((line = br.readLine()) != null) {
            line = line.replaceAll("\\s+$", "");
            if (line.isEmpty()) continue;

            char firstChar = line.replaceAll("^\\s+", "").charAt(0);

            if (currLetter == ' ') {
                currLetter = firstChar;
            }

            if (firstChar != currLetter) {
                // buat piece baru
                pieces.add(new Piece(colors[colorIndex] + currLetter + "\u001b[0m", row, col, currShape));
                currLetter = firstChar;
                currShape = new ArrayList<>();
                row = 0;
                col = 0;
                colorIndex++;
                colorIndex %= 16;
            }

            List<Integer> currShapeRow = new ArrayList<>();
            int currCol = 0;
            for (char c : line.toCharArray()) {
                if (c == ' ') {
                    currShapeRow.add(0);
                } else {
                    currShapeRow.add(1);
                }
                currCol++;
            }

            currShape.add(currShapeRow);
            col = Math.max(col, currCol);
            row++;
        }

        // menambahkan piece terakhir
        if (!currShape.isEmpty()) {
            pieces.add(new Piece(colors[colorIndex] + currLetter + "\u001b[0m", row, col, currShape));
        }

        br.close();



        return new Object[]{pieces, m, n, p};
    } catch (IOException | ArrayIndexOutOfBoundsException e) {
        return null;
    }
}
```

```java

// fungsi untuk menulis output ke file
public static void writeToFile(Board board, String filePath){
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))){
        for (List<String> row : board.shape){
            for (String s : row){
                writer.write(s.replaceAll("\\u001b\\[[0-9;]*m", "") + " ");
            }
            writer.newLine();
        }
        System.out.println("Output berhasil ditulis ke file: " + filePath);
    } catch (IOException e) {

        System.out.println("Output Gagal ditulis ke file: " + filePath);
    }
}

// fungsi untuk mengecek apakah piece bisa masuk ke board
public static boolean isMatch(Board board, List<List<Integer>> shape, int rowOffset, int colOffset){
    try {
        if ((rowOffset < 0) || (colOffset < 0)) {return false;}

        int cell;
        for (int i = 0; i < shape.size(); i++) {
            for (int j = 0; j < shape.get(0).size(); j++){
                cell = board.flag.get(rowOffset + i).get(colOffset + j) + shape.get(i).get(j);
                if (cell > 1) {return false;}
            }
        }
        return true;
    } catch (ArrayIndexOutOfBoundsException e) {

        return false;
    }
}

// fungsi utama, melakukan bruteforcing dan backtracking untuk menemukan solusi
public static Object[] bruteForce(Board board, List<Piece> pieces, int count) {
    for (int rowOffset = 0; rowOffset < board.row; rowOffset++){
        for (int colOffset = 0; colOffset < board.col; colOffset++){
            if (board.flag.get(rowOffset).get(colOffset) == 0) { // jika ditemukan space kosong pada board
                for (Piece currPiece : pieces){
                    String letter = currPiece.letter;
                    for (List<List<Integer>> shape : currPiece.getPositions()){
                        for (int rowPointer = 0; rowPointer < shape.size(); rowPointer++){
                            for (int colPointer = 0; colPointer < shape.get(0).size(); colPointer++){
                                count++;
                                if (isMatch(board, shape, rowOffset - rowPointer, colOffset - colPointer)){  // jika piece bisa masuk ke board
                                    board.install(shape, rowOffset - rowPointer, colOffset - colPointer, letter);

                                    List<Piece> nextPieces = new ArrayList<>();
                                    for (Piece pc : pieces){
                                        if (!pc.equals(currPiece)){
                                            nextPieces.add(pc);  // nextPieces berisi semua piece kecuali piece saat ini
                                        }
                                    }
                                    Object[] result = bruteForce(board, nextPieces, count);  // rekurens
                                    boolean isComplete = (boolean)result[0];
                                    count = (int)result[1];
                                    if (isComplete){
                                        return new Object[]{true, count};  // basis
                                    }
                                    board.uninstall(shape, rowOffset - rowPointer, colOffset - colPointer);
                                }
                            }
                        }
                    }
                }
                return new Object[]{false, count};
            }
        }
    }
    return new Object[]{true, count}; // basis
}
```

```java

    @SuppressWarnings("unchecked")  // tipe list dipastikan sesuai, ga perlu di cek
    public static void main(String[] args) {

        try {

            Scanner scanner = new Scanner(System.in);
            System.out.println("Masukkan nama file input : ");
            String filename = scanner.nextLine().strip();

            Object[] result = readFromFile("test/" + filename);

            List<Piece> pieces = (List<Piece>)  result[0];
            int m = (int) result[1];
            int n = (int) result[2];
            int p = (int) result[3];


            // membuat object board
            Board board = new Board(m, n);



            int count = 0;
            long startTime = System.currentTimeMillis();
            Object[] values = bruteForce(board, pieces, count);
            long endTime = System.currentTimeMillis();

            boolean isComplete = (boolean) values[0];
            count = (int) values[1];

            if (isComplete){
                System.out.println("\nBerhasil mengisi board\n");
                board.showBoardShape();
            } else {
                System.out.println("\nGagal mengisi board\n");
            }

            System.out.println("\nJumlah percobaan : " + count);
            System.out.println("Waktu eksekusi : " + (endTime - startTime) + " ms");

            if (isComplete){

                while (true){
                    System.out.println("Apakah anda ingin menyimpan solusi?(y/n)");
                    String ans = scanner.nextLine().strip();

                    if (ans.equals("y") | ans.equals("Y")) {
                        System.out.print("nama file: ");
                        String filePath = scanner.nextLine().strip();
                        writeToFile(board, "output/" + filePath);
                        break;
                    } else if (ans.equals("n") | ans.equals("N")){
                        break;
                    }

                }


            scanner.close();

            }
        } catch ( ArrayIndexOutOfBoundsException | NullPointerException | NumberFormatException e) {
            System.out.println("\nPastikan nama dan isi file input sudah benar!");
        }


    }
}

```

BAB IV TESTING

No	Test case		Penjelasan
	input	output	
1	test > input1.txt 3 3 2 DEFAULT AAA BBB BBB	Berhasil mengisi board A A A B B B B B B Jumlah percobaan : 2 Waktu eksekusi : 0 ms	Kasus sederhana
2	test > input2.txt 5 5 6 DEFAULT AA A BBB CC C DDDD D EEE EE FFF FFF	Berhasil mengisi board A A B B B A C C F F E E C F F E E D F F E D D D D Jumlah percobaan : 9345 Waktu eksekusi : 6 ms	Kasus 5 X 5
3	test > input3.txt 5 5 6 DEFAULT AAA A BBB CC C DDDD D EEE EE FFF FFF	Gagal mengisi board Jumlah percobaan : 13764040 Waktu eksekusi : 933 ms	Kasus gagal, board tidak berhasil diisi

4	test > input4.txt 8 8 4 DEFAULT AAAAAAAA AAAAAA AAAAAAA AAAAAAAA A AAAA AAAAAAAA A AAAA BB B BBBBBBBB CC C DDD	Berhasil mengisi board A A A A A A A A A A A A A A C C A A A A A A A C A A A A A A A A A D D D A A A A A A A A A A A A A B A A A A B B B B B B B B B B Jumlah percobaan : 385 Waktu eksekusi : 4 ms	Kasus bentuk piece aneh
5	test > input5.txt 5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG	Berhasil mengisi board A B B C C A A B D C E E E D D E E F F F G G G F F Jumlah percobaan : 897 Waktu eksekusi : 2 ms	Kasus berhasil

6	test > input6.txt 8 8 15 DEFAULT AAA AA BBB B CCC CC D DDD D E EEEE FFF F F GGGGG HHH II I JJ JJJ K KKK LLL LL MMM M N NN OOO	Berhasil mengisi board A A A B B B C C A A E E B C C C F F F E G H H H F I I E G J J J F I D E G K J J L D D D G K N N L L D M G K K N L L M M M O O O Jumlah percobaan : 29063 Waktu eksekusi : 21 ms	Kasus dengan banyak piece
7	test > input7.txt AKU SIGMA	Masukkan nama file input : input7.txt Pastikan nama dan isi file input sudah benar!	Kasus file yang tidak sesuai

8	test > input8.txt 10 12 8 DEFAULT AAAAAAAAA AAAAAAAAA AAAA AAAAAA AAAAAA AAAAAAAAAA AAAAAAAAAA AAAAAAAAA AAA AAA AAA AAA HHH H BBB BBBBB BBBBBB RR KKKKK VVV VVV III III NNN NNN	Berhasil mengisi board R A A A A A A A A A V V R A A A A A A A A A V V H H H B B B A A A A V V H B B B B B A A A A A A B B B B B B A A A A A A K A A A A A A A A A A A K A A A A A A A A A A A K A A A A A A A A A I I K A A A N N N A A A I I K A A A N N N A A A I I Jumlah percobaan : 5623548 Waktu eksekusi : 470 ms	Kasus amogus
9	test > custom1.txt 5 7 5 CUSTOM ...X... .XXXXX. XXXXXXX .XXXXX. ...X... A AAA BB BBB CCCC C D EEE E	Masukkan nama file input : custom1.txt Berhasil mengisi board A B B A A A B B B C C C C E E E D C E Jumlah percobaan : 15376 Waktu eksekusi : 17 ms	Kasus custom

BAB V LAMPIRAN

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	✔	

Github repository: [FityatulhaqRosyidi/Tucil1_13523116]

REFERENSI

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf

https://youtube.com/shorts/MWiPAS3wfGM?feature=shared

Java Tutorial