

Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force



Disusun Oleh:

Ahmad Wicaksono (13523121)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR ISI

1. LATAR BELAKANG	3
1.1. IQ Puzzler Pro	3
1.2. Algoritma Brute Force	3
1.3. Penyelesaian IQ Puzzler Pro Menggunakan Algoritma Brute Force	4
2. IMPLEMENTASI ALGORITMA DALAM BAHASA JAVA	4
2.1. File inputPuzzle.java	4
2.2. File rotation.java	4
2.3. File bruteForce.java	5
2.4. File outputPuzzle.java	5
2.5 File main.java	6
3. SOURCE CODE DALAM BAHASA JAVA	6
3.1. Repositori Github	6
3.2. Source Code Program	7
3.2.1. inputPuzzle.java	7
3.2.2. rotation.java	9
3.2.3. bruteForce.java	10
3.2.4. outputPuzzle.java	12
3.2.5. Main.java	13
4. INPUT DAN OUTPUT PROGRAM	14
4.1. Test Case 1 (case1.txt)	14
4.2. Test Case 2 (case2.txt)	14
4.3. Test Case 3 (case3.txt)	15
4.4. Test Case 4 (case4.txt)	16
4.5. Test Case 5 (case5.txt)	16
4.6. Test Case 6 (case6.txt)	16
4.7. Test Case 7 (case7.txt)	17
5. LAMPIRAN	18
6. REFERENSI	18

1. LATAR BELAKANG

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.

1.2. Algoritma Brute Force

Algoritma Brute Force merupakan metode pemecahan masalah yang dilakukan dengan mencoba semua kemungkinan solusi secara sistematis hingga menemukan hasil yang benar.. Cara kerja brute force adalah dengan meng-enumerasi semua kemungkinan solusi. Contoh penerapan algoritma ini dapat ditemukan dalam pencarian kata sandi dengan mencoba seluruh kombinasi karakter, pencarian pola dalam teks dengan membandingkan pola pada setiap posisi dalam teks, serta penyelesaian masalah *Traveling Salesman Problem (TSP)* dengan menghitung semua kemungkinan rute untuk menemukan yang paling optimal.

Meskipun brute force memiliki keunggulan dalam kemudahan implementasi dan jaminan menemukan solusi optimal, metode ini memiliki kelemahan utama berupa ketidakefisienan, terutama pada permasalahan dengan ruang solusi yang sangat besar. Waktu eksekusi yang cenderung lama membuat brute force kurang ideal untuk masalah berskala besar, sehingga sering kali digunakan sebagai pendekatan dasar sebelum menerapkan algoritma yang lebih efisien.

1.3. Penyelesaian IQ Puzzler Pro Menggunakan Algoritma Brute Force

Dalam menyelesaikan IQ Puzzle Pro, algoritma brute force dengan *backtracking* digunakan untuk mencoba semua kemungkinan penyusunan bentuk secara sistematis, sambil menghindari jalur yang tidak menghasilkan solusi valid. Algoritma ini bekerja dengan menempatkan setiap bagian puzzle di posisi yang memungkinkan dan memeriksa apakah susunan tersebut sesuai dengan aturan permainan. Jika suatu konfigurasi tidak valid, algoritma akan kembali ke langkah sebelumnya (*backtracking*) dan mencoba alternatif lain tanpa harus mengeksplorasi seluruh kemungkinan secara menyeluruh. Dengan cara ini, banyak percobaan yang tidak perlu dapat dieliminasi lebih awal, sehingga mengurangi jumlah kemungkinan yang harus diperiksa dibandingkan brute force murni. Meskipun pendekatan ini lebih efisien daripada brute force tanpa optimasi, kompleksitasnya tetap tinggi, terutama jika ruang solusi sangat besar. Oleh karena itu, metode ini lebih cocok untuk teka-teki dengan jumlah kemungkinan yang masih dapat dikendalikan.

2. IMPLEMENTASI ALGORITMA DALAM BAHASA JAVA

2.1. File inputPuzzle.java

File inputPuzzle.java menjadi sumber data untuk membaca dan memproses grid puzzle menjadi daftar matriks biner berdasarkan karakter unik yang terdapat dalam grid.

Nama Fungsi	Deskripsi
readInput	Membaca file input yang berisi dimensi dan label puzzle, lalu memprosesnya menjadi list puzzle berdasarkan karakter unik yang ada
processPuzzle	Menyaring bagian grid yang mengandung label tertentu, memangkas bagian kosong, dan mengubahnya menjadi format matriks biner untuk disimpan dalam daftar puzzle.

2.2. File rotation.java

File ini digunakan untuk menghasilkan semua transformasi unik dari sebuah puzzle dalam bentuk matriks, termasuk rotasi 90 derajat searah jarum jam dan pencerminan horizontal.

Nama Fungsi	Deskripsi
-------------	-----------

puzzChange	Menghasilkan daftar transformasi unik dari puzzle, termasuk empat rotasi dan pencerminan horizontal.
rotating	Memutar puzzle 90 derajat searah jarum jam.
mirroring	Mencerminkan puzzle secara horizontal (membalik urutan kolom).

2.3. File bruteForce.java

File ini digunakan untuk menyusun potongan puzzle ke dalam papan menggunakan algoritma brute force dengan mempertimbangkan semua rotasi dan pencerminan setiap potongan.

Nama Fungsi	Deskripsi
bruteForceSolve	Mencoba menyusun semua potongan puzzle pada papan menggunakan brute force dengan rekursi dan backtracking.
validPuzz	Memeriksa apakah potongan puzzle dapat ditempatkan di posisi (x, y) pada papan tanpa tumpang tindih.
putPuzz	Menempatkan atau menghapus potongan puzzle di papan dengan karakter tertentu (label atau . untuk menghapus).
initBoard	Menginisialisasi papan dengan karakter . dan mengurutkan potongan puzzle berdasarkan ukuran (dengan yang terbesar ditempatkan lebih dulu).
cnt1	Menghitung jumlah sel bernilai 1 dalam suatu potongan puzzle untuk menentukan ukurannya.

2.4. File outputPuzzle.java

File ini digunakan untuk menampilkan dan menyimpan hasil penyusunan puzzle dalam bentuk tampilan warna di terminal, gambar PNG, serta file teks.

Nama Fungsi	Deskripsi
getAnsiColor	Mengembalikan kode warna ANSI untuk tampilan warna pada huruf di terminal berdasarkan karakter.
getColor	Mengembalikan warna Color untuk digunakan dalam pembuatan gambar berdasarkan karakter.

printBoardWithColor	Menampilkan papan puzzle di terminal dengan warna sesuai karakter.
saveBoardImage	Menyimpan papan puzzle sebagai gambar PNG dengan warna yang sesuai.
saveBoardText	Menyimpan papan puzzle sebagai file teks dengan representasi karakter.

2.5 File main.java

File ini berfungsi sebagai program utama yang membaca input puzzle dari file, memprosesnya menggunakan algoritma brute-force untuk menemukan solusi, menampilkan hasilnya di terminal, serta memberikan opsi untuk menyimpan solusi dalam bentuk teks dan gambar.

3. SOURCE CODE DALAM BAHASA JAVA

3.1. Repositori Github

Repositori program dapat diakses melalui tautan Github ini
https://github.com/sonix03/Tucil1_13523121

3.2. Source Code Program

3.2.1. inputPuzzle.java

1. readInput

```
public static void readInput(String filename) throws FileNotFoundException {
    Scanner scanner = new Scanner(new File(filename));

    if (!scanner.hasNextInt()) {
        System.out.println(x:"Masukan Tidak Valid");
        scanner.close();
        return;
    }

    int M = scanner.nextInt();
    int N = scanner.nextInt();
    int P = scanner.nextInt();
    scanner.nextLine();

    if (scanner.hasNextLine()) {
        d = scanner.nextLine().trim();
    }

    List<String> lines = new ArrayList<>();
    while (scanner.hasNextLine()) {
        String line = scanner.nextLine();
        if (!line.trim().isEmpty()) {
            lines.add(line);
        }
    }
    scanner.close();
    labelSet.clear();
    for (String a : lines) {
        for (char chr : a.toCharArray()) {
            if (chr != ' ') {
                labelSet.add(chr);
            }
        }
    }
}

labels = new char[labelSet.size()];
int i = 0;
for (char chr : labelSet) {
    labels[i++] = chr;
}

for (char chr : labels) {
    processPuzzle(lines, chr);
}
```

2. processPuzzle

```
private static void processPuzzle(List<String> lines, char label) {
    int m = lines.size();
    int n = lines.stream().mapToInt(String::length).max().orElse(0);

    char[][] allGrid = new char[m][n];
    for (int r = 0; r < m; r++) {
        String line = lines.get(r);
        for (int c = 0; c < n; c++) {
            if (c < line.length()) {
                allGrid[r][c] = line.charAt(c);
            } else {
                allGrid[r][c] = ' ';
            }
        }
    }

    boolean[] validRow = new boolean[m];
    boolean[] validCol = new boolean[n];

    for (int r = 0; r < m; r++) {
        for (int c = 0; c < n; c++) {
            if (allGrid[r][c] == label) {
                validRow[r] = true;
                validCol[c] = true;
            }
        }
    }
}
```

```
int top = 0, bottom = m - 1, left = 0, right = n - 1;
while (top <= bottom && !validRow[top]) {
    top++;
}
while (bottom >= top && !validRow[bottom]) {
    bottom--;
}
while (left <= right && !validCol[left]) {
    left++;
}
while (right >= left && !validCol[right]) {
    right--;
}

int nRow = bottom - top + 1;
int nCol = right - left + 1;
int[][] trim = new int[nRow][nCol];

for (int r = 0; r < nRow; r++) {
    for (int c = 0; c < nCol; c++) {
        if (allGrid[top + r][left + c] == label) {
            trim[r][c] = 1;
        } else {
            trim[r][c] = 0;
        }
    }
}

puzzles.add(trim);
}
```


3.2.2. rotation.java

1. puzzChange

```
static List<int[][]> puzzChange(int[][] p) {
    Set<String> visit = new HashSet<>();
    List<int[][]> changes = new ArrayList<>();
    for (int i = 0; i < 4; i++) {
        p = rotating(p);
        String hashArr = Arrays.deepToString(p);
        if (visit.add(hashArr)) changes.add(p);
        int[][] mirror = mirroring(p);
        if (visit.add(Arrays.deepToString(mirror))) changes.add(mirror);
    }
    return changes;
}
```

2. rotating

```
static int[][] rotating(int[][] p) {
    int m = p.length;
    int n = p[0].length;
    int[][] rotated = new int[n][m];
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            rotated[j][m - 1 - i] = p[i][j];
    return rotated;
}
```

3. mirroring

```
static int[][] mirroring(int[][] p) {
    int m = p.length;
    int n = p[0].length;
    int[][] mirror = new int[m][n];
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            mirror[i][n - 1 - j] = p[i][j];
    return mirror;
}
```

3.2.3. bruteForce.java

1. bruteForceSolve

```
static boolean bruteForceSolve(int n) {
    if (n == puzzles.size()) {
        return true;
    }
    int[][] puzzle = puzzles.get(n);
    char label = labels[n];
    for (int row = 0; row <= M - puzzle.length; row++) {
        for (int col = 0; col <= N - puzzle[0].length; col++) {
            for (int[][] change : rotation.puzzChange(puzzle)) {
                count++;
                if (validPuzz(change, row, col)) {
                    putPuzz(change, row, col, label);
                    if (bruteForceSolve(n + 1)) { return true; }
                }
                putPuzz(change, row, col, val: '.');
            }
        }
    }
    return false;
}

static boolean validPuzz(int[][] p, int x, int y) {
    for (int i = 0; i < p.length; i++) {
        for (int j = 0; j < p[i].length; j++) {
            if (p[i][j] == 1) {
                int nx = x + i, ny = y + j;
                if (nx >= M || ny >= N || board[nx][ny] != '.') {
                    return false;
                }
            }
        }
    }
    return true;
}
```

2. validPuzz

```
static boolean validPuzz(int[][] p, int x, int y) {
    for (int i = 0; i < p.length; i++) {
        for (int j = 0; j < p[i].length; j++) {
            if (p[i][j] == 1) {
                int nx = x + i, ny = y + j;
                if (nx >= M || ny >= N || board[nx][ny] != '.') {
                    return false;
                }
            }
        }
    }
    return true;
}
```

3. putPuzz

```
static void putPuzz(int[][] p, int x, int y, char val) {
    for (int i = 0; i < p.length; i++) {
        for (int j = 0; j < p[i].length; j++) {
            if (p[i][j] == 1) {
                board[x + i][y + j] = val;
            }
        }
    }
}
```

4. initBoard

```
static void initBoard() {
    board = new char[M][N];
    for (char[] m : board) {
        Arrays.fill(m, val:'.');
    }

    // Sorting puzzles
    puzzles = new ArrayList<>(inputPuzzle.puzzles);
    puzzles.sort((a, b) -> Integer.compare(cnt1(b), cnt1(a)));

    // Sorting labels
    List<Character> labelList = new ArrayList<>();
    Map<int[][], Character> puzzleLabel = new HashMap<>();

    for (int i = 0; i < inputPuzzle.puzzles.size(); i++) {
        puzzleLabel.put(inputPuzzle.puzzles.get(i), inputPuzzle.labels[i]);
    }

    puzzles.forEach(p
        ->
        labelList.
        add(puzzleLabel.get(p)));

    labels = new char[labelList.size()];

    for (int i = 0; i < labelList.size(); i++) {
        labels[i] = labelList.get(i);
    }
}
```

5. cnt1

```
static int cnt1(int[][] p) {
    int count = 0;
    for (int[] m : p) {
        for (int cnt : m) {
            if (cnt == 1) {
                count++;
            }
        }
    }
    return count;
}
```

3.2.4. outputPuzzle.java

1. getAnsiColor

```
private static String getAnsiColor(char c) {  
    if (c == '.') return "\u001B[30m";  
    int i = (c - 'A') % ANSI.length;  
    return ANSI[i];  
}
```

2. getColor

```
private static Color getColor(char c) {  
    if (c == '.') return Color.BLACK;  
    int index = (c - 'A') % COLORS.length;  
    return COLORS[index];  
}
```

3. printBoardColor

```
public static void printBoardColor(char[][] c) {  
    if (c == null || c.length == 0) {  
        System.out.println(x:"Board kosong");  
        return;  
    }  
  
    for (char[] row : c) {  
        for (char n : row) {  
            System.out.print(getAnsiColor(n) + n + " " + RESET_ANSI);  
        }  
        System.out.println();  
    }  
}
```

4. saveBoardImage

```
public static void saveBoardImage(char[][] m, String name) {  
    int size = 50;  
    int width = m[0].length * size;  
    int height = m.length * size;  
  
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);  
    Graphics2D g = image.createGraphics();  
  
    for (int i = 0; i < m.length; i++) {  
        for (int j = 0; j < m[i].length; j++) {  
            g.setColor(getColor(m[i][j]));  
            g.fillRect(j * size, i * size, size, size);  
  
            g.setColor(Color.BLACK);  
            g.drawRect(j * size, i * size, size, size);  
        }  
    }  
    g.dispose();  
  
    File dir = new File(pathname:"../test/output");  
    if (!dir.exists()) {  
        dir.mkdirs();  
    }  
  
    File file = new File(dir, name);  
    try {  
        ImageIO.write(image, formatName:"png", file);  
        System.out.println("Output berhasil tersimpan di " + file.getAbsolutePath());  
    }  
    catch (Exception e) {  
        System.out.println("Gagal menyimpan gambar " + e.getMessage());  
    }  
}
```

5. saveBoardText

```
public static void saveBoardText(char[][] m, String name) {
    File dir = new File(pathname:"../test/output");
    if (!dir.exists()) {
        dir.mkdirs();
    }

    File outputFile = new File(dir, name);
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile))) {
        for (char[] row : m) {
            writer.write(new String(row));
            writer.newLine();
        }
        System.out.println("Output berhasil tersimpan di " + outputFile.getAbsolutePath());
    } catch (IOException e) {
        System.out.println("Gagal menyimpan output: " + e.getMessage());
    }
}
```

3.2.5. Main.java

1. Main

```
public static void main(String[] args) throws FileNotFoundException {
    Scanner scannerA = new Scanner(System.in);

    System.out.print(s:"Masukkan nama file: ");
    String dir = "../test/input/";
    String file = scannerA.nextLine();

    Scanner scannerB = new Scanner(new File(dir + file));

    bruteForce.M = scannerB.nextInt();
    bruteForce.N = scannerB.nextInt();
    int P = scannerB.nextInt();

    scannerB.nextLine();

    inputPuzzle.readInput(dir + file);

    bruteForce.initBoard(); // DEFAULT

    long start = System.currentTimeMillis();

    if (P > inputPuzzle.puzzles.size()) {
        System.out.println(x:"Nilai P lebih besar dari jumlah puzzle yang tersedia");
        System.out.println("Jumlah puzzle yang tersedia: " + inputPuzzle.puzzles.size());
        scannerA.close();
        scannerB.close();
        return;
    }
}
```

```
if (bruteForce.bruteForceSolve(n:0)) {
    long end = System.currentTimeMillis();
    outputPuzzle.printBoardColor(bruteForce.board);
    System.err.println();

    System.out.println("Waktu Pencarian: " + (end - start) + " ms");
    System.out.println();
    System.out.println("Banyak kasus yang ditinjau: " + bruteForce.count);
    System.out.println();

    System.out.println(x:"Apakah anda ingin menyimpan solusi? (ya/tidak)");
    String choice = scannerA.next();
    scannerA.nextLine();

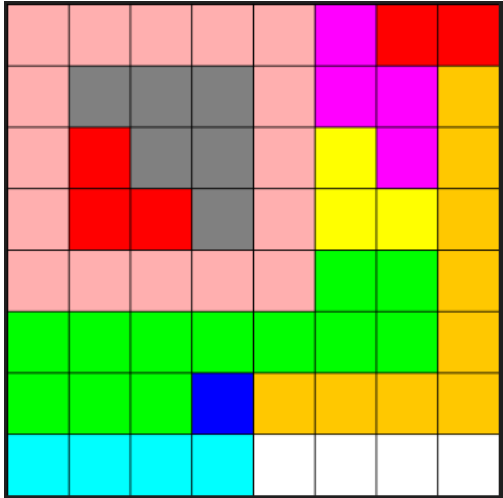
    if (choice.equals(anObject:"ya")) {
        System.out.print(s:"Masukkan nama file (tanpa format txt atau png): ");
        String name = scannerA.nextLine();

        outputPuzzle.saveBoardText(bruteForce.board, name + ".txt");
        outputPuzzle.saveBoardImage(bruteForce.board, name + ".png");
        System.out.println(x:"File berhasil disimpan");
    } else {
        System.out.println(x:"File tidak tersimpan");
    }
} else {
    System.out.println(x:"Tidak ada solusi");
}

scannerA.close();
scannerB.close();
}
```

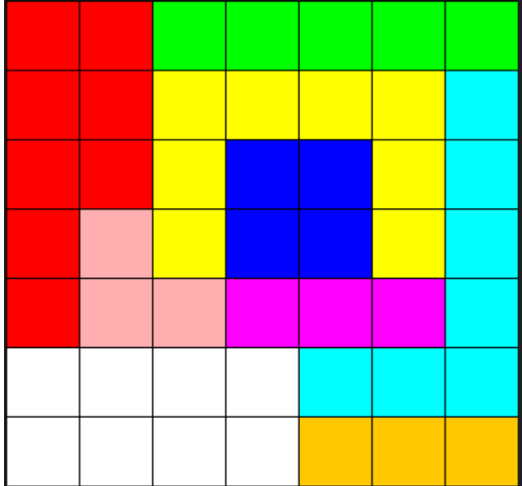
4. INPUT DAN OUTPUT PROGRAM

4.1. Test Case 1 (case1.txt)

Input	Output
8 8 12 DEFAULT A AA CC C BBBBB BBB D EE EE FFFF GGGG HHHH H H H H H IIII I I I I I I IIII KK JJJ JJ J LL LL	<pre> Masukkan nama file: case1.txt I I I I I E K K I J J J I E E H I A J J I C E H I A A J I C C H I I I I I L L H B B B B L L H B B B D H H H H F F F F G G G G Waktu Pencarian: 901 ms Banyak kasus yang ditinjau: 622299 Apakah anda ingin menyimpan solusi? (ya/tidak) </pre> 

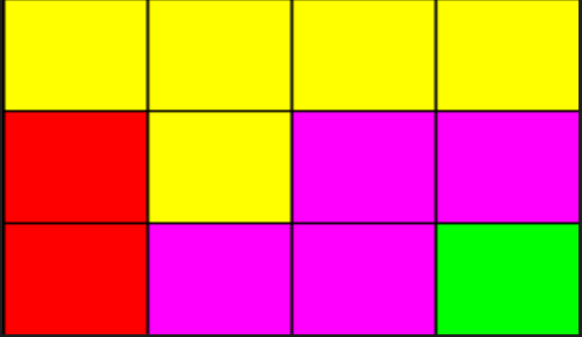
4.2. Test Case 2 (case2.txt)

Input	Output
7 7 9 DEFAULT	

AAA AAAAA BBBBB CCC C C CCC EEE FFF F F F F GGGG GGGG HHH I II DD DD	<pre> Masukkan nama file: case2.txt A A B B B B B A A C C C C F A A C D D C F A I C D D C F A I I E E E F G G G G F F F G G G G H H H Waktu Pencarian: 312 ms Banyak kasus yang ditinjau: 128563 Apakah anda ingin menyimpan solusi? (ya/tidak) </pre> 
---	---

4.3. Test Case 3 (case3.txt)

Input	Output
3 4 4 DEFAULT CCCC C AA B EE EE	<pre> Masukkan nama file: case3.txt C C C C A C E E A E E B Waktu Pencarian: 52 ms Banyak kasus yang ditinjau: 74 Apakah anda ingin menyimpan solusi? (ya/tidak) </pre>

	
--	--

4.4. Test Case 4 (case4.txt)

Input	Output
3 3 6 AA BB B BB CC	<pre>Masukkan nama file: case4.txt Nilai P lebih besar dari jumlah puzzle yang tersedia! Jumlah puzzle yang tersedia: 3</pre>

4.5. Test Case 5 (case5.txt)

Input	Output
3 3 3 DEFAULT AA AA AAA BB C	<pre>Masukkan nama file: case5.txt Tidak ada solusi</pre>

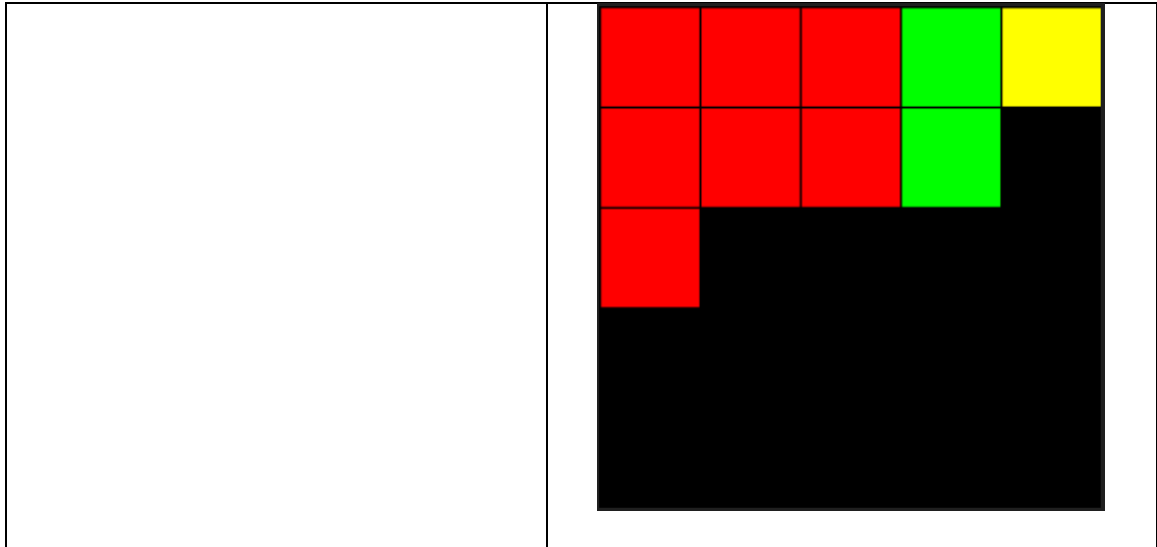
4.6. Test Case 6 (case6.txt)

Input	Output
5 5 8 DEFAULT A AA B BB C CC D	

DD EE EE FF FF F GGG	<p>Masukkan nama file: case6.txt</p> <pre> E E E F F A E E F F A A B F C D B B C C D D G G G </pre> <p>Waktu Pencarian: 380 ms</p> <p>Banyak kasus yang ditinjau: 252476</p> <p>Apakah anda ingin menyimpan solusi? (ya/tidak)</p> 
---	--

4.7. Test Case 7 (case7.txt)

Input	Output
5 5 3 DEFAULT AA AA AAA BB C	<p>Masukkan nama file: case7.txt</p> <pre> A A A B C A A A B . A </pre> <p>Waktu Pencarian: 42 ms</p> <p>Banyak kasus yang ditinjau: 13</p> <p>Apakah anda ingin menyimpan solusi? (ya/tidak)</p>



5. LAMPIRAN

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	V	
2	Program berhasil dijalankan	V	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	V	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	V	
5	Program memiliki Graphical User Interface (GUI)		V
6	Program dapat menyimpan solusi dalam bentuk file gambar	V	
7	Program dapat menyelesaikan kasus konfigurasi custom		V
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)	V	
9	Program dibuat oleh saya sendiri	V	

6. REFERENSI

- <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/Tucil1-Stima-2025.pdf>
- <https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/>
- <https://www.freecodecamp.org/news/brute-force-algorithms-explained/>
- <https://www.smartgames.eu/uk/one-player-games/iq-puzzler-pro-0>