Tugas Kecil 1 IF2211 Strategi Algoritma Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force

Oleh :
13523139 / Jonathan Kenan Budianto

INSTITUT TEKNOLOGI BANDUNG
2025

BAB I
LATAR BELAKANG DAN ALGORITMA

1.1 Permainan IQ Puzzle Pro

IQ Puzzler Pro adalah permainan teka-teki yang mengasah logika dan keterampilan berpikir spasial melalui berbagai tantangan 2D dan 3D. Cara bermainnya dimulai dengan memilih salah satu dari 120 tantangan yang tersedia dalam buku petunjuk, dengan tingkat kesulitan mulai dari mudah hingga ahli. Setiap tantangan memiliki susunan awal tertentu yang harus diikuti dengan menempatkan beberapa potongan puzzle di papan permainan sesuai petunjuk.

Tugas pemain adalah melengkapi papan dengan menyusun potongan yang tersisa hingga tidak ada celah yang tersisa. Permainan ini memiliki dua mode utama, yaitu mode 2D, di mana pemain mengisi papan datar, dan mode 3D, di mana pemain menyusun potongan puzzle untuk membentuk piramida di area khusus. Jika semua potongan telah terpasang dengan benar sesuai aturan tantangan, maka puzzle dianggap selesai. Jika belum, pemain harus mencoba kombinasi lain hingga menemukan solusi yang tepat. Dengan konsep yang sederhana tetapi menantang, IQ Puzzler Pro menjadi permainan yang menyenangkan sekaligus melatih kemampuan pemecahan masalah dan konsentrasi.

1.2 Brute Force

Brute force adalah metode pemecahan masalah yang mencoba semua kemungkinan solusi secara sistematis hingga menemukan jawaban yang benar. Pendekatan ini tidak memerlukan strategi kompleks, melainkan hanya mengandalkan pencarian menyeluruh dengan mencoba setiap opsi satu per satu. Karena mencakup semua kemungkinan, brute force selalu menemukan solusi jika solusi tersebut memang ada. Namun, kelemahannya adalah metode ini bisa sangat tidak efisien jika jumlah kemungkinan terlalu banyak, karena memerlukan waktu dan sumber daya yang besar.

Dalam kehidupan sehari-hari, brute force dapat diterapkan, misalnya, untuk membuka kunci kombinasi dengan mencoba setiap angka secara berurutan,

menebak kata sandi dengan mencoba berbagai kombinasi karakter, atau menyelesaikan teka-teki seperti Sudoku dengan menguji semua kemungkinan angka. Meskipun brute force mudah diterapkan dan sangat cocok untuk komputer yang mampu memproses jutaan kemungkinan dengan cepat, metode ini sering kali dianggap sebagai solusi terakhir karena kurang efisien dibandingkan dengan pendekatan yang lebih cerdas dan terstruktur.

1.3 Algoritma Penyelesaian Permainan IQ Puzzle Pro dengan Metode Brute Force

Dalam penyelesaian permainan IQ Puzzler Pro, metode brute force dapat digunakan untuk menemukan solusi dengan mencoba semua kemungkinan susunan potongan puzzle hingga mendapatkan kombinasi yang benar. Karena permainan ini melibatkan penyusunan berbagai bentuk potongan dalam ruang terbatas, brute force berarti secara sistematis mencoba setiap kemungkinan posisi dan orientasi potongan hingga semua bagian terpasang dengan sempurna.

Meskipun brute force dapat menjamin solusi, metode ini sangat tidak efisien jika diterapkan secara manual, karena jumlah kombinasi yang mungkin bisa sangat besar, terutama pada tantangan yang lebih sulit. Oleh karena itu, pemain biasanya mengandalkan logika, pola, dan intuisi spasial untuk menyaring kemungkinan yang tidak relevan, sehingga tidak perlu mencoba setiap kombinasi secara acak. Namun, dalam pemrograman, brute force bisa diterapkan untuk membuat algoritma pencarian solusi otomatis yang dapat menguji jutaan kemungkinan dengan cepat hingga menemukan susunan yang benar. Dengan demikian, meskipun brute force bukan metode ideal bagi manusia dalam menyelesaikan IQ Puzzler Pro, pendekatan ini tetap relevan dalam konteks komputasi atau analisis sistematis permainan.

BAB II
IMPLEMENTASI ALGORITMA BRUTE FORCE

Program pemgimplementasian algortima brute force dalam menyelesaikan permainan IQ Puzzle Pro menggunakan bahasa Java. Struktur program ini terdiri dari 4 File. Filenya terdiri dari **solve.java, Rotation.java**, **Conditions.java**, dan **Print.java**.

2.1 File solve.py

Fungsi	Deskripsi
main	Tujuan: Menjalankan program untuk menyelesaikan puzzle berdasarkan file input. Algoritma: 1. Mulai pencatatan waktu (startTime). 2. Memuat puzzle dari file menggunakan loadPuzzle. 3. Jika solusi ditemukan dengan solvePuzzle(0), panggil handleSolution untuk menangani penyimpanan hasil. 4. Jika tidak ada solusi, cetak "No solution found."
handleSolution	Tujuan: Menampilkan dan menyimpan solusi jika ditemukan. Algoritma: 1. Dapatkan string representasi dari board menggunakan Print.getBoardAsPlainString(boa

	rd). 2. Cetak board menggunakan Print.printBoard(board). 3. Hitung waktu yang diperlukan untuk mencari solusi dan jumlah kasus yang diuji. 4. Tanya pengguna apakah ingin menyimpan solusi. 5. Jika ya, simpan solusi ke dalam file teks menggunakan saveSolution. 6. Tanya lagi apakah ingin menyimpan solusi dalam bentuk gambar. 7. Jika ya, simpan solusi sebagai gambar menggunakan Print.saveSolutionAsImage.
saveSolution	Tujuan: Menyimpan solusi ke dalam file solusi.txt. Algoritma: 1. Gunakan FileWriter untuk menulis string solusi ke dalam file solusi.txt. 2. Jika terjadi error saat menyimpan, cetak pesan error.
loadPuzzle	Tujuan: Membaca puzzle dari file dan mempersiapkan board serta pieces. Algoritma: 1. Buka file dan baca ukuran board (N, M) serta jumlah jenis pieces (P). 2. Baca tipe board: - Jika DEFAULT, isi board dengan karakter “.” (kosong).

	- Jika CUSTOM, baca board dari file. 3. Cetak board yang telah di-load. 4. Inisialisasi listperhuruf (map untuk menyimpan bentuk pieces berdasarkan huruf). 5. Baca pieces dari file hingga mencapai jumlah P + 10. 6. Konversi setiap bentuk piece menjadi matriks dengan convertToMatrix dan simpan dalam pieces.
convertToMatrix	Tujuan: Mengubah daftar string yang merepresentasikan bentuk piece menjadi matriks karakter. Algoritma: 1. Hitung jumlah baris (rows) dan panjang maksimum kolom (cols). 2. Buat matriks ukuran rows × cols dan isi dengan karakter “.”. 3. Iterasi setiap baris dalam daftar: 4. Jika karakter pada indeks tertentu sesuai dengan jenishuruf, simpan dalam matriks. 5. Jika tidak, isi dengan “.”.
solve	Tujuan: Menyusun pieces ke dalam board menggunakan pencarian backtracking. Algoritma (Backtracking): 1. Jika semua pieces telah ditempatkan (pieceIndex ==

	pieces.size()), periksa apakah board sudah penuh (Conditions.isBoardFilled). 2. Ambil piece saat ini (pieces.get(pieceIndex)). 3. Dapatkan semua rotasi dan orientasi piece menggunakan Rotation.pemutaranPiece(piece). 4. Coba tempatkan setiap orientasi piece di setiap posisi (r, c) pada board: - Jika bisa ditempatkan (Conditions.canPlace), letakkan di board (Conditions.placePiece). - Panggil solvePuzzle(pieceIndex + 10) untuk mencoba piece berikutnya. - Jika berhasil, kembalikan true. - Jika tidak berhasil, hapus piece dari board (Conditions.removePiece) dan lanjutkan mencoba posisi lain. - Tingkatkan penghitung caseCount setiap kali melakukan percobaan baru. 5. Jika tidak ada penempatan yang valid, kembalikan false.

2.2 File Rotation.java

Fungsi	Deskripsi
rotate90	Tujuan:Melakukan rotasi 90° searah jarum jam pada matriks.

	Algoritma: 1. Buat matriks baru hasil dengan ukuran terbalik (cols × rows). 2. Iterasi setiap elemen (r, c) dari matrix: 3. Elemen pada baris r dan kolom c dipindahkan ke baris c dan kolom rows - r - 1 dalam hasil. 4. Kembalikan matriks hasil sebagai hasil rotasi.
reflectHorizontally	Tujuan: Melakukan refleksi horizontal (membalik kiri-kanan). Algoritma: 1. Buat matriks baru hasil dengan ukuran yang sama. 2. Iterasi setiap elemen (r, c) dari matrix: 3. Elemen di posisi (r, c) dipindahkan ke (r, cols - c - 1) dalam hasil. 4. Kembalikan hasil sebagai hasil refleksi.
saveSolution	Tujuan: Melakukan refleksi vertikal (membalik atas-bawah). Algoritma: 1. Buat matriks baru hasil dengan ukuran yang sama. 2. Iterasi setiap elemen (r, c) dari matrix: 3. Elemen di posisi (r, c) dipindahkan ke (rows - r - 1, c) dalam hasil. 4. Kembalikan hasil sebagai hasil refleksi.

pemutaranPiece	Tujuan: Menghasilkan semua orientasi unik dari sebuah bentuk dalam matriks, termasuk: - 4 Rotasi (0°, 90°, 180°, 270°). - 2 Refleksi (Horizontal & Vertikal). Algoritma: 1. Buat ListPemutaran untuk menyimpan semua orientasi. 2. Hasilkan 4 rotasi: - Simpan bentuk awal ke dalam daftar. - Ulangi 4 kali: ● Rotasi piece 90°. ● Tambahkan hasilnya ke ListPemutaran. 3. Hasilkan 2 refleksi dari setiap rotasi: - Untuk setiap elemen dalam ListPemutaran: - Untuk setiap elemen dalam ListPemutaran: ● Tambahkan hasil refleksi horizontal. ● Tambahkan hasil refleksi vertikal. 4. Hilangkan duplikasi: - Gunakan Set<String> untuk menyimpan hasil unik (mengonversi setiap matriks ke string). - Tambahkan hanya jika belum ada dalam Set. 5. Kembalikan daftar orientasi

	unik.
matrixToString	Tujuan: Fungsi ini mengubah matriks 2D char menjadi sebuah String dengan format setiap baris dipisahkan oleh karakter newline (\n). Ini berguna untuk: - Menyimpan representasi unik dari matriks dalam Set<String> (agar bisa digunakan untuk menghilangkan duplikasi). - Debugging (agar mudah melihat isi matriks dalam bentuk teks). - Mencetak matriks dengan format yang jelas. Algoritma: 1. Buat StringBuilder sb untuk menyusun string secara efisien. 2. Loop setiap baris row dalam matrix: - Loop setiap karakter ch dalam row, lalu tambahkan ke sb. - Setelah satu baris selesai, tambahkan karakter newline (\n) sebagai pemisah baris. Kembalikan sb.toString() sebagai hasil konversi.
copyMatrix	Tujuan: Membuat salinan independen dari matriks tanpa mengubah matriks asli. Algoritma: 1. Buat array 2D baru copy dengan jumlah baris yang sama dengan matrix.

	2. Iterasi setiap baris matrix[i]: - Gunakan Arrays.copyOf(matrix[i], matrix[i].length) untuk membuat salinan terpisah dari setiap baris. - Ini memastikan bahwa perubahan pada copy tidak mempengaruhi matrix asli. 3. Kembalikan copy sebagai hasil salinan.

2.3 File Print.java

Fungsi	Deskripsi
colorMap	Tujuan: Menyimpan peta warna ANSI yang digunakan untuk mencetak karakter dengan warna yang berbeda di terminal. Algoritma: 1. colorMap adalah Map<Character, String> yang memetakan setiap karakter (misalnya 'A', 'B', dll.) ke kode warna ANSI. 2. Warna-warna ini akan digunakan saat mencetak papan permainan (board) di terminal.
getBoardAsPlainString	Tujuan: Mengonversi papan permainan (board) menjadi string tanpa warna untuk tampilan sederhana. Algoritma:

	1. Menggunakan StringBuilder untuk membangun string. 2. Melakukan iterasi melalui setiap baris (board[i]). 3. Menambahkan setiap karakter ke StringBuilder. 4. Menambahkan newline ("\n") di akhir setiap baris. 5. Mengembalikan string hasil konversi.
saveSolutionAsImage	Tujuan: Menyimpan representasi papan permainan (board) sebagai gambar PNG. Algoritma: 1. Menentukan ukuran sel (cellSize = 50). 2. Menghitung dimensi gambar berdasarkan jumlah baris (N) dan kolom (M). 3. Membuat BufferedImage dengan mode TYPE_INT_ARGB. 4. Menginisialisasi Graphics2D untuk menggambar. 5. Membuat colorMap dengan warna Color yang berbeda untuk setiap karakter. 6. Melakukan iterasi melalui board: - Jika karakter bukan titik “.”, menggambar sel dengan warna yang sesuai. - Menggambar garis hitam sebagai batas setiap sel. 7. Menutup Graphics2D dan menyimpan gambar dalam format PNG.

2.4 File Conditions.java

Fungsi	Deskripsi
canPlace	Fungsi : Metode ini memeriksa apakah suatu potongan (piece) bisa diletakkan di papan (board) pada posisi yang ditentukan (r, c). Algoritma: 1. Dapatkan ukuran matriks piece dalam variabel rows (jumlah baris) dan cols (jumlah kolom). 2. Dapatkan ukuran matriks board dalam variabel N (jumlah baris papan) dan M (jumlah kolom papan). 3. Periksa batasan papan: - Jika menempatkan piece pada posisi (r, c) akan keluar dari batas papan (r + rows > N atau c + cols > M), kembalikan false (tidak bisa ditempatkan). 4. Periksa tabrakan dengan bagian papan yang sudah terisi: - Iterasi setiap sel dalam piece dan periksa apakah bagian tersebut akan bertabrakan dengan bagian papan yang sudah terisi (board[r + i][c + j] != '.'). - Jika ada satu saja bagian piece yang bukan '.' (kosong) dan bertemu dengan sel papan yang

	sudah terisi, maka kembalikan false. 5. Jika semua pemeriksaan lolos, kembalikan true, yang berarti piece dapat ditempatkan di lokasi yang diinginkan.
placePiece	Fungsi : Metode ini menempatkan piece ke dalam board pada posisi (r, c). Algoritma: 1. Dapatkan ukuran matriks piece dalam variabel rows dan cols. 2. Iterasi setiap elemen dalam piece: - Jika sel piece[i][j] bukan '.', salin ke posisi board[r + i][c + j] untuk meletakkannya di papan. 3. Setelah metode ini dijalankan, board akan menampilkan piece yang telah ditempatkan.
removePiece	Fungsi: Metode ini menghapus piece dari board dengan mengembalikan semua sel yang ditempati piece menjadi “.” (kosong). Algoritma: 1. Dapatkan ukuran piece dalam variabel rows dan cols. 2. Iterasi setiap elemen dalam piece: - Jika sel piece[i][j] bukan '.', ubah sel pada papan board[r + i][c + j] menjadi '.', sehingga area tersebut menjadi kosong kembali.

	3. Setelah metode ini dijalankan, board akan kembali seperti sebelum piece ditempatkan.
isBoardFilled	Fungsi: Metode ini memeriksa apakah papan (board) sudah penuh, yaitu tidak ada lagi sel '.' yang kosong. Algoritma : 1. Iterasi setiap sel dalam board: - Jika menemukan sel yang masih “.”, kembalikan false. 2. Jika semua sel sudah terisi, kembalikan true.

BAB III
SOURCE CODE PROGRAM

3.1 Source Code Program

solve.java

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

public class solve {
   private static int N, M, P;
   private static char[][] board;
   private static List<char[][]> pieces = new ArrayList<>();
   private static int caseCount = 0;
   private static long startTime;



   public static void main(String[] args) throws IOException {
       Scanner scanner = new Scanner(System.in);
       System.out.print("Masukkan nama file input : ");
       String filename = scanner.nextLine();
       String filepath =
"/Users/jonathankenanbudianto/Documents/coding/java/Tucil_stima_1_(IQ puzzler
pro)/test/input/" + filename;

       startTime = System.currentTimeMillis();
       loadPuzzle(filepath);

       if (solvePuzzle(0)) {
           handleSolution();
       } else {
           System.out.println("No solution found.");
       }
   }

   private static void handleSolution() throws IOException {
```

```
        String solusi = Print.getBoardAsPlainString(board); // Menggunakan warna
dalam ngeprint
        Print.printBoard(board);

        // Output waktu dan casecount
        long endTime = System.currentTimeMillis();
        System.out.println("Waktu pencarian: " + (endTime - startTime) + " ms");
        System.out.println("Banyak kasus yang ditinjau: " + caseCount);
        System.out.print("Apakah anda ingin menyimpan solusi? (ya/tidak): ");
        Scanner scanner = new Scanner(System.in);
        String response = scanner.nextLine();

        if (response.equalsIgnoreCase("ya")) {
            saveSolution(solusi);
            System.out.print("Apakah anda ingin menyimpan solusi sebagai gambar?
(ya/tidak): ");
            response = scanner.nextLine();
            if (response.equalsIgnoreCase("ya")) {
                Print.saveSolutionAsImage(board,
"/Users/jonathankenanbudianto/Documents/coding/java/Tucil_stima_1_(IQ puzzler
pro)/test/output/solusi.png");
                System.out.println("Solusi telah disimpan sebagai gambar
solusi.png");
            } else {
                System.out.println("Solusi tidak disimpan sebagai gambar.");
            }
        } else {
            System.out.println("Solusi tidak disimpan.");
        }
    }

    private static void saveSolution(String solusi) {
        try {
            FileWriter writer = new
FileWriter("/Users/jonathankenanbudianto/Documents/coding/java/Tucil_stima_1_(IQ
puzzler pro)/test/output/solusi.txt");
            writer.write(solusi);
            writer.close();
            System.out.println("Solusi telah disimpan dalam berkas solusi.txt");
        } catch (IOException e) {
```

```
            System.out.println("Terjadi kesalahan saat menyimpan solusi.");
            e.printStackTrace();
        }
    }

    private static void loadPuzzle(String filename) throws IOException {
        Scanner scanner = new Scanner(new File(filename));
        N = scanner.nextInt();
        M = scanner.nextInt();
        P = scanner.nextInt();
        scanner.nextLine(); // Pindah ke baris berikutnya

        String boardType = scanner.nextLine(); // Baca tipe board

        if (boardType.equals("DEFAULT")) {
        board = new char[N][M];
            for (int i = 0; i < N; i++) {
                for (int j = 0; j < M; j++) {
                    board[i][j] = '.';
                }
            }
        }
        else if (boardType.equals("CUSTOM")) {
            board = new char[N][M];
            for (int i = 0; i < N; i++) {
                String line = scanner.nextLine();
                for (int j = 0; j < M; j++) {
                    board[i][j] = line.charAt(j);
                }
            }
        }

        int jumlahjenishuruf = 0; // Penghitung jumlah jenis huruf yang sudah
diproses
        Map<Character, List<String>> listperhuruf = new LinkedHashMap<>();

        while (scanner.hasNextLine() && jumlahjenishuruf < P+1) {
            String line = scanner.nextLine();
            if (line.isEmpty()) continue; // Lewati baris kosong
```

```
            // Ambil huruf pertama sebagai label
            char jenishuruf = '\0';
            for (char ch : line.toCharArray()) {
                if (ch != ' ') {
                    jenishuruf = ch;
                    break;
                }
            }

            // Jika huruf baru ditemukan dan jumlah sudah mencapai P, berhenti
membaca
            if (!listperhuruf.containsKey(jenishuruf)) {
                if (jumlahjenishuruf >= P+10) break;
                listperhuruf.put(jenishuruf, new ArrayList<>());
                jumlahjenishuruf++;
            }

            listperhuruf.get(jenishuruf).add(line);
        }
        scanner.close();

        // Konversi setiap bentuk ke dalam matriks dan simpan ke daftar pieces
        for (Map.Entry<Character, List<String>> entry : listperhuruf.entrySet()) {
            char jenishuruf = entry.getKey();
            pieces.add(convertToMatrix(entry.getValue(), jenishuruf));
        }
    }

    private static char[][] convertToMatrix(List<String> listperhuruf, char
jenishuruf) {
        int rows = listperhuruf.size();
        int cols = 0;

        // Hitung panjang maksimal dari semua baris
        for (int i = 0; i < listperhuruf.size(); i++) {
            String spesificline = listperhuruf.get(i);
            if (spesificline.length() > cols) {
                cols = spesificline.length();
            }
        }
```

```
        char[][] matrix = new char[rows][cols];

        // Mengisi matriks dengan karakter yang sesuai
        for (int i = 0; i < rows; i++) {
            String spesificline = listperhuruf.get(i);
            for (int j = 0; j < cols; j++) {
                if (j < spesificline.length() && spesificline.charAt(j) ==
jenishuruf) {
                    matrix[i][j] = jenishuruf;
                } else {
                    matrix[i][j] = '.'; // Mengisi bagian kosong dengan titik
                }
            }
        }
        return matrix;
    }

    private static boolean solvePuzzle(int pieceIndex) {
        if (pieceIndex == pieces.size()) {
            return Conditions.isBoardFilled(board);
        }
        char[][] piece = pieces.get(pieceIndex);
        List<char[][]> ListPemutaran = Rotation.pemutaranPiece(piece);

        for (char[][] currentPiece : ListPemutaran) {
            for (int r = 0; r < N; r++) {
                for (int c = 0; c < M; c++) {
                    if (Conditions.canPlace(currentPiece, board, r, c)) {
                        Conditions.placePiece(currentPiece, board, r, c);
                        if (solvePuzzle(pieceIndex + 1)) return true;
                        Conditions.removePiece(currentPiece, board, r, c);
                        caseCount++;
                    }
                }
            }
        }
        return false;
    }
}
```

Rotation.java

```java
import java.util.*;

public class Rotation {

   public static char[][] rotate90(char[][] matrix) {
       int rows = matrix.length;
       int cols = matrix[0].length;
       char[][] hasil = new char[cols][rows];
       for (int r = 0; r < rows; r++) {
           for (int c = 0; c < cols; c++) {
               hasil[c][rows - r - 1] = matrix[r][c];
           }
       }
       return hasil;
   }

   public static char[][] reflectHorizontally(char[][] matrix) {
       int rows = matrix.length;
       int cols = matrix[0].length;
       char[][] hasil = new char[rows][cols];
       for (int r = 0; r < rows; r++) {
           for (int c = 0; c < cols; c++) {
               hasil[r][cols - c - 1] = matrix[r][c];
           }
       }
       return hasil;
   }

   public static char[][] reflectVertically(char[][] matrix) {
       int rows = matrix.length;
       int cols = matrix[0].length;
       char[][] hasil = new char[rows][cols];
       for (int r = 0; r < rows; r++) {
           for (int c = 0; c < cols; c++) {
               hasil[rows - r - 1][c] = matrix[r][c];
```

```
            }
        }
        return hasil;
    }

    public static List<char[][]> pemutaranPiece(char[][] piece) {
        List<char[][]> ListPemutaran = new ArrayList<>();

        for (int i = 0; i < 4; i++) {
            ListPemutaran.add(copyMatrix(piece));
            piece = rotate90(piece);
        }

        List<char[][]> mirror = new ArrayList<>();
        for (char[][] orientation : ListPemutaran) {
            mirror.add(reflectHorizontally(orientation));
            mirror.add(reflectVertically(orientation));
        }

        ListPemutaran.addAll(mirror);

        Set<String> uniqueOrientations = new HashSet<>();
        List<char[][]> uniqueListPemutaran = new ArrayList<>();

        for (char[][] orientation : ListPemutaran) {
            String orientationString = matrixToString(orientation);
            if (uniqueOrientations.add(orientationString)) {
                uniqueListPemutaran.add(orientation);
            }
        }

        return uniqueListPemutaran;
    }

    public static String matrixToString(char[][] matrix) {
        StringBuilder sb = new StringBuilder();
        for (char[] row : matrix) {
            for (char ch : row) {
                sb.append(ch);
            }
```

```
            sb.append('\n');
        }
        return sb.toString();
    }

    private static char[][] copyMatrix(char[][] matrix) {
        char[][] copy = new char[matrix.length][];
        for (int i = 0; i < matrix.length; i++) {
            copy[i] = Arrays.copyOf(matrix[i], matrix[i].length);
        }
        return copy;
    }
}
```

Print.java

```
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.*;
import javax.imageio.ImageIO;

public class Print {

    public static final Map<Character, String> colorMap = new HashMap<>();
    static {
        colorMap.put('A', "\u001B[31;1m"); // Merah terang
        colorMap.put('B', "\u001B[32;1m"); // Hijau terang
        colorMap.put('C', "\u001B[33;1m"); // Kuning terang
        colorMap.put('D', "\u001B[34;1m"); // Biru terang
        colorMap.put('E', "\u001B[35;1m"); // Ungu terang
        colorMap.put('F', "\u001B[36;1m"); // Cyan terang
        colorMap.put('G', "\u001B[91;1m"); // Merah terang
        colorMap.put('H', "\u001B[92;1m"); // Hijau terang
        colorMap.put('I', "\u001B[93;1m"); // Kuning terang
        colorMap.put('J', "\u001B[94;1m"); // Biru terang
```

```
        colorMap.put('K', "\u001B[95;1m"); // Ungu terang
        colorMap.put('L', "\u001B[96;1m"); // Cyan terang
        colorMap.put('M', "\u001B[97;1m"); // Putih terang
        colorMap.put('N', "\u001B[30;1m"); // Hitam terang
        colorMap.put('O', "\u001B[90;1m"); // Abu-abu terang
        colorMap.put('P', "\u001B[31m"); // Merah
        colorMap.put('Q', "\u001B[32m"); // Hijau
        colorMap.put('R', "\u001B[33m"); // Kuning
        colorMap.put('S', "\u001B[34m"); // Biru
        colorMap.put('T', "\u001B[35m"); // Ungu
        colorMap.put('U', "\u001B[36m"); // Cyan
        colorMap.put('V', "\u001B[37m"); // Putih
        colorMap.put('W', "\u001B[90m"); // Abu-abu
        colorMap.put('X', "\u001B[91m"); // Merah terang
        colorMap.put('Y', "\u001B[92m"); // Hijau terang
        colorMap.put('Z', "\u001B[93m"); // Kuning terang
        colorMap.put('.', "\u001B[37m"); // Putih (default)
    }
    public static final String RESET = "\u001B[0m"; // Reset warna ke default

    public static String getBoardAsPlainString(char[][] board) {
        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < board.length; i++) {
            for (int j = 0; j < board[i].length; j++) {
                sb.append(board[i][j]);
            }
            sb.append("\n");
        }
        return sb.toString();
    }

    public static void printBoard(char[][] board) {
        for (int i = 0; i < board.length; i++) {
            for (int j = 0; j < board[i].length; j++) {
                String color = colorMap.getOrDefault(board[i][j], "\u001B[37m"); //
Default putih
                System.out.print(color + board[i][j] + RESET);
            }
            System.out.println();
```

```
    }
}

public static void printPieces(List<char[][]> pieces) {
    System.out.println("Isi pieces:");
    for (int i = 0; i < pieces.size(); i++) {
        char[][] piece = pieces.get(i);
        System.out.println("[");
        for (int j = 0; j < piece.length; j++) {
            char[] row = piece[j];
            System.out.print("  " + Arrays.toString(row));
            System.out.println();
        }
        System.out.println("]");
    }
}

public static void printListPemutaran(List<char[][]> ListPemutaran) {
    int index = 0;
    for (char[][] orientation : ListPemutaran) {
        System.out.println("Orientation " + index + ":");
        for (char[] row : orientation) {
            for (char cell : row) {
                System.out.print(cell);
            }
            System.out.println();
        }
        System.out.println();
        index++;
    }
}

public static void printListPerHuruf(Map<Character, List<String>> listperhuruf) {
    for (Map.Entry<Character, List<String>> entry : listperhuruf.entrySet()) {
        System.out.println("Character: " + entry.getKey());
        for (String line : entry.getValue()) {
            System.out.println(line);
        }
        System.out.println();
    }
```

```
    }

    public static void saveSolutionAsImage(char[][] board, String filename) throws
IOException {
        int cellSize = 50; // ukuran cell di gambar
        int N = board.length;
        int M = board[0].length;
        BufferedImage image = new BufferedImage(M * cellSize, N * cellSize,
BufferedImage.TYPE_INT_ARGB);
        Graphics2D g = image.createGraphics();

        // Pendefinisian warna digambar
        Map<Character, Color> colorMap = new HashMap<>();
        colorMap.put('A', Color.RED);
        colorMap.put('B', Color.GREEN);
        colorMap.put('C', Color.BLUE);
        colorMap.put('D', Color.YELLOW);
        colorMap.put('E', Color.MAGENTA);
        colorMap.put('F', Color.CYAN);
        colorMap.put('G', Color.ORANGE);
        colorMap.put('H', Color.PINK);
        colorMap.put('I', Color.LIGHT_GRAY);
        colorMap.put('J', Color.DARK_GRAY);
        colorMap.put('K', Color.BLACK);
        colorMap.put('L', Color.WHITE);
        colorMap.put('M', Color.GRAY);
        colorMap.put('N', Color.DARK_GRAY);
        colorMap.put('O', Color.LIGHT_GRAY);
        colorMap.put('P', Color.RED.darker());
        colorMap.put('Q', Color.GREEN.darker());
        colorMap.put('R', Color.BLUE.darker());
        colorMap.put('S', Color.YELLOW.darker());
        colorMap.put('T', Color.MAGENTA.darker());
        colorMap.put('U', Color.CYAN.darker());
        colorMap.put('V', Color.ORANGE.darker());
        colorMap.put('W', Color.PINK.darker());
        colorMap.put('X', Color.LIGHT_GRAY.darker());
        colorMap.put('Y', Color.DARK_GRAY.darker());
        colorMap.put('Z', Color.BLACK.darker());
```

```
        // Menggambar papan
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                char cell = board[i][j];
                if (cell != '.') {
                    g.setColor(colorMap.getOrDefault(cell, Color.BLACK));
                    g.fillRect(j * cellSize, i * cellSize, cellSize, cellSize);
                }
                g.setColor(Color.BLACK);
                g.drawRect(j * cellSize, i * cellSize, cellSize, cellSize);
            }
        }

        g.dispose();
        ImageIO.write(image, "png", new File(filename));
    }
}
```

Conditions.java

```
public class Conditions {
    public static boolean canPlace(char[][] piece, char[][] board, int r, int c) {
        int rows = piece.length, cols = piece[0].length;
        int N = board.length, M = board[0].length;
        if (r + rows > N || c + cols > M) return false;

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (piece[i][j] != '.' && board[r + i][c + j] != '.') {
                    return false;
                }
            }
        }
        return true;
    }

    public static void placePiece(char[][] piece, char[][] board, int r, int c) {
        int rows = piece.length, cols = piece[0].length;
```

```
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (piece[i][j] != '.') {
                    board[r + i][c + j] = piece[i][j];
                }
            }
        }
    }

    public static void removePiece(char[][] piece, char[][] board, int r, int c) {
        int rows = piece.length, cols = piece[0].length;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (piece[i][j] != '.') {
                    board[r + i][c + j] = '.';
                }
            }
        }
    }
    public static boolean isBoardFilled(char[][] board) {
        for (int i = 0; i < board.length; i++) {
            for (int j = 0; j < board[i].length; j++) {
                if (board[i][j] == '.') {
                    return false;
                }
            }
        }
        return true;
    }
}
```

BAB IV
INPUT DAN OUTPUT PROGRAM

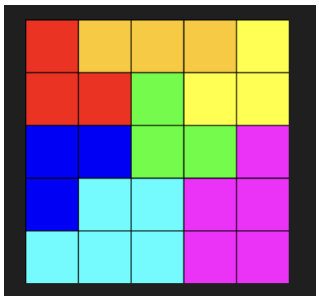
4.1 Test case 1

Input :

```
5 5 8
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
```

Output :

```
Masukkan nama file input : testcase.txt
AGGGD
AABDD
CCBBE
CFFEE
FFFEE
Waktu pencarian: 71 ms
Banyak kasus yang ditinjau: 7386
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi telah disimpan dalam berkas solusi.txt
Apakah anda ingin menyimpan solusi sebagai gambar? (ya/tidak): ya
Solusi telah disimpan sebagai gambar solusi.png
2025-02-24 11:45:28.586 java[93554:4926469] +[IMKClient subclass]: chose IMKClient_Modern
2025-02-24 11:45:28.586 java[93554:4926469] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```



```
AGGGD
AABDD
CCBBE
CFFEE
FFFEE
```

4.2 Test case 2

Input :

```
3 4 4
DEFAULT
A
A
A
AA
B
BB
C
CC
D
```

Output :

```
Masukkan nama file input : testcase3.txt
AAAA
ABCC
DBBC
Waktu pencarian: 23 ms
Banyak kasus yang ditinjau: 1
Apakah anda ingin menyimpan solusi dalam file? (ya/tidak): tidak
Solusi tidak disimpan.
```

4.3 Test case 3

Input :

```
5 5 8
DEFAULT
AA
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
```

(test case 1 namun ditambah piecenya)

Output :

```
Masukkan nama file input : testcase1.txt
No solution found.
```

4.4 Test case 4

Input :

```
3 4 4
DEFAULT
A
A
A
A
B
BB
C
CC
D
```

(test case 2 namun dikurangi pecenya)

Output :

```
Masukkan nama file input : testcase4.txt
No solution found.
```

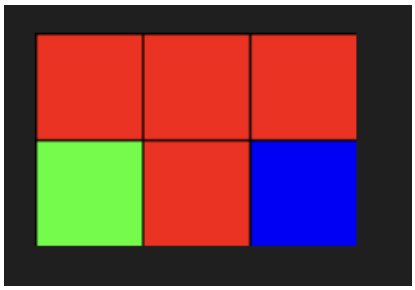
4.5 Test case 5

Input :

```
2 3 3
DEFAULT
AAA
 A
B
C
```

Output :

```
AAA
BAC
Waktu pencarian: 20 ms
Banyak kasus yang ditinjau: 1
Apakah anda ingin menyimpan solusi dalam file? (ya/tidak): ya
Solusi telah disimpan dalam berkas solusi.txt
Apakah anda ingin menyimpan solusi sebagai gambar? (ya/tidak): ya
Solusi telah disimpan sebagai gambar solusi.png
2025-02-24 12:46:20.021 java[95738:4990759] +[IMKClient subclass]: chose IMKClient_Modern
2025-02-24 12:46:20.021 java[95738:4990759] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```



```
AAA
BAC
```

4.6 Test case 6

Input :

```
2 3 4
DEFAULT
AAA
 A
B
C
D
```

Output :

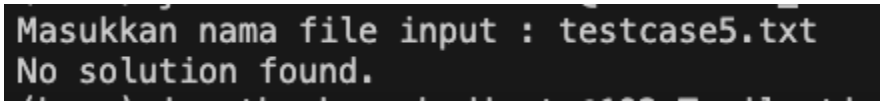
```
Masukkan nama file input : testcase5.txt
No solution found.
```

4.7 Test case 7

Input :

```
5 5 6
DEFAULT
AAA
A
B
C
D
GG
G
```

Output :



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	✓	