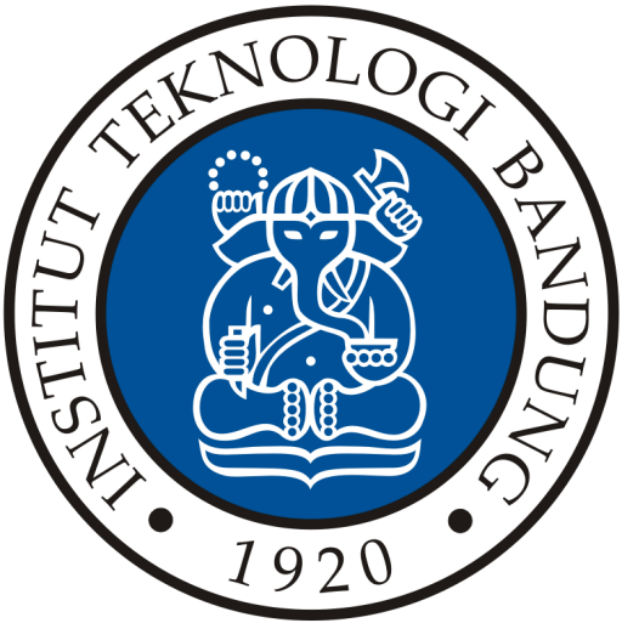
Laporan Tugas Kecil 1 IF2211 Strategi Algortima SEMESTER II 2024/2025

Pencarian Solusi IQ Puzzle Pro Menggunakan Metode Brute Force

Oleh

Qodri Azkarayan

NIM : 13523010



PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG

Februari 2025

DAFTAR ISI

BAB I

DESKRIPSI TUGAS

I.1 IQ Puzzler Pro

IQ Puzzler Pro adalah permainan papan yang diproduksi oleh perusahaan Smart Games dengan tujuan agar pemain dapat mengisi seluruh papan dengan piece atau blok puzzle yang telah disediakan. Permainan ini terdiri dari dua komponen utama, yaitu board (papan permainan) dan piece atau blok. Board merupakan komponen utama yang menjadi tempat dan tujuan permainan, di mana pemain harus mengisi seluruh area papan menggunakan blok yang tersedia. Sementara itu, piece atau 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 dalam keadaan kosong. Pemain dapat meletakkan blok puzzle sedemikian rupa sehingga tidak ada blok yang tumpang tindih, kecuali dalam kasus mode 3D. Setiap blok dapat diputar atau dicerminkan agar sesuai dengan papan permainan. Puzzle dianggap selesai jika papan terisi penuh dan seluruh blok berhasil ditempatkan tanpa ada yang tersisa.

Tugas dari program ini adalah menemukan setidaknya satu solusi dari permainan IQ Puzzler Pro dengan menggunakan algoritma Brute Force. Jika tidak ada solusi yang memungkinkan, program harus menampilkan bahwa solusi tidak ditemukan.

I.2 Algoritma Brute Force

Algoritma Brute Force adalah pendekatan pemecahan masalah yang mencoba semua kemungkinan solusi secara sistematis hingga menemukan solusi yang benar atau memastikan bahwa tidak ada solusi yang valid. Metode ini bekerja dengan mengeksplorasi seluruh ruang pencarian tanpa menggunakan optimasi atau heuristik tertentu, sehingga sangat mudah diimplementasikan dan dapat diterapkan pada berbagai jenis masalah.

Brute Force sering digunakan dalam berbagai bidang, termasuk pencarian pola dalam

teks, pemecahan sandi, optimasi sederhana, dan permainan puzzle seperti **IQ Puzzler Pro**. Dalam permainan ini, algoritma akan mencoba semua kemungkinan penempatan blok pada papan hingga menemukan solusi yang benar atau menyimpulkan bahwa tidak ada kombinasi yang memungkinkan.

Meskipun Brute Force selalu memberikan solusi jika ada, kelemahan utamanya adalah kompleksitas waktu yang tinggi, terutama jika jumlah kemungkinan yang harus diperiksa sangat besar. Oleh karena itu, dalam kasus dengan ruang pencarian yang luas, metode ini sering dianggap kurang efisien dibandingkan algoritma yang lebih cerdas.

\

BAB II

STRATEGI ALGORITMA

II.1 Rancangan Algoritma

Dengan algoritma Brute Force, program ini mencoba semua kemungkinan hingga menemukan satu solusi yang valid. Program bekerja dengan langkah berikut

1. Program dimulai dengan meminta pengguna untuk memasukkan nama file test case
    a. File tersebut berisikan:
        i. Ukuran papan (N x M)
        ii. Jumlah potongan puzzle (P)
        iii. Bentuk potongan huruf yang akan diisi ke dalam papan
    b. Program membaca dari file yang diberikan dan mengekstrak dimensi papan beserta jumlah potongan puzzle.
    c. Kemudian, program membaca potongan puzzle yang ada dan menyimpannya dalam bentuk `List<String>`.
    d. Objek Board: Papan berukuran N x M yang diinisialisasi dengan karakter '.' sebagai ruang kosong.
    e. Objek IQPuzzler: Menerima jumlah potongan puzzle dan objek Board. Dalam konstruktor, sebuah list `pieces` diinisialisasi untuk menyimpan semua bentuk potongan yang dibaca dari file.
    f. Metode `validasiInput` digunakan untuk membaca potongan-potongan puzzle dan menyimpannya ke dalam list `pieces`. Setiap potongan disimpan dengan karakter pembeda (AlphabetID) dan bentuknya dalam bentuk array 2D.
2. Algoritma brute force dimulai dalam metode 'permutasi()'.
    a. Basis Rekursi:
        i. Jika seluruh papan sudah terisi, solusi ditemukan dan dicetak.
        ii. Jika seluruh potongan telah diuji dan tidak ditemukan solusi, algoritma berhenti.

b. Pencarian Indeks Kosong: Jika sel papan pada posisi (x, y) tidak kosong, lanjutkan ke sel berikutnya (x, y+1).

c. Mencoba Semua Potongan Puzzle: Jika potongan belum digunakan, cari semua variasi dari potongan tersebut dengan memanggil generateTransformations().

d. Untuk setiap variasi yang ditemukan, diperiksa apakah potongan tersebut dapat dipasang pada posisi `(x, y)` menggunakan `matchAt().

    i. Jika potongan dapat dipasang:

        1. Potongan dipasang menggunakan `cariPermutasi()`.

        2. Ditandai sudah digunakan dengan `setUsed(true)`.

        3. Rekursi ke posisi berikutnya dengan `permutasi(x, y+1)`.

    ii. Jika tidak ditemukan solusi alias potongan tidak dapat dipasang:

        1. Lepaskan potongan dengan `restoreBoard()` dan ditandai sebagai tidak terpakai kembali.

3. Di dalam kelas `IQPuzzler`, status potongan dikelola menggunakan variabel boolean used

    a. Setiap potongan memiliki status apakah telah digunakan atau belum.

    b. Status ini diperiksa di awal setiap iterasi potongan dalam permutasi() untuk menghindari penggunaan ganda dalam satu solusi.

4. Metode generateTransformations() digunakan untuk menciptakan semua variasi potongan dengan melakukan rotasi 90 derajat hingga 360 derajat dan pencerminan horizontal.

5. Setelah semua posisi papan diperiksa, program melakukan validasi dengan metode `isBoardFull()` yang memastikan setiap bagian di board sudah terisi

6. Jika ditemukan solusi, program akan menanyakan apakah pengguna ingin menyimpannya

    a. Jika ya, metode `saveSolution()` di kelas `IQPuzzler` akan menuliskan solusi ke dalam file output yang diinginkan.

    b. Solusi yang disimpan akan mencerminkan susunan potongan yang valid di dalam papan

BAB III

IMPLEMENTASI DALAM BAHASA JAVA

III.1 Source Code

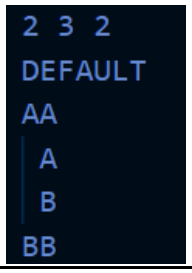
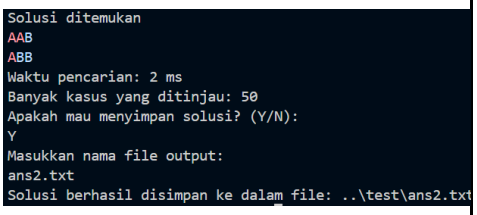
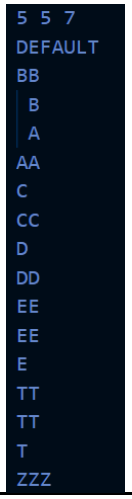
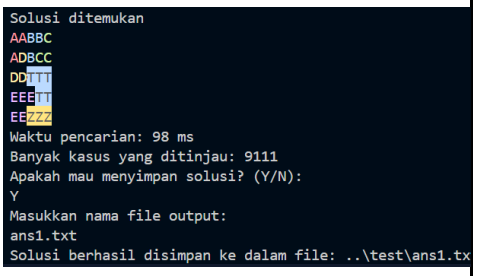
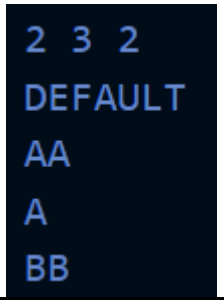
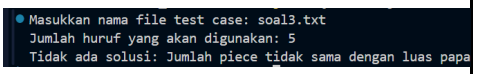
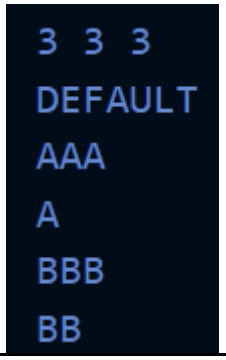
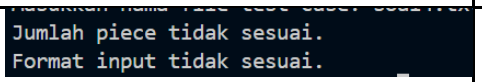
Tautan: https://github.com/qodriazka/Tucil_13523010.git

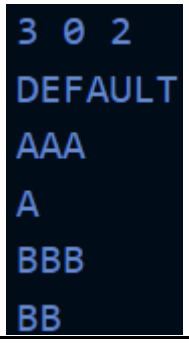
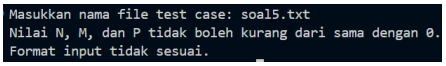
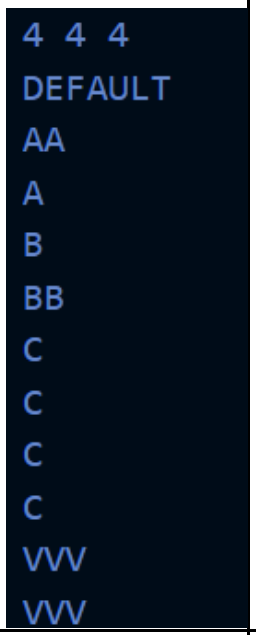
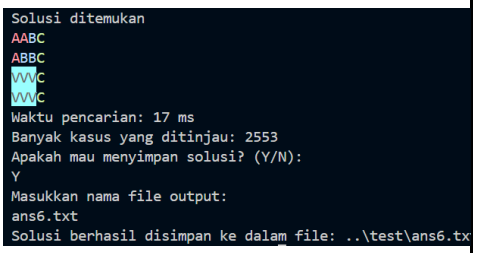
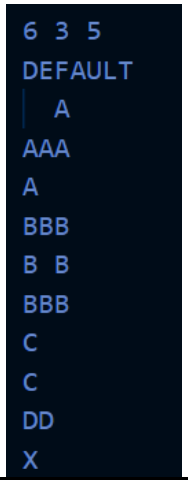
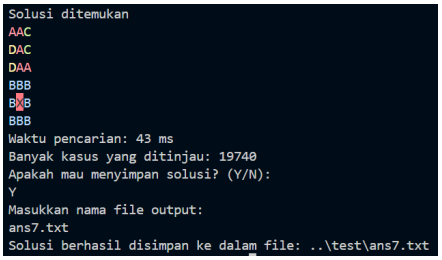
```
import java.io.*;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;

public class IQPuzzler {

static int N, M, P;
static int iterationCount = 0;
static char[][] papan;
static char[][] jawaban;
static List<String> pieces = new ArrayList<>();
static long startTime, endTime;
static boolean foundSolution = false;
static Map<Character, String> bentukPiece;
static Map<Character, Integer> banyakHuruf = new HashMap<>();
static List<char[][]> hasilPermutasi = new ArrayList<>();


@SuppressWarnings("resource")
Run | Debug
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    try {
        System.out.print("Masukkan nama file test case: ");
        String fileName = scanner.nextLine();
        if (!validasiInput(fileName)) {
            System.out.println("Format input tidak sesuai.");
            return;
        }
        int totalPieceCount = pieces.stream().mapToInt(piece -> piece.replace("\n", "").replace(".", "").length()).sum();
        System.out.println("Jumlah huruf yang akan digunakan: " + totalPieceCount);
        if (totalPieceCount != N * M) {
            System.out.println("Tidak ada solusi: Jumlah piece tidak sama dengan luas papan.");
            return;
        }
        System.out.println("\nBentuk setiap piece:");
        for (Map.Entry<Character, String> entry : bentukPiece.entrySet()) {
            System.out.println("Piece '" + entry.getKey() + "':");
```

```
        System.out.println("\nBentuk setiap piece:");
        for (Map.Entry<Character, String> entry : bentukPiece.entrySet()) {
            System.out.println("Piece '" + entry.getKey() + "':");
            System.out.println(entry.getValue());
        }
        System.out.println("\nBanyak huruf setiap piece:");
        for (Map.Entry<Character, Integer> entry : banyakHuruf.entrySet()) {
            System.out.println("Piece '" + entry.getKey() + "' memiliki " + entry.getValue() + " huruf.");
        }
        startTime = System.currentTimeMillis();
        permutasi(papan, bentukPiece, banyakHuruf, indeks:0);
        endTime = System.currentTimeMillis();
        System.out.println("Waktu pencarian: " + (endTime - startTime) + " ms");
        System.out.println("Banyak kasus yang ditinjau: " + iterationCount);
        System.out.println("Apakah mau menyimpan solusi? (Y/N): ");
        String save = scanner.nextLine();

        if (save.equalsIgnoreCase("Y")) {
            System.out.println("Masukkan nama file output: ");
            String outputFileName = scanner.nextLine();
            saveSolution(outputFileName, jawaban);
        }

        scanner.close();

    } catch (Exception e) {
        System.err.println("Error: " + e.getMessage());
    }
}

public static char[][] copyBoard(char[][] papan) {
    char[][] newBoard = new char[papan.length][papan[0].length];
    for (int i = 0; i < papan.length; i++) {
        newBoard[i] = papan[i].clone();
    }
    return newBoard;
```

```java
public static void restoreBoard(char[][] papan, char[][] original) {
    for (int i = 0; i < papan.length; i++) {
        System.arraycopy(original[i], srcPos:0, papan[i], destPos:0, original[i].length);
    }
}

@SuppressWarnings("resource")
public static boolean validasiInput(String fileName) throws IOException {
    bentukPiece = new HashMap<>();
    Path filePath = Paths.get(first:"../test", fileName);
    BufferedReader br = new BufferedReader(new FileReader(filePath.toFile()));
    String[] dims = br.readLine().split(regex:" ");
    if (dims.length != 3) {
        System.out.println(x:"Jumlah angka di baris pertama tidak sesuai.");
        return false;
    }
    N = Integer.parseInt(dims[0]);
    M = Integer.parseInt(dims[1]);
    P = Integer.parseInt(dims[2]);
    if (N <= 0 || M <= 0 || P <= 0) {
        System.out.println(x:"Nilai N, M, dan P tidak boleh kurang dari sama dengan 0.");
        return false;
    }
    papan = new char[N][M];
    for (char[] row : papan) Arrays.fill(row, val:'.');

    String caseType = br.readLine();
    if (!(caseType.equals(anObject:"DEFAULT") || caseType.equals(anObject:"CUSTOM") || caseType.equals(anObject:"PYRAMID"))) {
        System.out.println(x:"Jenis kasus tidak sesuai.");
        return false;
    }
    Set<Character> uniquePieces = new HashSet<>();
    String line;
    while ((line = br.readLine()) != null) {
        if (line.isEmpty()) continue;
        line = line.replaceAll(regex:" ", replacement:".");
        int l=0;
        while(line.charAt(l) == '.' || line.charAt(l) == ' '){
            l++;
        }
        char pieceChar = line.charAt(l);
        if (pieceChar != '.' && pieceChar != ' ') {
            uniquePieces.add(pieceChar);
            bentukPiece.putIfAbsent(pieceChar, value:"");
        }
        bentukPiece.put(pieceChar, bentukPiece.get(pieceChar) + (bentukPiece.get(pieceChar).isEmpty() ? "" : "\n") + line);
    }
    for (Map.Entry<Character, String> entry : bentukPiece.entrySet()) {
        char key = entry.getKey();
        String shape = entry.getValue();
        pieces.add(shape);
        banyakHuruf.put(key, banyakHuruf.getOrDefault(key, defaultValue:0) + shape.replace(target:"\n", replacement:"").replace(target:".", repl…"").length());
    }
    if (uniquePieces.size() != P){
        System.out.println(x:"Jumlah piece tidak sesuai.");
        return false;
    }
    br.close();
    return true;
}

public static void permutasi(char[][] papan, Map<Character, String> bentukPiece, Map<Character, Integer> banyakHuruf, int indeks) {
    if (banyakHuruf.isEmpty() || indeks >= banyakHuruf.size()) return;

    List<Character> hurufList = new ArrayList<>(banyakHuruf.keySet());
    char huruf = hurufList.get(indeks);
    int jumlah = banyakHuruf.get(huruf);
    Map<Character, Set<String>> bentukPieceTransformasi = new HashMap<>();
    Set<String> allForms = new HashSet<>();

    String[] shapeArray = bentukPiece.get(huruf).split(regex:"\n");
    allForms.addAll(generateTransformations(shapeArray));
    bentukPieceTransformasi.put(huruf, allForms);
```

BAB IV TESTING

NO	Test Case		Penjelasan
	Input	Output	
1.			Kasus 2x3 Sederhana
2.			Kasus 5x5 valid
3.			Kasus piece kurang
4.			Kasus jumlah piece tidak sesuai dengan P

5.			Kasus terdapat nilai NMP yang sama dengan 0
6.			Kasus 4x4 piece besar
7.			Kasus piece unik (bolong)

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	✓	