

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



**Disusun Oleh :
Alfian Hanif Fitria Yustanto 13523073**

**Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2025**

Daftar Isi

Daftar Isi.....	2
BAB I	
Deskripsi Masalah.....	4
1.1. Algoritma Brute Force.....	4
1.2 IQ Puzzler Pro.....	4
BAB II	
Algoritma.....	5
2.1. Batasan dan Aturan IQ Puzzler Pro.....	5
2.2. Algoritma Penyelesaian IQ Puzzler Pro.....	5
BAB III	
Implementasi Algoritma dalam Bahasa Java.....	8
3.1. File main.java.....	8
3.2. File IO.java.....	8
3.3. File Board.java.....	9
3.4. File Puzzle.java.....	10
BAB IV	
Source Code Program.....	12
4.1. Main.java.....	12
4.1.2. app().....	12
4.2. IO.java.....	13
4.2.1. getBoardInfo().....	13
4.2.2. getBoardType().....	13
4.2.3. getPuzzle().....	14
4.2.4. saveBoardTxt().....	15
4.2.4. saveBoardImage().....	16
4.3. Board.java.....	17
4.3.1. initiateMap().....	17
4.3.2. clearPuzzle().....	17
4.3.3. show().....	18
4.3.4. addPuzzle().....	19
4.3.5. isFull().....	20
4.3.6. isFail().....	20
4.3.7. solve().....	21
4.4. Puzzle.java.....	22
4.4.1. initiateShape().....	22
4.4.2. insertShapeValue().....	22
4.4.3. show().....	23

4.4.4. reset()	23
4.4.5. rotate90()	24
4.4.6. mirror()	24
4.5. Pranala GitHub	25
BAB IV	
Pengujian Program	26
5.1. Test case 1	26
5.2. Test case 2	27
5.3. Test case 3	29
5.4. Test case 4	30
5.5. Test case 5	32
5.6. Test case 6	34
5.7. Test case 7	36
BAB V	
Lampiran	38
BAB VI	
Referensi	39

BAB I

Deskripsi Masalah

1.1. Algoritma Brute Force

Algoritma brute force merupakan salah satu metode penyelesaian masalah yang paling sederhana. Pendekatan ini bersifat langsung (straightforward) tanpa mempertimbangkan efisiensi dalam proses penyelesaiannya.

Istilah "force" dalam brute force mengindikasikan bahwa algoritma ini lebih mengandalkan kekuatan pemrosesan dibandingkan optimasi strategi penyelesaian. Meskipun tidak selalu efisien, algoritma brute force mudah dipahami dan diimplementasikan. Namun, dalam kasus dengan ruang pencarian yang besar, pendekatan ini dapat menjadi sangat lambat dan tidak praktis dibandingkan metode yang lebih optimal seperti divide and conquer atau dynamic programming. Beberapa contoh penerapan algoritma brute force meliputi:

- **Selection Sort**
Algoritma pengurutan yang mencari elemen terkecil dan menempatkannya pada posisi yang sesuai.
- **Bubble Sort**
Algoritma pengurutan yang membandingkan dan menukar elemen berdekatan secara berulang hingga data terurut.
- **Sequential Search**
Metode pencarian data dengan memeriksa setiap elemen dalam struktur data secara berurutan hingga menemukan nilai yang dicari.
- **Brute Force String Matching**
Algoritma pencocokan string dengan membandingkan setiap karakter dalam teks dengan pola secara langsung.

1.2 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 dari permainan IQ Puzzler Pro terdiri dari:

- **Board (Papan)**
Board merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area papan menggunakan blok-blok yang telah disediakan.
- **Blok Puzzle/ Puzzle 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.

BAB II

Algoritma

2.1. Batasan dan Aturan IQ Puzzler Pro

IQ puzzler Pro memiliki beberapa aturan dan batasan untuk dapat kemudian diselesaikan. Batasan terdiri dari cara penempatan puzzle yang diperbolehkan dan aturan kondisi selesai puzzle. Berikut batasan dan aturan yang terdapat pada IQ Puzzler Pro ini,

- IQ Puzzler Pro dikatakan selesai jika board telah sepenuhnya terisi dan seluruh puzzle telah digunakan.
- Tidak diperbolehkan meletakkan puzzle di luar/melewati papan.
- Tidak diperbolehkan meletakkan puzzle secara bertumpuk dalam kondisi tipe board “DEFAULT”.
- Diperbolehkan melakukan rotasi dan pencerminan terhadap puzzle.

2.2. Algoritma Penyelesaian IQ Puzzler Pro

Penyelesaian permainan IQ Puzzler Pro dapat dilakukan melalui pendekatan brute force dengan mencoba seluruh kemungkinan susunan puzzle yang diperbolehkan. Untuk lebih detail, berikut langkah-langkah penyelesaiannya,

1. Siapkan konfigurasi puzzle dan board yang akan digunakan. Konfigurasi yang disiapkan berupa ukuran board $M \times N$, jumlah puzzle P , dan bentuk untuk setiap puzzle.
2. Cari slot kosong pertama pada papan untuk dijadikan acuan. Proses ini dilakukan dengan melakukan iterasi pada setiap slot pada papan mulai dari titik (0,0) sampai ditemukan slot kosong pada papan. Slot kosong ini akan menjadi acuan untuk meletakkan puzzle.
3. Pasang puzzle pada acuan. Puzzle yang dipasang pada board harus memenuhi aturan permainan seperti puzzle belum digunakan sebelumnya, puzzle tidak bertumpuk dengan puzzle lain, dan puzzle muat pada papan. Jika puzzle tidak dapat dipasang, ubah orientasi puzzle, hal ini dapat dilakukan dengan melakukan rotasi sebesar 90 derajat dan melakukan pencerminan pada puzzle sampai seluruh orientasi yang mungkin telah dicoba. Jika puzzle masih gagal dipasang setelah mencoba seluruh orientasi, maka ulangi langkah ini dengan puzzle lain yang belum digunakan.
4. Jika tidak ada puzzle yang dapat dipasang pada acuan, maka lakukan *backtrack*. *Backtrack* dapat dilakukan dengan cara mengambil puzzle terakhir yang telah dipasang pada board kemudian mencoba memasang kembali puzzle tersebut dengan orientasi berbeda. Jika puzzle tidak dapat dipasang setelah mencoba seluruh orientasi, maka ganti puzzle. Jika tidak ada puzzle yang memenuhi, lakukan langkah ini kembali.
5. Lakukan kembali langkah pada poin ke-2 sampai permainan selesai. Permainan selesai ketika papan penuh dan seluruh puzzle digunakan (sukses) atau papan tidak penuh dan seluruh kemungkinan susunan puzzle yang diperbolehkan sudah diperiksa (gagal).

Algoritma penyelesaian dapat dituliskan dalam pseudocode berikut,

```

PROCEDURE solve(input puzzles: array of Puzzle)
  KAMUS LOKAL
  id ← 0
  placedPuzzle ← array of size P initialized to 0
  status ← false
  fail ← false
  count ← 0

  ALGORITMA
  DO
    status ← false

    // Coba menempatkan puzzle pada slot kosong pertama
    i traversal [0 ... length of puzzles - 1]
      IF puzzles[i] is not used THEN
        WHILE NOT addPuzzle(puzzles[i]) AND NOT status DO
          count ← count + 1

          // Ubah orientasi puzzle jika gagal
          IF puzzles[i].rotateAmount = 4 THEN
            puzzles[i].mirror()
          ELSE IF puzzles[i].rotateAmount < 8 THEN
            puzzles[i].rotate90()
          ELSE
            puzzles[i].reset()
            BREAK

        // Jika puzzle berhasil ditempatkan, hentikan pencarian
        IF puzzles[i] is used THEN
          status ← true
          placedPuzzle[id] ← i
          id ← id + 1
          BREAK

      ELSE
        count ← count + 1

    // Backtracking jika gagal meletakkan puzzle
    WHILE NOT status DO
      IF id > 0 THEN
        id ← id - 1

        tempId ← placedPuzzle[id]
        count ← count + 1
        clearPuzzle(puzzles[tempId].name)
        puzzles[tempId].used ← false
        newPlacementFound ← false

        // Coba orientasi lain sebelum mengganti puzzle
        WHILE NOT newPlacementFound DO
          count ← count + 1
          IF puzzles[tempId].rotateAmount = 4 THEN
            puzzles[tempId].mirror()
          ELSE IF puzzles[tempId].rotateAmount < 8 THEN
            puzzles[tempId].rotate90()

```

```

        ELSE
            puzzles[tempId].reset()

            // Kondisi fail
            IF id = 0 AND tempId = P - 1 THEN
                fail ← true
            BREAK

        IF addPuzzle(puzzles[tempId]) THEN
            placedPuzzle[id] ← tempId
            id ← id + 1
            newPlacementFound ← true
            status ← true
            BREAK

    // Jika semua orientasi gagal, coba puzzle berikutnya
    IF NOT newPlacementFound THEN
        newId TRAVERSAL [tempId + 1 ... length of puzzles - 1]
        IF NOT puzzles[newId].used THEN
            WHILE NOT addPuzzle(puzzles[newId]) AND NOT
newPlacementFound DO
                count ← count + 1
                IF puzzles[newId].rotateAmount = 4 THEN
                    puzzles[newId].mirror()
                ELSE IF puzzles[newId].rotateAmount < 8 THEN
                    puzzles[newId].rotate90()
                ELSE
                    puzzles[newId].reset()
                    BREAK

            IF puzzles[newId].used THEN
                placedPuzzle[id] ← newId
                id ← id + 1
                newPlacementFound ← true
                status ← true
                BREAK

    WHILE NOT isFull() AND NOT fail
        ShowBoard()

```

BAB III

Implementasi Algoritma dalam Bahasa Java

3.1. File main.java

File ini berisi class main program dari aplikasi ini, berfungsi untuk sebagai antarmuka utama saat eksekusi program.

Method	Deskripsi
app()	Berfungsi sebagai penghubung antara objek dan menjadi antarmuka utama program.

3.2. File IO.java

File ini berisi class IO yang berfungsi untuk melakukan load konfigurasi file .txt dan menyimpan hasil program.

Atribut	Type Data	Deskripsi
fileName	String	Atribut untuk menyimpan nama file.
fileScanner	Scanner	Atribut scanner untuk membaca file.
inputScanner	Scanner	Atribut scanner untuk membaca input CLI.

Metode	Type	Deskripsi
getFileName()	public String	Mengembalikan nama file yang digunakan.
openFile()	public void	Membuka file yang ditentukan oleh fileName.
closeFile()	public void	Menutup scanner file setelah digunakan.
readFileName()	public static String	Membaca nama file dari input CLI dan memastikan file ada di direktori test/input/.
getBoardInfo()	public int[]	Membaca informasi konfigurasi board dari file.
getBoardType()	public String	Mengembalikan tipe board berdasarkan isi file.

getPuzzle(int, int, int)	public Puzzle[]	Membaca dan mengembalikan daftar puzzle dari file berdasarkan jumlah dan ukuran.
saveBoardTxt(Board)	public void	Menyimpan representasi teks board ke dalam file .txt.
saveBoardImage(Board, long, long, long)	public void	Menyimpan representasi visual board sebagai gambar .png.

3.3. File Board.java

File ini merupakan file yang digunakan untuk menyimpan kelas Board. Kelas board merupakan implementasi dari board pada permainan.

Atribut	Tipe Data	Deskripsi
map	char[][]	Matriks 2D yang merepresentasikan papan permainan dengan karakter sebagai elemen.
M	int	Jumlah baris papan permainan.
N	int	Jumlah kolom papan permainan.
P	int	Jumlah total puzzle yang tersedia.
type	String	Jenis papan permainan.

Metode	Tipe	Deskripsi
Board(int, int, int, String)	Konstruktor	Menginisialisasi papan permainan dengan ukuran M x N, jumlah puzzle P, dan tipe papan type.
initiateMap()	void	Mengisi seluruh papan dengan karakter % sebagai tanda kosong.
clearPuzzle(char)	void	Menghapus puzzle tertentu dari papan dengan mengganti karakter yang sesuai menjadi %.
show()	void	Menampilkan papan permainan di terminal dengan warna berbeda untuk setiap huruf puzzle.
findRefCor()	int[]	Mencari koordinat pertama yang kosong (%) pada papan, mengembalikan array [baris, kolom].

addPuzzle(Puzzle)	Boolean	Menambahkan objek Puzzle ke papan dengan mencari posisi referensi yang sesuai. Mengembalikan true jika berhasil, false jika gagal.
isFull()	Boolean	Memeriksa apakah papan sudah penuh dengan puzzle.
isFail(Puzzle[])	boolean	Memeriksa apakah permainan gagal dengan melihat apakah semua puzzle sudah digunakan tetapi papan tidak penuh.
solve(Puzzle[])	long[]	Merupakan inti dari penyelesaian permainan. Menjalankan algoritma brute force untuk menyusun puzzle pada papan. Mengembalikan array [jumlah percobaan, durasi (ms), status (0 = gagal, 1 = sukses)]. Perhitungan percobaan dilakukan pada setiap pemanggilan fungsi addPuzzle dan kondisi perbandingan lain.

3.4. File Puzzle.java

File ini merupakan file yang digunakan untuk menyimpan kelas Puzzle. Kelas Puzzle merupakan implementasi dari puzzle pada permainan.

Atribut	Tipe Data	Deskripsi
used	boolean	Menunjukkan apakah puzzle sudah digunakan atau belum.
row	int	Jumlah baris pada puzzle.
col	int	Jumlah kolom pada puzzle.
shape	char[][]	Representasi bentuk puzzle dalam bentuk array 2D karakter.
originShape	char[][]	Bentuk asli puzzle sebelum transformasi.
originCol	int	Jumlah kolom asli sebelum transformasi.
originRow	int	Jumlah baris asli sebelum transformasi.
name	char	Nama atau identifikasi puzzle.
rotateAmount	int	Jumlah rotasi yang telah dilakukan pada puzzle.

used	boolean	Menunjukkan apakah puzzle sudah digunakan atau belum.
------	---------	---

Metode	Tipe	Deskripsi
Puzzle()	-	Konstruktor untuk menginisialisasi puzzle dengan ukuran tertentu.
initiateShape()	void	Mengisi seluruh bentuk puzzle dengan karakter % sebagai tanda kosong.
insertShapeValue()	void	Memasukkan nilai ke dalam shape berdasarkan string yang diberikan.
show()	void	Menampilkan bentuk puzzle ke terminal.
compress()	void	Menghapus ruang kosong di bagian bawah dan kanan puzzle untuk mendapatkan ukuran sebenarnya.
reset()	void	Mengembalikan puzzle ke bentuk aslinya sebelum transformasi dilakukan.
isOrigin()	Boolean	Memeriksa apakah bentuk puzzle saat ini sesuai dengan bentuk aslinya (originShape).
transpose()	Puzzle	Membalikkan baris dan kolom puzzle, menukar posisi elemen ($\text{shape}[i][j] \rightarrow \text{shape}[j][i]$).
swapCol()	void	Menukar dua kolom pada bentuk puzzle.
rotate90()	void	Memutar puzzle sebesar 90 derajat searah jarum jam.
mirror()	void	Mencerminkan puzzle terhadap sumbu vertikal.
isMirrorOrigin()	boolean	Memeriksa apakah hasil pencerminan puzzle sama dengan bentuk aslinya.

BAB IV

Source Code Program

4.1. Main.java

4.1.2. app()

```
1 public static void app() {
2     String fileName;
3     int[] boardInfo;
4     String boardType;
5     Puzzle[] puzzleList;
6     boolean run = true;
7
8     while (run) {
9         System.out.println("\u001B[34m[INFO]\u001B[0m" + " : Masukkan konfigurasi puzzle dan papan !");
10        System.out.println("\u001B[33m[WARNING]\u001B[0m" + " : Pastikan Input memiliki .txt (XXX.txt)...");
11
12        fileName = IO.readFileName();
13        System.out.println();
14        IO io = new IO(fileName);
15        boardInfo = io.getBoardInfo();
16        boardType = io.getBoardType();
17
18        Board board = new Board(boardInfo[0], boardInfo[1], boardInfo[2], boardType);
19        board.initiateMap();
20        puzzleList = new Puzzle[boardInfo[2]];
21        puzzleList = io.getPuzzle(boardInfo[2], boardInfo[0], boardInfo[1]);
22        long[] stats;
23        System.out.println();
24        System.out.printf("\u001B[34m[INFO]\u001B[0m" + " : Informasi board...\n\n" +
25            "
26            +-----+
27            |           |
28            | M       | %s | \n" +
29            | N       | %s | \n" +
30            | P       | %s | \n" +
31            | Tipe    | %s | \n" +
32            +-----+
33            boardInfo[0], boardInfo[1],
34            boardInfo[2], boardType);
35        System.out.printf("\u001B[34m[INFO]\u001B[0m" + " : Hasil Penyelesaian board...\n\n");
36        stats = board.solve(puzzleList);
37
38        boolean validOpt = false;
39        System.out.printf("\n\u001B[34m[INFO]\u001B[0m" + " : Simpan hasil ? (y/n)\n");
40        do {
41            System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
42            String option = inputScanner.nextLine();
43
44            if (option.equals("y")) {
45                System.out.printf("\n\u001B[34m[INFO]\u001B[0m" +
46                    " : Simpan dalam bentuk ? (1/2/3)\n [1] .txt\n [2] .png\n [3] .txt dan .png\n");
47                do {
48                    System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
49                    option = inputScanner.nextLine();
50
51                    if (option.equals("1")) {
52                        io.saveBoardTxt(board, stats[0], stats[1], stats[2]);
53                        validOpt = true;
54                    } else if (option.equals("2")) {
55                        io.saveBoardImage(board, stats[0], stats[1], stats[2]);
56                        validOpt = true;
57                    } else if (option.equals("3")) {
58                        io.saveBoardTxt(board, stats[0], stats[1], stats[2]);
59                        io.saveBoardImage(board, stats[0], stats[1], stats[2]);
60                        validOpt = true;
61                    } else {
62                        validOpt = false;
63                        System.out.println("\u001B[33m[WARNING]\u001B[0m" + " : Pastikan Input benar ! (1/2/3)...");
64                    }
65                } while (!validOpt);
66            } else if (option.equals("n")) {
67                validOpt = true;
68            } else {
69                validOpt = false;
70                System.out.println("\u001B[33m[WARNING]\u001B[0m" + " : Pastikan Input benar ! (y/n)...");
71            }
72        } while (!validOpt);
73        System.out.println("\n\u001B[34m[INFO]\u001B[0m" + " : Keluar program ? (y/n)");
74        validOpt = false;
75        do {
76            System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
77            String option = inputScanner.nextLine();
78            if (option.equals("y")) {
79                validOpt = true;
80                run = false;
81            } else if (option.equals("n")) {
82                validOpt = true;
83            } else {
84                validOpt = false;
85                System.out.println("\u001B[33m[WARNING]\u001B[0m" + " : Pastikan Input benar ! (y/n)...");
86            }
87        } while (!validOpt);
88    }
89
90    System.out.println("\n\u001B[34m[INFO]\u001B[0m" + " : dadah !");
91
92 }
```

4.2. IO.java

4.2.1. getBoardInfo()

```
1 public int[] getBoardInfo() {
2     int[] info = new int[3];
3     boolean stat = true;
4     info[0] = -99999;
5     info[1] = -99999;
6     info[2] = -99999;
7     openFile();
8     info[0] = fileScanner.nextInt();
9     info[1] = fileScanner.nextInt();
10    info[2] = fileScanner.nextInt();
11    closeFile();
12    for (int i = 0; i < 3; i++) {
13        if (info[i] < 0) {
14            stat = false;
15        }
16    }
17    if (stat) {
18        System.out.println("\u001B[32m[SUCCESS]\u001B[0m"
19            + " : Konfigurasi (M/N/P) pada board valid...");
20    } else {
21        System.out.println("\u001B[31m[ERROR]\u001B[0m" + " : Konfigurasi (M/N/P) pada board tidak valid...");
22    }
23    return info;
24 }
```

4.2.2. getBoardType()

```
1 public String getBoardType() {
2     String type;
3     openFile();
4     fileScanner.nextLine();
5     type = fileScanner.next();
6     if (type.equals("DEFAULT")) {
7         System.out.println("\u001B[32m[SUCCESS]\u001B[0m"
8             + " : Konfigurasi tipe board valid...");
9     } else {
10        System.out.println("\u001B[31m[ERROR]\u001B[0m" + " : Konfigurasi tipe board tidak tersedia...");
11    }
12    return type;
13 }
```

4.2.3. getPuzzle()

```
1 public Puzzle[] getPuzzle(int amount, int M, int N) {
2     Puzzle[] resultPuzzles = new Puzzle[amount];
3     openFile();
4     fileScanner.nextLine(); // Skip baris pertama (info board)
5     fileScanner.nextLine(); // Skip baris kedua (tipe board)
6
7     int size = Math.max(M * N, M * N);
8
9     String line = null;
10
11     for (int i = 0; i < amount; i++) {
12         if (line == null) {
13             if (!fileScanner.hasNextLine()) {
14                 System.out.println("\u001B[31m[ERROR]\u001B[0m" + " : Konfigurasi Puzzle tidak valid...");
15                 break;
16             }
17             line = fileScanner.nextLine();
18         }
19
20         char currentCharacter = line.trim().charAt(0);
21         Puzzle currentPuzzle = new Puzzle(size, size);
22         currentPuzzle.initiateShape();
23         currentPuzzle.name = currentCharacter;
24
25         int row = 0;
26         do {
27             currentPuzzle.insertShapeValue(line, size, row);
28             row++;
29
30             if (row > size) {
31                 System.out.println("\u001B[31m[ERROR]\u001B[0m" + " : Konfigurasi Puzzle tidak valid...");
32                 break;
33             }
34
35             if (fileScanner.hasNextLine()) {
36                 String nextLine = fileScanner.nextLine();
37                 String nextLineCC = nextLine.trim();
38                 if (!nextLine.isEmpty() && nextLineCC.charAt(0) == currentCharacter) {
39                     line = nextLine;
40                 } else {
41                     line = nextLine;
42                     break;
43                 }
44             } else {
45                 line = null;
46                 break;
47             }
48         } while (true);
49
50         currentPuzzle.compress();
51         currentPuzzle.originShape = currentPuzzle.shape;
52         resultPuzzles[i] = currentPuzzle;
53     }
54
55     closeFile();
56     System.out.println("\u001B[32m[SUCCESS]\u001B[0m"
57         + " : Konfigurasi Puzzle selesai...");
58     return resultPuzzles;
59 }
```

4.2.4. saveBoardTxt()


```
1 public void saveBoardTxt(Board board, long attempt, long duration, long status) {
2     Scanner scanner = new Scanner(System.in);
3     String directory = "test/output/txt";
4     File folder = new File(directory);
5
6     if (!folder.exists()) {
7         folder.mkdirs();
8     }
9
10    String filename;
11    File file;
12
13    while (true) {
14        System.out.println("\n\u001B[34m[INFO]\u001B[0m" + " : Masukkan nama file !");
15        System.out.println("\u001B[33m[WARNING]\u001B[0m" + " : Pastikan menambahkan .txt (XXX.txt) !");
16        System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
17        filename = scanner.nextLine();
18        file = new File(directory, filename);
19
20        if (!file.exists()) {
21            break;
22        } else {
23            System.out.println("\u001B[31m[ERROR]\u001B[0m" + " : File sudah tersedia ! ubah nama file !...");
24        }
25    }
26
27    try (FileWriter writer = new FileWriter(file)) {
28        for (int i = 0; i < board.M; i++) {
29            for (int j = 0; j < board.N; j++) {
30                writer.write(board.map[i][j]);
31                writer.write(" ");
32            }
33            writer.write("\n");
34        }
35        writer.write("\n");
36        if (status == 1) {
37            writer.write("Status : Sukses\n");
38        } else {
39            writer.write("Status : Gagal\n");
40        }
41        writer.write("Percobaan : ");
42        writer.write(String.valueOf(attempt));
43        writer.write("\n");
44        writer.write("Durasi : ");
45        writer.write(String.valueOf(duration));
46        writer.write("ms");
47        writer.write("\n");
48
49        System.out.println("\u001B[32m[SUCCESS]\u001B[0m"
50            + " : File disimpan pada test/output/txt/" + filename);
51    } catch (IOException e) {
52        e.printStackTrace();
53    }
54
55 }
```

4.2.4. saveBoardImage()

```
1 public void saveBoardImage(Board board, long attempt, long duration, long status) {
2     char[][] map = board.map;
3     Scanner scanner = new Scanner(System.in);
4     int cellSize = 50;
5     int padding = 50;
6     int textHeight = 100;
7
8     int width = (board.N * cellSize) + (2 * padding);
9     int height = (board.M * cellSize) + (2 * padding) + textHeight;
10
11     BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
12     Graphics2D g2d = image.createGraphics();
13
14     g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
15
16     g2d.setColor(Color.WHITE);
17     g2d.fillRect(0, 0, width, height);
18
19     Color[] colors = {
20         Color.RED, Color.GREEN, Color.YELLOW, Color.BLUE, Color.MAGENTA, Color.CYAN,
21         Color.PINK, Color.ORANGE, Color.LIGHT_GRAY, Color.DARK_GRAY, Color.GRAY, Color.WHITE,
22         new Color(128, 0, 0), new Color(0, 128, 0), new Color(0, 0, 128), new Color(128, 128, 0),
23         new Color(128, 0, 128), new Color(0, 128, 128), new Color(192, 192, 192), new Color(255, 165, 0)
24     };
25
26     for (int i = 0; i < board.M; i++) {
27         for (int j = 0; j < board.N; j++) {
28             char ch = map[i][j];
29
30             if (ch >= 'A' && ch <= 'Z') {
31                 g2d.setColor(colors[(ch - 'A') % colors.length]);
32             } else {
33                 g2d.setColor(Color.WHITE);
34             }
35
36             int x = padding + (j * cellSize);
37             int y = padding + (i * cellSize);
38
39             g2d.fillRect(x, y, cellSize, cellSize);
40
41             g2d.setColor(Color.BLACK);
42             g2d.drawRect(x, y, cellSize, cellSize);
43
44             g2d.setColor(Color.BLACK);
45             g2d.setFont(new Font("Arial", Font.BOLD, 30));
46             g2d.drawString(String.valueOf(ch), x + 15, y + 35);
47         }
48     }
49
50     // font
51     g2d.setColor(Color.BLACK);
52     g2d.setFont(new Font("Arial", Font.BOLD, 20));
53     FontMetrics fm = g2d.getFontMetrics();
54
55     int lineSpacing = 25; // Jarak antar teks
56     int baseY = height - textHeight + (fm.getAscent() / 2) + 10; // Awal posisi teks
57
58     // teks Status
59     String statText = (status == 1) ? "Status: Sukses " : "Status: Gagal";
60     int textWidth = fm.stringWidth(statText);
61     int textX = (width - textWidth) / 2;
62     g2d.drawString(statText, textX, baseY);
63
64     // teks Percobaan
65     String attemptText = "Percobaan: " + attempt;
66     textWidth = fm.stringWidth(attemptText);
67     textX = (width - textWidth) / 2;
68     g2d.drawString(attemptText, textX, baseY + lineSpacing);
69
70     // teks Durasi
71     String durationText = "Durasi: " + duration + "ms";
72     textWidth = fm.stringWidth(durationText);
73     textX = (width - textWidth) / 2;
74     g2d.drawString(durationText, textX, baseY + (2 * lineSpacing));
75
76     g2d.dispose();
77
78     // validasi
79     File directory = new File("test/output/image");
80     if (!directory.exists()) {
81         directory.mkdirs();
82     }
83
84     // Save as PNG
85     File outputFile;
86     while (true) {
87         System.out.println("\u001B[34m[INFO]\u001B[0m" + " : Masukkan nama file !");
88         System.out.println("\u001B[33m[WARNING]\u001B[0m" + " : Pastikan menambahkan .png (XXX.png) !");
89         System.out.print("\u001B[38;5;214m[INPUT]\u001B[0m" + " : ");
90         String filename = scanner.nextLine();
91         outputFile = new File(directory, filename);
92
93         if (!outputFile.exists()) {
94             break;
95         } else {
96             System.out.println("\u001B[31m[ERROR]\u001B[0m" + " : File sudah tersedia ! ubah nama file !...");
97         }
98     }
99     try {
100         ImageIO.write(image, "png", outputFile);
101         System.out.println("\u001B[32m[SUCCESS]\u001B[0m"
102             + " : File disimpan pada " + outputFile);
103     } catch (IOException e) {
104         e.printStackTrace();
105     }
106 }
107 }
```



4.3. Board.java

4.3.1. initiateMap()



```
1  public void initiateMap() {
2      int i;
3      int j;
4      for (i = 0; i < this.M; i++) {
5          for (j = 0; j < this.N; j++) {
6              this.map[i][j] = '%';
7          }
8      }
9  }
```

4.3.2. clearPuzzle()



```
1  public void clearPuzzle(char puzzleName) {
2      int i;
3      int j;
4      for (i = 0; i < this.M; i++) {
5          for (j = 0; j < this.N; j++) {
6              if (this.map[i][j] == puzzleName) {
7                  this.map[i][j] = '%';
8              }
9          }
10     }
11 }
```


4.3.3. show()

```
1 public void show() {
2     String[] colors = {
3         "\u001B[31m", "\u001B[32m", "\u001B[33m", "\u001B[34m", "\u001B[35m", "\u001B[36m",
4         "\u001B[91m", "\u001B[92m", "\u001B[93m", "\u001B[94m", "\u001B[95m", "\u001B[96m",
5         "\u001B[97m", "\u001B[90m", "\u001B[37m", "\u001B[31m", "\u001B[32m", "\u001B[33m",
6         "\u001B[34m", "\u001B[35m", "\u001B[36m", "\u001B[91m", "\u001B[92m", "\u001B[93m",
7         "\u001B[94m", "\u001B[95m"
8     };
9
10    String reset = "\u001B[0m";
11
12    for (int i = 0; i < this.M; i++) {
13        for (int j = 0; j < this.N; j++) {
14            char ch = this.map[i][j];
15            if (ch >= 'A' && ch <= 'Z') {
16                int index = ch - 'A';
17                System.out.printf("%s%s%s ", colors[index], ch, reset);
18            } else {
19                System.out.printf("%s ", ch);
20            }
21        }
22        System.out.println();
23    }
24 }
```

4.3.4. addPuzzle()


```
1 public Boolean addPuzzle(Puzzle puzzle) {
2     int[] refCor = findRefCor();
3     int x = refCor[1];
4     int y = refCor[0];
5
6     if (puzzle.used) {
7         return false;
8     }
9
10    if (x == -1 && y == -1) {
11        return false;
12    }
13
14    int i;
15    int j;
16
17    // find offset on puzzle shape
18    int offset = 0;
19    for (j = 0; j < puzzle.col; j++) {
20        if (puzzle.shape[0][j] == '%') {
21            offset++;
22        } else {
23            break;
24        }
25    }
26
27    x -= offset;
28    if (x < 0) {
29        return false;
30    }
31
32    if (puzzle.row + y > this.M) {
33        return false;
34    }
35
36    if (puzzle.col + x > this.N) {
37        return false;
38    }
39
40    for (i = y; (i - y) < puzzle.row; i++) {
41        for (j = x; (j - x) < puzzle.col; j++) {
42            if (this.map[i][j] != '%' && puzzle.shape[i - y][j - x] != '%') {
43                clearPuzzle(puzzle.name);
44                return false;
45            }
46            if (puzzle.shape[i - y][j - x] != '%') {
47                this.map[i][j] = puzzle.shape[i - y][j - x];
48            }
49        }
50    }
51    puzzle.used = true;
52    return true;
53 }
```

4.3.5. isFull()



```
1  public Boolean isFull() {
2      int i;
3      int j;
4
5      for (i = 0; i < this.M; i++) {
6          for (j = 0; j < this.N; j++) {
7              if (this.map[i][j] == '%') {
8                  return false;
9              }
10         }
11     }
12     return true;
13 }
```

4.3.6. isFail()



```
1  public boolean isFail(Puzzle[] puzzles) {
2      boolean allUsed = true;
3      for (int i = 0; i < this.P; i++) {
4          if (puzzles[i].used == false) {
5              allUsed = false;
6          }
7      }
8      if (isFull() && allUsed) {
9          return false;
10     } else {
11         return true;
12     }
13 }
```

4.3.7. solve()


```

1 public long[] solve(Puzzle[] puzzles) {
2     long startTime = System.currentTimeMillis();
3     int id = 0;
4     int[] placedPuzzle = new int[this.P];
5     boolean status;
6     boolean fail = false;
7     int count = 0;
8
9     do {
10         status = false;
11
12         // coba tambah puzzle
13         for (int i = 0; i < puzzles.length; i++) {
14             if (!puzzles[i].used) {
15                 while (!addPuzzle(puzzles[i]) && !status) {
16                     count++;
17
18                     // ubah orientasi klo gagal
19                     if (puzzles[i].rotateAmount == 4) {
20                         puzzles[i].mirror();
21                     } else if (puzzles[i].rotateAmount < 8) {
22                         puzzles[i].rotate90();
23                     } else {
24                         puzzles[i].reset();
25                         break;
26                     }
27                 }
28                 // kalo puzzle berhasil stop nyari
29                 if (puzzles[i].used) {
30                     status = true;
31                     placedPuzzle[id++] = i;
32                     break;
33                 }
34             } else {
35                 count++;
36             }
37         }
38
39         // backtrack kalo gagal menambah puzzle
40         while (!status) {
41             if (id > 0) {
42                 id--;
43             }
44             // hapus puzzle terakhir
45             int tempId = placedPuzzle[id];
46             count++;
47             clearPuzzle(puzzles[tempId].name);
48             puzzles[tempId].used = false;
49
50             boolean newPlacementFound = false;
51
52             // Coba orientasi lain sebelum ganti puzzle
53             while (!newPlacementFound) {
54                 count++;
55                 if (puzzles[tempId].rotateAmount == 4) {
56                     puzzles[tempId].mirror();
57                 } else if (puzzles[tempId].rotateAmount < 8) {
58                     puzzles[tempId].rotate90();
59                 } else {
60                     puzzles[tempId].reset();
61                     if (id == 0 && tempId == this.P - 1) {
62                         // here cok
63                         fail = true;
64                     }
65                     break;
66                 }
67             }
68
69             if (addPuzzle(puzzles[tempId])) {
70                 placedPuzzle[id++] = tempId;
71                 newPlacementFound = true;
72                 status = true;
73                 break;
74             }
75
76             // Kalo semua orientasi gagal, coba puzzle berikutnya
77             if (!newPlacementFound) {
78                 for (int newId = tempId + 1; newId < puzzles.length; newId++) {
79                     if (!puzzles[newId].used) {
80                         while (!addPuzzle(puzzles[newId]) && !newPlacementFound) {
81                             count++;
82                             if (puzzles[newId].rotateAmount == 4) {
83                                 puzzles[newId].mirror();
84                             } else if (puzzles[newId].rotateAmount < 8) {
85                                 puzzles[newId].rotate90();
86                             } else {
87                                 puzzles[newId].reset();
88                                 break;
89                             }
90                         }
91
92                         if (puzzles[newId].used) {
93                             placedPuzzle[id++] = newId;
94                             newPlacementFound = true;
95                             status = true;
96                             break;
97                         }
98                     }
99                 }
100             }
101         }
102     } while (!status && !fail);
103
104     long endTime = System.currentTimeMillis();
105     long duration = endTime - startTime;
106
107     if (fail || !isFail(puzzles)) {
108         System.out.printf(
109             "%n+-----+\\n" + "Status : Gagal\\n" + "Percobaan : %s\\n" + "Durasi : %sms\\n"
110             + "-----+\\n",
111             count, duration);
112     } else {
113         show();
114         System.out.printf(
115             "%n+-----+\\n" + "Status : Sukses\\n" + "Percobaan : %s\\n"
116             + "-----+\\n" + "Durasi : %sms\\n"
117             + "-----+\\n",
118             count, duration);
119     }
120
121     long[] res = new long[3];
122     res[0] = count;
123     res[1] = duration;
124     if (fail || !isFail(puzzles)) {
125         res[2] = 0;
126     } else {
127         res[2] = 1;
128     }
129
130     return res;
131 }

```


4.4. Puzzle.java

4.4.1. initiateShape()




```
1  public void initiateShape() {
2      for (int i = 0; i < this.row; i++) {
3          for (int j = 0; j < this.col; j++) {
4              this.shape[i][j] = '%';
5          }
6      }
7  }
```

4.4.2. insertShapeValue()




```
1  public void insertShapeValue(String line, int size, int row) {
2      int i;
3      for (i = 0; i < line.length(); i++) {
4          if (line.charAt(i) != ' ') {
5              this.shape[row][i] = line.charAt(i);
6          } else {
7              this.shape[row][i] = '%';
8          }
9      }
10 }
```

4.4.3. show()



```
1 public void show() {
2     for (int i = 0; i < this.row; i++) {
3         for (int j = 0; j < this.col; j++) {
4             System.out.printf("%s", this.shape[i][j]);
5         }
6         System.err.println();
7     }
8 }
9
```

4.4.4. reset()



```
1 public void reset() {
2     this.col = this.originCol;
3     this.row = this.originRow;
4     this.shape = this.originShape;
5     this.rotateAmount = 0;
6     this.used = false;
7 }
```

4.4.5. rotate90()

```
1 public void rotate90() {
2     Puzzle M = new Puzzle(this.row, this.col);
3     M.shape = this.shape;
4
5     Puzzle R = M.transpose();
6     int m = R.col;
7
8     for (int i = 0; i <= (m / 2) - 1; i++) {
9         R.swapCol(i, m - i - 1);
10    }
11    this.row = R.row;
12    this.col = R.col;
13    this.shape = R.shape;
14    this.rotateAmount += 1;
15 }
```

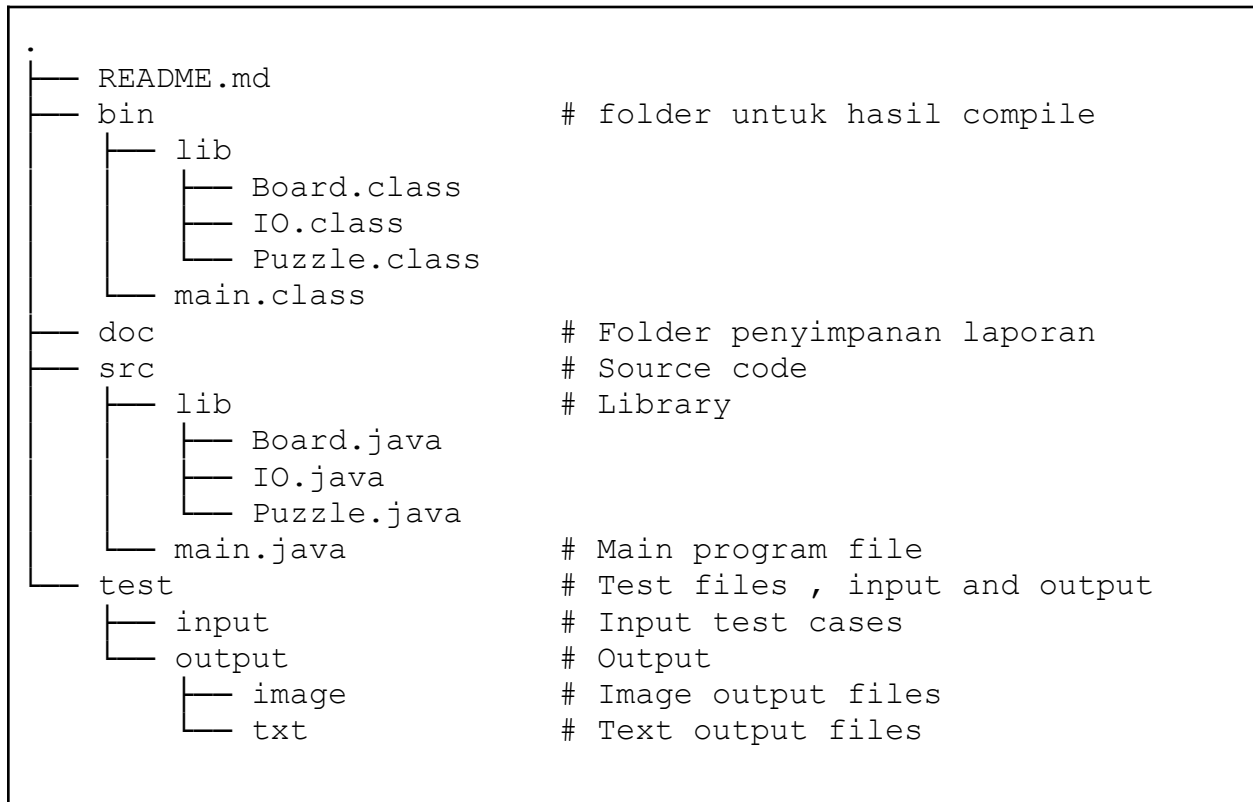
4.4.6. mirror()

```
1 public void mirror() {
2     int i;
3     int j;
4     char[][] temp = new char[this.row][this.col];
5     for (i = 0; i < this.row; i++) {
6         for (j = 0; j < this.col; j++) {
7             temp[i][j] = this.shape[i][this.col - 1 - j];
8         }
9         for (j = 0; j < this.col; j++) {
10            this.shape[i][j] = temp[i][j];
11        }
12    }
13    this.rotateAmount += 1;
14 }
```


4.5. Pranala GitHub

https://github.com/AlfianHanifFY/Tucil1_13523073.git

4.6. Struktur Program



- README.md berisi informasi umum program.
- Folder bin berisi hasil *compile* program.
- Folder src berisi *source code* program.
- Algoritma brute force untuk penyelesaian puzzle terdapat pada src/lib/board.java pada method solve().
- Folder test berisi file input dan output program.
- Input berupa file txt terletak pada test/input.
- Output berupa file txt terletak pada test/output/txt.
- Output berupa file .png terletak pada test/output/image.

BAB IV

Pengujian Program

5.1. Test case 1

Masukan (.txt)
5 5 7 DEFAULT A AA B BB C CC D DD EE EE E FF FF F GGG
Luaran (.txt dan CLI)
 <pre>[INFO] : Informasi board... +-----+ M : 5 N : 5 P : 7 Tipe : DEFAULT +-----+ [INFO] : Hasil Penyelesaian board... A B B C C A A B C D E E E D D E E F F F G G G F F +-----+ Status : Sukses Percobaan : 460 Durasi : 1ms +-----+</pre>
A B B C C A A B C D E E E D D E E F F F

G G G F F

Status : Sukses
Percobaan : 460
Durasi : 1ms

Luaran (.png)

A	B	B	C	C
A	A	B	C	D
E	E	E	D	D
E	E	F	F	F
G	G	G	F	F

Status: Sukses
Percobaan: 460
Durasi: 1ms

5.2. Test case 2

Masukan (.txt)

3 3 1
DEFAULT
BBB
B B
BBB

Luaran (.txt dan CLI)

```
[INFO] : Informasi board...
```

```
+-----+  
| M      : 3  
| N      : 3  
| P      : 1  
| Tipe   : DEFAULT  
+-----+
```

```
[INFO] : Hasil Penyelesaian board...
```

```
+-----+  
Status   : Gagal  
Percobaan : 29  
Durasi   : 0  
+-----+
```

B B B

B % B

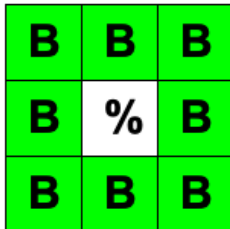
B B B

Status : Gagal

Percobaan : 29

Durasi : 0ms

Luaran (.png)



Status: Gagal

Percobaan: 29

Durasi: 0ms

Penjelasan

Kasus gagal dikarenakan puzzle yang kurang saat dimasukkan kedalam board.
File .png dan .txt menunjukkan pengecekan terakhir puzzle.

5.3. Test case 3

Masukan (.txt)
3 3 3 DEFAULT AA A BBB CC C
Luaran (.txt dan CLI)
 <pre>[INFO] : Informasi board... +-----+ M : 3 N : 3 P : 3 Tipe : DEFAULT +-----+ [INFO] : Hasil Penyelesaian board... A A B C A B C C B +-----+ Status : Sukses Percobaan : 7 Durasi : 0ms +-----+</pre>
A A B C A B C C B Status : Sukses Percobaan : 7 Durasi : 0ms
Luaran (.png)

A	A	B
C	A	B
C	C	B

Status: Sukses
Percobaan: 7
Durasi: 0ms

5.4. Test case 4

Masukan (.txt)

```
3 3 3
DEFAULT
AA
AA
AAA
BB
C
```

Luaran (.txt dan CLI)

```
[INFO] : Informasi board...

+-----+
| M      : 3 |
| N      : 3 |
| P      : 3 |
| Tipe   : DEFAULT |
+-----+

[INFO] : Hasil Penyelesaian board...

+-----+
| Status  : Gagal |
| Percobaan : 3799 |
| Durasi   : 3ms  |
+-----+
```

C % %
% % %
% % %

Status : Gagal
Percobaan : 3799
Durasi : 3ms

Luaran (.png)

C	%	%
%	%	%
%	%	%

Status: Gagal
Percobaan: 3799
Durasi: 3ms

Penjelasan

Kasus gagal dikarenakan board penuh namun terdapat puzzle yang belum digunakan.
File .png dan .txt menunjukkan pengecekan terakhir puzzle.

5.5. Test case 5

Masukan (.txt)
7 7 9 DEFAULT AAA AAAAA BBBBB CCC C C CCC DD DD EEE FFF F F F F GGGG GGGG HHH I II
Luaran (.txt dan CLI)

[INFO] : Informasi board...

```
+-----+
| M      : 7 |
| N      : 7 |
| P      : 9 |
| Tipe   : DEFAULT |
+-----+
```

[INFO] : Hasil Penyelesaian board...

```
A A A C C C B
A A A A A C B
F I E E E C B
F I I C C C B
F D D H H H B
F D D G G G G
F F F G G G G
```

```
+-----+
Status      : Sukses
Percobaan   : 609044
Durasi      : 192ms
+-----+
```

```
A A A C C C B
A A A A A C B
F I E E E C B
F I I C C C B
F D D H H H B
F D D G G G G
F F F G G G G
```

Status : Sukses
Percobaan : 609044
Durasi : 192ms

Luaran (.png)

A	A	A	C	C	C	B
A	A	A	A	A	C	B
F	I	E	E	E	C	B
F	I	I	C	C	C	B
F	D	D	H	H	H	B
F	D	D	G	G	G	G
F	F	F	G	G	G	G

Status: Sukses
Percobaan: 609044
Durasi: 192ms

5.6. Test case 6

Masukan (.txt)

```

8 8 12
DEFAULT
A
A
BBBBB
  BBB
C
CC
D
EE
  EE
FFFF
GGGG
HHHH
H
H
H
H
H
IIIII
I  I
I  I
I  I
IIIII

```

JJJ
JJ
J
KK
K
LL
LL

Luaran (.txt dan CLI)

```
[INFO] : Informasi board...

+-----+
| M      : 8 |
| N      : 8 |
| P      : 12|
| Tipe   : DEFAULT |
+-----+

[INFO] : Hasil Penyelesaian board...

A B B B B B C D
A E E B B B C C
F G E E H H H H
F G I I I I I H
F G I J J J I H
F G I J J K I H
L L I J K K I H
L L I I I I I H

+-----+
| Status   : Sukses |
| Percobaan : 27833  |
| Durasi    : 20ms   |
+-----+
```

A B B B B B C D
A E E B B B C C
F G E E H H H H
F G I I I I I H
F G I J J J I H
F G I J J K I H
L L I J K K I H
L L I I I I I H

Status : Sukses
Percobaan : 27833
Durasi : 20ms

Luaran (.png)

A	B	B	B	B	B	C	D
A	E	E	B	B	B	C	C
F	G	E	E	H	H	H	H
F	G	I	I	I	I	I	H
F	G	I	J	J	J	I	H
F	G	I	J	J	K	I	H
L	L	I	J	K	K	I	H
L	L	I	I	I	I	I	H

Status: Sukses
Percobaan: 27833
Durasi: 20ms

5.7. Test case 7

Masukan (.txt)

```
8 8 4
DEFAULT
CCCC
C C
C C
CCCC
DD
DD
AAAAAAAA
A A
A A
A A
A A
A A
A A
AAAAAAAA
BBBBBB
B B
B B
B B
B B
BBBBBB
```

Luaran (.txt dan CLI)

```
[INFO] : Informasi board...

+-----+
| M      : 8 |
| N      : 8 |
| P      : 4 |
| Tipe   : DEFAULT |
+-----+

[INFO] : Hasil Penyelesaian board...

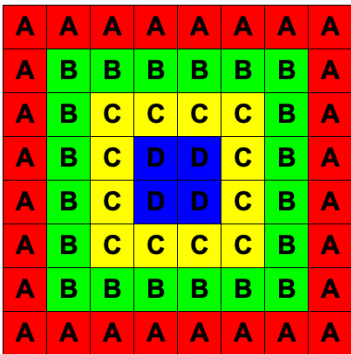
A A A A A A A A
A B B B B B B A
A B C C C C B A
A B C D D C B A
A B C D D C B A
A B C C C C B A
A B B B B B B A
A A A A A A A A

+-----+
| Status  : Sukses |
| Percobaan : 17425 |
| Durasi   : 25ms   |
+-----+
```

A A A A A A A A
A B B B B B B A
A B C C C C B A
A B C D D C B A
A B C D D C B A
A B C C C C B A
A B B B B B B A
A A A A A A A A

Status : Sukses
Percobaan : 17425
Durasi : 25ms

Luaran (.png)



Status: Sukses
Percobaan: 17425
Durasi: 25ms

BAB V

Lampiran

Link Github : https://github.com/AlfianHanifFY/Tucil1_13523073.git

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 <i>Graphical User Interface</i> (GUI)		V
6	Program dapat menyimpan solusi dalam bentuk file gambar	V	
7	Program dapat menyelesaikan kasus konfigurasi <i>custom</i>		V
8	Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		V
9	Program dibuat oleh saya sendiri	V	

BAB VI

Referensi

- [1] Munir, R., *Algoritma Brute Force (Bagian 1)*, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, ITB, 2025. [Online]. Tersedia: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf). [Diakses pada 21 Februari 2025].
- [2] Levitin, A., *Introduction to the Design & Analysis of Algorithms*, edisi ke-3. Pearson, 2012.