

# **APLIKASI PENYELESAIAN IQ PUZZLER PRO DENGAN ALGORITMA BRUTE FORCE**

**LAPORAN TUGAS KECIL 01**  
**Mata Kuliah Strategi Algoritma**  
**IF 2211**



Oleh :  
Ivan Wirawan                      13523046

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**JL. GANESHA 10, BANDUNG 40132**

**2025**

## Daftar Isi

Daftar Isi.....	1
Deskripsi Masalah.....	2
Teori Singkat.....	3
Kode dan Algoritma Program.....	4
1. Algoritma Utama (Main).....	4
2. Algoritma Fungsi selectFile.....	6
3. Algoritma Fungsi pickColor.....	7
4. Algoritma Fungsi getANSI.....	7
5. Algoritma Fungsi chooseFormatGUI.....	8
7. Algoritma Fungsi exportImage.....	9
8. Algoritma Fungsi exportText.....	11
9. Algoritma Fungsi solve.....	12
10. Algoritma Fungsi checkBoard.....	13
11. Algoritma Fungsi canPlace.....	13
12. Algoritma Fungsi placeBlok.....	14
13. Algoritma Fungsi convertMatrix.....	15
14. Algoritma Fungsi generateVariants.....	16
15. Algoritma Fungsi rotate.....	17
17. Algoritma Fungsi printSolution.....	18
Eksperimen.....	19
1. Kasus Eksperimen 1.....	19
2. Kasus Eksperimen 2.....	20
3. Kasus Eksperimen 3.....	20
4. Kasus Eksperimen 4.....	21
5. Kasus Eksperimen 5.....	21
6. Kasus Eksperimen 6.....	22
7. Kasus Eksperimen 7.....	23
Penutup.....	24
1. Kesimpulan.....	24
Lampiran.....	25

## Deskripsi Masalah

IQ Puzzler Pro adalah sebuah permainan teka-teki besutan perusahaan asal Amerika Serikat yaitu Smart Games. Dalam permainan ini, tujuan akhir dari pemain adalah mengisi papan kosong dengan potongan-potongan yang diberikan. Permainan ini memiliki dua buah komponen penting yaitu papan dan blok. Saat memulai permainan, diberikan sebuah papan kosong yang ukurannya telah ditentukan sebelumnya. Selain papan kosong, pemain diberikan beberapa macam potongan blok yang memiliki bentuk unik dengan jumlah yang sesuai untuk mengisi papan.

Pemain harus memulai permainan dengan meletakkan potongan dengan orientasi sesuai agar bisa mengisi papan dengan seluruh blok secara penuh. Pemain dapat melakukan rotasi atau mencerminkan blok sebelum meletakkannya ke dalam papan. Pemain harus berstrategi atau mencoba secara berulang urutan dan orientasi yang tepat setiap potongan agar bisa mengisi papan secara penuh tanpa ada blok yang tersisa.

Contoh dari permainan dapat dilihat di gambar bawah ini.



Gambar 1. Contoh Permainan IQ Puzzler Pro

## Teori Singkat

Menurut Kamus Besar Bahasa Indonesia (KBBI), algoritma adalah suatu prosedur sistematis untuk menyelesaikan masalah matematika dalam langkah-langkah terbatas atau urutan pengambilan keputusan yang logis untuk memecahkan masalah tersebut. Dalam bidang komputasi, algoritma digunakan sebagai istilah cara penyelesaian sebuah masalah yang dikomputasikan. Algoritma ini akan dijalankan oleh komputer dan membantu manusia dengan mengerjakannya lebih cepat.

Strategi algoritma adalah sebuah pendekatan umum untuk memecahkan persoalan secara algoritmik. Melalui strategi algoritma, tujuan utamanya adalah memecahkan sebuah persoalan, dimana kesangkilan tidak dipertimbangkan bergantung dengan strategi yang dianut. Sebuah algoritma dapat dibedakan menjadi dua buah bergantung dengan solusi yang diinginkan. Jenis pertama adalah algoritma yang memberikan hasil akhir saja. Jenis kedua adalah algoritma yang memberikan langkah penyelesaiannya sebagai solusi akhir.

Dalam menentukan strategi yang sesuai, dapat dilakukan analisis algoritma. Melalui analisis ini, kompleksitas waktu dan ruang dari algoritma akan diuji dari segi kesangkilannya. Ada berbagai jenis algoritma yang sering digunakan dalam menyelesaikan permasalahan komputasi. Setiap algoritma memiliki karakteristik dari cara penyelesaiannya begitu juga dengan kompleksitas waktu dan ruangnya. Contoh algoritma tersebut adalah Algoritma Brute-Force, Algoritma Greedy, Algoritma Divide and Conquer, Algoritma Backtracking, Algoritma Branch and Bound, dan lain-lain

Algoritma Brute Force adalah algoritma pendekatan lurus untuk memecahkan suatu persoalan. Di dalam algoritma ini, fokusnya adalah menyelesaikan masalah bukan membuat algoritma yang efektif dan efisien. Algoritma ini tidak mementingkan kompleksitas ruang dan waktu yang dibutuhkan untuk menyelesaikan masalah. Di sisi lain, algoritma ini menjamin adanya sebuah solusi yang tepat dalam persoalan. Akan tetapi, banyak waktu dan ruang yang terbuang secara sia-sia. Algoritma ini bukanlah algoritma yang paling sangkil, namun algoritma ini pasti memberikan solusi yang tepat bagi permasalahan yang sedang dihadapi.

## Kode dan Algoritma Program

### 1. Algoritma Utama (Main)

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1  public static void main(String[] args)
2  {
3      File selectedFile = selectFile();
4      if (selectedFile == null)
5      {
6          System.out.println("Tidak ada file yang dipilih.");
7          return;
8      }
9
10     try
11     {
12         Scanner scanner = new Scanner(selectedFile);
13         if(!scanner.hasNextLine())
14         {
15             System.out.println("File kosong.");
16             return;
17         }
18         String[] firstLine = scanner.nextLine().split(" ");
19         if(firstLine.length != 3 || !String.join(" ",
20 firstLine).matches("\\s*\\d+\\s+\\d+\\s+\\d+\\s*"))
21         {
22             System.out.println("Format file tidak valid.");
23             return;
24         }
25         try
26         {
27             N = Integer.parseInt(firstLine[0]);
28             M = Integer.parseInt(firstLine[1]);
29             P = Integer.parseInt(firstLine[2]);
30         }
31         catch (NumberFormatException e)
32         {
33             System.out.println("Format Dimensi tidak valid.");
34             return;
35         }
36         if (N <= 0 || M <= 0 || P <= 0)
37         {
38             System.out.println("Dimensi harus lebih besar dari 0.");
39             return;
40         }
41
42         String type = scanner.nextLine().trim();
43         if(N*M < P)
44         {
45             System.out.println("Jumlah kotak tidak mencukupi.");
46             return;
47         }
48         else if (!type.equals("DEFAULT"))
49         {
50             System.out.println("Tipe soal tidak valid.");
51             return;
52         }
53     }
54     else
55     {
56         // ... (rest of the code)
57     }
58 }

```

Gambar 2. Tangkapan Layar Kode Main (1)

```

1  {
2      List<String> allLines = new ArrayList<>();
3
4      while (scanner.hasNextLine())
5      {
6          allLines.add(scanner.nextLine());
7      }
8      scanner.close();
9
10     int i = 0;
11     while (i < allLines.size())
12     {
13         ArrayList<String> currentPiece = new ArrayList<>();
14         char currentChar = allLines.get(i).charAt(0);
15
16
17         if (!colorMap.containsKey(currentChar))
18         {
19             colorMap.put(currentChar, pickColor());
20         }
21
22         while (i < allLines.size() && allLines.get(i).charAt(0) ==
currentChar)
23         {
24             currentPiece.add(allLines.get(i));
25             i++;
26         }
27
28         pieces.add(convertMatrix(currentPiece));
29     }
30     board = new char[M][N];
31     for (char[] row : board) Arrays.fill(row, '.');
32     startTime = System.currentTimeMillis();
33     if (solve(0))
34     {
35         printSolution();
36         long endTime = System.currentTimeMillis();
37         System.out.println("Waktu proses: " + (endTime - startTime) + " ms");
38         System.out.println("Jumlah percobaan: " + attempts);
39         chooseFormatGUI();
40     } else
41     {
42         System.out.println("Tidak ditemukan solusi.");
43     }
44 }
45
46 catch (Exception e)
47 {
48     e.printStackTrace();
49 }
50 }

```

Gambar 3. Tangkapan Layar Kode Main (2)

Strategi algoritma yang diimplementasikan di dalam program penyelesaian IQ Puzzler Pro adalah algoritma *brute-force*. Konsep dari algoritma ini adalah penyelesaian secara lurus tanpa mengindahkan kesanggikan baik dari segi kompleksitas ruang maupun waktu. Secara mudah, algoritma program ini adalah mencoba seluruh bentuk potongan blok sehingga tercipta suatu susunan dimana blok terisi penuh oleh potongan yang ada.

Di dalam implementasinya, program dimulai dari memilih berkas masukan melalui *Graphical User Interface* (GUI), pengguna bisa memilih *path* ke berkas soal yang menjadi soal. Diasumsikan bahwa berkas soal sudah valid dengan format yang sesuai spesifikasi program. Setelah berhasil memilih berkas, program akan membaca seluruh komponen yang ada. Contohnya adalah nilai panjang, lebar, jumlah komponen, dan setiap potongan blok. Agar mempermudah inspeksi, saat pembacaan potongan, setiap potongan akan diberikan warna khusus melalui algoritma yang sudah ditentukan sebelumnya. Kumpulan warna ini akan disimpan dengan sebuah *map* yang kuncinya adalah setiap karakter. Bentuk potongan dari teks yang dibaca akan diubah menjadi matriks dan disimpan dalam *list*.

Setelah itu, program akan menginisiasi pembuatan papan kosong dengan panjang dan lebar sesuai dengan masukan. Program akan memulai algoritma penyelesaian setelah memulai perhitungan. Jika ditemukan sebuah solusi bagi teka-teki dalam soal, maka bentuk hasil akhir akan ditampilkan di dalam *terminal*. Informasi waktu proses dan jumlah percobaan yang dilakukan juga akan ditampilkan. Setelah itu, pengguna akan diberikan opsi untuk melakukan *export* solusi menjadi sebuah gambar atau file teks kembali melalui GUI. Apabila tidak ditemukan solusi, maka program akan mencetak pesan tersebut di *terminal*.

## 2. Algoritma Fungsi selectFile

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1  public static File selectFile()
2  {
3      JFileChooser fileChooser = new JFileChooser();
4      fileChooser.setDialogTitle("Pilih File Puzzle Input");
5      fileChooser.setFileFilter(new FileNameExtensionFilter("Text Files (*.txt)",
6      "txt"));
7
8      int userSelection = fileChooser.showOpenDialog(null);
9      if (userSelection == JFileChooser.APPROVE_OPTION)
10     {
11         return fileChooser.getSelectedFile();
12     }
13     return null;
14 }

```

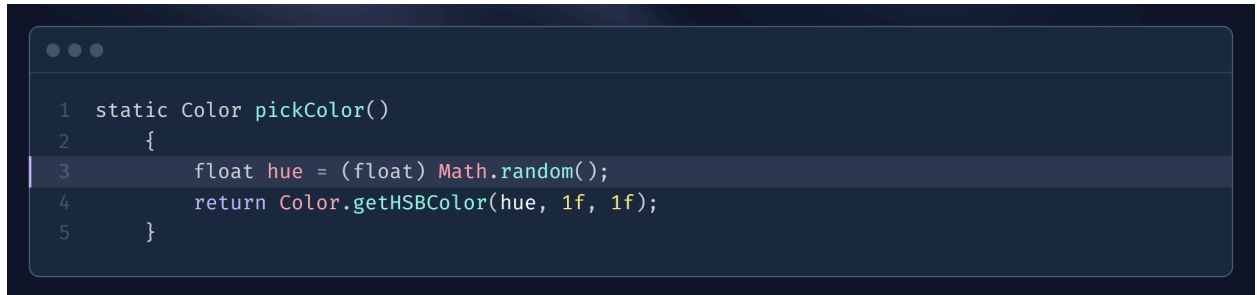
Gambar 4. Tangkapan Layar Kode selectFile

Algoritma yang digunakan dalam fungsi ini hanyalah memilih direktori yang ingin dituju, file soal, dan menyimpan file tersebut. Jika tidak ada *file* yang dipilih, maka fungsi akan mengembalikan nilai null. Di dalam kode program, kaskas yang digunakan adalah *JFileChooser* yaitu sebuah komponen di dalam Java Swing. Komponen ini akan

menampilkan dialog pemilihan berkas kepada pengguna. Dengan ini, pengguna dapat memilih secara langsung direktori beserta file yang ingin dituju.

### 3. Algoritma Fungsi pickColor

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1 static Color pickColor()
2 {
3     float hue = (float) Math.random();
4     return Color.getHSBColor(hue, 1f, 1f);
5 }

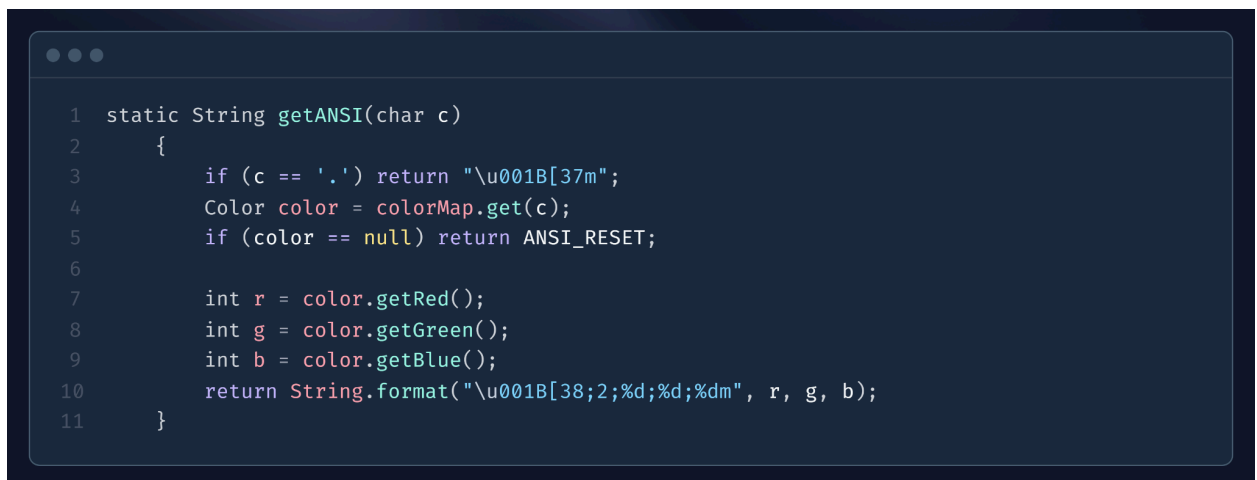
```

Gambar 5. Tangkapan Layar Kode pickColor

Fungsi ini berguna untuk menghasilkan sebuah warna acak. Di dalam fungsi ini, program akan menghasilkan sebuah angka acak yang menjadi warna dasar. Setelah itu, nilai tersebut akan dikonversi menjadi sebuah data warna dengan kecerahan maksimal. Tujuan dari fungsi ini adalah memberikan warna agar pengguna bisa melihat solusi dengan lebih jelas dari segi potongan blok.

### 4. Algoritma Fungsi getANSI

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1 static String getANSI(char c)
2 {
3     if (c == '.') return "\u001B[37m";
4     Color color = colorMap.get(c);
5     if (color == null) return ANSI_RESET;
6
7     int r = color.getRed();
8     int g = color.getGreen();
9     int b = color.getBlue();
10    return String.format("\u001B[38;2;%d;%d;%dm", r, g, b);
11 }

```

Gambar 6. Tangkapan Layar Kode getANSI

Fungsi ini digunakan sebagai fungsi pembantu dalam menuliskan solusi persoalan. Sebelumnya, warna akan diberikan acak melalui fungsi nomor 3. Di dalam fungsi nomor 4 ini, program akan membaca nilai-nilai Red-Green-Blue (RGB), dari warna teks yang sudah diberikan sebelumnya. Setelah mendapatkannya, program ini akan mengembalikan sebuah *string* yang merupakan format pewarnaan teks menggunakan



ANSI. Solusi dari permasalahan akan ditampilkan di terminal secara berwarna menggunakan ANSI. Apabila teks yang dibaca tidak berwarna, maka program akan mengembalikan warna dasar yaitu putih.

## 5. Algoritma Fungsi chooseFormatGUI

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1 static void chooseFormatGUI()
2 {
3     String[] options = {"File Teks (TXT)", "File Gambar (JPG)", "Keduanya", "Tidak
Disimpan"};
4     int choice = JOptionPane.showOptionDialog
5     (
6         null,
7         "Apakah Anda ingin mengexport dalam bentuk file txt atau gambar?",
8         "Export Format",
9         JOptionPane.DEFAULT_OPTION,
10        JOptionPane.QUESTION_MESSAGE,
11        null,
12        options,
13        options[0]
14    );
15
16    if (choice == 3 || choice == JOptionPane.CLOSED_OPTION)
17    {
18        return;
19    }
20
21    try
22    {
23        if (choice == 0 || choice == 2)
24        {
25            exportGUI("txt");
26        }
27        if (choice == 1 || choice == 2)
28        {
29            exportGUI("jpg");
30        }
31    }
32    catch (IOException e)
33    {
34        System.out.println("Error: " + e.getMessage());
35    }
36 }

```

Gambar 7. Tangkapan Layar Kode chooseFormatGUI

Fungsi ini digunakan sebagai tampilan kepada pengguna untuk melakukan *export* solusi dalam bentuk file teks ataupun gambar. Program menggunakan kaskas `JOptionPane` sebagai *User Interface* (UI) sehingga tampilan lebih menarik dan lebih mudah. Setelah menampilkan pilihan dan pesan, program akan memanggil fungsi lainnya (nomor 6)

sesuai dengan pengguna. Pengguna bisa memilih tidak melakukan *export* sama sekali, bentuk gambar, bentuk teks, dan keduanya.

#### 6. Algoritma Fungsi exportGUI

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1  static void exportGUI(String ext) throws IOException
2  {
3      JFileChooser fileChooser = new JFileChooser();
4      fileChooser.setDialogTitle("Export Solution as " + ext.toUpperCase());
5      fileChooser.setFileFilter(new FileNameExtensionFilter(
6          ext.equals("txt") ? "Text File (*.txt)" : "JPEG Image (*.jpg)",
7          ext
8      ));
9
10     if (fileChooser.showSaveDialog(null) == JFileChooser.APPROVE_OPTION)
11     {
12         String path = fileChooser.getSelectedFile().getPath();
13         if (!path.toLowerCase().endsWith("." + ext))
14         {
15             path += "." + ext;
16         }
17
18         if (ext.equals("jpg"))
19         {
20             exportImage(path);
21         } else
22         {
23             exportText(path);
24         }
25         System.out.println("Solution exporte to: " + path);
26     }
27 }

```

Gambar 7. Tangkapan Layar Kode exportGUI

Fungsi ini merupakan fungsi lanjutan dari fungsi sebelumnya. Setelah pengguna memilih format yang diinginkan, maka program akan meminta pengguna untuk memilih direktori tempat pengguna ingin menyimpan hasilnya. Setelah berhasil memilih direktori, maka program akan memanggil fungsi lainnya yang sesuai dengan permintaan pengguna (gambar atau teks). Terakhir, program akan memberi pesan letak *path* penyimpanan di dalam *terminal*.

#### 7. Algoritma Fungsi exportImage

Kode algoritma ini dapat dilihat dari gambar di bawah ini.

```

1 static void exportImage(String path) throws IOException
2 {
3     int cell = 30;
4     int font = 20;
5     BufferedImage image = new BufferedImage(N * cell, M * cell,
6     BufferedImage.TYPE_INT_RGB);
7     Graphics2D g2d = image.createGraphics();
8     g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
9     RenderingHints.VALUE_ANTIALIAS_ON);
10    g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
11    RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
12
13    g2d.setColor(Color.WHITE);
14    g2d.fillRect(0, 0, N * cell, M * cell);
15
16    g2d.setFont(new Font("Arial", Font.BOLD, font));
17    FontMetrics metrics = g2d.getFontMetrics();
18
19    for (int i = 0; i < M; i++) {
20        for (int j = 0; j < N; j++) {
21            char c = board[i][j];
22            if (c != '.') {
23                g2d.setColor(colorMap.get(c));
24                g2d.fillRect(j * cell, i * cell, cell, cell);
25
26                String charStr = String.valueOf(c);
27                int textWidth = metrics.stringWidth(charStr);
28                int textHeight = metrics.getHeight();
29                int x = j * cell + (cell - textWidth) / 2;
30                int y = i * cell + (cell + textHeight) / 2 - metrics.getDescent();
31
32                g2d.setColor(Color.WHITE);
33                g2d.drawString(charStr, x, y);
34            }
35            g2d.setColor(Color.BLACK);
36            g2d.drawRect(j * cell, i * cell, cell, cell);
37        }
38    }
39
40    g2d.dispose();
41    ImageIO.write(image, "jpg", new File(path));
42 }

```

Gambar 8. Tangkapan Layar Kode exportImage

Fungsi ini bertujuan untuk melakukan *export* dalam bentuk gambar menggunakan kakas dari Java yaitu Graphics2D. Pada awalnya, algoritma ini akan membuat sebuah *buffer* dengan jumlah baris dan kolom yang sesuai dengan ukurannya. Ukuran dari setiap sel dan tulisan serta gaya tulisan sudah ditampilkan sebelumnya. Pada awalnya, program

akan mengisi gambar dengan warna putih sebagai latar belakang. Setelah itu, program akan menggambar matriks setiap potongan (sel) dengan format yang sesuai. Program akan membaca warna yang sebelumnya telah ditentukan agar warna di hasil lainnya juga sama. Terakhir, program akan menggambarkan garis-garis batas. Setelah menyelesaikan proses pembuatan, maka program akan menyimpan *file* gambar tersebut di tempat yang telah dipilih *user*.

## 8. Algoritma Fungsi exportText

Kode algoritma ini dapat dilihat dari gambar di bawah ini.

```

1 static void exportText(String path) throws IOException
2 {
3     try (PrintWriter writer = new PrintWriter(new FileWriter(path)))
4     {
5         for (char[] row : board)
6         {
7             writer.println(new String(row));
8         }
9
10        writer.println("\nInformasi Proses Penyelesaian:");
11        writer.printf("Waktu proses: %d ms%n", System.currentTimeMillis() -
12 startTime);
13        writer.printf("Jumlah percobaan: %d%n", attempts);
14    }
15 }

```

Gambar 9. Tangkapan Layar Kode exportText

Fungsi ini bertujuan untuk melakukan *export dalam* bentuk teks. Setelah pengguna memilih letak repositori penyimpanan, program akan menuliskan isi dari papan yang telah berisi solusi ke dalamnya. Selain itu, program juga menambahkan informasi proses penyelesaian berapa waktu proses dan percobaan. Hasilnya adalah sebuah *file* teks yang berisikan solusi soal, dan informasi tambahannya ke dalam repositori yang telah ditentukan.

## 9. Algoritma Fungsi solve

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1  static boolean solve(int idx)
2  {
3      if (idx >= pieces.size()) return checkBoard();
4
5      char[][] blok = pieces.get(idx);
6      List<char[][]> transformations = generateVariants(blok);
7      for (char[][] variant : transformations)
8      {
9          for (int i = 0; i <= M - variant.length; i++)
10         {
11             for (int j = 0; j <= N - variant[0].length; j++)
12             {
13
14                 if (canPlace(variant, i, j))
15                 {
16                     attempts++;
17                     placeBlok(variant, i, j, true);
18                     if (solve(idx + 1)) return true;
19                     placeBlok(variant, i, j, false);
20                 }
21             }
22         }
23     }
24     return false;
25 }

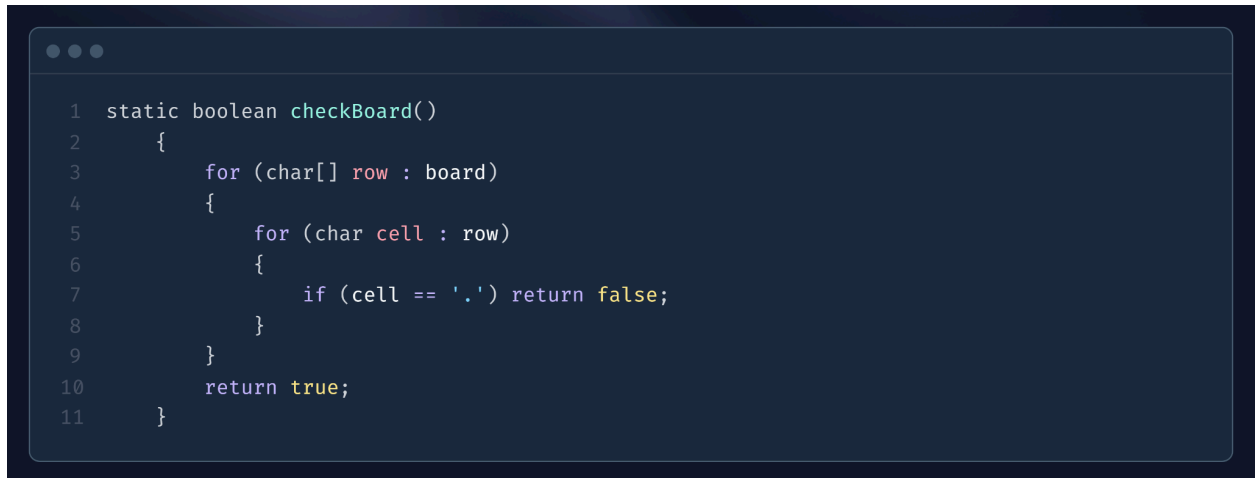
```

Gambar 10. Tangkapan Layar Kode solve

Fungsi ini merupakan fungsi utama di dalam program ini. Secara singkat, fungsi ini akan mencoba seluruh kemungkinan susunan blok agar bisa termuat dalam papan dengan tepat. Batas yang ditetapkan algoritma ini adalah apabila potongan blok sudah habis terpakai, dan papan sudah terisi penuh. Program akan mencoba terus berbagai kemungkinan, dengan mencobakan variasi (melalui rotasi dan membalik) peletakan blok dalam setiap jenisnya. Sebelumnya, potongan akan dimasukkan ke dalam fungsi nomor 14 untuk mencari kemungkinan variasinya. Program akan mencoba dari setiap variasi yang tersedia di dalam *list* matriks potongan. Apabila bisa diletakkan, maka program akan melanjutkan ke potongan selanjutnya. Jika tidak, maka program akan mengulang ke potongan sebelumnya lagi. Program dirancang secara rekursif sehingga setiap kemungkinan akan dicoba kepada setiap letak. Di dalam setiap percobaan, akan dihitung juga jumlah percobaan yang dilakukan agar dapat dilihat. Program ini akan mengembalikan nilai *true* atau *false* yang menandakan apakah teka-teki yang diberikan memiliki solusi atau tidak.

## 10. Algoritma Fungsi checkBoard

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1 static boolean checkBoard()
2 {
3     for (char[] row : board)
4     {
5         for (char cell : row)
6         {
7             if (cell == '.') return false;
8         }
9     }
10    return true;
11 }

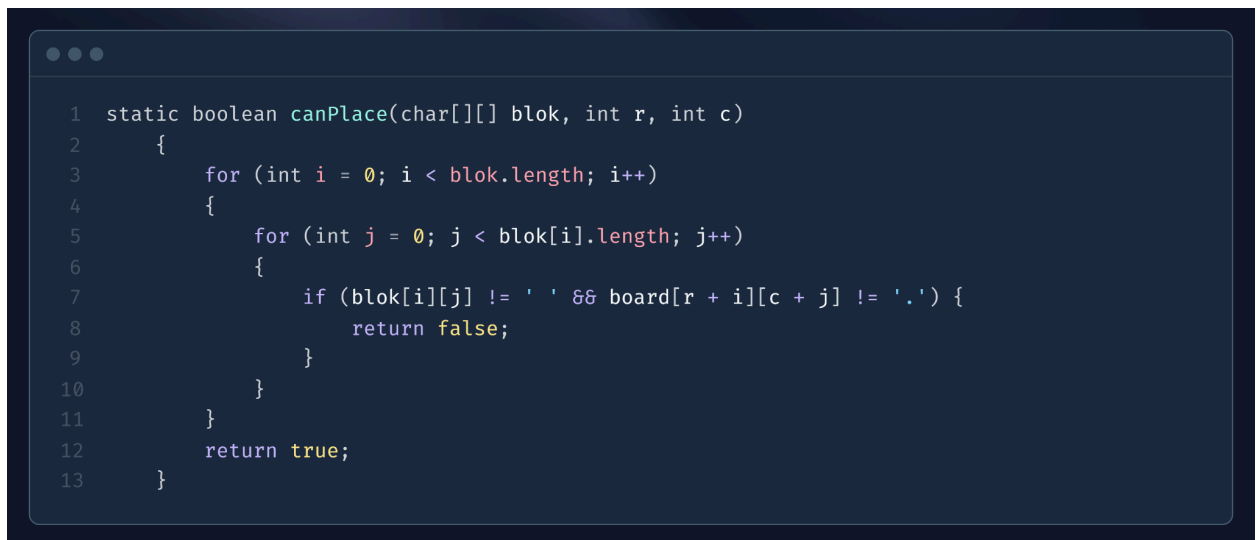
```

Gambar 11. Tangkapan Layar Kode checkBoard

Fungsi ini bertujuan untuk menentukan apakah papan permainan sudah terisi penuh atau tidak. Program ini akan memeriksa seluruh sel secara terperinci, apabila masih ada yang kosong (ditandai dengan karakter '.'), maka akan mengembalikan nilai *false*. Apabila sudah penuh, maka akan mengembalikan nilai *true*.

## 11. Algoritma Fungsi canPlace

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1 static boolean canPlace(char[][] blok, int r, int c)
2 {
3     for (int i = 0; i < blok.length; i++)
4     {
5         for (int j = 0; j < blok[i].length; j++)
6         {
7             if (blok[i][j] != ' ' && board[r + i][c + j] != '.') {
8                 return false;
9             }
10        }
11    }
12    return true;
13 }

```

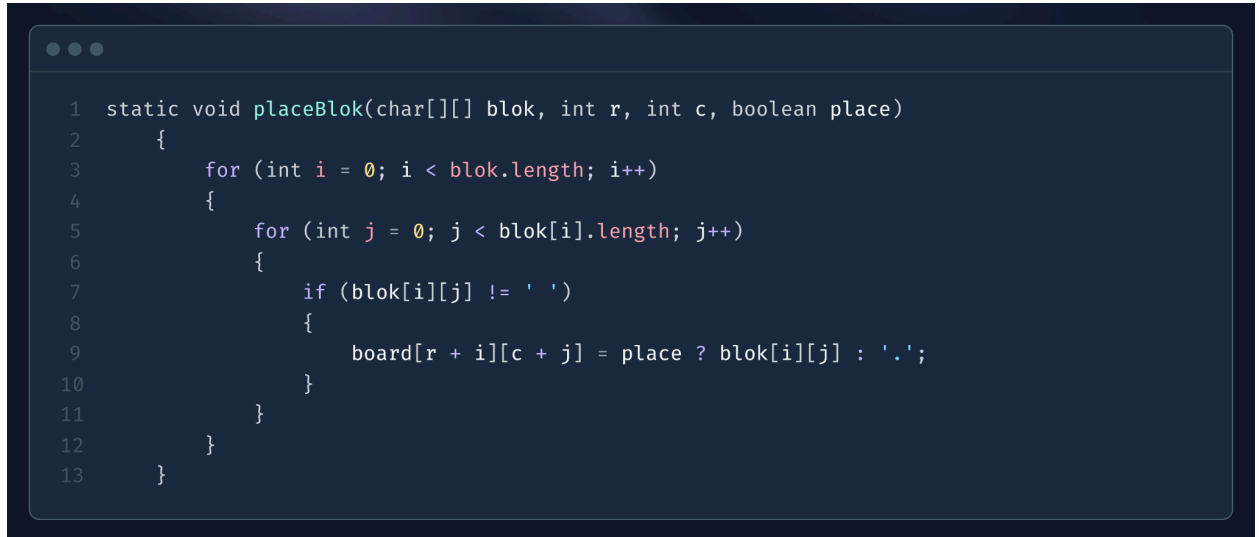
Gambar 12. Tangkapan Layar Kode canPlace

Fungsi ini bertujuan untuk mengecek apakah potongan bisa ditempatkan. Algoritmanya adalah dengan melakukan pengecekan terhadap titik pusat dan titik yang akan ditempati blok. Apabila titik pusat tidak berada di luar papan dan di seluruh titik yang akan ditempati merupakan kosong (ditandai dengan karakter '.'), maka akan

mengembalikan hasil *true*. Hal ini berarti bahwa potongan blok dapat diletakkan di koordinat tersebut. Jika telah melalui seluruh iterasi dan tidak berhasil, maka program akan mengembalikan nilai *false*.

## 12. Algoritma Fungsi placeBlok

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



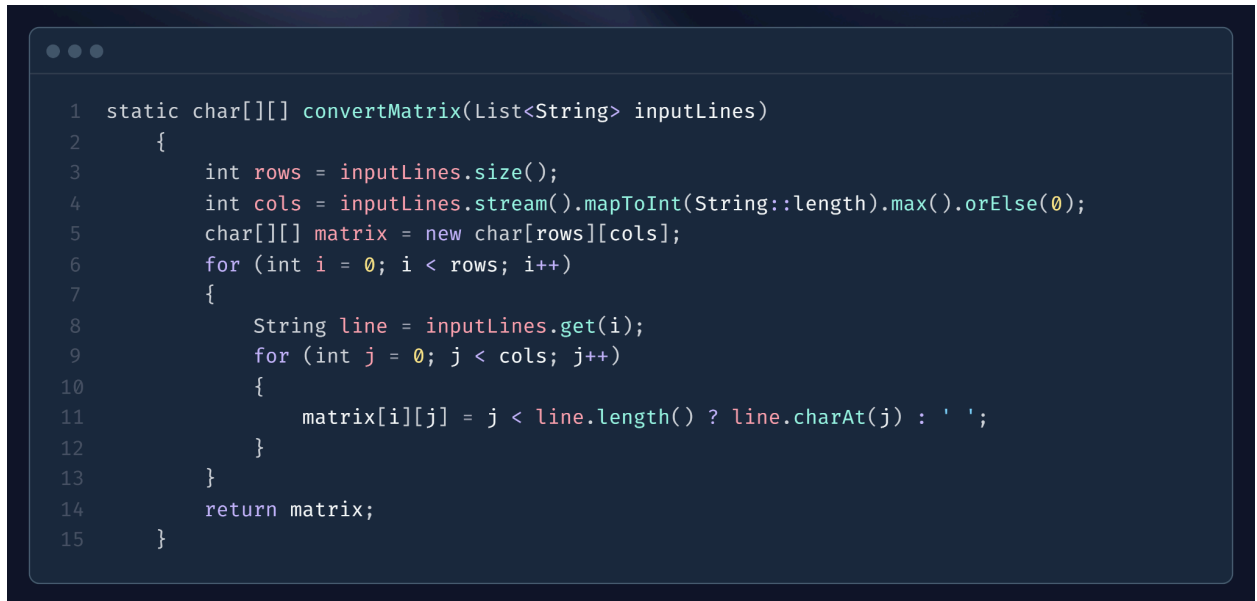
```
1 static void placeBlok(char[][] blok, int r, int c, boolean place)
2 {
3     for (int i = 0; i < blok.length; i++)
4     {
5         for (int j = 0; j < blok[i].length; j++)
6         {
7             if (blok[i][j] != ' ')
8             {
9                 board[r + i][c + j] = place ? blok[i][j] : '.';
10            }
11        }
12    }
13 }
```

Gambar 13. Tangkapan Layar Kode placeBlok

Fungsi ini bertujuan untuk meletakkan potongan blok di dalam papan permainan. Setelah dikonfirmasi melalui fungsi nomor 11, maka program dapat meletakkan atau menghapus potongan pada papan. Dengan iterasi ke dimensi panjang dan lebar potongan, apabila ada karakter (tidak kosong) pada blok, maka akan diletakkan ke dalam matriks papan permainan. Akan tetapi, program ini juga bisa menghapus potongan apabila adanya kesalahan letak. Penghapusan ini berguna di saat program kembali mengulang untuk mencoba variasi susunan lainnya.

### 13. Algoritma Fungsi convertMatrix

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```
1 static char[][] convertMatrix(List<String> inputLines)
2 {
3     int rows = inputLines.size();
4     int cols = inputLines.stream().mapToInt(String::length).max().orElse(0);
5     char[][] matrix = new char[rows][cols];
6     for (int i = 0; i < rows; i++)
7     {
8         String line = inputLines.get(i);
9         for (int j = 0; j < cols; j++)
10        {
11            matrix[i][j] = j < line.length() ? line.charAt(j) : ' ';
12        }
13    }
14    return matrix;
15 }
```

Gambar 14. Tangkapan Layar Kode convertMatrix

Algoritma ini berfungsi sebagai pengubah bacaan berkas soal dari teks menjadi sebuah matriks. Program akan menginisiasi pembuatan matriks dengan jumlah kolom dan baris yang sesuai dengan soal. Kemudian membaca jumlah elemen serta jumlah maksimalnya untuk menentukan panjang dan lebar. Untuk bagian yang kosong di dalam teks, akan diisi dengan sebuah spasi kosong agar dimensi baris dan kolom tetap sesuai dengan spesifikasi.



## 14. Algoritma Fungsi generateVariants

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1  static List<char[][]> generateVariants(char[][] blok)
2  {
3
4      if (blok.length == 1 && blok[0].length == 1)
5      {
6          return Collections.singletonList(blok);
7      }
8
9      Set<String> seen = new HashSet<>();
10     List<char[][]> transformations = new ArrayList<>();
11     char[][] current = blok;
12
13     for (int i = 0; i < 4; i++)
14     {
15         if (seen.add(Arrays.deepToString(current))) transformations.add(current);
16         current = rotate(current);
17     }
18
19     current = flip(blok);
20     for (int i = 0; i < 4; i++)
21     {
22         if (seen.add(Arrays.deepToString(current))) transformations.add(current);
23         current = rotate(current);
24     }
25     return transformations;
26 }

```

Gambar 15. Tangkapan Layar Kode generateVariants

Fungsi ini bertujuan untuk menyimpan berbagai macam bentuk potongan blok. Variasi bentuk blok didapatkan melalui fungsi nomor 15 dan fungsi nomor 16. Untuk blok dengan ukuran 1\*1, akan dikembalikan secara langsung tanpa dirotasi maupun dicerminkan. Ada dua variabel penting yaitu Set yang berisikan string (bernama seen) dan List yang berisikan matriks (bernama transformations). Set yang berisikan string tersebut digunakan untuk mencegah adanya elemen duplikasi daripada potongan blok. Caranya adalah dengan menggunakan fungsi deepToString, dimana sebuah array multidimensi disimpan melalui sebuah hash. Apabila ada array yang sama, maka hasil dari proses hash akan sama. Disinilah program akan mengecek keberadaan array yang sama melalui set seen yang telah ada. Kemudian, list transformations berfungsi sebagai list penyimpan matriks variasi potongan yang ada. Di dalam program, setiap potongan akan dirotasi selama 90 derajat sebanyak 4 kali dan dicerminkan di setiap rotasi. Maka, potongan akan memiliki 8 jenis bentuk pada umumnya. Fungsi ini akan mengembalikan list berisi variasi potongan yang dapat dicobakan di papan.

## 15. Algoritma Fungsi rotate

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1  static char[][] rotate(char[][] matrix)
2  {
3      int rows = matrix.length, cols = matrix[0].length;
4      char[][] rotated = new char[cols][rows];
5      for (int i = 0; i < rows; i++)
6      {
7          for (int j = 0; j < cols; j++)
8          {
9              rotated[j][rows - 1 - i] = matrix[i][j];
10         }
11     }
12     return rotated;
13 }
14

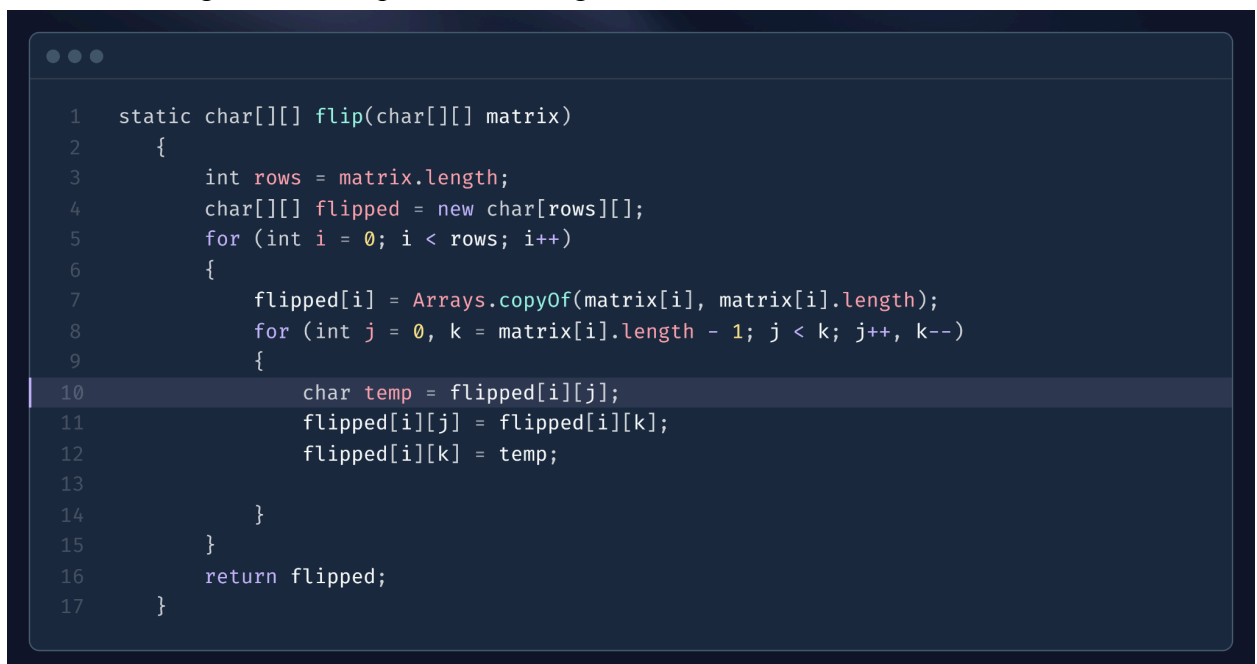
```

Gambar 16. Tangkapan Layar Kode rotate

Fungsi ini bertujuan untuk menghasilkan rotasi dari potongan blok dalam bentuk matriks. Program akan membuat sebuah matriks baru dengan jumlah kolom awal sebagai baris baru dan baris lama sebagai kolom baru. Fungsi ini akan merotasi potongan sebanyak 90 derajat. Setelah itu, program akan mengembalikan bentuk matriks yang telah dirotasi.

## 16. Algoritma Fungsi flip

Kode algoritma ini dapat dilihat dari gambar di bawah ini.



```

1  static char[][] flip(char[][] matrix)
2  {
3      int rows = matrix.length;
4      char[][] flipped = new char[rows][];
5      for (int i = 0; i < rows; i++)
6      {
7          flipped[i] = Arrays.copyOf(matrix[i], matrix[i].length);
8          for (int j = 0, k = matrix[i].length - 1; j < k; j++, k--)
9          {
10             char temp = flipped[i][j];
11             flipped[i][j] = flipped[i][k];
12             flipped[i][k] = temp;
13         }
14     }
15     return flipped;
16 }
17


```

Gambar 17. Tangkapan Layar Kode flip

Program ini bertujuan untuk menghasilkan cerminan dari potongan blok dalam bentuk matriks. Sama seperti fungsi nomor 15, fungsi ini akan membuat sebuah matriks baru. Akan tetapi, nilai baris dan kolomnya tetap sesuai. Pada awalnya, program akan menyalin isi seluruh matriks awal. Setelah itu, program akan melakukan cerminan dari matriks tersebut kemudian mengembalikan hasilnya.

## 17. Algoritma Fungsi printSolution

Kode algoritma ini dapat dilihat dari gambar di bawah ini.

A screenshot of a code editor with a dark background. It displays a Java function named `printSolution()`. The function is static and void. It uses two nested for loops: the outer loop iterates over each row of the `board` array, and the inner loop iterates over each cell in the current row. Inside the inner loop, the code calls `System.out.print(getANSI(cell) + cell + ANSI_RESET);` to print each cell with its corresponding ANSI color code. After the inner loop finishes for a row, the code calls `System.out.println();` to move to the next line. The code is numbered from 1 to 12 on the left side of the editor.

```
1 static void printSolution()
2 {
3     for (char[] row : board)
4     {
5         for (char cell : row)
6         {
7             System.out.print(getANSI(cell) + cell + ANSI_RESET);
8         }
9     }
10    System.out.println();
11 }
12 }
```

Gambar 18. Tangkapan Layar Kode printSolution

Fungsi ini bertujuan untuk mencetak hasil solusi ke dalam *terminal*. Algoritma yang digunakan adalah melakukan iterasi dari setiap baris dan kolom di dalam papan, kemudian mencetak setiap sel dengan warna yang telah disesuaikan.

## Eksperimen

### 1. Kasus Eksperimen 1

```
Tucil1_13523046 > test > ≡ input1.txt
1  5 5 7
2  DEFAULT
3  A
4  AA
5  B
6  BB
7  C
8  CC
9  D
10 DD
11 EE
12 EE
13 E
14 FF
15 FF
16 F
17 GGG
```

Gambar 19. Input Eksperimen 1

```
AGGGD
AABDD
CCBBE
CFFEE
FFFE
Waktu proses: 64 ms
Jumlah percobaan: 7393
Solution exparte to: C:\Users\ivanw\OneDrive\Documents\AAA\output1.txt
Solution exparte to: C:\Users\ivanw\OneDrive\Documents\AAA\output1.jpg
```

Gambar 20. Output Eksperimen 1

## 2. Kasus Eksperimen 2

```
Tucil1_13523046 > test > ≡ input2.txt
1 2 2 3
2 DEFAULT
3 A
4 BB
5 C
```

Gambar 21. Input Eksperimen 2

```
AC
BB
Waktu proses: 2 ms
Jumlah percobaan: 3
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output2.txt
Solution exported to: C:\Users\ivanw\OneDrive\Documents\output2.jpg
□
```

Gambar 22. Output Eksperimen 2

## 3. Kasus Eksperimen 3

```
Tucil1_13523046 > test > ≡ input3.txt
1 3 3 9
2 DEFAULT
3 A
4 B
5 C
6 D
7 E
8 F
9 G
10 H
11 I
```

Gambar 23. Input Eksperimen 3

```
ABCDD
ABCCC
ABBBB
AAAAA
Waktu proses: 4 ms
Jumlah percobaan: 4
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output4.txt
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output4.jpg
□
```

Gambar 24. Output Eksperimen 3

#### 4. Kasus Eksperimen 4

```
Tucil1_13523046 > test > ≡ input4.txt
1 5 4 4
2 DEFAULT
3 A
4 A
5 A
6 AAAAA
7 B
8 B
9 BBBB
10 C
11 CCC
12 DD
```

Gambar 25. Input Eksperimen 4

```
ABCDD
ABCCC
ABBBB
AAAAA
Waktu proses: 4 ms
Jumlah percobaan: 4
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output4.txt
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output4.jpg
```

Gambar 26. Output Eksperimen 4

#### 5. Kasus Eksperimen 5

```
Tucil1_13523046 > test > ≡ input5.txt
1 3 3 3
2 DEFAULT
3 A
4 AAA
5 B
6 C
7 CCC
```

Gambar 27. Input Eksperimen 5

```
CCC
ABC
AAA
Waktu proses: 2 ms
Jumlah percobaan: 12
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output5.txt
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output5.jpg
```

Gambar 28. Output Eksperimen 5

## 6. Kasus Eksperimen 6

```
Tucil1_13523046 > test > ≡ input6.txt
1 2 5 4
2 DEFAULT
3 AA
4 B
5 B
6 C
7 C
8 DD
9 DD
```

Gambar 29. Input Eksperimen 6

```
AA
BC
BC
DD
DD
Waktu proses: 2 ms
Jumlah percobaan: 4
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output6.txt
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output6.jpg
```

Gambar 30. Output Eksperimen 6

## 7. Kasus Eksperimen 7

```
Tucil1_13523046 > test > ☰ input7.txt
1      8 4 7
2      DEFAULT
3      AAA
4      A
5      BB
6      BBB
7      BBB
8      C
9      DD
10     E
11     EE
12     EE
13     FF
14     FF
15     G
16     G
17     GGG
18     GGG
```

Gambar 31. Input Eksperimen 7

```
AAABBCFF
AGGBBBFF
DGGBBBEE
DGGGEEEE
Waktu proses: 16 ms
Jumlah percobaan: 683
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output7.txt
Solution exported to: C:\Users\ivanw\OneDrive\Documents\AAA\output7.jpg
```

Gambar 32. Output Eksperimen 7



## Penutup

### 1. Kesimpulan

Melalui implementasi program yang telah disimpulkan, dapat disimpulkan bahwa algoritma *brute force* menjanjikan sebuah solusi yang tepat meskipun tidak efektif dan efisien. Dalam kasus menyelesaikan IQ Puzzler Pro, penyelesaian dilakukan tanpa adanya optimisasi sama sekali. Cara pandang yang lurus untuk menyelesaikan permainan ini adalah dengan mencoba segala bentuk dan cara. Hal tersebutlah yang mendasari algoritma *brute force*. Kompleksitas waktu dan ruang tidak terlalu dipedulikan dalam strategi tersebut, fokusnya terletak pada penyelesaian masalah. Strategi ini dapat diaplikasikan untuk permasalahan-permasalahan yang tidak kompleks di segi ruang dan waktu. Dengan menerapkan algoritma bertipe *brute force* pada persoalan-persoalan tersebut, jawaban akan disuguhkan dengan cepat mengingat kemampuan komputer saat ini. Akan tetapi, apabila digunakan dalam permasalahan kompleks, maka daya dan waktu yang dikerahkan akan jauh lebih besar dan cenderung sia-sia.

Dalam konteks penyelesaian permainan IQ Puzzler pro, algoritma yang diterapkan dalam program ini menjamin sebuah solusi yang tepat. Akan tetapi, masih banyak optimisasi dan pengembangan yang dapat diaplikasikan ke dalam algoritma ini agar lebih sangkil. Kompleksitas waktu dan ruang sudah seharusnya menjadi prioritas bagi para pengembang perangkat lunak. Program yang mampu menyelesaikan masalah dengan efektif dan efisien adalah program yang baik dan layak untuk digunakan.

## Lampiran

### Referensi :

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/stima24-25.htm>

### Tautan Repository :

[https://github.com/ivan-wirawan/Tucil1\\_13523046](https://github.com/ivan-wirawan/Tucil1_13523046)

### Tabel Spesifikasi Program :

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