

Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force



Disusun oleh:

13523153 – Muhammad Farrel Wibowo

INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

Bab I.....	3
Deskripsi Masalah dan Algoritma.....	3
1.1 Algoritma Brute Force.....	3
1.2 IQ Puzzler Pro	3
1.3 Algoritma Penyelesaian IQ Puzzler Pro dengan pendekatan Brute Force.....	3
1.4 Pseudocode	5
.....	8
Bab II	9
Source Code Program	9
2.1. Repository Program	9
2.2. Source Code Program	9
.....	13
Bab IV Masukan dan Luaran Program.....	15
Bab V Lampiran.....	20

Bab I

Deskripsi Masalah dan Algoritma

1.1 Algoritma Brute Force

Algoritma Brute Force adalah sebuah pendekatan lurus dan langsung dalam menyelesaikan suatu persoalan. Metode ini bekerja dengan mencoba semua kemungkinan solusi hingga menemukan jawaban yang benar. Pendekatan ini dikenal karena kesederhanaannya dalam implementasi, di mana algoritma secara sistematis dan eksplisit mengeksplorasi semua opsi yang tersedia tanpa menggunakan teknik optimasi atau heuristik. Brute force memecahkan masalah dengan cara yang jelas, mudah dipahami, dan sering kali dapat langsung ditulis berdasarkan pernyataan masalah atau konsep yang terlibat dalam penyelesaiannya. Pendekatan ini cocok digunakan ketika jumlah kemungkinan solusi masih dalam batas yang wajar, namun bisa menjadi tidak efisien untuk persoalan dengan ruang pencarian yang sangat besar, karena waktu komputasi yang meningkat secara eksponensial. Meskipun demikian, brute force sering kali menjadi pendekatan pertama yang digunakan sebelum metode yang lebih efisien dikembangkan.

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 penting dari permainan IQ Puzzler Pro terdiri dari: 1. Board (Papan) – Board merupakan komponen utama yang menjadi tujuan permainan dimana pemain harus mampu mengisi seluruh area papan menggunakan blok-blok yang telah disediakan. 2. Blok/Piece – Blok adalah komponen yang digunakan pemain untuk mengisi papan kosong hingga terisi penuh. Setiap blok memiliki bentuk yang unik dan semua blok harus digunakan untuk menyelesaikan puzzle.

1.3 Algoritma Penyelesaian IQ Puzzler Pro dengan pendekatan Brute Force

1. Pengguna memasukkan nama file yang berisi konfigurasi permainan. File ini akan dibaca menggunakan fungsi `filemanager.readFile(namaFile)`, yang akan mengembalikan data berupa ukuran papan ($N \times M$) dan daftar blok yang akan ditempatkan.
2. Jika file berhasil dibaca, program akan menampilkan ukuran papan permainan ($N \times M$) serta jumlah blok yang harus ditempatkan. Jika file tidak ditemukan atau formatnya salah, program akan memberikan peringatan kepada pengguna. Program kemudian akan membaca daftar blok yang kemudian akan mengonversi setiap blok menjadi array karakter (`char[][]`).
3. Program akan mencetak daftar blok yang telah diproses agar pengguna dapat melihat apakah blok yang akan digunakan sudah benar.
4. Program akan mencari semua kemungkinan penempatan blok pada papan permainan dengan metode brute force dan backtracking. Misalnya, jika sebuah blok dapat ditempatkan di beberapa posisi pada papan, maka program akan mencoba posisi pertama terlebih dahulu. Jika solusi tidak ditemukan dengan posisi tersebut, maka program akan melakukan backtracking

dan mencoba posisi lainnya hingga menemukan konfigurasi yang valid atau menyimpulkan bahwa tidak ada solusi yang mungkin.

5. program akan menghasilkan semua kemungkinan rotasi dan pencerminan blok. Misalnya, jika sebuah blok dapat diletakkan dalam posisi tanpa rotasi, rotasi 90° , 180° , 270° atau pencerminan horizontal, maka program akan mencoba setiap kemungkinan tersebut satu per satu. Jika blok dalam suatu orientasi tidak dapat ditempatkan, maka program akan mencoba orientasi lainnya sebelum beralih ke posisi berikutnya di papan.
6. Untuk menentukan apakah suatu blok dapat ditempatkan, program akan mengecek apakah posisi tersebut bertabrakan dengan blok lain atau keluar dari batas papan. Jika blok dapat ditempatkan, maka program akan melanjutkan pencarian untuk blok berikutnya hingga semua blok berhasil ditempatkan atau semua kemungkinan telah dicoba tanpa menemukan solusi.
7. program akan menghapus blok terakhir yang ditempatkan sebelum melanjutkan pencarian. Jika sebuah blok telah ditempatkan tetapi konfigurasi tidak menghasilkan solusi yang valid, maka program akan melakukan backtracking, menghapus blok tersebut, dan mencoba menempatkannya di posisi lain. Proses ini akan terus berulang hingga semua kemungkinan telah dieksplorasi.
8. program akan menampilkan solusi dalam bentuk papan permainan yang telah diisi dengan semua blok yang berhasil ditempatkan. Solusi ini akan ditampilkan di terminal dengan warna yang berbeda untuk setiap blok, serta dapat disimpan dalam bentuk file teks (.txt)

1.4 Pseudocode

```
FUNCTION solvePuzzle(blockIndex):  
    IF blockIndex == total_blocks THEN  
        IF isBoardFull() THEN  
            RETURN true // Semua blok telah ditempatkan, solusi ditemukan  
        END IF  
        RETURN false // Papan belum penuh, backtracking  
    END IF  
  
    block ← blocks[blockIndex] // Ambil blok saat ini  
  
    FOR row FROM 0 TO (N - block.height):  
        FOR col FROM 0 TO (M - block.width):  
            FOR each transformedBlock IN generateTransformations(block):  
                IF canPlaceBlock(board, transformedBlock, row, col) THEN  
                    blockChar ← getChar(transformedBlock)  
                    placeBlock(board, transformedBlock, row, col, blockChar)  
                    attemptCount ← attemptCount + 1 // Hitung percobaan  
  
                    IF solvePuzzle(blockIndex + 1) THEN  
                        RETURN true // Solusi ditemukan  
                    END IF  
  
                    removeBlock(board, transformedBlock, row, col) // Backtrack  
                END IF  
            END FOR  
        END FOR  
    END FOR  
  
    RETURN false // Jika semua kemungkinan gagal, kembali ke blok sebelumnya  
END FUNCTION
```

```

FUNCTION canPlaceBlock(board, block, row, col):
    IF row + block.height > board.height OR col + block.width > board.width THEN
        RETURN false // Cegah keluar batas
    END IF

    FOR i FROM 0 TO block.height:
        FOR j FROM 0 TO block.width:
            IF block[i][j] != '_' AND board[row + i][col + j] != '_' THEN
                RETURN false // Bertabrakan dengan blok lain
            END IF
        END FOR
    END FOR
    RETURN true
END FUNCTION

FUNCTION placeBlock(board, block, row, col, blockChar):
    FOR i FROM 0 TO block.height:
        FOR j FROM 0 TO block.width:
            IF block[i][j] != '_' THEN
                board[row + i][col + j] ← blockChar
            END IF
        END FOR
    END FOR
END FUNCTION

FUNCTION removeBlock(board, block, row, col):
    FOR i FROM 0 TO block.height:
        FOR j FROM 0 TO block.width:
            IF block[i][j] != '_' THEN
                board[row + i][col + j] ← '_'
            END IF
        END FOR
    END FOR
END FUNCTION

```

```

FUNCTION isBoardFull():
    FOR i FROM 0 TO N:
        FOR j FROM 0 TO M:
            IF board[i][j] == '_' THEN
                RETURN false // Masih ada ruang kosong
            END IF
        END FOR
    END FOR
    RETURN true
END FUNCTION

FUNCTION generateTransformations(piece):
    transformations ← []
    currentPiece ← piece
    FOR i FROM 0 TO 3: // Rotasi 0°, 90°, 180°, 270°
        IF currentPiece NOT IN transformations THEN
            ADD currentPiece TO transformations
        END IF
        currentPiece ← rotate(currentPiece)
    END FOR
    mirroredPiece ← flip(piece)
    FOR i FROM 0 TO 3:
        IF mirroredPiece NOT IN transformations THEN
            ADD mirroredPiece TO transformations
        END IF
        mirroredPiece ← rotate(mirroredPiece)
    END FOR

    RETURN transformations
END FUNCTION

FUNCTION flip(block):
    row ← length of block
    col ← length of block[0]
    flipped ← 2D array of size [row][col]

    FOR i FROM 0 TO row - 1:
        FOR j FROM 0 TO col - 1:
            flipped[i][col - 1 - j] ← block[i][j] // Pencerminan horizontal
        END FOR
    END FOR
    RETURN flipped
END FUNCTION

```

```
FUNCTION rotate(block):  
    row ← length of block  
    col ← length of block[0]  
    rotated ← 2D array of size [col][row]  
  
    FOR i FROM 0 TO row - 1:  
        FOR j FROM 0 TO col - 1:  
            rotated[j][row - 1 - i] ← block[i][j] // Rotasi 90° searah jarum jam  
        END FOR  
    END FOR  
  
    RETURN rotated  
END FUNCTION
```


Bab II

Source Code Program

2.1. Repository Program

https://github.com/faawibowo/Tucil1_13523153.git

2.2. Source Code Program

2.2.1 app.java

```
1  import java.util.List;
2  import java.util.Scanner;
3
4  import module.filemanager;
5  import module.block;
6
7  public class App {
8      private static char[][] board;
9      private static List<char[][]> blocks;
10     private static int N, M; // Ukuran papan
11     private static int attemptCount = 0; // Jumlah percobaan
12
13     public static void main(String[] args) throws Exception {
14         Scanner input = new Scanner(System.in);
15         // while (true) {
16             System.out.println("Welcome to IQ Puzzler Solver!");
17             System.out.println("Masukkan Nama File: ");
18             String namaFile = input.nextLine();
19             String data = filemanager.readFile(namaFile);
20             N = filemanager.readN(data);
21             M = filemanager.readM(data);
22
23             System.out.println("M:" + filemanager.readM(data));
24             System.out.println("N:" + filemanager.readN(data));
25             System.out.println("P:" + filemanager.readP(data));
26
27             board = new char[filemanager.readM(data)][filemanager.readN(data)];
28             for (int i = 0; i < filemanager.readM(data); i++) {
29                 for (int j = 0; j < filemanager.readN(data); j++) {
30                     board[i][j] = '_';
31                 }
32             }
33
34             List<char[][]> rawblocks = filemanager.parseBlocks(data);
35
36             System.out.println("List Blok :");
37             for (int i = 0; i < rawblocks.size(); i++) {
38                 char[][] block = rawblocks.get(i);
39                 char x = filemanager.findFirstNonSpaceCharacter(block[0]);
40                 System.out.println("Blok " + (char) (x) + ":");
41                 filemanager.printBlock(rawblocks.get(i));
42                 System.out.println();
43             }
44
45             blocks = filemanager.convertBlock(rawblocks);
46             long startTime = System.nanoTime();
47
48             boolean foundSolution = solvePuzzle(0);
49
50             long endTime = System.nanoTime();
51             double executionTimeMs = (endTime - startTime) / 1_000_000.0;
52
53             // **Cetak hasil**
54             if (foundSolution) {
55                 System.out.println("Solusi ditemukan:");
56                 printColorBoard();
57             } else {
58                 System.out.println("Tidak ada solusi.");
59             }
60
61             System.out.println("Total percobaan: " + attemptCount);
62             System.out.printf("Waktu eksekusi: %.3f ms\n", executionTimeMs);
63
64             System.out.println("Apakah ingin save file? (y/n)");
65             String save = input.nextLine();
66             if (save.equals("y")) {
67                 System.out.println("Masukkan nama file: ");
68                 String saveFile = input.nextLine();
69                 String BoardData = filemanager.board2String(board);
70                 filemanager.saveFile(saveFile, BoardData);
71             }
72     }
```

```

1 public static boolean solvePuzzle(int pieceIndex) {
2     if (pieceIndex == blocks.size()) {
3         if (isBoardFull()) {
4             return true;
5         }
6         return false; // Jika papan belum penuh, backtrack
7     }
8
9     char[][] piece = blocks.get(pieceIndex);
10
11     // Coba semua posisi di papan
12     for (int row = 0; row <= N - piece.length; row++) {
13         for (int col = 0; col <= M - piece[0].length; col++) {
14             for (char[][] transformedPiece : block.generateTransformations(piece)) {
15
16                 if (canPlaceBlock(board, transformedPiece, row, col)) {
17                     char blockChar = filemanager.getChar(transformedPiece);
18                     placeBlock(board, transformedPiece, row, col, blockChar);
19                     attemptCount++;
20                     if (solvePuzzle(pieceIndex + 1)) {
21                         return true;
22                     }
23
24                     removeBlock(board, transformedPiece, row, col);
25
26                 }
27             }
28         }
29     }
30
31     return false; // Jika semua kemungkinan gagal
32 }
33
34 public static boolean canPlaceBlock(char[][] board, char[][] piece, int row, int col) {
35     if (row + piece.length > board.length || col + piece[0].length > board[0].length) {
36         return false;
37     }
38
39     for (int i = 0; i < piece.length; i++) {
40         for (int j = 0; j < piece[0].length; j++) {
41             if (piece[i][j] != '_' && board[row + i][col + j] != '_') {
42                 return false;
43             }
44         }
45     }
46     return true;
47 }
48
49 public static void placeBlock(char[][] board, char[][] piece, int row, int col, char blockChar) {
50     for (int i = 0; i < piece.length; i++) {
51         for (int j = 0; j < piece[0].length; j++) {
52             if (piece[i][j] != '_') {
53                 board[row + i][col + j] = blockChar;
54             }
55         }
56     }
57 }
58

```

```

1 public static void removeBlock(char[][] board, char[][] piece, int row, int col) {
2     for (int i = 0; i < piece.length; i++) {
3         for (int j = 0; j < piece[0].length; j++) {
4             if (piece[i][j] != '_') {
5                 board[row + i][col + j] = '_';
6             }
7         }
8     }
9 }
10
11 public static void printBoard() {
12     for (char[] row : board) {
13         System.out.println(new String(row));
14     }
15     System.out.println();
16 }
17
18 public static void printColorBoard() {
19     for (char[] row : board) {
20         for (char cell : row) {
21             if (cell == '_') {
22                 System.out.print(" ");
23             } else {
24                 int index = cell - 'A';
25                 if (index >= 0 && index < 26) {
26                     System.out.print(COLORS[index] + cell + RESET + " ");
27                 } else {
28                     System.out.print(cell + " ");
29                 }
30             }
31         }
32         System.out.println();
33     }
34     System.out.println();
35 }
36
37 public static boolean isBoardFull() {
38     for (int i = 0; i < N; i++) {
39         for (int j = 0; j < M; j++) {
40             if (board[i][j] == '_') {
41                 return false;
42             }
43         }
44     }
45     return true;
46 }
47
48 public static final String[] COLORS = {
49     "\u001B[31m", // A - Merah
50     "\u001B[32m", // B - Hijau
51     "\u001B[33m", // C - Kuning
52     "\u001B[34m", // D - Biru
53     "\u001B[35m", // E - Magenta
54     "\u001B[36m", // F - Cyan
55     "\u001B[91m", // G - Terang Merah
56     "\u001B[92m", // H - Terang Hijau
57     "\u001B[93m", // I - Terang Kuning
58     "\u001B[94m", // J - Terang Biru
59     "\u001B[95m", // K - Terang Magenta
60     "\u001B[96m", // L - Terang Cyan
61     "\u001B[90m", // M - Abu-abu Gelap
62     "\u001B[97m", // N - Putih Terang
63     "\u001B[38;5;208m", // O - Oranye
64     "\u001B[38;5;202m", // P - Merah Bata
65     "\u001B[38;5;82m", // Q - Hijau Neon
66     "\u001B[38;5;226m", // R - Kuning Neon
67     "\u001B[38;5;21m", // S - Biru Laut
68     "\u001B[38;5;201m", // T - Pink
69     "\u001B[38;5;123m", // U - Biru Tosca
70     "\u001B[38;5;214m", // V - Kuning Keemasan
71     "\u001B[38;5;99m", // W - Ungu Tua
72     "\u001B[38;5;124m", // X - Merah Darah
73     "\u001B[38;5;190m",
74     "\u001B[38;5;50m"
75 };
76 public static final String RESET = "\u001B[0m";
77
78 }

```

2.2.2. FileManager.java

```
1 package module;
2
3 import java.io.File;
4 import java.io.FileNotFoundException;
5 import java.util.ArrayList;
6 import java.util.List;
7 import java.util.Scanner;
8 import java.util.regex.Matcher;
9 import java.util.regex.Pattern;
10
11 public class filemanager {
12     private static String inputPath = "/test/input/";
13
14     public static int readM(String data) {
15         return extractNumber(data, 0);
16     }
17
18     public static int readN(String data) {
19         return extractNumber(data, 1);
20     }
21
22     public static int readP(String data) {
23         return extractNumber(data, 2);
24     }
25
26     private static int extractNumber(String data, int index) {
27         Pattern pattern = Pattern.compile("\\d+");
28         Matcher matcher = pattern.matcher(data);
29         ArrayList<Integer> numbers = new ArrayList<>();
30
31         while (matcher.find()) {
32             numbers.add(Integer.parseInt(matcher.group()));
33         }
34
35         return (numbers.size() > index) ? numbers.get(index) : -1;
36     }
37
38     public static String readFile(String nama) {
39         StringBuilder result = new StringBuilder();
40         try {
41             String curDir = System.getProperty("user.dir");
42             File myObj = new File(curDir + inputPath + nama + ".txt");
43             Scanner myReader = new Scanner(myObj);
44             while (myReader.hasNextLine()) {
45                 String data = myReader.nextLine();
46                 result.append(data).append("\n");
47             }
48             myReader.close();
49         } catch (FileNotFoundException e) {
50             System.out.println("File Tidak Dapat Ditemukan.");
51         }
52         return result.toString();
53     }
54
55     public static List<char[][]> parseBlocks(String input) {
56         String[] lines = input.split("\n");
57         List<char[][]> blockList = new ArrayList<>();
58         int i = 2;
59
60         while (i < lines.length) {
61             List<char[]> tempBlock = new ArrayList<>();
62
63             tempBlock.add(lines[i].toCharArray());
64             i++;
65
66             while (i < lines.length && isSameBlock(tempBlock, lines[i])) {
67                 tempBlock.add(lines[i].toCharArray());
68                 i++;
69             }
70
71             char[][] block = new char[tempBlock.size()][];
72             for (int j = 0; j < tempBlock.size(); j++) {
73                 block[j] = tempBlock.get(j);
74             }
75             blockList.add(block);
76         }
77
78         return blockList;
79     }
80 }
```

```

1 public static boolean isSameBlock(List<char[]> tempBlock, String newLine) {
2     if (tempBlock.isEmpty() || newLine.isEmpty()) {
3         return false;
4     }
5     char[] firstBlockLine = tempBlock.get(0);
6     char firstChar = findFirstNonSpaceCharacter(firstBlockLine);
7     char newFirstChar = findFirstNonSpaceCharacter(newLine.toCharArray());
8
9     return firstChar == newFirstChar;
10 }
11
12 public static char findFirstNonSpaceCharacter(char[] arr) {
13     for (char c : arr) {
14         if (c != ' ')
15             return c;
16     }
17     return ' ';
18 }
19
20 public static char getChar(char[][] block) {
21     for (char[] row : block) {
22         for (char c : row) {
23             if (c != '_') {
24                 return c;
25             }
26         }
27     }
28     return '_';
29 }
30
31 public static void printBlock(char[][] block) {
32     for (char[] row : block) {
33         System.out.println(new String(row));
34     }
35 }
36
37 public static int maxLength(char[][] blocks) {
38     int max = 0;
39     for (char[] block : blocks) {
40         if (block.length > max) {
41             max = block.length;
42         }
43     }
44     return max;
45 }
46
47 public static List<char[][]> convertBlock(List<char[][]> blocks) {
48     List<char[][]> newBlocks = new ArrayList<>();
49
50     for (char[][] block : blocks) {
51         int maxLen = maxLength(block);
52         char[][] newBlock = new char[block.length][maxLen];
53         for (int i = 0; i < block.length; i++) {
54             System.arraycopy(block[i], 0, newBlock[i], 0, block[i].length);
55             for (int j = block[i].length; j < maxLen; j++) {
56                 newBlock[i][j] = '_';
57             }
58             for (int j = 0; j < maxLen; j++) {
59                 if (newBlock[i][j] == ' ') {
60                     newBlock[i][j] = '_';
61                 }
62             }
63         }
64         newBlocks.add(newBlock);
65     }
66     return newBlocks;
67 }
68
69 public static void saveFile(String fileName, String data) {
70     try {
71         String curDir = System.getProperty("user.dir");
72         File myObj = new File(curDir + "/test/output/" + fileName + ".txt");
73         myObj.createNewFile();
74         java.io.FileWriter myWriter = new java.io.FileWriter(myObj);
75         myWriter.write(data);
76         myWriter.close();
77     } catch (Exception e) {
78         System.out.println("An error occurred.");
79         e.printStackTrace();
80     }
81 }
82
83 public static String board2String(char[][] board) {
84     StringBuilder sb = new StringBuilder();
85     for (char[] row : board) {
86         sb.append(row).append("\n");
87     }
88     return sb.toString();
89 }
90 }

```

2.2.3. block.java

```
1  package module;
2
3  import java.util.ArrayList;
4  import java.util.Arrays;
5  import java.util.HashSet;
6  import java.util.List;
7  import java.util.Set;
8
9  public class block {
10     public static char[][] rotate(char[][] block) {
11         int row = block.length;
12         int col = block[0].length;
13         char[][] rotated = new char[col][row];
14         for (int i = 0; i < row; i++) {
15             for (int j = 0; j < col; j++) {
16                 rotated[j][row - 1 - i] = block[i][j];
17             }
18         }
19         return rotated;
20     }
21
22     public static char[][] flip(char[][] block) {
23         int row = block.length;
24         int col = block[0].length;
25         char[][] flipped = new char[row][col];
26         for (int i = 0; i < row; i++) {
27             for (int j = 0; j < col; j++) {
28                 flipped[i][col - 1 - j] = block[i][j];
29             }
30         }
31         return flipped;
32     }
33
34     // Menghasilkan semua kemungkinan rotasi dan pencerminan untuk suatu blok
35     public static List<char[][]> generateTransformations(char[][] piece) {
36         List<char[][]> transformations = new ArrayList<>();
37         Set<String> seen = new HashSet<>(); // Menghindari duplikasi
38
39         char[][] currentPiece = piece;
40         for (int i = 0; i < 4; i++) { // Coba rotasi 0°, 90°, 180°, 270°
41             if (seen.add(Arrays.deepToString(currentPiece))) {
42                 transformations.add(currentPiece);
43             }
44             currentPiece = rotate(currentPiece);
45         }
46
47         currentPiece = flip(piece);
48         for (int i = 0; i < 4; i++) {
49             if (seen.add(Arrays.deepToString(currentPiece))) {
50                 transformations.add(currentPiece);
51             }
52             currentPiece = rotate(currentPiece);
53         }
54
55         return transformations;
56     }
57 }
58
59
```

Bab IV Masukan dan Luaran Program

TC1

```
test > input > tc1.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 XXX
18 |
```

Solusi ditemukan:


```
A B B C C
A A B C E
F D D E E
F F D E E
F F X X X
```

Total percobaan: 561


Waktu eksekusi: 37.744 ms

Apakah ingin save file? (y/n)

TC2

```
test > input >  tc2.txt
 1  5 5 25
 2  DEFAULT
 3  A
 4  B
 5  C
 6  D
 7  E
 8  F
 9  G
10  H
11  I
12  J
13  K
14  L
15  M
16  N
17  O
18  P
19  Q
20  R
21  S
22  T
23  U
24  V
25  W
26  X
27  Y
28
```

```
Solusi ditemukan:
A B C D E
F G H I J
K L M N O
P Q R S T
U V W X Y

Total percobaan: 25
Waktu eksekusi: 3.050 ms
Apakah ingin save file? (y/n)

```


TC3

```
test > input > tc3.txt
1 4 4 3
2 DEFAULT
3 A
4 AA
5 BB
6 B
7 CC
```

```
Tidak ada solusi.
Total percobaan: 6110
Waktu eksekusi: 34.656 ms
Apakah ingin save file? (y/n)

```

TC4

```
test > input > tc4.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
18
```

```
Solusi ditemukan:  
A B B C C  
A A B C E  
F D D E E  
F F D E E  
F F G G G  
  
Total percobaan: 107301  
Waktu eksekusi: 2244.396 ms  
Apakah ingin save file? (y/n)
```

TC5

```
test > input > tc5.txt  
1 4 4 5  
2 DEFAULT  
3 A  
4 AA  
5 B  
6 BB  
7 C  
8 CC  
9 DDDD  
10 EEEE  
11 |
```

```
Tidak ada solusi.  
Total percobaan: 7476  
Waktu eksekusi: 110.221 ms  
Apakah ingin save file? (y/n)  
|
```

TC6

```
test > input > tc6.txt
1 6 6 8
2 DEFAULT
3 A
4 AA
5 AAA
6 B
7 BB
8 BBB
9 C
10 CC
11 CCC
12 D
13 DD
14 DDD
15 EEE
16 EEE
17 EEE
18
```

```
Tidak ada solusi.
Total percobaan: 7476
Waktu eksekusi: 110.221 ms
Apakah ingin save file? (y/n)

```

TC7

```
test > input > tc7.txt
1 3 3 5
2 DEFAULT
3 A
4 AA
5 BB
6 CC
7 CC
8 D
9 DD
10 E
11 EE
12 F
13 FF
14
15
```

```
Tidak ada solusi.
Total percobaan: 78
Waktu eksekusi: 4.161 ms
Apakah ingin save file? (y/n)

```

v

Bab V Lampiran

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