LAPORAN TUGAS KECIL 1

IF2211 - STRATEGI ALGORITMA

Disusun oleh :

Ferdin Arsenarendra Purtadi – 13523117

Program Studi Teknik Informatika

Institut Teknologi Bandung

2025

1. Deskripsi Tugas

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:

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

b) 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.

Tugas anda adalah menemukan cukup satu solusi dari permainan IQ Puzzler Pro dengan menggunakan algoritma Brute Force, atau menampilkan bahwa solusi tidak ditemukan jika tidak ada solusi yang mungkin dari puzzle.

2. Spesifikasi tugas

Spesifikasi Wajib

- Buatlah program sederhana dalam bahasa Java yang mengimplementasikan algoritma Brute Force untuk mencari solusi dalam permainan IQ Puzzler Pro.
- Algoritma brute force yang diimplementasikan harus bersifat “murni”, tidak boleh memanfaatkan heuristik.
- Papan yang perlu diisi mulanya akan selalu kosong.
- Sebuah blok puzzle bisa saja dirotasi maupun dicerminkan sebelum diletakan pada papan.
- Input:

  program akan memberikan pengguna sebuah prompt untuk memilih file test case berekstensi .txt, kemudian program membaca file test case tersebut yang berisi :

  1. Dimensi Papan terdiri atas dua buah variabel N dan M yang membentuk papan berdimensi NxM.
  2. Banyak blok puzzle direpresentasikan oleh variabel integer P.
  3. Jenis kasus sebuah variabel string S yang digunakan untuk mengidentifikasi kasus konfigurasi, hanya mungkin bernilai salah satu diantara DEFAULT/CUSTOM/PYRAMID.

4. Bentuk blok puzzle yang dilambangkan oleh konfigurasi Character berupa huruf. Akan ada P buah blok puzzle berbeda yang dibentuk oleh P buah huruf berbeda. Character yang digunakan adalah huruf A-Z dalam kapital.

```
N M P
S
puzzle_1_shape
puzzle_2_shape
…
puzzle_P_shape
```

Output:

1. Tampilkan konfigurasi blok puzzle yang berhasil mengisi papan. Gunakan print berwarna untuk menunjukkan blok puzzle dengan jelas. Pastikan setiap blok puzzle berbeda memiliki warna berbeda. Beri tahu pengguna apabila puzzle tidak memiliki solusi.
2. Waktu eksekusi program dalam milisecond (tidak termasuk waktu membaca masukan dan menyimpan solusi, cukup waktu pencarian oleh algoritma).
3. Banyak kasus atau jumlah iterasi yang ditinjau oleh algoritma brute force.
4. Prompt untuk menyimpan solusi dalam sebuah berkas berekstensi .txt (Struktur untuk file output dibebaskan).

3. Program

3.1 Algoritma Brute Force

Dalam mengerjakan tugas ini, digunakan algoritma brute force untuk memecahkan penyusunan puzzle.

Program pada awalnya melakukan pembacaan file yang menerima informasi ukuran papan, banyak piece puzzle, dan bentuk piece. Program kemudian membuat 8 macam orientasi dari tiap bentuk piece, 8 orientasi ini meliputi 1 bentuk awal, 3 bentuk hasil rotasi, serta 4 hasil pencerminan dari bentuk asli dan rotasinya. Piece disimpan dalam sebuah array of Piece, sedangkan orientasinya disimpan dalam matrix of character yang masuk dalam kelas Piece. Berikut algoritma pembuatan orientasi dari masing-masing piece :

```
Function makeOrientations(piece, shape):
    height ← size(shape)
    width ← 0
    for row in shape
```

```
            width ← MAX(width, length(row))
        end for

        for i ← 0 to height-1
            for j ← 0 to length(shape[i])-1
                piece.orientations[0][i][j] ← shape[i][j]
            end for
        end for

        piece.height[0] ← height
        piece.width[0] ← width
        piece.validOrientation[0] ← True

        for base ← 0 to 4 step 4
            for rot ← 1 to 3
                curr ← base + rot
                prev ← base + rot - 1
                newHeight ← piece.width[prev]
                newWidth ← piece.height[prev]

                for i ← 0 to piece.height[prev]-1
                    for j ← 0 to piece.width[prev]-1
                        piece.orientations[curr][j][newWidth - 1 - i] ←
piece.orientations[prev][i][j]
                    end for
                end for

                piece.height[curr] ← newHeight
                piece.width[curr] ← newWidth
                piece.validOrientation[curr] ← True
            end for
            if base = 0 then
                for i ← 0 to height-1
                    for j ← 0 to width-1
                        piece.orientations[4][i][width - 1 - j] ←
piece.orientations[0][i][j]
                    end for
                end for

                piece.height[4] ← height
                piece.width[4] ← width
                piece.validOrientation[4] ← True
            end if
        end for

        for i ← 0 to MAX_ORIENTATIONS-1
            if not piece.validOrientation[i] then continue
            for j ← 0 TO i-1
                if not piece.validOrientation[j] then continue
                if isSameOrientation(piece, i, j) then
                    piece.validOrientation[i] ← False
                    break
                end if
```

```
            end for
        end for
end function
```

Selanjutnya program menginisialisasi papan permainan puzzle. Program mengisinya dengan titik '.' sebagai inisiasi papan. Berikut algoritmanya :

```
function initializeBoard():
    for i ← 0 to N-1
        for j ← 0 to M-1
            board[i][j] ← '.'
        end for
    end for
end function
```

Setelah papan diinisiasi, program akan mulai melakukan pemecahan masalah penyusunan piece puzzle.

Fungsi solvePuzzle memanfaatkan algoritma brute force dalam menyusun tiap piecenya. Pertama fungsi akan mengecek apakah tiap piece telah digunakan, informasi apakah piece telah digunakan atau belum disimpan dalam sebuah array boolean 'used[]', jika ditemukan satu elemen dalam array tersebut yang bernilai False maka program akan melanjutkan ke pengecekan apakah papan sudah terisi semua oleh piece (penuh). Berikut algoritma pengecekan penggunaan piece :

```
allUsed ← TRUE
for isUsed in used
    if not isUsed then
        allUsed ← False
        break
    end if
end for
```

Selanjutnya program akan mengecek apakah semua titik di papan sudah terisi oleh semua piece puzzle. Jika pada papan masih ditemukan titik, maka papan masih belum penuh sehingga program akan mengisinya dan menandainya dengan startX dan startY. Berikut algoritmanya:

```
startX, startY ← -1
if allUsed then
    boardFull ← True
    for  i ← 0 to N-1
        for  j ← 0 to M-1
            if board[i][j] = '.'  then
                startX ← i
                startY ← j
                boardFull ← False
                break
```

```
                end if
            end for
        end for
        return boardFull
end if
```

Jika masih ditemukan bagian kosong pada papan, maka program akan mencoba satu-satu piece yang belum digunakan beserta semua orientasinya untuk dipasangkan ke papan, dimulai dari penanda yang telah dibuat sebelumnya (startX, startY). Berikut algoritmanya :

```
    for p ← 0 to P-1
        if used[p] then continue
        for o ← 0 to MAX_ORIENTATIONS-1
            if not pieces[p].validOrientation[o] then continue

            casesChecked ← casesChecked + 1

            if canPlacePiece(startX, startY, pieces[p], o) then
                used[p] ← True
                placePiece(startX, startY, pieces[p], o, pieces[p].letter)

                if solvePuzzle() then return True

                used[p] ← False
                placePiece(startX, startY, pieces[p], o, '.')
            end if
        end for
end for
```

Pada bagian ini, program akan memeriksa setiap piece yang disimpan dalam array boolean ‘used’, program akan mencari piece yang belum digunakan. jika ditemukan piece yang belum digunakan maka program akan mengecek apakah ada orientasi dari piece tersebut yang bisa dipasangkan dalam papan. Pada bagian ini kasus yang diperiksa bertambah 1. Pengecekkan dilakukan dengan menggunakan fungsi canPlacePiece dengan algoritma sebagai berikut :

```
    function canPlacePiece(x, y, piece, orientation):
        if not piece.validOrientation[orientation] then
            return False
        end if

        for i ← 0 to piece.height[orientation]-1
            for j ← 0 to piece.width[orientation]-1
                if piece.orientations[orientation][i][j] = piece.letter then
                    if x + i >= N or y + j >= M or board[x + i][y + j] ≠ '.' then
                        return false
                    end if
```

```
                    end if
                end for
    end for
```

Jika orientasi dari piece ini bisa dipasangkan pada papan, maka piece akan dipasangkan ke papan dan kondisi piece tersebut dianggap sudah terpakai. Kemudian program akan mencoba menyelesaikan puzzle dengan piece baru yang telah diletakkan. Jika pada akhirnya puzzle tidak terselesaikan dengan dipasangnya piece ini, maka program akan backtrack ke piece yang baru saja terpasang lalu menjadikan kondisi piece tersebut kembali menjadi tidak terpakai dan mengganti bagian yang sudah terisi sebelumnya dengan titik (mengosongkan kembali bagian piece yang sebelumnya sudah dipasang) kemudian program akan mengecek orientasi berikutnya hingga semua piece puzzle telah dicek.

Saat semua pengecekan telah selesai dan ditemukan bahwa papan belum penuh atau ada piece yang belum digunakan, maka fungsi solvePuzzle akan mengembalikan nilai False. Jika semua piece telah digunakan dan papan penuh maka fungsi akan mengembalikan nilai True.

Setelah semua algoritma pemasangan piece telah dilakukan, maka selanjutnya program akan menampilkan lama waktu penyelesaian puzzle (atau waktu hingga semua kasus telah dicek) dan banyaknya kasus yang telah dicek ke pengguna. Jika ditemukan konfigurasi yang tepatt, maka program akan menampilkan papan beserta isinya.

3.2 Source Code

3.2.1 Kode keseluruhan

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

public class PuzzleGame {
//Algoritma Brute Force
//Main Program
}
```

3.2.2 Kelas Piece

```
        static class Piece {
            char letter;
            char[][][]          orientations          =          new
```

```
        char[MAX_ORIENTATIONS][MAX_SHAPE][MAX_SHAPE];
            int[] height = new int[MAX_ORIENTATIONS];
            int[] width = new int[MAX_ORIENTATIONS];
            boolean[] validOrientation = new
        boolean[MAX_ORIENTATIONS];
    }
```

3.2.3 Inisialisasi Papan Puzzle

```
        static void initializeBoard() {
            board = new char[N][M];
            for (int i = 0; i < N; i++) {
                for (int j = 0; j < M; j++) {
                    board[i][j] = '.';
                }
            }
        }
```

3.2.4 Cek Apakah Potongan Puzzle Bisa Diletakkan

```
        static boolean canPlacePiece(int x, int y, Piece piece, int
        orientation) {
            if (!piece.validOrientation[orientation]) return false;

            for (int i = 0; i < piece.height[orientation]; i++) {
                for (int j = 0; j < piece.width[orientation]; j++) {
                    if (piece.orientations[orientation][i][j] == piece.letter) {
                        if (x + i >= N || y + j >= M || board[x + i][y + j] != '.')
        {
                            return false;
                        }
                    }
                }
            }
            return true;
    }
```

3.2.5 Meletakkan Potongan Puzzle ke Papan

```
        static void placePiece(int x, int y, Piece piece, int orientation, char
        value) {
            for (int i = 0; i < piece.height[orientation]; i++) {
                for (int j = 0; j < piece.width[orientation]; j++) {
                    if (piece.orientations[orientation][i][j] == piece.letter) {
                        board[x + i][y + j] = value;
                    }
                }
            }
    }
```

3.2.6 Membuat Orientasi dari Tiap Potongan Puzzle

```
        static void makeOrientations(Piece piece, List<String> shape) {
```

```
            int height = shape.size();
            int width = 0;
            for (String row : shape) {
               width = Math.max(width, row.length());
            }
            for (int i = 0; i < shape.size(); i++) {
               String row = shape.get(i);
               for (int j = 0; j < row.length(); j++) {
                  piece.orientations[0][i][j] = row.charAt(j);
               }
            }
            piece.height[0] = height;
            piece.width[0] = width;
            piece.validOrientation[0] = true;

            for (int base = 0; base < 8; base += 4) {
               for (int rot = 1; rot < 4; rot++) {
                  int curr = base + rot;
                  int prev = base + rot - 1;

                  int newHeight = piece.width[prev];
                  int newWidth = piece.height[prev];

                  for (int i = 0; i < piece.height[prev]; i++) {
                     for (int j = 0; j < piece.width[prev]; j++) {
                        piece.orientations[curr][j][newWidth - 1 - i] =
piece.orientations[prev][i][j];
                     }
                  }

                  piece.height[curr] = newHeight;
                  piece.width[curr] = newWidth;
                  piece.validOrientation[curr] = true;
               }

               if (base == 0) {
                  for (int i = 0; i < height; i++) {
                     for (int j = 0; j < width; j++) {
                        piece.orientations[4][i][width - 1 - j] =
                           piece.orientations[0][i][j];
                     }
                  }

                  piece.height[4] = height;
                  piece.width[4] = width;
                  piece.validOrientation[4] = true;
               }
            }

            for (int i = 0; i < MAX_ORIENTATIONS; i++) {
               if (!piece.validOrientation[i]) continue;
               for (int j = 0; j < i; j++) {
                  if (!piece.validOrientation[j]) continue;
```

```
                    if (isSameOrientation(piece, i, j)) {
                        piece.validOrientation[i] = false;
                        break;
                    }
                }
            }
        }

        static boolean isSameOrientation(Piece piece, int o1, int o2) {
            if (piece.height[o1] != piece.height[o2] || piece.width[o1] !=
    piece.width[o2])
                return false;

            for (int i = 0; i < piece.height[o1]; i++) {
                for (int j = 0; j < piece.width[o1]; j++) {
                    if (piece.orientations[o1][i][j] !=
    piece.orientations[o2][i][j])
                        return false;
                }
            }
            return true;
    }
```

3.2.7 Menyelesaikan Puzzle

```
        static boolean solvePuzzle() {
            boolean allUsed = true;
            for (boolean isUsed : used) {
                if (!isUsed) {
                    allUsed = false;
                    break;
                }
            }
            if (allUsed) {
                boolean boardFull = true;
                for (int i = 0; i < N && boardFull; i++) {
                    for (int j = 0; j < M; j++) {
                        if (board[i][j] == '.') {
                            boardFull = false;
                            break;
                        }
                    }
                }
                return boardFull;
            }
            int startX = -1, startY = -1;
            boolean found = false;
            for (int i = 0; i < N && !found; i++) {
                for (int j = 0; j < M; j++) {
                    if (board[i][j] == '.') {
                        startX = i;
                        startY = j;
                        found = true;
```

```
                        break;
                    }
                }
            }

            if (!found) return false;

            for (int p = 0; p < P; p++) {
                if (used[p]) continue;

                for (int o = 0; o < MAX_ORIENTATIONS; o++) {
                    if (!pieces[p].validOrientation[o]) continue;

                    casesChecked++;

                    if (canPlacePiece(startX, startY, pieces[p], o)) {
                        used[p] = true;
                       placePiece(startX, startY, pieces[p], o, pieces[p].letter);

                        if (solvePuzzle()) return true;

                        used[p] = false;
                        placePiece(startX, startY, pieces[p], o, '.');
                    }
                }
            }

            return false;
    }
```

3.2.8 Mencetak Papan

```
        static void printBoard() {
            Map<Character, Integer> colorMap = new HashMap<>();
            int colorIndex = 0;

            for (int i = 0; i < N; i++) {
                for (int j = 0; j < M; j++) {
                    char c = board[i][j];
                    if (c != '.') {
                        if (!colorMap.containsKey(c)) {
                         colorMap.put(c, colorIndex++ % COLORS.length);
                        }
                        System.out.print(COLORS[colorMap.get(c)] + c +
        RESET);
                    } else {
                        System.out.print(c);
                    }
                }
                System.out.println();
            }
    }
```

3.2.9 Menyimpan Solusi Puzzle

```
        static void saveSolution(String filename, long solveTime) {
            try (PrintWriter writer = new PrintWriter(new
        FileWriter("../test/" + filename))) {
                writer.println(N + " " + M + " " + P);
                writer.println();

                for (int i = 0; i < N; i++) {
                   for (int j = 0; j < M; j++) {
                      writer.print(board[i][j]);
                   }
                   writer.println();
                }
                writer.println("\n");
                writer.println("Hasil: ");
                writer.println("Waktu penyelesaian: " + solveTime + "
        ms");
                writer.println("Kasus yang diperiksa: " + casesChecked);

            } catch (IOException e) {
                System.out.println("Error saat menyimpan solusi: " +
        e.getMessage());
            }
    }
```

3.2.10 Membaca File Input

```
        static boolean readPuzzle(String filename) {
            try (BufferedReader reader = new BufferedReader(new
        FileReader("../test/" + filename))) {
                String line;

                line = reader.readLine();
                while (line != null && line.trim().isEmpty()) {
                   line = reader.readLine();
                }

                String[] dims = line.split(" ");

                if (dims.length != 3) {
                   System.out.println("Informasi puzzle tidak lengkap!
        Harap dicek!\n");
                   return false;
                }

                boolean checkIfInteger = true;
                for (String dim : dims) {
                   try {
                      Integer.parseInt(dim);
                   } catch (NumberFormatException e) {
                      checkIfInteger = false;
                      break;
                   }
```

```
            }

            if (!checkIfInteger) {
            System.out.println("Informasi Puzzle tidak dalam angka!
Coba dicek!\n");
                return false;
            }

            N = Integer.parseInt(dims[0]);
            M = Integer.parseInt(dims[1]);
            P = Integer.parseInt(dims[2]);

            if (N < 1) {
             System.out.println("Error: Jumlah baris papan tidak valid!
Setidaknya papan memiliki 1 baris!\n");
                return false;
            }
            if (M < 1) {
                System.out.println("Error: Jumlah kolom papan tidak
valid! Setidaknya kolom memiliki 1 baris!\n");
                return false;
            }
            if (P < 0 || P > 26) {
                System.out.println("Error: Banyaknya potongan puzzle
tidak valid! Harus di antara 0 sampai 26!\n");
                return false;
            }

            line = reader.readLine();
            while (line != null && line.trim().isEmpty()) {
                line = reader.readLine();
            }

            if (line == null || !line.equals("DEFAULT")) {
             System.out.println("Error: Tipe papan tidak valid! Coba
dicek!\n");
                return false;
            }

            initializeBoard();
            pieces = new Piece[P];
            used = new boolean[P];

            int filePiece = 0;
            String nextPieceFirstLine = null;

            while (true) {
                if (nextPieceFirstLine != null) {
                    line = nextPieceFirstLine;
                    nextPieceFirstLine = null;
                } else {
                    line = reader.readLine();
                }
```

```
                if (line == null) {
                    break;
                }

                if (line.trim().isEmpty()) {
                    continue;
                }

                if (filePiece >= P) {
                    System.out.println("Error: Terlalu banyak potongan
puzzle! Seharusnya hanya " + P + " potongan.\n");
                    return false;
                }

                pieces[filePiece] = new Piece();
                List<String> shape = new ArrayList<>();

                pieces[filePiece].letter = line.charAt(0);
                shape.add(line);

                while ((line = reader.readLine()) != null) {
                    if (line.trim().isEmpty()) {
                        break;
                    }

                    if (line.length() > 0 &&
Character.isLetter(line.charAt(0)) &&
                        line.charAt(0) != pieces[filePiece].letter) {
                        nextPieceFirstLine = line;
                        break;
                    }

                    shape.add(line);
                }

                makeOrientations(pieces[filePiece], shape);
                filePiece++;
            }

            if (filePiece < P) {
            System.out.println("Error: Kekurangan potongan puzzle!
Seharusnya ada " + P + " potongan, tapi hanya ada\n" + filePiece);
                return false;
            }

            System.out.println("Potongan puzzle berhasil dibaca!\n");
            return true;

        } catch (IOException e) {
            System.out.println("Error saat membaca file input: " +
e.getMessage());
            return false;
```

```
            }
    }
```

3.2.11 Main Program

```
        static void showMainMenu() {
            while (true) {
             System.out.println("====== WELCOME TO ======\n");
                System.out.println("=== IQ Puzzle Solver ===\n");
                System.out.println("1. Selesaikan Puzzle");
                System.out.println("2. Keluar\n");
                System.out.print("Pilih (1-2): ");

                String choice = scanner.nextLine();

                switch (choice) {
                    case "1":
                        playPuzzle();
                        break;
                    case "2":
                    System.out.println("Terima kasih sudah mampir!\n");
                        return;
                    default:
                        System.out.println("Opsi tidak valid! Masukan opsi
yang tersedia!\n");
                }
            }
        }

        static void playPuzzle() {
          System.out.print("Masukkan nama file puzzle lengkap
dengan .txt: ");
            String filename = scanner.nextLine();
            try {
                if (!readPuzzle(filename)) {
                    return;
                }

                System.out.println("Menyelesaikan puzzle...\n");
                long startTime = System.currentTimeMillis();
                boolean solved = solvePuzzle();
                long endTime = System.currentTimeMillis();
                long solveTime = endTime - startTime;

                if (solved) {
                    System.out.println("Puzzle Selesai!\n");
                    printBoard();
                    System.out.println("\n");
                System.out.println("Waktu penyelesaian: " + solveTime
+ " ms");
                    System.out.println("Kasus yang diperiksa: " +
casesChecked);
```

```
                System.out.print("Apakah solusi ingin disimpan?
        (ya/tidak): \n");
                String save = scanner.nextLine();
                if (save.equalsIgnoreCase("ya")) {
                   System.out.print("Enter output filename: ");
                   String outFile = scanner.nextLine();
                   if (!outFile.toLowerCase().endsWith(".txt")) {
                      outFile += ".txt";
                   }

                   saveSolution(outFile, solveTime);
                  System.out.println("Solusi tersimpan di \n" + outFile);
                   }
               } else {
               System.out.println("Puzzle tidak dapat terselesaikan!\n");
                  System.out.println("Waktu pengerjaan: " + solveTime
        + " ms");
                  System.out.println("Kasus yang diperiksa: " +
        casesChecked);
                }
            } catch (Exception e) {
               System.out.println("Error saat pemecahan puzzle: " +
        e.getMessage());
            }
         }

         public static void main(String[] args) {
            showMainMenu();
            scanner.close();
   }
```

4. Test Case

a. Kasus default

a) Input

File input default untuk puzzle solver disimpan dalam file default_tc.txt seperti berikut :

```
5 5 7
DEFAULT
A
AA
B
BB
C
CC
D
DD
EE
EE
E
FF
FF
F
GGG
```

Gambar 4.1.1 Input Test Case 1

b) Output

Berikut adalah hasil yang diberikan oleh program puzzle solver dari file input default_tc.txt yang kemudian disimpan dalam file output1.txt.

```
====== WELCOME TO ======

=== IQ Puzzle Solver ===

1. Selesaikan Puzzle
2. Keluar

Pilih (1-2): 1
Masukkan nama file puzzle lengkap dengan .txt: default_tc.txt
Potongan puzzle berhasil dibaca!

Menyelesaikan puzzle...

Puzzle Selesai!

ABBCC
AABDC
EEEDD
EEFFF
GGGFF


Waktu penyelesaian: 1 ms
Kasus yang diperiksa: 805
Apakah solusi ingin disimpan? (ya/tidak):
ya
Enter output filename: output1
Solusi tersimpan di
output1.txt
```

Gambar 4.1.2 Output Test Case 1

b. Kasus terlalu banyak potongan puzzle

a) Input

File input yang memiliki potongan puzzle yang ada dalam file lebih banyak daripada yang diberikan dalam informasi papan di line pertama file txt. File input ini disimpan dengan nama manypcs.txt seperti berikut :

```
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  HHHHH
```

Gambar 4.2.1 Input Test Case 2

b) Output

Berikut adalah hasil yang diberikan oleh program puzzle solver dari file input manypcs.txt.

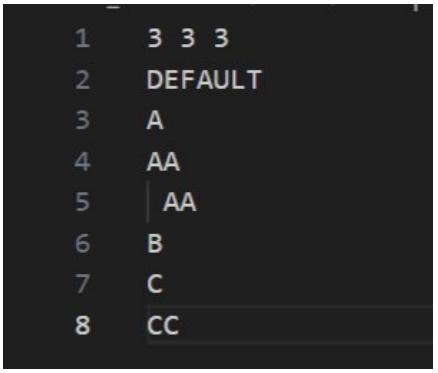
```
====== WELCOME TO ======

=== IQ Puzzle Solver ===

1. Selesaikan Puzzle
2. Keluar

Pilih (1-2): 1
Masukkan nama file puzzle lengkap dengan .txt: manypcs.txt
Error: Terlalu banyak potongan puzzle! Seharusnya hanya 7 potongan.
```

Gambar 4.2.2 Output Test Case 2

c. Kasus terdapat spasi di dalam bentuk

a) Input

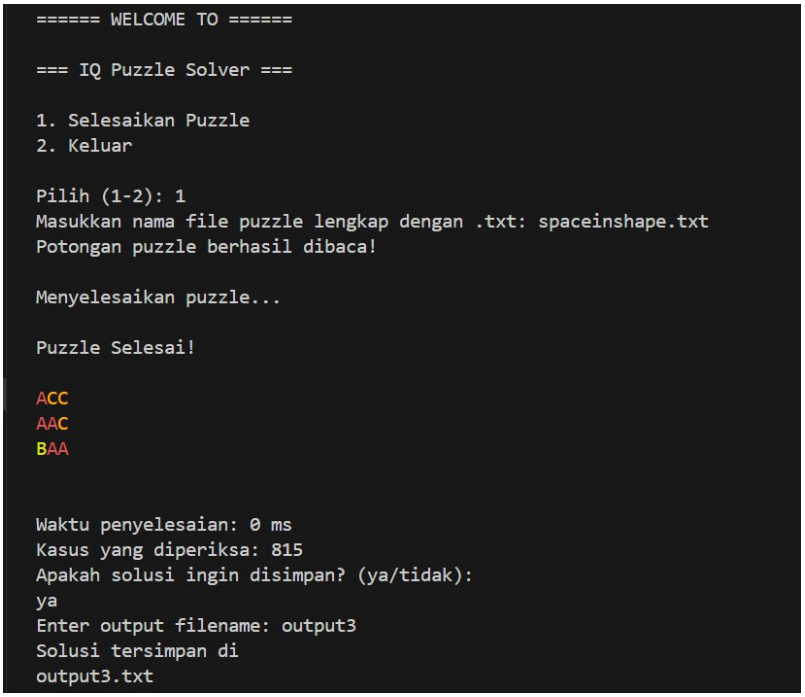
File input memiliki potongan puzzle yang terdapat spasi dalam bentuk-bentuknya. File input ini disimpan dengan nama spaceinshape.txt seperti berikut



Gambar 4.3.1 Input Test Case 3

b) Output

Berikut adalah hasil yang diberikan oleh program puzzle solver dari file input spaceinshape.txt kemudian disimpan dalam file output3.txt.

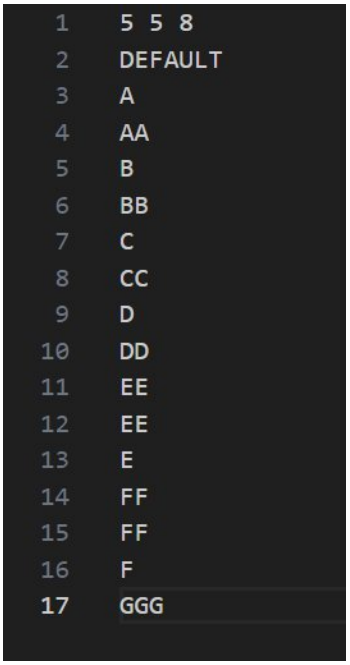


Gambar 4.3.2 Output Test Case 3

d. Kasus kekurangan potongan puzzle

a) Input

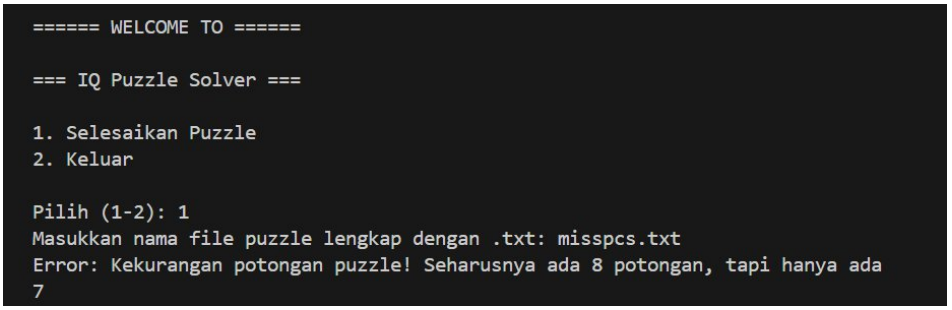
File input yang memiliki potongan puzzle yang ada dalam file lebih sedikit daripada yang diberikan dalam informasi papan di line pertama file txt. File input ini disimpan dengan nama misspcs.txt seperti berikut :



Gambar 4.4.1 Input Test Case 4

b) Output

Berikut adalah hasil yang diberikan oleh program puzzle solver dari file input misspcs.txt.

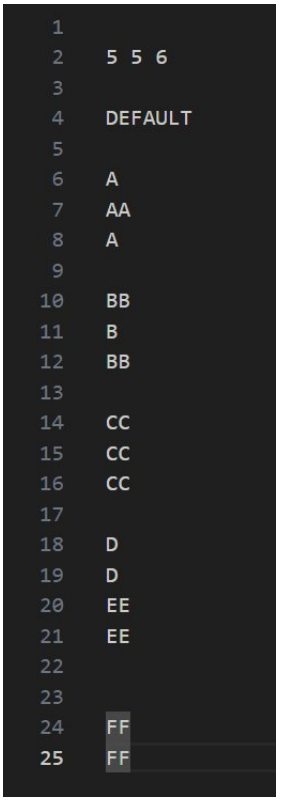


Gambar 4.4.2 Output Test Case 4

e. Kasus terdapat line kosong dalam file

a) Input

File input memiliki potongan puzzle yang terdapat line kosong di antara tiap line potongan puzzle ataupun informasi puzzle. File input ini disimpan dengan nama spacingline.txt seperti berikut :



Gambar 4.5.1 Input Test Case 5

b) Output

Berikut adalah hasil yang diberikan oleh program puzzle solver dari file input spacingline.txt kemudian disimpan dalam file output5.txt.

```
====== WELCOME TO ======

=== IQ Puzzle Solver ===

1. Selesaikan Puzzle
2. Keluar

Pilih (1-2): 1
Masukkan nama file puzzle lengkap dengan .txt: spacingline.txt
Potongan puzzle berhasil dibaca!

Menyelesaikan puzzle...

Puzzle Selesai!

ABBCC
AABCC
ABBCC
DEEFF
DEEFF


Waktu penyelesaian: 0 ms
Kasus yang diperiksa: 8
Apakah solusi ingin disimpan? (ya/tidak):
ya
Enter output filename: output5
Solusi tersimpan di
output5.txt
```

Gambar 4.5.2 Output Test Case 5

f. Kasus informasi papan tidak valid

a) Input

File input memiliki informasi puzzle yang tidak berbentuk angka (integer). File input ini disimpan dengan nama unmatchtypeinfo.txt seperti berikut :

```
1   aku suka stima
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 4.6.1 Input Test Case 6

b) Ouput

Berikut adalah hasil yang diberikan oleh program puzzle solver dari file input unmatchtypeinfo.txt.

```
====== WELCOME TO ======

=== IQ Puzzle Solver ===

1. Selesaikan Puzzle
2. Keluar

Pilih (1-2): 1
Masukkan nama file puzzle lengkap dengan .txt: unmatchtypeinfo.txt
Informasi Puzzle tidak dalam angka! Coba dicek!
```

Gambar 4.6.2 Output Test Case 6

g. Kasus tidak memiliki solusi

a) Input

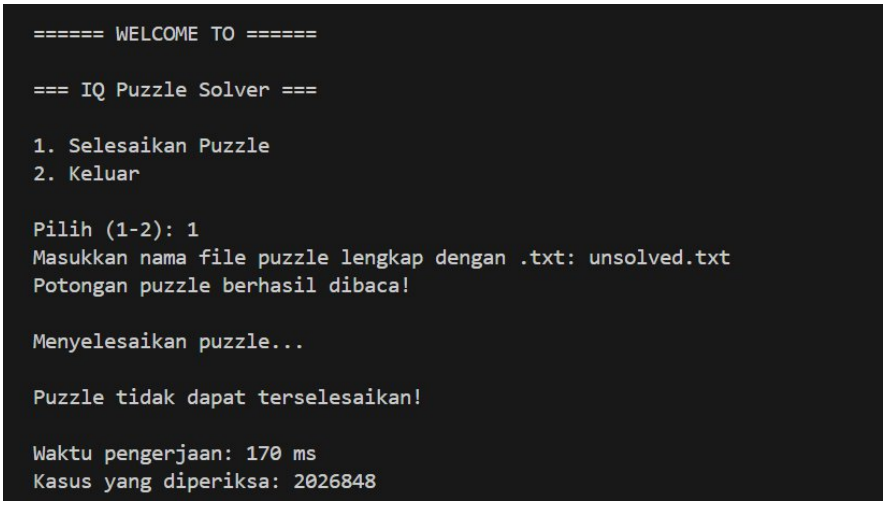
File input default untuk puzzle solver disimpan dalam file unsolved.txt yang diharapkan tidak ditemukan solusinya seperti berikut :

```
1   5 5 8
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  HHHHH
```

Gambar 4.7.1 Input Test Case 7

b) Output

Berikut adalah hasil yang diberikan oleh program puzzle solver dari file input unsolved.txt.



Gambar 4.7.2 Output Test Case 7

Lampiran

Repository Github

https://github.com/Ferdin-Arsenic/Tucil1_13523117.git

Checklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Solusi yang diberikan program benar dan mematuhi aturan permainan	✓	
4. Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	✓	
5. Program memiliki Graphical User Interface (GUI)		✓
6. Program dapat menyimpan solusi dalam bentuk file gambar		✓
7. Program dapat menyelesaikan kasus konfigurasi custom		✓
8. Program dapat menyelesaikan kasus konfigurasi Piramida (3D)		✓
9. Program dibuat oleh saya sendiri	✓	