TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA

Penyelesaian **IQ Puzzler Pro** dengan Algoritma Brute Force

Disusun oleh:

Aria Judhistira 13523112

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA - KOMPUTASI

INSTITUT TEKNOLOGI BANDUNG

2025

DAFTAR ISI

BAB I DESKRIPSI MASALAH DAN ALGORITMA PENYELESAIAN

1.1 Deskripsi Permasalahan

IQ Puzzler Pro merupakan suatu permainan untuk mengasah kemampuan berpikir dan kreativitas. Tujuan utama dari permainan ini adalah pemain harus mengisi seluruh papan dengan *piece* atau blok yang tersedia. Pada implementasi ini, papan permainan selalu dimulai dengan keadaan kosong, setiap *piece* dapat dirotasikan dan dicerminkan, serta permainan dinyatakan selesai jika dan hanya jika semua *piece* sudah diletak dan papan permainan terisi penuh.

1.2 Algoritma Penyelesaian

Penyelesaian permasalahan dalam permainan IQ Puzzler Pro menggunakan algoritma *brute force*. Berikut adalah langkah-langkah penyelesaian dengan mengimplementasi konsep algoritma tersebut.

1. Program meminta input user berupa file berekstensi .txt yang berisi spesifikasi permasalahan, yakni dimensi papan, jumlah *piece*, konfigurasi permainan, dan bentuk setiap *piece* yang akan digunakan.
2. Setelah menginisialisasi dan menyimpan semua nilai input pada file ke dalam variabel objek, program memulai algoritma pencarian solusi dan perhitungan waktu.
3. Pencarian solusi dilakukan secara rekursif. Basis rekursif adalah jika setiap *piece* sudah digunakan dan setiap posisi pada papan terpenuhi, maka algoritma pencarian solusi dinyatakan selesai.
4. Program mengambil *piece* pertama yang sebelumnya dibaca dari input, kemudian menjabarkan semua kemungkinan rotasi dan pencerminan *piece* tersebut dan menyimpannya dalam suatu *list*.
5. Program mencari posisi yang valid yang dimulai dari indeks (0,0) pada papan yang direpresentasikan sebagai matriks. Orientasi *piece* yang digunakan adalah orientasi pertama yang terdaftar pada *list* orientasi *piece*. Apabila *piece* tidak dapat ditempatkan pada indeks yang dipilih, program akan memilih orientasi *piece* berikutnya pada *list* orientasi *piece*. Jika tidak ada orientasi yang memenuhi, program akan mencoba indeks berikutnya secara traversal, kemudian mencoba

orientasi *piece* pertama pada *list* kembali. Setiap kali program melakukan iterasi, program mencatatnya sebagai salah satu kasus yang ditinjau.

6. Jika posisi yang diperiksa adalah posisi valid, program menempatkan *piece* pada posisi tersebut dan secara rekursif mengulang prosedur pencarian solusi (dari langkah 3) untuk *piece* berikutnya yang terdaftar.
7. Bila semua kemungkinan orientasi *piece* pada setiap posisi matriks tidak memungkinkan, program akan melakukan *backtracking* ke *piece* sebelumnya yang kemudian akan mencoba kombinasi orientasi dan posisi berikutnya.
8. Apabila setiap kemungkinan sudah dicoba dan tidak ada yang memenuhi, program akan menyatakan puzzle tidak memiliki solusi dan tidak dapat diselesaikan.
9. Pada akhir program, program menunjukkan solusi yang ditemukan, waktu pencarian yang dibutuhkan, dan banyak kasus yang ditinjau. Program juga meminta pengguna untuk memilih untuk menyimpan hasil solusi ke dalam file atau tidak. Jika pengguna memilih untuk menyimpan hasil solusi, program akan menyimpan output pada terminal ke dalam file berekstensi .txt dan gambar solusi dalam file berekstensi .png.

Berikut adalah *pseudocode* untuk algoritma utama yang tertera dalam Solver.java dalam menyelesaikan persoalan puzzle.

```
function puzzleSolver(board:char[][], pieces:List of Piece, idxPiece:int,
numCases:List of Integer)
      if idxPiece = size(piece) and isBoardFull(board) then
            -> true
      if idxPiece = size(piece) and not isBoardFull(board) then
            -> false
      if idxPiece =/= size(piece) and      isBoardFull(board) then
            -> false

      currPiece <- pieces[idxPiece]
      pieceName <- currPiece.getPieceName()
      pieceCombinations <- getPieceCombinations(currPiece)
```

```
i traversal [0..board.length]
      j traversal [0..board[0].length]
            for each currComb in pieceCombinations:
                  numCases[0] <- numCases[0] + 1
                  if isPosValid(board, currComb, idxCol, idxRow)
                  then
                        placePiece(board,   currComb,   idxCol,   idxRow,
                        pieceName)
                        idxPiece <- idxPiece + 1
                        if    puzzleSolver(board,    pieces,    idxPiece,
                        numCases) then
                              -> true
                        else
                              removePiece(board,     currComb,     idxCol,
                              idxRow)
-> false
```

BAB II IMPLEMENTASI DALAM JAVA

2.1 Main.java

Kelas Main berisi driver utama untuk menjalankan seluruh program. Fungsi ini merupakan fungsi utama yang memanggil fungsi-fungsi dan kelas-kelas lainnya untuk menjalankan program. Program dimulai dengan meminta aksi pengguna, kemudian meminta file input dari pengguna. Setelah dimasukkan, program akan memberikan *feedback* jika terdapat kesalahan (contoh: file tidak tersedia) atau langsung memberikan hasil algoritma jika tidak terdapat kesalahan. Selanjutnya, pengguna dapat menyimpan hasil jawaban algoritma dalam bentuk file, baik dalam bentuk .txt maupun .png.

2.2 Solver.java

Kelas ini berisi algoritma utama dalam menyelesaikan persoalan IQ Puzzler Pro. Fungsi puzzleSolver mencari kemungkinan posisi *piece* secara rekursif hingga papan terpenuhi dan semua *piece* terpasang atau salah satu dari kedua syarat tersebut tidak terpenuhi.

```java
package Game;
import Utils.Utils;
import java.util.*;

public class Solver {

    // Fungsi utama untuk menyelesaikan
    public static boolean puzzleSolver(char[][] board, List<Piece> pieces, int idxPiece, int[] numCases) {
        // Kasus basis jika semua puzzle sudah terisi pada board
        if (idxPiece == pieces.size() && Utils.isBoardFull(board)) return true;

        if (idxPiece == pieces.size() && !Utils.isBoardFull(board)) return false; // Semua piece ditaruh tapi papan belum penuh

        if (idxPiece != pieces.size() && Utils.isBoardFull(board)) return false; // Papan sudah penuh padahal piece belum semua ditaruh

        // Kasus rekursif
        Piece currPiece = pieces.get(idxPiece);
        char pieceName = currPiece.getPieceName();
        List<List<int[]>> pieceCombinations = Utils.getPieceCombinations(currPiece);

        for (int idxRow = 0; idxRow < board.length; idxRow++) {
            for (int idxCol = 0; idxCol < board[0].length; idxCol++) {
                for (List<int[]> currComb : pieceCombinations) { // Iterasi setiap kemungkinan piece
                    numCases[0] += 1;
                    if (Utils.isPosValid(board, currComb, idxCol, idxRow)) {
                        Utils.placePiece(board, currComb, idxCol, idxRow, pieceName);
                        if (puzzleSolver(board, pieces, idxPiece+1, numCases)) {
                            return true;
                        } else Utils.removePiece(board, currComb, idxCol, idxRow); // Kasus backtrack jika piece berikutnya tidak bisa ditaruh
                    }
                }
            }
        }
        return false;
    }
}
```

2.3 Piece.java

Kelas ini merupakan representasi sebuah *piece*. Atribut yang dimiliki meliputi List yang berisi koordinat-koordinat *piece* dalam bentuk (*x*, *y*) dan nama dari *piece* dalam bentuk karakter. Berikut adalah daftar metode-metode dalam kelas ini beserta deskripsinya.

Nama Fungsi	Deskripsi
Piece	Konstruktor untuk membuat objek Piece.
rotatePiece90	Merotasikan koordinat-koordinat pada Piece sebesar 90 derajat.
mirrorPiece	Mencerminkan koordinat-koordinat pada Piece secara horizontal.
getCells	Mengembalikan List berisi koordinat Piece
getPieceName	Mengembalikan karakter nama Piece
setCells	Menetapkan koordinat baru pada Piece

```java
package Game;
import java.util.*;

public class Piece {
    List<int[]> cells;
    char pieceName;

    public Piece(List<int[]> cells, char name) {
        this.cells = cells;
        this.pieceName = name;
    }

    // Method untuk merotasikan piece 90 derajat clockwise
    public void rotatePiece90() {
        List<int[]> rotatedCells = new ArrayList<>();

        // Rotasi setiap koordinat piece
        for (int[] coordinate : cells) {
            // (x,y) -> (-y, x)
            int[] newCoordinate = new int[]{-coordinate[1], coordinate[0]};
            rotatedCells.add(newCoordinate);
        }

        // Cari nilai minimum x dan y
        int minX = Integer.MAX_VALUE;
        int minY = Integer.MAX_VALUE;
        for (int[] coordinate : rotatedCells) {
            minX = Math.min(minX, coordinate[0]);
            minY = Math.min(minY, coordinate[1]);
        }

        // Sesuaikan koordinat agar nilai x dan y minimum menjadi 0
        List<int[]> normalizedCells = new ArrayList<>();
        for (int[] coordinate : rotatedCells) {
            normalizedCells.add(new int[]{coordinate[0] - minX, coordinate[1] - minY});
        }

        this.cells = normalizedCells;
    }
```

```java
    // Method untuk mencerminkan koordinat piece
    public void mirrorPiece() {
        List<int[]> mirroredCells = new ArrayList<>();

        // Mirror setiap koordinat piece
        for (int[] coordinate : cells) {
            // (x,y) -> (-x, y)
            int[] mirroredCoordinate = new int[]{-coordinate[0], coordinate[1]};
            mirroredCells.add(mirroredCoordinate);
        }

        // Cari nilai minimum row sebagai offset
        int minRow = Integer.MAX_VALUE;
        for (int[] coordinate : mirroredCells) {
            minRow = Math.min(minRow, coordinate[0]);
        }

        // Sesuaikan koordinat agar nilai row minimum menjadi 0
        List<int[]> normalizedMirroredCells = new ArrayList<>();
        for (int[] coordinate : mirroredCells) {
            normalizedMirroredCells.add(new int[]{coordinate[0] - minRow, coordinate[1]});
        }

        this.cells = normalizedMirroredCells;
    }

    // Method untuk mendapatkan atribut piece
    public List<int[]> getCells() {
        return this.cells;
    }
    public char getPieceName() {
        return this.pieceName;
    }

    // Method untuk set atribut piece
    public void setCells(List<int[]> cells) {
        this.cells = cells;
    }
}
```

2.4 IOHandler.java

Kelas ini berisi fungsi-fungsi untuk menerima dan membaca input serta melakukan output, baik ke dalam CLI (Command-Line Interface) maupun ke dalam file. Berikut adalah fungsi-fungsi dengan deskripsinya.

Nama Fungsi	Deskripsi
inputFile	Mengembalikan String berupa path menuju file yang ingin dibaca
readInput	Membaca masukan pada input file, kemudian menyimpan nilai-nilai input dengan mengisi objek gameConfig
writeOutputFile	Menulis hasil algoritma ke dalam sebuah file berekstensi .txt

writeOutputTerminal	Menampilkan hasil algoritma ke dalam terminal pengguna
outputAsImage	Membentuk gambar berdasarkan hasil algoritma dan menyimpannya dalam bentuk .png
getFileName	Mengembalikan nama file yang dimasukkan oleh pengguna

2.5 GameConfig.java

Kelas ini digunakan untuk mengatur konfigurasi permainan dengan menyimpan informasi berupa dimensi papan permainan, jumlah *piece*, jenis puzzle, daftar yang berisi semua objek Piece yang terdaftar, dan status penyelesaian. Berikut adalah deskripsi untuk setiap fungsi-fungsinya

Nama Fungsi	Deskripsi
GameConfig	Konstruktor untuk menginisialisasi objek dari kelas ini.
setGameConfig	Prosedur untuk mengisi atribut-atribut kelas GameConfig.
isPosEmpty	Memeriksa apakah posisi (*i, j*) kosong atau tidak.
Metode-metode Getter	Mengambil nilai atribut yang tersimpan dalam suatu instansi GameConfig.
Metode-metode Setter	Menetapkan nilai atribut yang baru pada objek.
isInputValid	Memeriksa apakah input dari file memungkinkan permainan atau tidak.

2.6 Utils.java

Kelas ini berisi fungsi-fungsi pembantu atau *helper* untuk membantu fungsi utama puzzleSolver dalam mencari solusi. Selain itu, kelas ini juga berisi beberapa fungsi untuk

membantu validasi input dan map untuk pewarnaan output. Berikut adalah daftar fungsi-fungsi dan deskripsinya.

Nama Fungsi	Deskripsi
getPieceCombinations	Mengembalikan semua kombinasi rotasi dan pencerminan piece dalam bentuk sebuah List.
isPosValid	Mengecek apakah piece dapat menempatkan posisi (i,j) .
placePiece	Menaruh piece pada posisi (i,j).
removePiece	Menghapuskan piece pada posisi (i,j).
isBoardFull	Mengecek apakah papan sudah penuh atau belum.
printPieces	Menampilkan koordinat semua piece yang tercatat
getTotalPieceCoords	Mengembalikan total koordinat setiap piece
getTotalPos	Mengembalikan semua posisi yang valid pada papan

```java
public class Utils {

    // Method untuk mencatat semua kemungkinan rotasi dan pencerminan piece
    public static List<List<int[]>> getPieceCombinations(Piece piece) {
        List<List<int[]>> combinations = new ArrayList<>(); // List berisi semua kemungkinan rotasi dan pencerminan piece

        // Rotasi piece
        for (int i = 0; i < 4; i++) {
            piece.rotatePiece90();
            List<int[]> rotatedPiece = piece.getCells();
            combinations.add(rotatedPiece);
        }

        // Pencerminan piece
        piece.mirrorPiece();

        // Piece dirotasikan kembali
        for (int i = 0; i < 4; i++) {
            piece.rotatePiece90();
            List<int[]> rotatedPiece = piece.getCells();
            combinations.add(rotatedPiece);
        }

        return combinations;
    }

    public static boolean isPosValid(char[][] board, List<int[]> pieceCells, int idxCol, int idxRow) {
        for (int[] coordinate : pieceCells) {
            int row = idxRow + coordinate[1];
            int col = idxCol + coordinate[0];
            if (row < 0 || row >= board.length || col < 0 || col >= board[0].length || board[row][col] != '.') {
                return false;
            }
        }
        return true;
    }

    public static void placePiece(char[][] board, List<int[]> pieceCells, int idxCol, int idxRow, char pieceName) {
        for (int[] coordinate : pieceCells) {
            int row = idxRow + coordinate[1];
            int col = idxCol + coordinate[0];
            board[row][col] = pieceName;
        }
    }

    public static void removePiece(char[][] board, List<int[]> pieceCells, int idxCol, int idxRow) {
        for (int[] coordinate : pieceCells) {
            int row = idxRow + coordinate[1];
            int col = idxCol + coordinate[0];
            board[row][col] = '.';
        }
    }
```

```java
    // Method untuk mengecek apakah papan sudah penuh atau belum
    public static boolean isBoardFull(char[][] board) {
        for (int idxRow = 0; idxRow < board.length; idxRow++) {
            for (int idxCol = 0; idxCol < board[0].length; idxCol++) {
                if (board[idxRow][idxCol] == '.') return false;
            }
        }
        return true;
    }


    // Debug method untuk print pieces
    public static void printPieces(List<Piece> pieces) {
        for (Piece piece : pieces) {
            System.out.println("Piece: " + piece.getPieceName());
            for (int[] cell : piece.getCells()) {
                System.out.println(cell[0] + " " + cell[1]);
            }
        }
    }

    // Method untuk mendapatkan semua jumlah potong piece
    public static int getTotalPieceCoords(List<Piece> pieces) {
        int totalCoords = 0;
        for (Piece piece : pieces) {
            totalCoords += piece.getCells().size();
        }
        return totalCoords;
    }

    // Method untuk mendapatkan semua posisi valid pada papan
    public static int getTotalPos(char[][] board) {
        int totalPos = 0;
        for (int idxRow = 0; idxRow < board.length; idxRow++) {
            for (int idxCol = 0; idxCol < board[0].length; idxCol++) {
                if (board[idxRow][idxCol] == '.') totalPos++;
            }
        }
        return totalPos;
    }

}
```

BAB III TEST KASUS PROGRAM

3.1 Test Case 1

Tampilan test case (.txt)

test1.txt

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

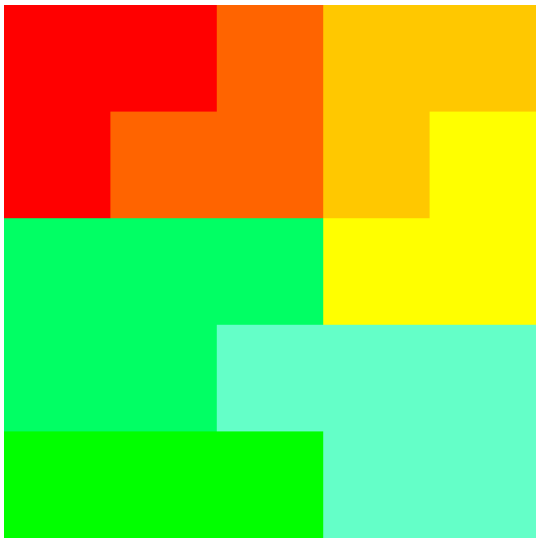
Tampilan output (CLI)

```
--------------------------------------------------------------------------------
                    IQ Puzzler Pro (Java Edition)
--------------------------------------------------------------------------------
Silakan pilih:
1. Mulai permainan
2. Keluar
Pilihan: 1
Masukkan nama file input: test1.txt
AABCC
ABBCD
EEEDD
EEFFF
GGGFF

Waktu pencarian: 3 ms

Banyak kasus yang ditinjau: 5690 kasus

Apakah Anda ingin menyimpan solusi? (y/n):
```

Tampilan output (File)

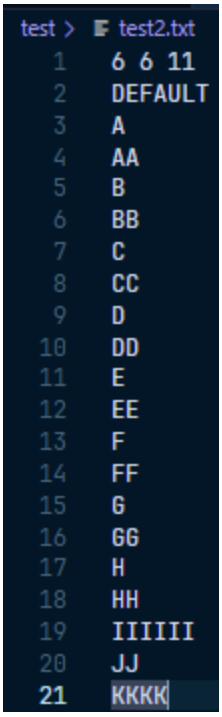
test > results > y.txt

```
1   AABCC
2   ABBCD
3   EEEDD
4   EEFFF
5   GGGFF
6
7   Waktu pencarian: 3 ms
8   Banyak kasus yang ditinjau: 5690 kasus
```

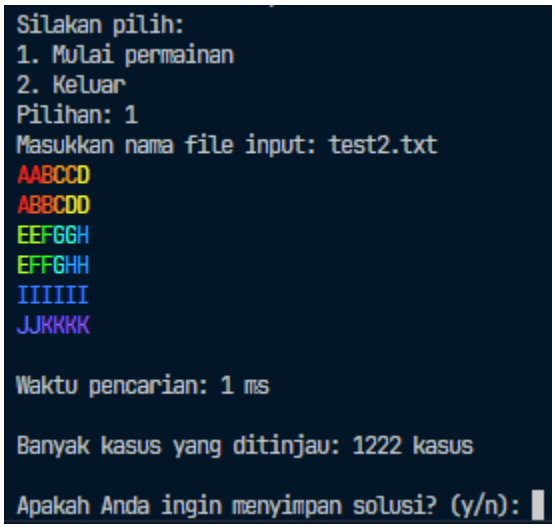


3.2 Test Case 2

Tampilan test case (.txt)



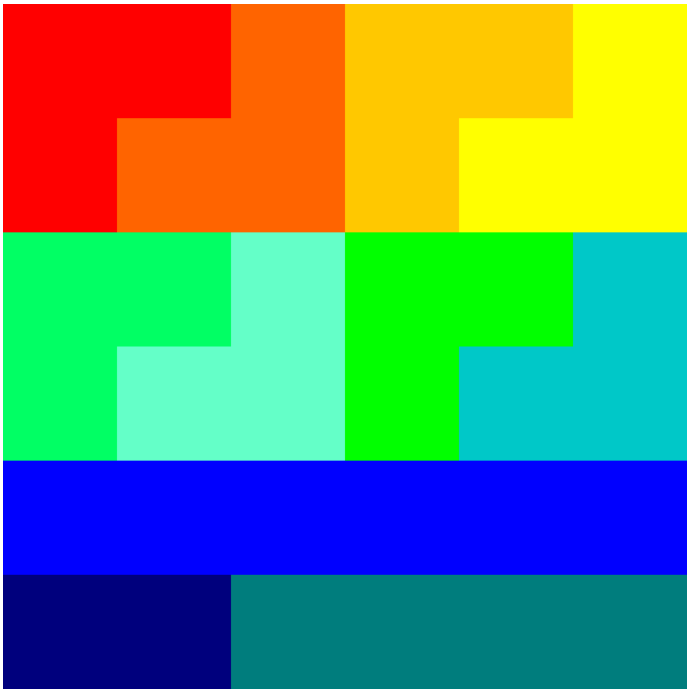
Tampilan output (CLI)



Tampilan output (File)

test > results > test2.txt

```
AABCCD
ABBCDD
EEFGGH
EFFGHH
IIIIII
JJKKKK

Waktu pencarian: 1 ms
Banyak kasus yang ditinjau: 1222 kasus
```



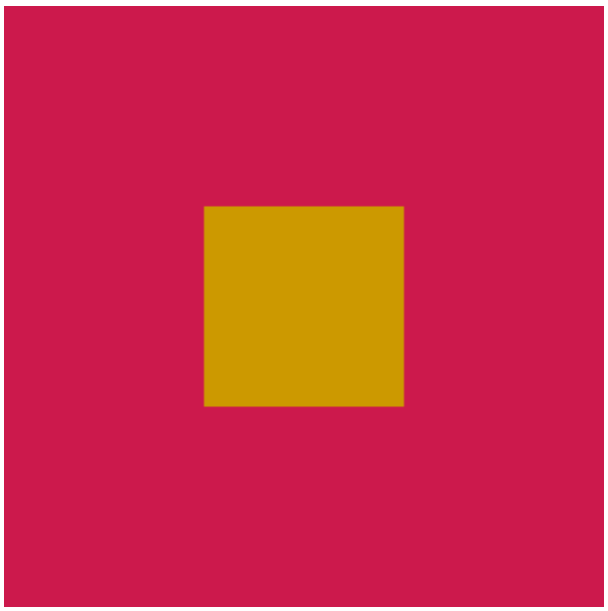
3.3 Test Case 3

Tampilan test case (.txt)
test > test3.txt 3 3 2 DEFAULT ZZZ Z Z ZZZ P
Tampilan output (CLI)

```
Silakan pilih:
1. Mulai permainan
2. Keluar
Pilihan: 1
Masukkan nama file input: test3.txt
ZZZ
ZPZ
ZZZ

Waktu pencarian: 0 ms

Banyak kasus yang ditinjau: 34 kasus

Apakah Anda ingin menyimpan solusi? (y/n):
```

Tampilan output (File)

```
test > results > test3.txt
1   ZZZ
2   ZPZ
3   ZZZ
4
5   Waktu pencarian: 0 ms
6   Banyak kasus yang ditinjau: 34 kasus
```



3.4 Test Case 4

Tampilan test case (.txt)

test > test4.txt

```
6 7 10
DEFAULT
KKK
KK
LLL
 L
MMM
M
MMM
OO
OO
P
P
P
QQQQQQ
R
RRR
 R
SS
S
SS
T
UU

```

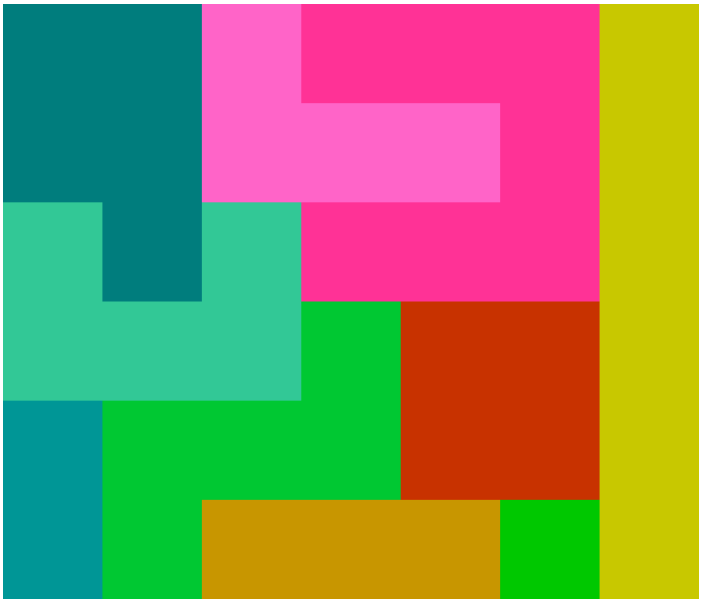
Tampilan output (CLI)

```
Silakan pilih:
1. Mulai permainan
2. Keluar
Pilihan: 1
Masukkan nama file input: test4.txt
KKLMMMQ
KKLLLMQ
SKSMMMQ
SSSROOQ
URRROOQ
URPPPTQ

Waktu pencarian: 3001 ms

Banyak kasus yang ditinjau: 272841837 kasus

Apakah Anda ingin menyimpan solusi? (y/n):
```

Tampilan output (File)

test > results > test4.txt

```
KKLMMMQ
KKLLLMQ
SKSMMMQ
SSSROOQ
URRROOQ
URPPPTQ

Waktu pencarian: 3001 ms
Banyak kasus yang ditinjau: 272841837 kasus
```

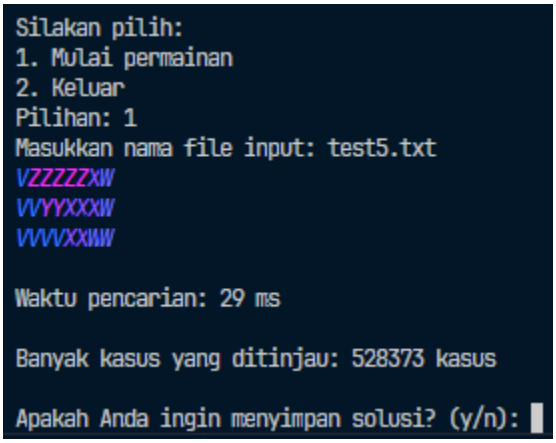
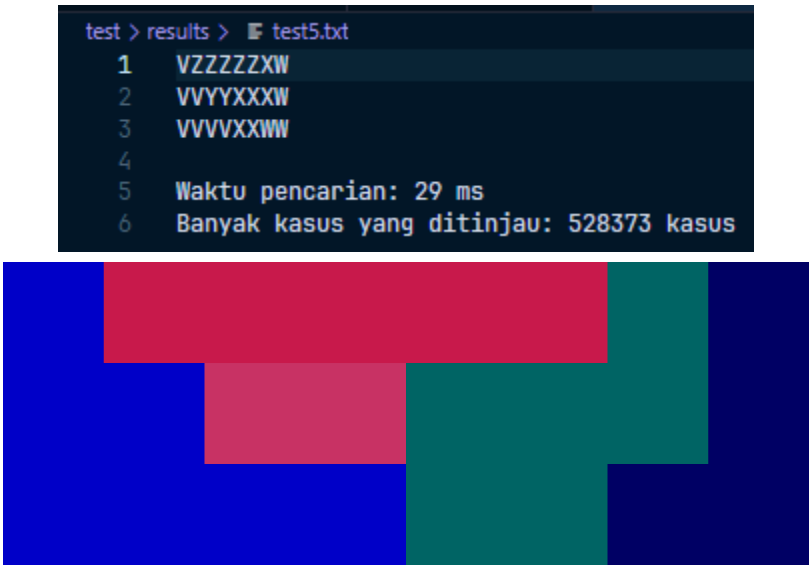


3.5 Test Case 5

Tampilan test case (.txt)

test > test5.txt

```
3 8 5
DEFAULT
V
VV
VVVV
WWW
W
X
XXX
XX
YY
ZZZZZ
```

Tampilan output (CLI)

Tampilan output (File)


3.6 Test Case 6

Tampilan test case (.txt)

test > test6.txt

```
4 6 7
DEFAULT
AAAA
A
BB
B
CCC
C C
CCC
DD
EE
E
FF
G
```

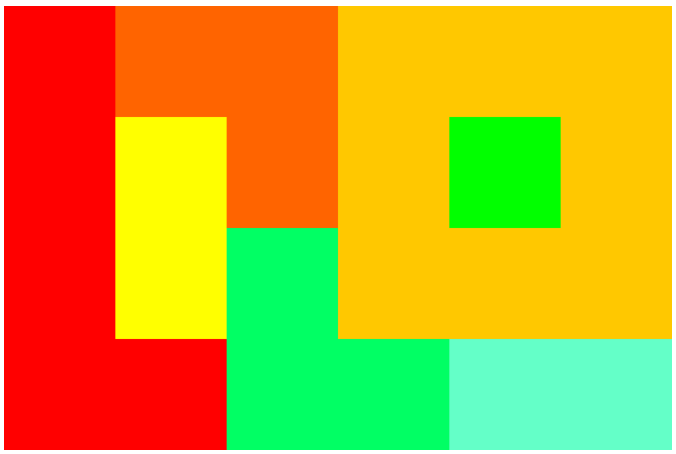
Tampilan output (CLI)

```
Silakan pilih:
1. Mulai permainan
2. Keluar
Pilihan: 1
Masukkan nama file input: test6.txt
ABBCCC
ADBCGC
ADECCC
AAEEFF

Waktu pencarian: 17 ms

Banyak kasus yang ditinjau: 234708 kasus

Apakah Anda ingin menyimpan solusi? (y/n):
```

Tampilan output (File)

test > results > test6.txt

```
ABBCCC
ADBCGC
ADECCC
AAEEFF

Waktu pencarian: 17 ms
Banyak kasus yang ditinjau: 234708 kasus
```



3.7 Test Case 7

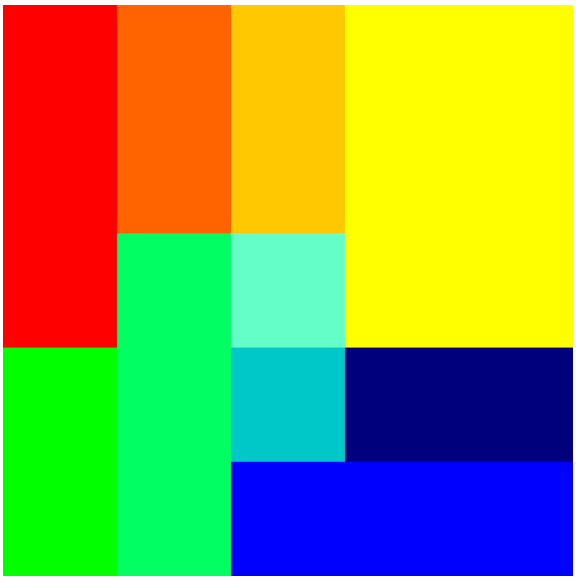
Tampilan test case (.txt)
test > test7.txt 1 5 5 10 2 DEFAULT 3 AAA 4 BB 5 CC 6 DD 7 DD 8 DD 9 EEE 10 F 11 GG 12 H 13 III 14 JJ 15
Tampilan output (CLI)

```
Silakan pilih:
1. Mulai permainan
2. Keluar
Pilihan: 1
Masukkan nama file input: test7.txt
ABCDD
ABCDD
AEFDD
GEHJJ
GEIII

Waktu pencarian: 0 ms

Banyak kasus yang ditinjau: 821 kasus

Apakah Anda ingin menyimpan solusi? (y/n):
```

Tampilan output (File)

test > results > test7.txt

```
1  ABCDD
2  ABCDD
3  AEFDD
4  GEHJJ
5  GEIII
6
7  Waktu pencarian: 0 ms
8  Banyak kasus yang ditinjau: 821 kasus
```



BAB IV KESIMPULAN

Program implementasi permainan IQ Puzzler Pro ini menggunakan konsep algoritma *brute force* untuk menemukan solusi. Program dimulai dari pembacaan input, dilanjutkan dengan penyelesaian menggunakan algoritma *brute force*. Algoritma ini melibatkan mencari kombinasi posisi dan orientasi *piece* pada papan yang valid secara rekursif untuk setiap *piece*. Apabila suatu *piece* tidak dapat ditempatkan sama sekali, program akan melakukan *backtracking* ke *piece* sebelumnya dengan mengubah orientasi atau posisinya. Setelah mendapatkan hasilnya, program dapat menyimpan hasil solusi ke dalam file berbentuk .txt dan .png.

BAB V LAMPIRAN

5.1 Pranala Repositori GitHub

Berikut adalah pranala menuju repositori GitHub yang berisi sumber kode: https://github.com/TukangLas21/Tucil1_13523112

5.2 Checklist Tabel

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