# Algoritma:

Algoritma yang digunakan adalah dengan menggunakan Teknik bruteforce. Algoritma ini meletakkan piece secara urut dari sebuah pojok ke pojok lainnya. Apabila bisa meletakkan piece, maka akan diletakkan piece berikutnya. Namun apabila tidak bisa meletakkan piece-nya, maka akan dicoba memutar/mencerminkan/membalik piece tersebut. Apabila ternyata masih tidak bisa, maka akan dicoba diganti dengan piece lainnya yang belum pernah dicoba di tempat tersebut. Proses ini akan dilakukan terus menerus, hingga semua jenis kemungkinan telah dicoba, atau telah ditemukan suatu susunan yang tepat.

# Source code:

- Includes:

```
package src;
import java.util.Scanner;
import java.util.ArrayList;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
```

- Src untuk print waktu (memanggil startTimer di awal, kemudian memanggil stopTimer saat ingin berhenti):

```
    private static long startTime;

    public static void startTimer() {
        startTime = System.currentTimeMillis();
    }

    public static void stopTimer() {
        long endTime = System.currentTimeMillis();
        System.out.println("\nExecution time: " + (endTime - startTime) + " ms");
    }
```

- A

```java
public static void saveFile(int[][][]board, int N, int M, int isPyramid) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Save answer to file? (Y/N): ");
        String saveResponse = scanner.nextLine();
        while(!(saveResponse.equalsIgnoreCase("Y") || saveResponse.equalsIgnoreCase("N"))) {
            System.out.println("Invalid input. Please enter Y or N.");
            System.out.print("Save answer to file? (Y/N): ");
            saveResponse = scanner.nextLine();
        }
        if (saveResponse.equalsIgnoreCase("Y")) {
            System.out.print("Enter filename(without .txt): ");
            String filename = scanner.nextLine();
            File file = new File("save/" + filename + ".txt");

            if (file.exists()) {
                System.out.print("File already exists. Overwrite? (Y/N): ");
                String overwriteResponse = scanner.nextLine();
                while(!(overwriteResponse.equalsIgnoreCase("Y") || overwriteResponse.equalsIgnoreCase("N"))) {
                    System.out.println("Invalid input. Please enter Y or N.");
                    System.out.print("File already exists. Overwrite? (Y/N): ");
                    overwriteResponse = scanner.nextLine();
                }
                if (!overwriteResponse.equalsIgnoreCase("Y")) {
                    System.out.println("File not saved.");
                    scanner.close();
                    return;
                }
            }

            try (FileWriter writer = new FileWriter(file)) {
                for (int a = 0; a < (isPyramid == 1 ? N : 1); a++) {
                    for (int b = 0; b < M; b++) {
                        for (int c = 0; c < N; c++) {
                            if (board[a][b][c] == -1) {
                                writer.write("  ");
                            } else {
                                char letter = (char) ('A' + board[a][b][c] - 1);
                                writer.write(letter + " ");
                            }
                        }
                        writer.write("\n");
                    }
                }
                System.out.println("File saved successfully.");
            } catch (IOException e) {
                System.out.println("Error while saving the file: " + e.getMessage());
            }
        } else {
            System.out.println("File not saved.");
        }
        scanner.close();
    }
```

- Src untuk print papan:

```java
public static void printBoard(int [][][] board, int N, int M, int isPyramid){
    String[] colors = {
        "\u001B[31m", "\u001B[32m", "\u001B[33m", "\u001B[34m", "\u001B[35m", "\u001B[36m",
        "\u001B[91m", "\u001B[92m", "\u001B[93m", "\u001B[94m", "\u001B[95m", "\u001B[96m",
        "\u001B[37m", "\u001B[90m", "\u001B[97m", "\u001B[30m", "\u001B[41m", "\u001B[42m",
        "\u001B[43m", "\u001B[44m", "\u001B[45m", "\u001B[46m", "\u001B[47m", "\u001B[100m",
        "\u001B[101m", "\u001B[102m"
    };
    int ma;
    if(isPyramid == 0){
        ma = 1;
    }else{
        ma = N;
    }
    for (int a = 0; a < ma; a++){
        if(isPyramid == 1){
            System.out.println("Layer " + (a+1));
        }
        for (int b = 0; b < N; b++){
            for(int c= 0 ; c < M; c++){
                if(board[a][b][c] == -1){
                    System.out.print("  ");
                }else{
                    char letter = (char) ('A' + board[a][b][c] - 1);
                    String color = colors[(board[a][b][c] + 25) % 26];
                    System.out.print(color + letter + " \u001B[0m");
                }
            }
            System.out.println();
        }
    }
}
```

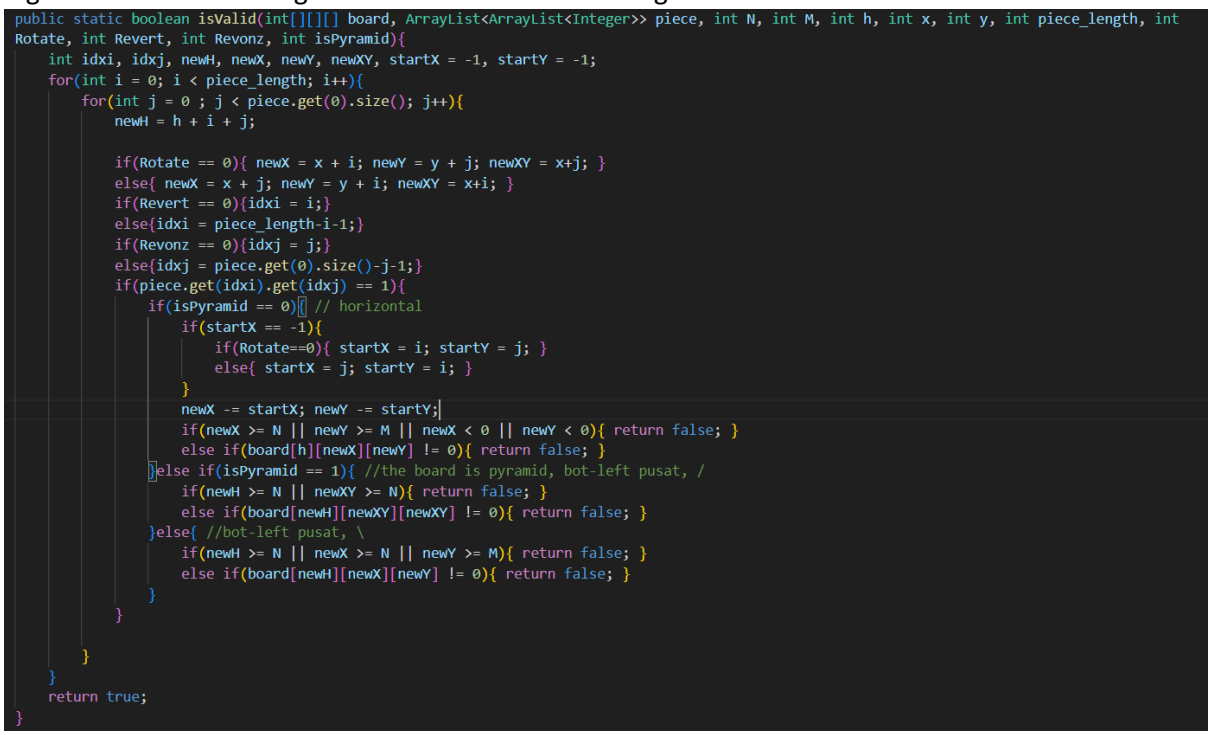
- Fungsi untuk mengecek validitas string pada piece/block:

```java
public static char blockStringChecker(String s){
    char validity = ']';
    for(int i = 0; i < s.length(); i++){
        if(s.charAt(i) != ' '){
            if(s.charAt(i)== ']'){
                System.out.println("Error: Invalid block input");
                System.exit(0);
            }else if(validity == ']'){
                validity = s.charAt(i);
            }else{
                if(validity != s.charAt(i)){
                    System.out.println("Error: Invalid block input");
                    System.exit(0);
                }
            }
        }
    }
    if (validity < 'A' || validity > 'Z') { // cek A-Z
        System.out.println("Error: Invalid block input");
        System.exit(0);
    }
    return (validity);
}
```

- Prosedur untuk meletakkan/mengambil piece/block dari papan:

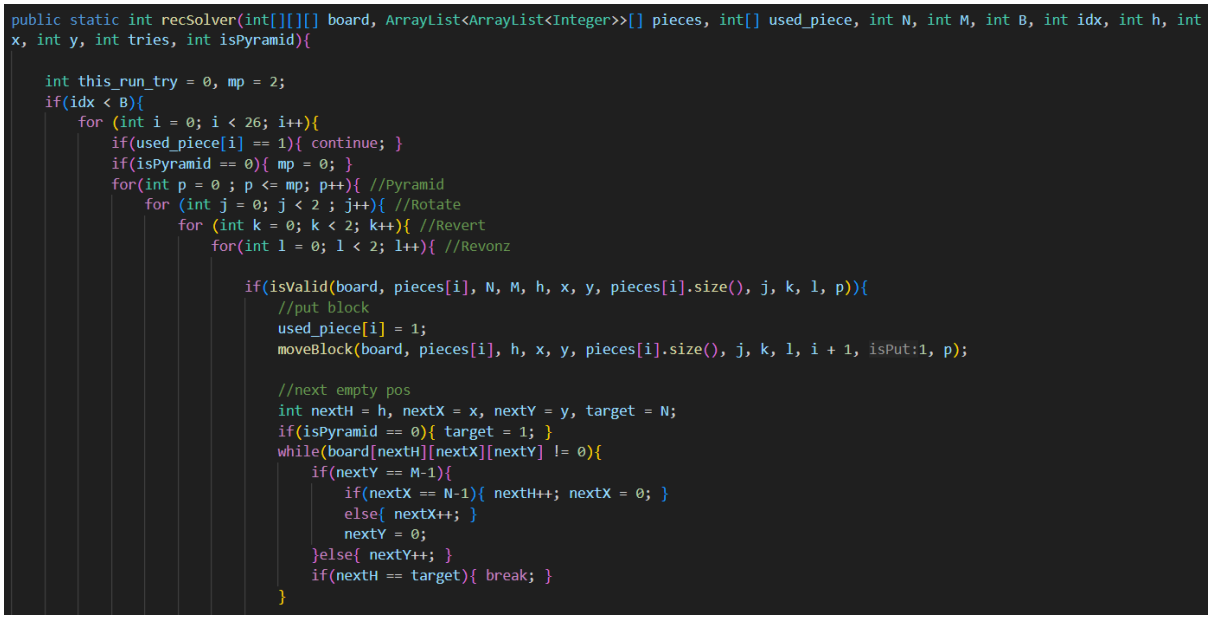
```java
public static void moveBlock(int[][][] board, ArrayList<ArrayList<Integer>> piece, int h, int x, int y, int piece_length, int Rotate, int
Revert, int Revonz, int blockNumber, int isPut, int isPyramid){
    int num = blockNumber, idxi, idxj, newX, newY, newXY, newH, startX = -1, startY = -1;
    if(isPut == 0){ num = 0; }
    for(int i = 0; i < piece_length; i++){
        for(int j = 0 ; j < piece.get(0).size(); j++){
            newH = h + i + j;

            if(Rotate == 0){ newX = x + i; newY = y + j; newXY = x+j; }
            else{ newX = x + j; newY = y + i; newXY = x+i; }

            if(Revert == 0){idxi = i;}
            else{idxi = piece_length-i-1;}

            if(Revonz == 0){idxj = j;}
            else{idxj = piece.get(0).size()-j-1;}

            if(piece.get(idxi).get(idxj) == 1){
                if(isPyramid == 0){ // horizontal
                    if(startX == -1){
                        if(Rotate==0){ startX = i; startY = j; }
                        else{ startX = j; startY = i; }
                    }
                    newX -= startX; newY -= startY;
                    board[h][newX][newY] = num;
                }else if(isPyramid == 1){ //the board is pyramid, bot-left pusat, /
                    board[newH][newXY][newXY] = num;
                }else{ //bot-left pusat, \
                    board[newH][newX][newY] = num;
                }
            }

        }
    }
}
```

- Fungsi untuk mengecek apakah bisa meletakkan piece/block pada papan. Fungsi ini digunakan SEBELUM fungsi meletakkan block di atas digunakan:



```
public static boolean isValid(int[][][] board, ArrayList<ArrayList<Integer>> piece, int N, int M, int h, int x, int y, int piece_length, int
Rotate, int Revert, int Revonz, int isPyramid){
    int idxi, idxj, newH, newX, newY, newXY, startX = -1, startY = -1;
    for(int i = 0; i < piece_length; i++){
        for(int j = 0 ; j < piece.get(0).size(); j++){
            newH = h + i + j;

            if(Rotate == 0){ newX = x + i; newY = y + j; newXY = x+j; }
            else{ newX = x + j; newY = y + i; newXY = x+i; }
            if(Revert == 0){idxi = i;}
            else{idxi = piece_length-i-1;}
            if(Revonz == 0){idxj = j;}
            else{idxj = piece.get(0).size()-j-1;}
            if(piece.get(idxi).get(idxj) == 1){
                if(isPyramid == 0){ // horizontal
                    if(startX == -1){
                        if(Rotate==0){ startX = i; startY = j; }
                        else{ startX = j; startY = i; }
                    }
                    newX -= startX; newY -= startY;
                    if(newX >= N || newY >= M || newX < 0 || newY < 0){ return false; }
                    else if(board[h][newX][newY] != 0){ return false; }
                }else if(isPyramid == 1){ //the board is pyramid, bot-left pusat, /
                    if(newH >= N || newXY >= N){ return false; }
                    else if(board[newH][newXY][newXY] != 0){ return false; }
                }else{ //bot-left pusat, \
                    if(newH >= N || newX >= N || newY >= M){ return false; }
                    else if(board[newH][newX][newY] != 0){ return false; }
                }
            }

        }
    }
    return true;
}
```

- Fungsi solver yang menggunakan konsep rekursif. Fungsi ini sangat penting untuk mencari solusinya:



```
public static int recSolver(int[][][] board, ArrayList<ArrayList<Integer>>[] pieces, int[] used_piece, int N, int M, int B, int idx, int h, int
x, int y, int tries, int isPyramid){

    int this_run_try = 0, mp = 2;
    if(idx < B){
        for (int i = 0; i < 26; i++){
            if(used_piece[i] == 1){ continue; }
            if(isPyramid == 0){ mp = 0; }
            for(int p = 0 ; p <= mp; p++){ //Pyramid
                for (int j = 0; j < 2 ; j++){ //Rotate
                    for (int k = 0; k < 2; k++){ //Revert
                        for(int l = 0; l < 2; l++){ //Revonz

                            if(isValid(board, pieces[i], N, M, h, x, y, pieces[i].size(), j, k, l, p)){
                                //put block
                                used_piece[i] = 1;
                                moveBlock(board, pieces[i], h, x, y, pieces[i].size(), j, k, l, i + 1, isPut:1, p);

                                //next empty pos
                                int nextH = h, nextX = x, nextY = y, target = N;
                                if(isPyramid == 0){ target = 1; }
                                while(board[nextH][nextX][nextY] != 0){
                                    if(nextY == M-1){
                                        if(nextX == N-1){ nextH++; nextX = 0; }
                                        else{ nextX++; }
                                        nextY = 0;
                                    }else{ nextY++; }
                                    if(nextH == target){ break; }
                                }
```

Kemudian akan di-cek apakah papan sudah penuh atau belum. Apabila papan sudah penuh:

```java
//solution found
if(nextH == target){

    if(idx == B-1){
        System.out.println("\nSolution found.\nTries: " + tries + "\nSolution:\n");
        stopTimer();

        //print board
        printBoard(board, N, M, isPyramid);
        saveFile(board, N, M, isPyramid);

    }else{ //kalo kelebihan piece(s)
        System.out.println("\nTest Case invalid.\nTries: " + tries);
        stopTimer();

        System.out.println("Reason: board full with spare piece(s).");

        //print board
        printBoard(board, N, M, isPyramid);

        //PIECE ga kepake

        String[] colors = {
            "\u001B[31m", "\u001B[32m", "\u001B[33m", "\u001B[34m", "\u001B[35m", "\u001B[36m",
            "\u001B[91m", "\u001B[92m", "\u001B[93m", "\u001B[94m", "\u001B[95m", "\u001B[96m",
            "\u001B[37m", "\u001B[90m", "\u001B[97m", "\u001B[30m", "\u001B[41m", "\u001B[42m",
            "\u001B[43m", "\u001B[44m", "\u001B[45m", "\u001B[46m", "\u001B[47m", "\u001B[100m",
            "\u001B[101m", "\u001B[102m"
        };

        System.out.println("Unused piece(s):");
        for(int a = 0; a < B; a++){
            if(used_piece[a] == 0){
                for(int b = 0; b < pieces[a].size(); b++){
                    for(int c = 0 ; c < pieces[a].get(b).size() ; c++){
                        if(pieces[a].get(b).get(c) == 1){
                            // System.out.print(a+1);
                            char letter = (char) ('A' + a);
                            String color = colors[(a) % 26];
                            System.out.print(color + letter + " \u001B[0m");
                        }else{
                            System.out.print(" ");
                        }
                    }
                    System.out.println("");
                }
                System.out.println("");
            }
        }
    }
    System.exit(0);
}
```

Bila papan belum penuh, maka dilanjutkan dengan proses rekursif:

```
                        //recursive
                        this_run_try += recSolver(board, pieces, used_piece, N, M, B, idx+1, nextH, nextX, nextY, tries + this_run_try,
                        isPyramid);
                        //take block
                        used_piece[i] = 0;
                        moveBlock(board, pieces[i], h, x, y, pieces[i].size(), j, k, l, i + 1, isPut:0, p);
                        // System.out.println("Valid move: " + i + " " + j + " " + k + " " + l);
                    }else{
                        // System.out.println("Invalid move: " + i + " " + j + " " + k + " " + l);
                        this_run_try++;
                    }
                }
            }

        }

    }
}
return this_run_try;
}
```

- MAIN

```java
public static void main(String[] args){
    try {
        scanner = new Scanner(System.in);
        System.out.print("Enter the filename (must be in the test folder): ");
        String filename = scanner.nextLine();
        input = new Scanner(new File("test/" + filename + ".txt"));
        // scanner.close(); // Do not close the scanner here
    } catch (FileNotFoundException e) {
        System.out.println("File not found: " + e.getMessage());
        System.exit(1);
    }
    int N = input.nextInt();
    int M = input.nextInt();
    int B = input.nextInt();
    input.nextLine(); // consume the newline character
    String board_type = input.nextLine();
    int[][][] board;
    //board set-up
    int startx = -1, starty = -1;
    if(board_type.equals("DEFAULT")){
        board = new int[1][N][M];
        for (int i = 0; i < N; i++){
            for (int j = 0; j < M; j++){
                board[0][i][j] = 0;
            }
        }
        startx = 0;
        starty = 0;
    }else if(board_type.equals("CUSTOM")){
        startx = -1; starty = -1;
        board = new int[1][N][M];
        String c;
        for (int i = 0; i < N; i++){
            if (!input.hasNextLine()) {
                System.out.println("Invalid input: not enough rows for custom board");
                System.exit(0);
            }
            c = input.nextLine();
            if (c.length() != M) {
                System.out.println("Invalid input: row length does not match M");
                System.exit(0);
            }
            for (int j = 0; j < M; j++){
                if(c.charAt(j) == 'X'){
                    if(startx == -1){
                        startx = i;
                        starty = j;
                    }
                    board[0][i][j] = 0;
                }else if(c.charAt(j) == '.'){
                    board[0][i][j] = -1;
                }else{
                    System.out.println("Invalid input: invalid character in custom board");
                    System.exit(0);
                }
            }
        }

    }else if(board_type.equals("PYRAMID")){
        board = new int[N][N][M]; //top left always valid
        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                for (int k = 0; k < M; k++){
                    if(j<=i && k<=i){
                        board[i][j][k] = 0;
                    }else{
                        board[i][j][k] = -1;
                    }
                }
            }
```

```
            }
        }
    }else{//error, invalid\
        board = new int[0][0][0];
        System.out.println("Invalid board type");
        System.exit(0);
    }

    //pieces set-up
    int[] used_piece = new int[26];
    for (int i = 0; i < 26; i++){
        used_piece[i] = 1;
    }

    @SuppressWarnings("unchecked")
    ArrayList<ArrayList<Integer>>[] piece = new ArrayList[26];
    for (int i = 0; i < 26 ; i++) {
        piece[i] = new ArrayList<>();
    }
    String piece_layer;
    char pos = ']', temp = ']';
    int pos_idx = 0;
    boolean START = true;
    while(input.hasNextLine()){  //read pieces
        piece_layer = input.nextLine();
        //cek piece ke-
        if (START){
            START = false;
            pos = blockStringChecker(piece_layer);
            used_piece[pos-'A'] = 0;
            pos_idx++;
        }else{
            temp = blockStringChecker(piece_layer);
            if(temp != pos){
                pos = temp;
                pos_idx++;
                if(used_piece[pos-'A'] == 0){
                    System.out.println("Invalid input: duplicate piece");
                    System.exit(0);
                }
                used_piece[pos-'A'] = 0;
            }
        }

        //debug
        System.out.println("OK"+pos_idx);

        //kelebihan piece
        if(pos_idx > B){
            System.out.println("Invalid input: too many pieces");
            System.exit(0);
        }

        ArrayList<Integer> tempPiece = new ArrayList<>();
        for (int i = 0; i < piece_layer.length(); i++){
            if(piece_layer.charAt(i) == ' '){
                tempPiece.add(0);
            }else{
                tempPiece.add(1);
            }
        }
        int pos_num = pos - 'A';
        piece[pos_num].add(tempPiece);


    }

    System.out.println("Input read successfully.\n");
    input.close();
```

```java
//sqauring matirx
for (int i = 0; i < 26; i++){
    int maks = 0;
    for (int j = 0; j < piece[i].size(); j++){
        if(piece[i].get(j).size() > maks){
            maks = piece[i].get(j).size();
        }
    }
    for (int j = 0; j < piece[i].size(); j++){
        while(piece[i].get(j).size() < maks){
            piece[i].get(j).add(0);
        }
    }
}

//print pieces (DEBUG)
for (int i = 0; i < 26; i++){
    if(used_piece[i] == 1){
        continue;
    }
    System.out.println("Piece " + (i+1) + ":");
    for (int j = 0; j < piece[i].size(); j++){
        for (int k = 0; k < piece[i].get(j).size(); k++){
            System.out.print(piece[i].get(j).get(k) + " ");
        }
        System.out.println();
    }
}
```

```java
//print the board
System.out.println("Board:");
int isPyramid = 0;
if(board_type.equals("PYRAMID")){
    isPyramid = 1;
}
printBoard(board, N, M, isPyramid);

//solve
if (startx == -1 && !(board_type.equals("PYRAMID"))) {
    System.out.println("Error: No starting point found");
    System.exit(0);
}

int tries = -1;
startTimer();
if(board_type.equals("DEFAULT") || board_type.equals("CUSTOM")){
    // System.out.print(startx + " " + starty);
    tries = recSolver(board, piece, used_piece, N, M, B, idx:0, h:0, startx, starty, tries:1, isPyramid);
}else{  //PYRAMID
    tries = recSolver(board, piece, used_piece, N, M, B, idx:0, h:0, x:0, y:0, tries:1, isPyramid);
}
System.out.println("No solution found.\nTries: " + tries);
scanner.close();
stopTimer();
System.exit(0);

//---------------------
    }
}
```

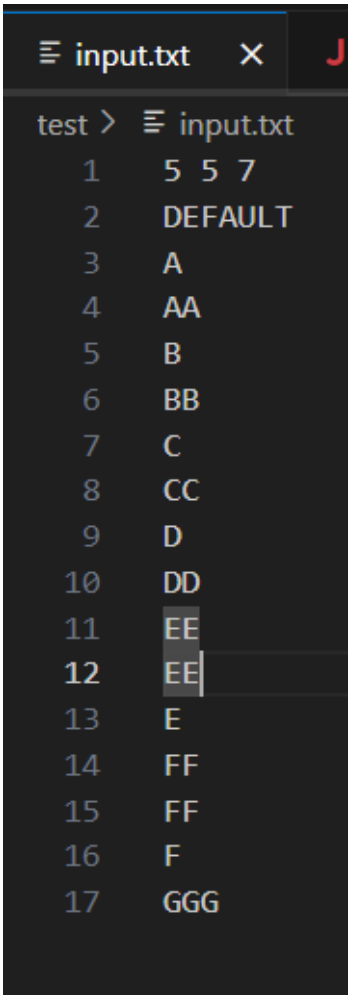
- 

# Input/Output:

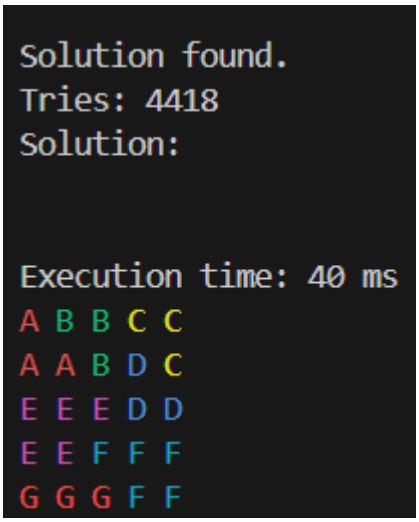
1. Gambar memilih nama file

```
PS C:\CODING\CODING\Java\tubes\STIMA-Tucil1> java src/IqPuzzlePro.java
Enter the filename (must be in the test folder): input
```

2. Input DEFAULT 5 5 7




Output DEFAULT 5 5 7




Menyimpan dalam bentuk .txt

```
Save answer to file? (Y/N): y
Enter filename(without .txt): test1
File saved successfully.
```

test1.txt U

save > test1.txt

```
1  A B B C C
2  A A B D C
3  E E E D D
4  E E F F F
5  G G G F F
6
```

3. Input DEFAULT 3 3 3

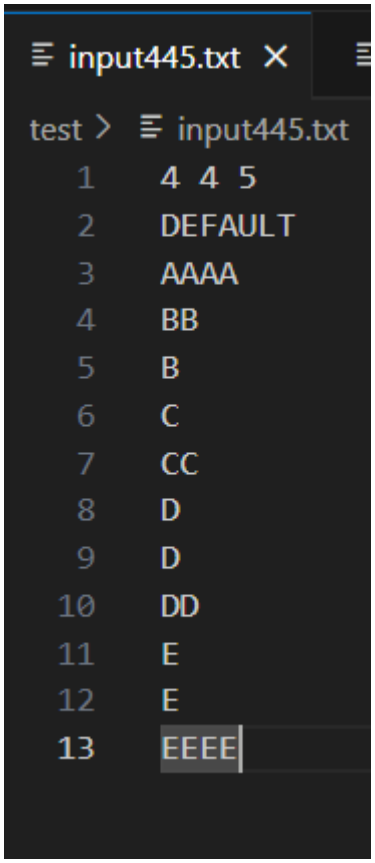
test > input333.txt

```
1  3 3 3
2  DEFAULT
3    AA
4  AA
5  AA
6  B
7  CC
```

Hasil

```
Solution found.
Tries: 1
Solution:


Execution time: 1 ms
B A A
A A C
A A C
```

4. Input DEFAULT 4 4 5, Kelebihan piece/block




```
Test Case invalid.
Tries: 4208

Execution time: 34 ms
Reason: board full with spare piece(s).
A A A A
E B C C
E B B C
E E E E
Unused piece(s):
D
D
D D
```

5. Input CUSTOM 5 5 5

inputC555.txt U

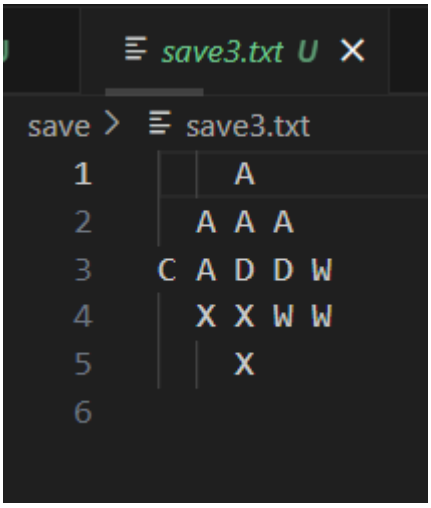
st > inputC555.txt

```
5 5 5
CUSTOM
..X..
.XXX.
XXXXX
.XXXX
..X..
 A
AAA
A
W
WW
C
DD
XX
X
```

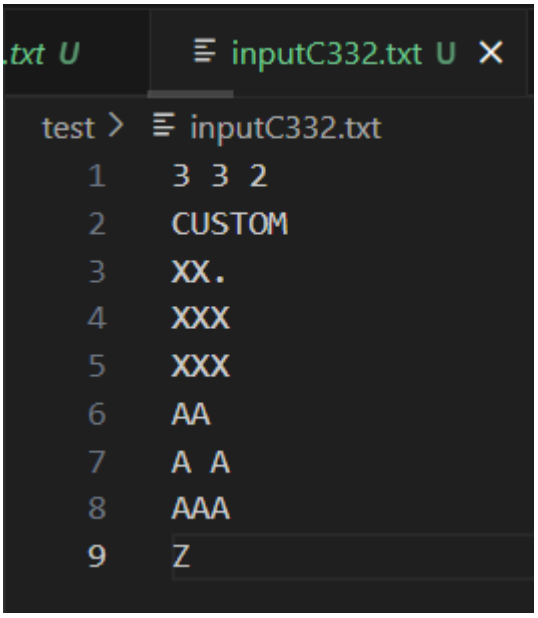
Hasil:

```
Solution found.
Tries: 2
Solution:


Execution time: 0 ms
    A
  A A A
C A D D W
  X X W W
    X
```

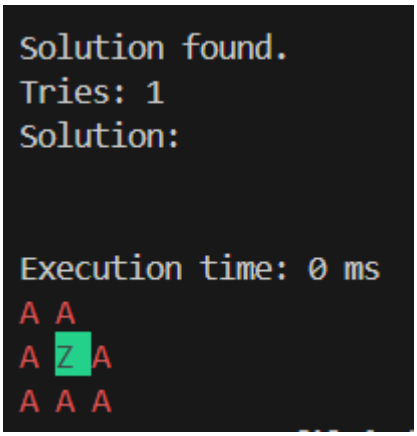
Hasil dalam file:



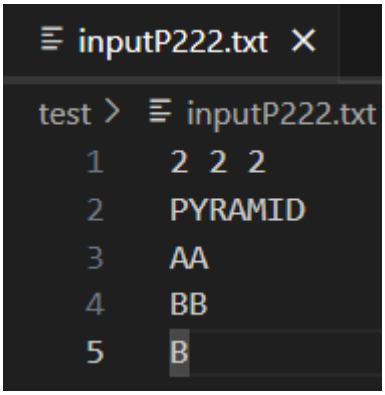

6. Input CUSTOM 3 3 2



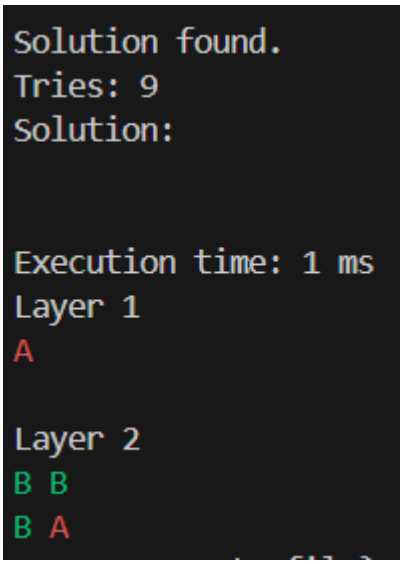

Hasil:



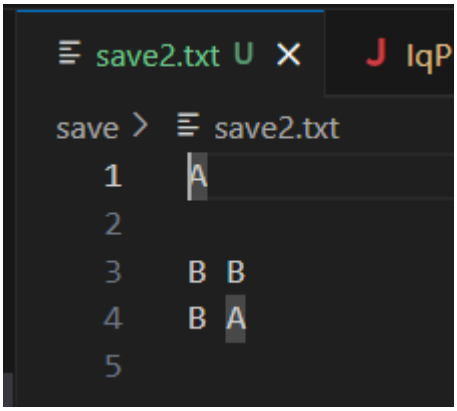

7. Input PIRAMIDA 2 2 2



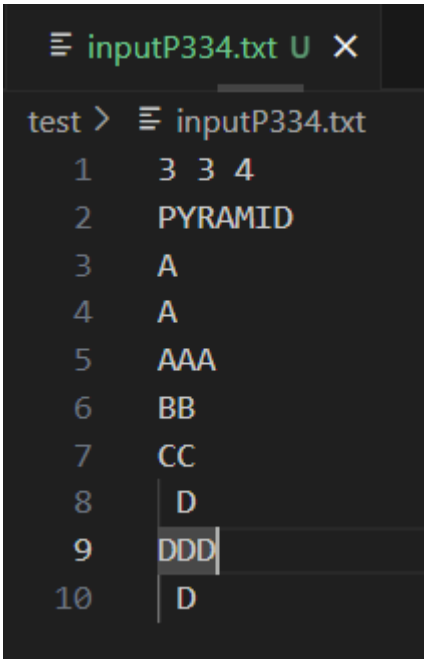

Hasil Terminal



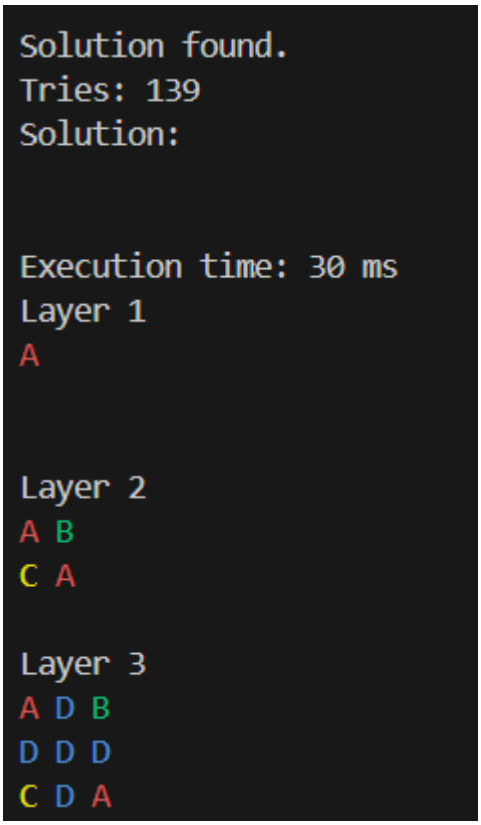

Hasil .txt



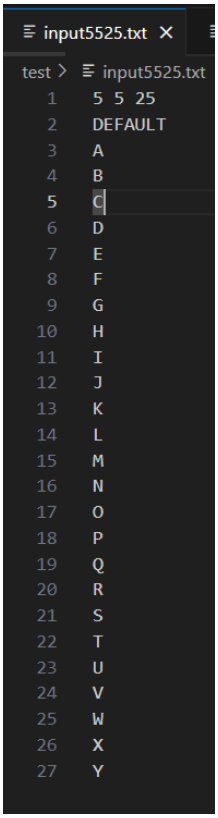

8. Input PIRAMIDA 3 3 4



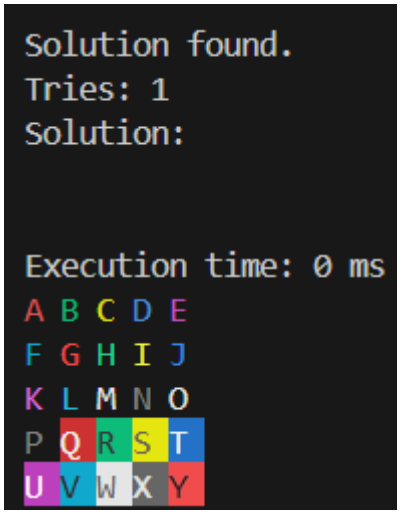

Hasil:




9. Input DEFAULT 5 5 25




Hasil:




# Repository: https://github.com/mimiCrai/Tucil1_13523054

# LAMPIRAN:

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