IF2123 STRATEGI ALGORITMA
TUGAS KECIL 1

Oleh:
13523024 - Richard Christian

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2025

DAFTAR ISI

BAB I DESKRIPSI MASALAH **3**
Spesifikasi Wajib **3**
Algoritma Brute Force **4**
Puzzle IQ Puzzler Pro **4**
BAB II IMPLEMENTASI **5**
Kelas Main **5**
Kelas Input **6**
Kelas Piece **8**
Kelas Board **10**
Kelas Brute **12**
BAB III TEST CASE **14**
LAPORAN **18**
LAMPIRAN **18**
I. Link Repository 18
II. Link Referensi 18

BAB I DESKRIPSI MASALAH

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**.

- **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).

Algoritma Brute Force

Algoritma Brute Force adalah algoritma yang memecahkan masalah secara *straightforward* untuk memecahkan persoalan. Pemecahan masalah dengan algoritma brute force

biasanya sangat sederhana, langsung, jelas caranya, dan mudah dipahami. Penulisan algoritma brute force biasanya berdasarkan pada pernyataan persoalan dan konsep yang dilibatkan.

Algoritma Brute Force memiliki berbagai kelemahan, seperti kurangnya adaptabilitas dan kebutuhan kekuatan komputasi yang biasanya lebih tinggi dibandingkan algoritma yang lain. Waktu pengerjaan algoritma brute force juga biasanya meningkat secara eksponensial sehingga kurang sesuai untuk permasalahan yang kompleks. Walaupun begitu, algoritma brute force tetap dapat digunakan untuk memecahkan permasalahan yang sederhana karena mudahnya penerapan algoritma ini.

Puzzle IQ Puzzler Pro

Dalam puzzle IQ Puzzler Pro, pemain akan mendapatkan sebuah papan, umumnya berukuran 11x5 dengan 12 blok yang dapat disimpan di dalam papan. Blok akan mempunyai bentuk yang unik dan pemain perlu menempatkan blok pada papan hingga semua blok habis dan tidak ada tempat tersisa pada papan. Pada algoritma ini, diasumsikan bahwa papan dimulai dengan kosong.

Kompleksitas dari permainan ini datang dari banyaknya tempat yang bisa diisi blok, dan bentuk blok yang unik sehingga setiap blok dapat diputar dan dibalikkan. Ini berarti setiap blok dapat ditempatkan di berbagai tempat pada papan, dengan konfigurasi yang bervariasi. Dalam algoritma ini, solusi yang akan dicari hanya untuk konfigurasi 2 Dimensi.

BAB II IMPLEMENTASI

Implementasi program dikerjakan menggunakan bahasa pemrograman *Java,* dengan pendekatan *object oriented programming.* File input pada program perlu dimasukkan dalam folder test, dan output akan disimpan pada folder yang sama.

Kelas Main

Kelas Main berisi fungsi:
Void main(String[] args)
Fungsi ini berfungsi sebagai Main dari program ketika dijalankan. Main menggabungkan kelas-kelas lain sehingga menjadi satu program, sekaligus menerima input dan output terminal dari user.

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print(s:"Enter input file name (with .txt, located in 'test/' folder): ");
        String filename = scanner.nextLine();
        String filePath = "test/" + filename;

        Input input = new Input();
        input.readInputFromFile(filePath);
        int npieces = input.getNpiece();
        if (input.getPieces().size() < npieces){
            System.out.println(x:"Not enough pieces");
        } else {
        int[] dimensions = input.getBoardDim();
        Board board = new Board(dimensions[0], dimensions[1]);
        Brute brute = new Brute(board, input.getPieces());

        long startTime = System.currentTimeMillis();
        boolean solved = brute.solve(index:0);
        long endTime = System.currentTimeMillis();
        long timeTaken = endTime - startTime;

        if (solved) {
            board.printBoard();
        } else {
            board.printBoard();
            System.out.println(x:"No solutions found.");
        }
        System.out.println("Attempts: " + brute.getAttempts());
        System.out.println("Time taken: " + timeTaken + " ms");

            Output.writeToFile(board.getGrid(), brute.getAttempts(), timeTaken);
        }
            scanner.close();
    }
}
```

Kelas Input

Kelas Input berisi fungsi:

1. Input()

Fungsi ini adalah konstruktor dari kelas Input. Menggunakan ArrayList untuk memudahkan penambahan Piece kedalam list.

```
public Input() {
    this.pieces = new ArrayList<>();
}
```

2. void main(String[] args()

Fungsi ini digunakan untuk pengetesan awal pada input file, dan dapat digunakan untuk mengecek hasil bacaan diluar menjalankan program utama.

```
public static void main(String[] args) {
    Input input = new Input();
    input.readInputFromFile(filename:"input.txt");
    input.printStoredPieces();
}
```

3. void readInputFromFIle(String Filename)

Fungsi ini menerima lokasi file dengan filename, kemudian menggunakan scanner untuk membaca isi file .txt. Pembacaan dimulai dengan mengambil 3 integer teratas, dan menyimpannya dalam n, m, dan p. nxm merepresentasikan ukuran papan, dan p merepresentasikan jumlah piece yang ada. Pembacaan mode diloncat karena hanya mode default yang tersedia. Pembacaan piece lalu dilakukan secara iteratif dan disimpan dalam list Piece. Rotasi dan Mirroring pada piece diset 0 pada saat awal dimasukkan.

```java
public void readInputFromFile(String filename) {
    //header awalan
    try (Scanner scanner = new Scanner(new File(filename))) {
        n = scanner.nextInt();
        m = scanner.nextInt();
        p = scanner.nextInt();
        scanner.nextLine();
        scanner.nextLine();
        List<int[]> shapeCoords = new ArrayList<>();
        char currentChar = '#';
        int i = 0;
        while (scanner.hasNextLine()) { //baca isi
            String line = scanner.nextLine();
            if(line.trim().isEmpty())
                continue;
            char firstChar = line.trim().charAt(index:0);

            if (currentChar== '#') {
            currentChar = firstChar;
            }
            if(firstChar != currentChar) {
                pieces.add(new Piece(currentChar, shapeCoords, rot:0, mir:0));
                shapeCoords = new  ArrayList<>();
                currentChar = firstChar;
                i = 0;
                if (pieces.size() > p-1) { //kalo kelebihan (tetep diitung yang ga lebih)
                    System.err.println(x:"Too many pieces");
                    break;
                }
            }

            for (int col =0; col < line.length(); col++) {
                if (line.charAt(col) == currentChar) {
                    shapeCoords.add(new int[]{i, col});
                }
            }
            i++  ;
         }
        if (!shapeCoords.isEmpty()) {
            pieces.add(new Piece(currentChar, shapeCoords, rot:0, mir:0));
        }
    } catch (FileNotFoundException e) {
        System.err.println(x:"File Not Found");
    }

    }
```

4. List<Piece> getPieces()

Fungsi mengembalikan list piece

```java
public List<Piece> getPieces() {
    return pieces;
}
```

5. Int[] getBoardDim()

Fungsi mengembalikan Dimensi Board

```java
public int[] getBoardDim() {
    return new int[]{n, m};
}
```

6. Void printStoredPieces()

Fungsi untuk mengecek piece yang terbaca, dengan menggunakan bantuan fungsi printshape dan menampilkan rotasi dan mirroring pada piece.

```
public void printStoredPieces() {
    for (Piece piece : pieces) {
        System.out.println(piece.getShape() + " Rot: " +piece.getRot() + " Mir: " +piece.getMir());
        piece.printShape();
        System.out.println();}
}
```

7. int getNpiece()

Mengembalikan jumlah piece yang seharusnya ada sesuai p pada file input.

```
public int getNpiece() {
    return  p;
}
```

Kelas Piece

Kelas Piece berisi fungsi:

1. Piece(char shape, List<int[]> coords, int rot, int mir)

Konstruktor Piece, menerima huruf piece, koordinat piece, dan rotasi dan mirroring pada piece.

```
public Piece(char shape, List<int[]> coords, int rot, int mir) {
    this.shape = shape;
    this.coords = new ArrayList<>(coords);
    this.rot = rot;
    this.mir = mir;
}
```

2. Char getShape()

Mengembalikan huruf piece.

```
public char getShape() {
    return shape;
}
```

3. Int getRot()

Mengembalikan rotasi pada piece.

```
public int getRot() {
    return rot;
}
```

4. Int getMir()

Mengembalikan mirroring pada piece.

```
public int getMir() {
    return mir;
}
```

5. List<int[]> getCoords()

Mengembalikan bentuk piece dalam koordinat.

```
public List<int[]> getCoords() {
    return coords;
}
```

6. void rotate()

Mengembalikan piece yang sudah diputar 90 derajat, menambahkan 1 pada rotasi piece.

```
public void rotate() {
    List<int[]> coord2 = new ArrayList<>();
    for (int[] coord : coords) {
        coord2.add(new int[]{coord[1],  -coord[0]});
    }
    coords = coord2;
    rot = (rot +1)%4;
    norm();
}
```

7. Void mirror()

Mengembalikan piece yang sudah dibalikkan, menambah mirroring 1 pada piece.

```
public void mirror() {
    List<int[]> coord2 = new ArrayList<>();
    for (int[] coord : coords) {
        coord2.add(new int[]{coord[0],  -coord[1]});
    }
    coords = coord2;
    mir = 1 -mir;
    norm();
    }
```

8. Void norm()

Menormalisasi piece pada 0,0, sebagai bantuan dalam fungsi rotate dan mirror.

```java
public void norm() {
    if (coords.isEmpty()) {
        return;
    }
    int minRow = Integer.MAX_VALUE;
    int minCol = Integer.MAX_VALUE;


    for (int[] coord : coords) {
        minRow = Math.min(minRow, coord[0]);
        minCol = Math.min(minCol, coord[1]);
    }
    for (int[] coord : coords) {
        coord[0] -= minRow;
        coord[1] -= minCol;
    }
}
```

9. Void printShape()

Mencetak koordinat pada piece, untuk membantu pada pengetesan awal.

```java
public void printShape() {
    for (int[] coord : coords) {
        System.out.println(coord[0]+ "," +  coord[1]);
    }
}
```

Kelas Board

Kelas Board berisi fungsi:

1. Board(int rows, int cold)

Konstruktor Board, menerima ukuran board

```java
public Board(int rows, int cols) {
    this.rows = rows;
    this.cols = cols;
    this.grid = new char[rows][cols];
    for (char[] row : grid) Arrays.fill(row,val:'.');
}
```

.

2. boolean canPlace(Piece piece, int r, int c)

Mengecek apakah sebuah piece dapat disimpan dalam koordinat pada board. Menggunakan looping dari koordinat piece pada papan, dan mengecek bila semua posisi kosong.

```
public boolean canPlace(Piece piece, int r, int c) {
    for (int[] coord : piece.getCoords()) {
        int nr = r +coord[0], nc = c +coord[1];
        if (nr < 0 || nr >= rows|| nc < 0 || nc >= cols|| grid[nr][nc] != '.') {
            return false;}
    }
    return true;
}
```

3. void placePiece(Piece piece, int r, int c)

Menyimpan piece pada papan.

```
public void placePiece(Piece piece, int r, int c) {
    for (int[] coord : piece.getCoords()){
        grid[r + coord[0]][c + coord[1]] = piece.getShape();
    }
}
```

4. void removePiece(Piece piece, int r, int c)

Mengeluarkan piece pada papan, papan kembali kosong pada posisi piece.

```
public void removePiece(Piece piece, int r, int c) {
    for (int[] coord : piece.getCoords()){
        grid[r + coord[0]][c + coord[1]] = '.';
    }
}
```

5. void printBoard()

Mencetak papan bermain, dengan bantuan ANSI untuk mengeluarkan output berwarna pada terminal.

```
public void printBoard() {
    System.out.println(x:"\nCurrent Board:");
    for (char[] row : grid) {
        for (char cell : row) {
                if (cell == '.') {
                    System.out.print(cell + " ");
                } else {
                    String color = COLORS[(cell - 'A') % COLORS.length];
                    System.out.print(color + cell + RESET + " ");
                }
        }
        System.out.println();
    }
}
```

6. int[] topLeft(Piece piece)

Memastikan bahwa piece dipasang dari kiri atas papan, bertujuan untuk meningkatkan efisiensi brute force dengan mengurangi iterasi yang sudah pasti tidak valid.

```
public int[] topLeft(Piece piece) {
    for (int r = 0; r <rows; r++) {
        for (int c = 0; c <cols; c++) {
            if (canPlace(piece, r, c)){
                return new int[]{r, c};}
        }
    }
    return null;
}
```

7. boolean isFull()

Mengecek bila papan sudah terisi penuh.

```
public boolean isFull() {
    for (char[] row : grid) {
        for (char cell : row) {
            if (cell =='.') {return false ;}
        }
    }
    return true;
}
```

8. char[][] getGrid()

Mengembalikan grid permainan.

```
public char[][] getGrid() {
    return grid;
}
```

Kelas Brute

Kelas Brute berisi fungsi:

1. Brute(Board board, List<Piece> pieces)

Konstruktor Brute untuk proses algoritma brute force utama.

2. boolean solve(int index)

Proses brute force utama. Program didesain secara brute force dengan tambahan rekursif untuk mengurangi iterasi yang sudah pasti salah. Iterasi dilakukan dengan mengiterasi sejumlah piece, dimana urutan piece ditukar dengan menggunakan bantuan Collections.swap(). Urutan piece lalu dimasukkan satu per satu pada papan, dengan mengubah rotasi dan mirroring dari piece bila tidak sesuai. Piece lalu disimpan pada bagian kiri atas yang memungkinkan.

Sebelum Piece dimasukkan, dicek bahwa tidak ada baris yang kosong diatas piece tersebut. Ini memungkinkan karena setiap piece disimpan dari kiri atas, sehingga bila ada kekosongan diatas

baris disisipkannya piece, maka iterasi tersebut pastillah salah, sehingga return false digunakkan untuk keluar dari proses tersebut. Bila tidak ada kekosongan, maka piece disimpan pada papan, dan perhitungan attempt ditambah satu. “Processing…” dicetak untuk memastikan program berjalan. Rekursi ini lalu dilakukan pada setiap piece hingga ditemukan solusi dimana setiap piece tersimpan, atau tidak ditemukan solusi setelah setiap kemungkinan penyimpanan telah dicoba.

Karena program yang bersifat brute force, dengan kompleksitas yang tinggi, maka program dapat memakan waktu yang sangat lama untuk mengisi papan yang besar dengan piece yang banyak. Ini dikarenakan kemungkinan yang ditelusuri merupakan perkalian ukuran papan, jumlah dari setiap piece, kemungkinan urutan piece, dan kemungkinan orientasi piece.

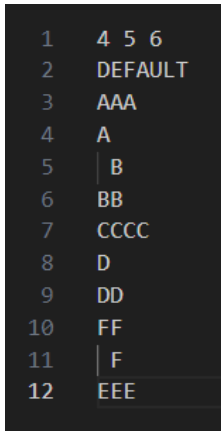
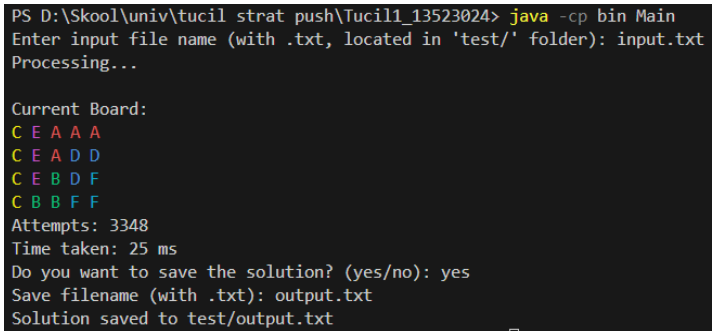
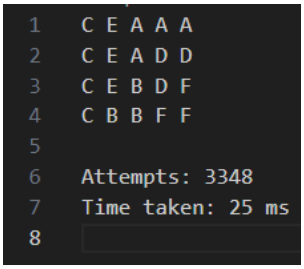
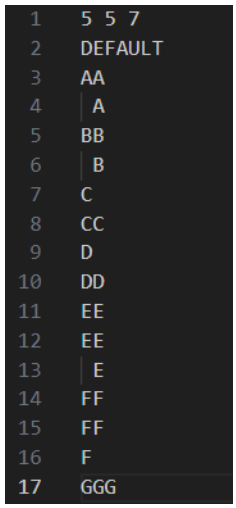
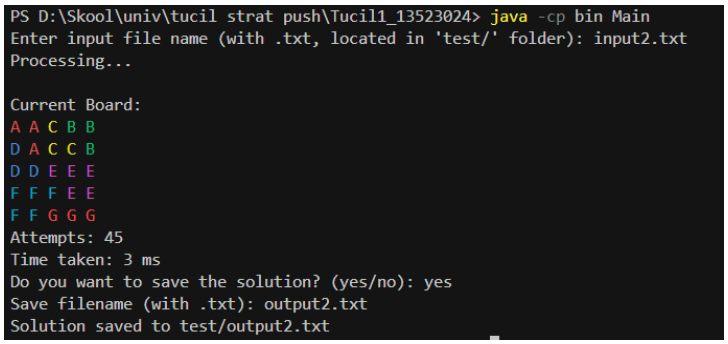
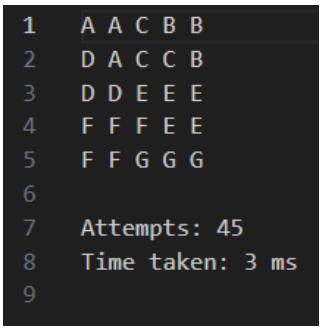
```java
public boolean solve(int index) {
    if (index >= pieces.size()) {
        if (board.isFull()) {
            return true;
        }
    }
    for (int i = index; i < pieces.size(); i++) {
        Collections.swap(pieces, index, i); //nuker dari list pieces

        Piece piece = pieces.get(index); //ngambil piece
        for (int rot = 0; rot < 4; rot++) {
            for (int mir = 0; mir < 2; mir++) {
                int[] bestPos = board.topLeft(piece);
                if (bestPos != null) {
                    int r = bestPos[0], c = bestPos[1];

                    for (int k = 0; k < r; k++) {
                        for (int nc = 0; nc < board.cols; nc++) {
                            if (board.grid[k][nc] == '.') {return false;} //cek baris atas terisi
                        }
                    }

                    board.placePiece(piece, r, c);
                    attempts++;
                    if (attempts == 2) {
                        System.out.println(x:"Processing... "); //biar keliatan jalan
                    }
                    if (solve(index + 1)) {return true;}
                    board.removePiece(piece, r, c);
                }
                piece.mirror();
            }
            piece.rotate();
        }

        Collections.swap(pieces, index, i);
    }
    return false;
}
```

3. long getAttempts()

Fungsi ini digunakan untuk mengembalikan attempts pada proses brute force.

```java
public long getAttempts() {
    return attempts;
}
```

BAB III TEST CASE

No	Input	Terminal	Output
1			
2			

<table>
<tr>
<td>3</td>
<td>

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

</td>
<td>

```
PS D:\Skool\univ\tucil strat push\Tucil1_13523024> java -cp bin Main
Enter input file name (with .txt, located in 'test/' folder): input2a.txt
Processing...

Current Board:
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
No solutions found.
Attempts: 34207
Time taken: 91 ms
Do you want to save the solution? (yes/no): yes
Save filename (with .txt): output2a.txt
Solution saved to test/output2a.txt
```

</td>
<td>

```
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .

Attempts: 34207
Time taken: 91 ms

```

</td>
</tr>
<tr>
<td>4</td>
<td>

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

</td>
<td>

```
PS D:\Skool\univ\tucil strat push\Tucil1_13523024> java -cp bin Main
Enter input file name (with .txt, located in 'test/' folder): input2b.txt
Not enough pieces
```

</td>
<td>Tidak ada output</td>
</tr>
</table>

5

```
6 6 10
DEFAULT
A
AA
B
BB
C
CC
C
D
DD
EE
EE
E
F
FF
F
GGGG
HHH
III
I
JJ
J
```

```
PS D:\Skool\univ\tucil strat push\Tucil1_13523024> java -cp bin Main
Enter input file name (with .txt, located in 'test/' folder): input3.txt
Processing...

Current Board:
A A G G G G
A B F E E E
B B F F E E
D D F H H H
I D C C C J
I I I C J J
Attempts: 1912
Time taken: 24 ms
Do you want to save the solution? (yes/no): yes
Save filename (with .txt): output3.txt
Solution saved to test/output3.txt
```

```
A A G G G G
A B F E E E
B B F F E E
D D F H H H
I D C C C J
I I I C J J

Attempts: 1912
Time taken: 24 ms

```

6

```
4 8 9
DEFAULT
AA
 A
BBB
CC
 CC
DD
E
EEEE
E
FFF
GGG
 G
HHH
III
I
```

```
PS D:\Skool\univ\tucil strat push\Tucil1_13523024> java -cp bin Main
Enter input file name (with .txt, located in 'test/' folder): input4.txt
Processing...

Current Board:
A A B B B F F F
E A H H H I I I
E E E E G C C I
E D D G G G C C
Attempts: 299010
Time taken: 872 ms
Do you want to save the solution? (yes/no): yes
Save filename (with .txt): output4.txt
Solution saved to test/output4.txt
```

```
A A B B B F F F
E A H H H I I I
E E E E G C C I
E D D G G G C C

Attempts: 299010
Time taken: 872 ms

```

6

```
1 26 26
DEFAULT
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
Z
```

```
PS D:\Skool\univ\tucil strat push\Tucil1_13523024> java -cp bin Main
Enter input file name (with .txt, located in 'test/' folder): input5.txt
Processing...

Current Board:
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 Z
Attempts: 26
Time taken: 1 ms
Do you want to save the solution? (yes/no): yes
Save filename (with .txt): output5.txt
Solution saved to test/output5.txt
```

```
A B C D E F G H I J K

Attempts: 26
Time taken: 1 ms

L M N O P Q R S T U V W X Y Z
```

LAPORAN

-

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	✓	

LAMPIRAN

I. Link Repository
https://github.com/doober22/Tucil1_13523024.git

II. Link Referensi
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/stima24-25.htm