# IF2211 – K01

## TUGAS KECIL 1 STRATEGI ALGORITMA

## PROGRAM IQ PUZZLER PRO

Felix Chandra 13523012

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2024**

# Daftar Pustaka

# Bab 1

# Deskripsi Masalah

Tugas Besar 1 untuk mata kuliah Strategi Algoritma dengan kode mata kuliah IF2211 memberikan permasalahan berupa pembentukan suatu program menggunakan bahasa pemrograman Java, yang merupakan bahasa pemrograman dengan orientasi objek, untuk menyelesaikan permainan IQ Puzzler Pro.

IQ Puzzler Pro merupakan 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.

**(Sumber: https://www.smartgamesusa.com)**

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

Program diminta untuk 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.

# Bab 2

# Penjelasan Program

Program menyelesaikan masalah menggunakan algoritma brute force dimana program mencoba semua kemungkinan dari posisi block untuk menemukan solusi dari testcase IQ puzzler pro yang diberikan.

Penyelesaian dimulai dengan cara menerima input berupa file .txt yang berisikan ukuran papan, jumlah blocks dan bentuk block. Format file adalah sebagai berikut:
N M P
puzzle_1_shape
puzzle_2_shape
…
puzzle_P_shape

perhatikan bahwa tidak seperti pada spesifikasi saya tidak meminta inputan berupa S karena program saya hanya menyelesaikan puzzle default bukan pyramid dan custom.

blocks disimpan menggunakan kelas piece yang terdiri dari id berupa character (A,B,C,....,P) dan List yang menyimpan semua kemungkinan bentuk dari blocks yang nantinya diperoleh dari rotasi dan refleksi.

```
class Piece {
    char id;
    List<List<Coordinate>> transformations;
```

```java
private List<Coordinate> rotate(List<Coordinate> coords, int rotations) {
    List<Coordinate> rotated = new ArrayList<>();
    for (Coordinate coord : coords) {
        int r = coord.r;
        int c = coord.c;
        for (int i = 0; i < rotations; i++) {
            int newR = c;
            int newC = -r;
            r = newR;
            c = newC;
        }
        rotated.add(new Coordinate(r, c));
    }
    return normalize(rotated);
}

private List<Coordinate> reflect(List<Coordinate> coords, boolean horizontal) {
    List<Coordinate> reflected = new ArrayList<>();
    for (Coordinate coord : coords) {
        int r = coord.r;
        int c = coord.c;
        if (horizontal) {
            c = -c;
        } else {
            r = -r;
        }
        reflected.add(new Coordinate(r, c));
    }
    return normalize(reflected);
}
```

Setelah setiap blocks diperoleh, dilakukan transformasi pada setiap blocks yaitu dirotasi 90,180,270 derajat dan juga direfleksikan secara vertikal dan horizontal untuk menghasilkan semua kemungkinan bentuk.

Total area blocks akan dibandingkan dengan luas papan. Apabila tidak sama program akan berhenti dan mengeluarkan pesan “`Tidak ada solusi karena total luas blok puzzle tidak sama dengan luas papan.`“

```java
int totalArea = 0;
for (Piece piece : pieces) {
    totalArea += piece.transformations.get(0).size();
}
if (totalArea != N * M) {
    System.out.println("Tidak ada solusi karena total luas blok puzzle tidak sama dengan luas papan.");
    return;
}
```

Apabila ukuran total luas blocks dan papan sama dilanjutkan dengan pencarian solusi yang dilakukan dengan Algoritma backtracking untuk mencoba semua kemungkinan penempatan blok puzzle di papan.

Langkah-langkahnya:

1. Cari Posisi Kosong

Program mencari posisi kosong pertama di papan menggunakan metode findEmpty

```java
private int[] findEmpty(char[][] board) {
    for (int r = 0; r < N; r++) {
        for (int c = 0; c < M; c++) {
            if (board[r][c] == '\0') {
                return new int[]{r, c};
            }
        }
    }
    return null;
}
```

2.Coba Tempatkan Blok Puzzle

Untuk setiap blok puzzle yang tersedia, program mencoba menempatkannya di posisi kosong dengan semua transformasi yang mungkin.

Jika blok puzzle dapat ditempatkan (tidak bertabrakan dengan blok lain dan masih dalam batas papan), program akan menempatkannya.

```java
private boolean canPlace(char[][] board, int r, int c, List<Coordinate> coordinates) {
    for (Coordinate coord : coordinates) {
        int nr = r + coord.r;
        int nc = c + coord.c;
        if (nr < 0 || nr >= N || nc < 0 || nc >= M || board[nr][nc] != '\0') {
            return false;
        }
    }
    return true;
}

private void place(char[][] board, int r, int c, List<Coordinate> coordinates, char id) {
    for (Coordinate coord : coordinates) {
        board[r + coord.r][c + coord.c] = id;
    }
}
```

3.Rekursi

Program memanggil dirinya sendiri secara rekursif untuk mencoba menempatkan blok puzzle berikutnya.

4.Backtrack

Jika penempatan blok puzzle tidak mengarah ke solusi, program akan menghapus blok tersebut dan mencoba transformasi atau blok puzzle lainnya.

```java
private boolean backtrack(List<Piece> remaining, char[][] board) {
    if (remaining.isEmpty()) {
        solutionBoard = board;
        return true;
    }

    int[] pos = findEmpty(board);
    if (pos == null) {
        return false;
    }
    int r = pos[0], c = pos[1];

    for (int i = 0; i < remaining.size(); i++) {
        Piece piece = remaining.get(i);
        List<Piece> newRemaining = new ArrayList<>(remaining);
        newRemaining.remove(i);

        for (List<Coordinate> transformation : piece.transformations) {
            cases++;
            if (canPlace(board, r, c, transformation)) {
                char[][] newBoard = copy(board);
                place(newBoard, r, c, transformation, piece.id);
                if (backtrack(newRemaining, newBoard)) {
                    return true;
                }
            }
        }
    }
    return false;
}
```

5.Solusi Ditemukan

Jika semua blok puzzle berhasil ditempatkan dan papan terisi penuh, program menampilkan hasil/solusi, jumlah iterasi dan juga waktu yang digunakan untuk menjalankan program serta menyimpan solusi dalam bentuk file.txt jika diminta oleh pengguna. Apabila tidak diperoleh maka akan mengeluarkan pesan “Tidak Ada Solusi”.

```java
if (solver.solve()) {
    solver.printSolution();
    System.out.print("Apakah anda ingin menyimpan solusi? (ya/tidak) ");
    String answer = br.readLine().trim().toLowerCase();
    if (answer.equals("ya")) {
        System.out.print("Masukkan nama file output: ");
        String outputFile = br.readLine();
        solver.saveSolution(outputFile);
    }
} else {
    System.out.println("Puzzle tidak memiliki solusi.");
}
```

# Bab 3

# Eksperimen

**Test case 1:**

stima > src > a.txt

```
1   5 5 7
2   A
3   AA
4   B
5   BB
6   C
7   CC
8   D
9   DD
10  EE
11  EE
12  E
13  FF
14  FF
15  F
16  GGG
```

**Solusi test case 1:**

```
felix@Mac src % cd "/Users/felix/Documents/Java/st
Masukkan nama file input: a.txt
Waktu pencarian: 7 ms
Banyak kasus yang ditinjau: 41254
A B B C C
A A B D C
E E E D D
E E F F F
G G G F F
Apakah anda ingin menyimpan solusi? (ya/tidak) ya
Masukkan nama file output: SolusiA.txt
```

ma > src > SolusiA.txt

```
1   ABBCC
2   AABDC
3   EEEDD
4   EEFFF
5   GGGFF
```

**Test case 2:**

```
stima > src > b.txt
1   3 3 3
2   A
3    A
4     A
5   BB
6   B
7   C
8   CC
```

**Solusi test case 2:**

```
Masukkan nama file input: b.txt
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 16
A B B
A C B
A C C
Apakah anda ingin menyimpan solus
```

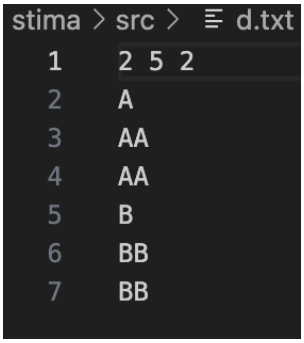
**Test case 3:**

```
stima > src > c.txt
1   4 4 4
2   A
3   AA
4   AAA
5   B
6   BB
7   C
8   C
9   C
10  D
11   D
12    D
13     D
```

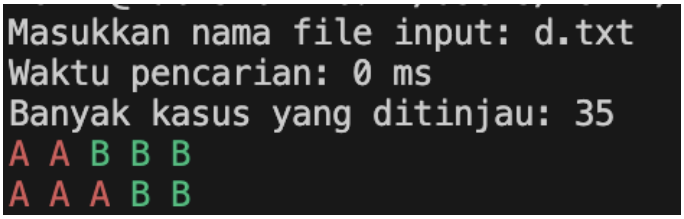
**Solusi test case 3:**

```
Masukkan nama file input: c.txt
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 11
A B B C
A A B C
A A A C
D D D D
Apakah anda ingin menyimpan solus
```

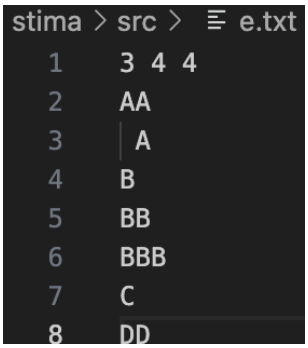
**Test case 4:**



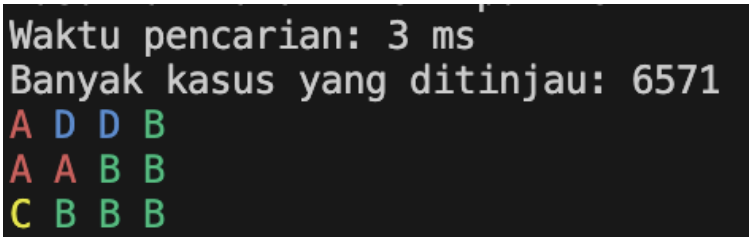

**Solusi test case 4:**




**Test case 5:**




**Solusi test case 5:**




**Test case 6:**

stima > src > f.txt

```
4 4 4
FFF
F
FFF
F
AA
B
B
B
BB
C
```

**Solusi test case 6:**

```
Masukkan nama file input: f.txt
Waktu pencarian: 2 ms
Banyak kasus yang ditinjau: 5520
F C B B
F F F B
F A A B
F F F B
```

**Test case 7:**

stima > src > g.txt

```
5 5 7
AAA
A A
AAA
B
C
C
C
CC
CC
DDD
EE
FF
F
G
```

**Solusi test case 7:**

```
Masukkan nama file input: g.txt
Waktu pencarian: 13 ms
Banyak kasus yang ditinjau: 10680
A A A B E
A G A F E
A A A F F
C C C C C
C C D D D
```

**Test case 8:**

stima > src > h.txt

```
1   2 2 2
2   A
3   A
4   A
5   B
```

puzzle tidak bisa diselesaikan karena block A berukuran 3x1 sedangkan ukuran papan hanya 2x2

**Solusi test case 8:**

```
Masukkan nama file input: h.txt
Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 168
Puzzle tidak memiliki solusi.
```

**Test case 9 (berbeda dengan testcase 1):**

stima > src > i.txt

```
1   5 5 7
2   A
3   AA
4   B
5   BB
6   C
7   CC
8   D
9   DD
10  EE
11  EE
12  E
13  FF
14  FF
15  F
16  GGGG
```

puzzle tidak dapat diselesaikan karena ukuran papan hanya 25 unit sedangkan total luas block seluas 26 unit

**Solusi test case 9:**

```
felix@Mac src % cd "/Users/felix/Documents/Java/stima/src/" && javac Main.jav
Masukkan nama file input: i.txt
Tidak ada solusi karena total luas blok puzzle tidak sama dengan luas papan.
felix@Mac src %
```

# Bab 4

# Penutup

## 1. Kesimpulan

Melalui tugas kecil 1 mata kuliah Strategi Algorima, Penulis menjadi lebih mengerti dengan algoritma bruteforce dan dapat mengimplementasikannya ke dalam program untuk menyelesaikan berbagai macam permasalahan.

# Lampiran

## 1. Tabel Deskripsi Program

| 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 | ✓ | |

## 2. Tautan *repository*

https://github.com/Felix-Chandra-13523012/Tucil1_13523012