

Laporan Tugas Kecil 1
IF2211 Strategi Algoritma

Zulfaqqar Nayaka Athadiansyah — 13523094

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

Daftar Isi

Daftar Tabel	2
Daftar Gambar	3
1 Pendahuluan	5
1.1 Deskripsi Masalah	5
1.2 Struktur Program	5
2 Algoritma	6
2.1 Variabel/Atribut	6
2.2 Prakomputasi (<i>Precomputing</i>)	6
2.3 <i>Backtracking</i>	7
2.4 Integrasi	7
3 Eksperimen	9
3.1 Kasus Uji 1: Berkas txt kosong	9
3.2 Kasus Uji 2: Berkas txt tidak lengkap	9
3.3 Kasus Uji 3: Jumlah keping dalam berkas txt lebih sedikit dari p	10
3.4 Kasus Uji 4: Jumlah keping dalam berkas txt lebih banyak dari p	10
3.5 Kasus Uji 5: Tidak ada solusi	10
3.6 Kasus Uji 6: Papan DEFAULT (1)	11
3.7 Kasus Uji 7: Papan DEFAULT (2) dan fitur menyimpan ke berkas txt baru	11
3.8 Kasus Uji 8: Papan DEFAULT (3)	12
3.9 Kasus Uji 9: Papan DEFAULT (4)	13
3.10 Kasus Uji 10: Papan DEFAULT (5)	13
3.11 Kasus Uji 11: Papan DEFAULT (6)	14
3.12 Kasus Uji 11: Papan DEFAULT (7)	14
4 Lampiran	16
4.1 Pranala Repositori	16
4.2 Kode Sumber	16
4.3 Tabel Kelengkapan	26

Daftar Tabel

1 Daftar fitur yang diimplementasikan dalam program 26

Daftar Gambar

1	<i>Directory tree kode sumber program</i>	5
2	Spesifikasi laptop yang digunakan untuk pengujian	9
3	Hasil kasus uji 1	9
4	Hasil kasus uji 2	9
5	Hasil kasus uji 3	10
6	Hasil kasus uji 4	10
7	Hasil kasus uji 5	11
8	Hasil kasus uji 6	11
9	Hasil kasus uji 7	12
10	Hasil kasus uji 8	12
11	Hasil kasus uji 9	13
12	Hasil kasus uji 10	14
13	Hasil kasus uji 11	14
14	Hasil kasus uji 12	15

Daftar Kode

1	Prakomputasi dalam algoritma <i>brute force</i>	6
2	Main.java	16
3	Keping.java	16
4	Papan.java	18
5	Data.java	20
6	ReadTXT.java	21
7	WriteTXT.java	24
8	Solver.java	25

1 Pendahuluan

1.1 Deskripsi Masalah

Di dalam Tugas Kecil 1 ini, kami diminta untuk membuat program pemecahan dari permainan papan IQ Puzzler Pro menggunakan algoritma *brute force* yang ditulis dalam bahasa pemrograman Java. Secara konsep, permainan ini mirip seperti *puzzle pentomino*.

Program akan menerima masukan berupa file .txt dengan konfigurasi tertentu. Baris pertama memuat nilai n , m , dan p . Di sini, n adalah jumlah baris, m adalah jumlah kolom, dan p adalah jumlah keping yang tersedia. Baris kedua memuat informasi tentang jenis papan. Pada program saya, jenis papan yang tersedia hanyalah DEFAULT. Selanjutnya, terdapat p buah keping yang masing-masing direpresentasikan oleh satu huruf alfabet.

Dalam program ini, papan permainan dan keping *puzzle* dimodelkan sebagai matriks $n \times m$ berelemen char. Supaya sejalan dengan kaidah pemrograman berorientasi objek, keduanya diimplementasikan sebagai kelas yang memiliki metode-metode yang akan membantu kita dalam mengimplementasikan algoritma pencarian dengan *brute force*.

1.2 Struktur Program

```
/src
├── Main.java
├── classes
│   ├── Data.java
│   ├── Keping.java
│   ├── Papan.java
│   └── Solver.java
└── utils
    └── ReadTXT.java
```

Gambar 1: *Directory tree kode sumber program*

2 Algoritma

Kode sumber utuh untuk algoritma *brute force* yang diterapkan untuk mencari solusi IQ Puzzler Pro dapat dilihat di berkas `Solver.java` yang juga dilampirkan di bagian terakhir dokumen ini.

2.1 Variabel/Atribut

Didefinisikan kelas `Solver` untuk menampung fungsi yang akan kita gunakan untuk menyelesaikan persoalan. Kelas ini mempunyai sejumlah atribut:

```

1  package src.classes;
2
3  import java.util.*;
4
5  public class Solver {
6  private Papan papan;
7  private final HashMap<Character, Keping> kepingings;
8  private final List<Character> kepingOrder;
9  private final HashMap<Character, List<Keping>> transformationsMap;
10 private int nIterasi;
11 private long waktu;
12
13
14 public Solver(Papan papan, Data data) {
15     this.papan = papan;
16     this.kepingings = data.getKepingings();
17     this.kepingOrder = data.getKepingOrder();
18     this.transformationsMap = precomputeTransformations();
19     this.nIterasi = 0;
20     this.waktu = 0;
21 }

```

Objek `papan` mewakili papan dari permainan ini, sementara `kepingings` berfungsi menampung seluruh keping *puzzle* yang tersedia, `kepingOrder` untuk menyimpan urutan keping supaya mempermudah *traversal* pada `kepingings`, `transformationsMap` untuk menampung hasil *precomputing* terhadap variasi orientasi (akibat rotasi maupun refleksi), serta `nIterasi` dan `waktu` yang secara berturut-turut menyimpan banyaknya iterasi dan panjang durasi yang dibutuhkan hingga solusi didapatkan.

2.2 Prakomputasi (*Precomputing*)

Precomputing atau prakomputasi saya lakukan untuk menyimpan variasi-variasi yang suatu kepingan punya akibat rotasi atau refleksi. Karena ada 4 sudut rotasi yang berlaku (0° , 90° , 180° , dan 270°) serta 2 macam bentuk untuk masing-masing rotasi karena adanya refleksi, maka tiap keping mempunyai 8 macam rupa.

Prakomputasi menghindarkan kita dari redundansi. Sebuah persegi, misalnya, akan mempunyai bentuk yang sama persis ketika dirotasikan maupun direfleksikan. Dengan menyimpan ke dalam `HashMap`, kita dapat menghemat sumber daya yang digunakan.

Barangkali orang dapat berargumen kalau ini adalah trik untuk mengoptimasi algoritma sehingga algoritmanya bukan *brute force* murni. Padahal, dengan metode ini kita tetap melakukan *exhaustive search*. Kita benar-benar mengeksplorasi segala kemungkinan yang ada. Bedanya, di sini kita memastikan bahwa kemungkinan yang sudah diperiksa tidak perlu diperiksa lagi selanjutnya.

Beberapa contoh yang bukan *brute force* murni adalah jika kita mengurutkan keping-keping dari yang terbesar dahulu untuk dicoba, atau menggunakan beberapa trik yang sudah umum dikenal untuk menyelesaikan *exact cover problem*, seperti *dancing links/Algorithm X* yang dicetuskan oleh The GOAT Donald Knuth.

Dalam program saya, prakomputasi diimplementasikan dalam fungsi `precomputeTransformations()`:

Kode 1: Prakomputasi dalam algoritma *brute force*

```

1  private HashMap<Character, List<Keping>> precomputeTransformations() {
2      HashMap<Character, List<Keping>> map = new HashMap<>();
3      for (Character key : keplings.keySet()) {
4          Keping original = keplings.get(key);
5          Set<String> done = new HashSet<>();
6          List<Keping> precompute = new ArrayList<>();
7
8          for (int rot = 0; rot < 4; rot++) {
9              Keping kepingDiputar = original.copyKeping();
10             for (int r = 0; r < rot; r++) kepingDiputar.putar();
11
12             for (int flip = 0; flip < 2; flip++) {
13                 Keping kepingDicermin = kepingDiputar.copyKeping();
14                 if (flip == 1) kepingDicermin.cermin();
15
16                 String keyRep = Arrays.deepToString(kepingDicermin.getBentuk());
17                 if (!done.contains(keyRep)) {
18                     done.add(keyRep);
19                     precompute.add(kepingDicermin);
20                 }
21             }
22         }
23         map.put(key, precompute);
24     }
25     return map;
26 }

```

2.3 Backtracking

Untuk mengeksplorasi seluruh kemungkinan, kita menggunakan *backtracking*. Di sini kita memeriksa mulai dari pojok kiri atas papan (row = 0; col = 0) hingga pojok kanan bawah papan, jika kepingnya muat. Hal tersebut diperiksa dengan menggunakan metode `isPlacementValid()`. Jika suatu keping muat untuk ditambahkan pada suatu titik di papan, kita akan menambahkannya dengan metode `tambahKeping()`. Ketika seluruh keping digunakan atau tidak ada ruang lagi untuk menambah keping yang tersisa, kita akan berjalan mundur (*backtracking*) dengan menghapus keping terbaru yang kita tambahkan dengan menggunakan metode `hapusKeping()`.

```

1  public boolean solve(int idx) {
2      if (idx >= keplings.size()) {
3          return papan.isPenuh();
4      }
5      Character currKey = kepingOrder.get(idx);
6      for (Keping keping : transformationsMap.get(currKey)) {
7          for (int row = 0; row < papan.getN(); row++) {
8              for (int col = 0; col < papan.getM(); col++) {
9                  nIterasi++;
10                 if (papan.isPlacementValid(keping, row, col)) {
11                     papan.tambahKeping(keping, row, col);
12                     if (solve(idx + 1)) return true;
13                     papan.hapusKeping(keping, row, col);
14                 }
15             }
16         }
17     }
18     return false;
19 }

```

2.4 Integrasi

Fungsi berikut merupakan integrasi dari seluruh fungsi yang dibutuhkan untuk menyelesaikan IQ Puzzler Pro.

```

1  public String solve() {
2      long start = System.currentTimeMillis();

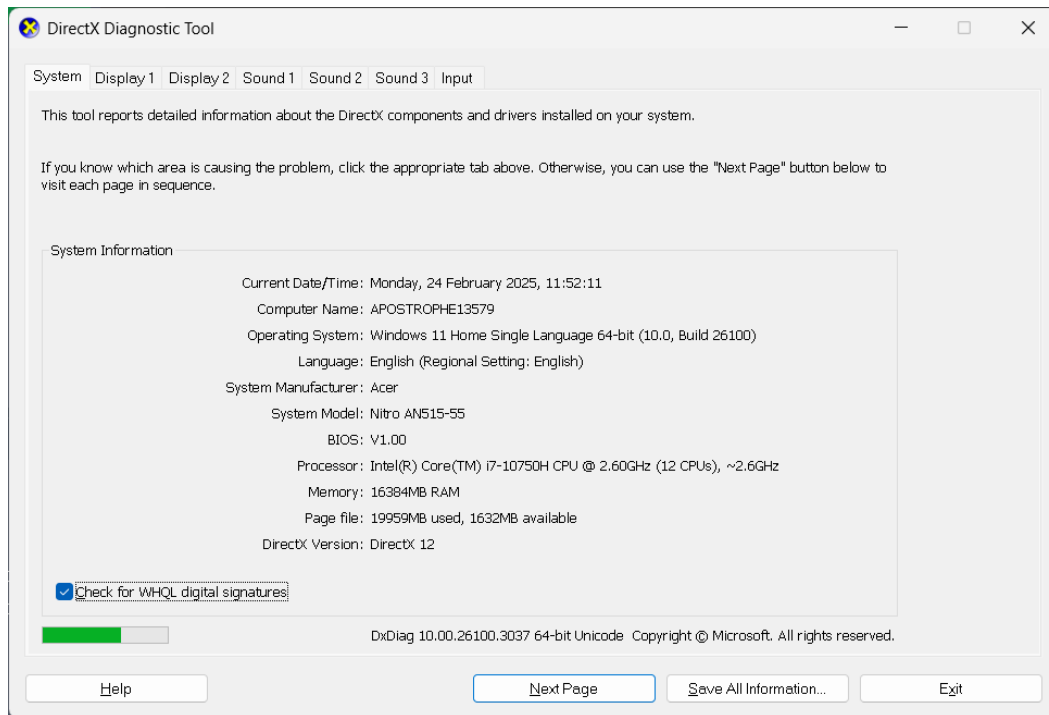
```



```
3     boolean solved = solve(0);
4     long end = System.currentTimeMillis();
5     this.waktu = end - start;
6     String hasil = "";
7
8     if (solved && papan.isPenuh()) {
9         papan.printPapan();
10        System.out.println("Berhasil menyelesaikan puzzle!");
11        hasil += "Berhasil menyelesaikan puzzle!\n";
12
13        System.out.println("Hasil:");
14        hasil += "Hasil:\n";
15
16        for (int i = 0; i < papan.getN(); i++) {
17            for (int j = 0; j < papan.getM(); j++) {
18                hasil += papan.getPetak()[i][j];
19                hasil += " ";
20            }
21            hasil += "\n";
22        }
23
24    } else {
25        System.out.println("Gagal menyelesaikan puzzle :(");
26    }
27
28    System.out.println("Jumlah iterasi: " + this.nIterasi);
29    hasil += "Jumlah iterasi: " + this.nIterasi + "\n";
30
31    System.out.println("Waktu yang dibutuhkan: " + this.waktu + " ms");
32    hasil += "Waktu yang dibutuhkan: " + this.waktu + " ms\n";
33    return hasil;
34 }
35 }
```

3 Eksperimen

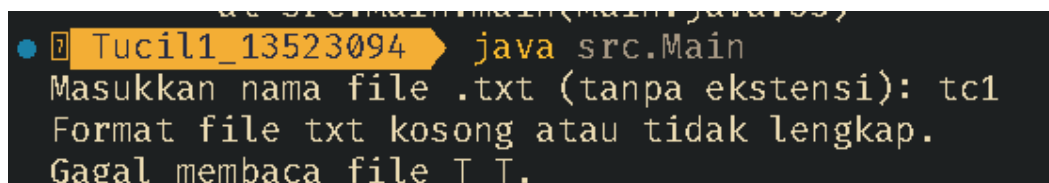
Kasus-kasus uji berikut disimpan dalam folder `test` di repositori. Pengujian dijalankan di laptop Acer Nitro dengan kecepatan CPU 2.6 GHz dan kapasitas RAM 16 GB. Program dikompilasi dengan Java versi 8 *update* 441.



Gambar 2: Spesifikasi laptop yang digunakan untuk pengujian

3.1 Kasus Uji 1: Berkas txt kosong

Diberikan berkas `tc1.txt` kosong, program memberikan keluaran sebagai berikut.



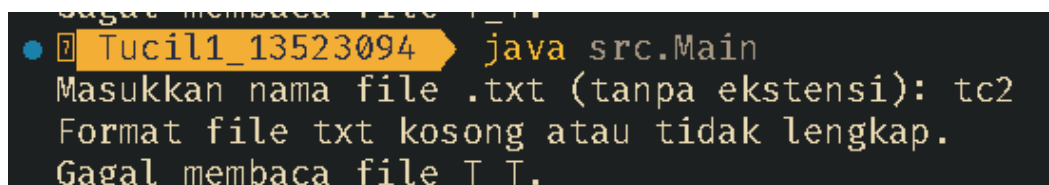
Gambar 3: Hasil kasus uji 1

3.2 Kasus Uji 2: Berkas txt tidak lengkap

Diberikan berkas `tc2.txt` tidak lengkap sebagai berikut:

```
6 5 3
DEFAULT
```

Keluarannya adalah:



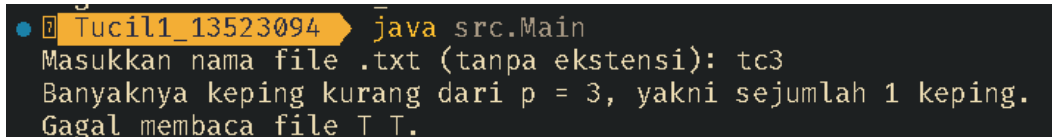
Gambar 4: Hasil kasus uji 2

3.3 Kasus Uji 3: Jumlah keping dalam berkas txt lebih sedikit dari p

Diberikan berkas tc3.txt sebagai berikut:

```
6 5 3
DEFAULT
P
PPP
P
P
```

Keluarannya adalah:



```
● Tucil1_13523094 java src.Main
Masukkan nama file .txt (tanpa ekstensi): tc3
Banyaknya keping kurang dari p = 3, yakni sejumlah 1 keping.
Gagal membaca file T T.
```

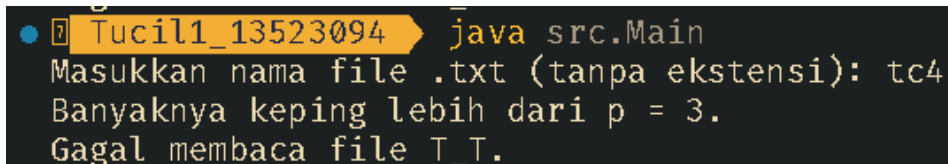
Gambar 5: Hasil kasus uji 3

3.4 Kasus Uji 4: Jumlah keping dalam berkas txt lebih banyak dari p

Diberikan berkas tc4.txt sebagai berikut:

```
6 5 3
DEFAULT
P
PPP
P
P
X
XXX
X X
Y
YYYY
Y
SSSS
S SS
```

Keluarannya adalah:



```
● Tucil1_13523094 java src.Main
Masukkan nama file .txt (tanpa ekstensi): tc4
Banyaknya keping lebih dari p = 3.
Gagal membaca file T T.
```

Gambar 6: Hasil kasus uji 4

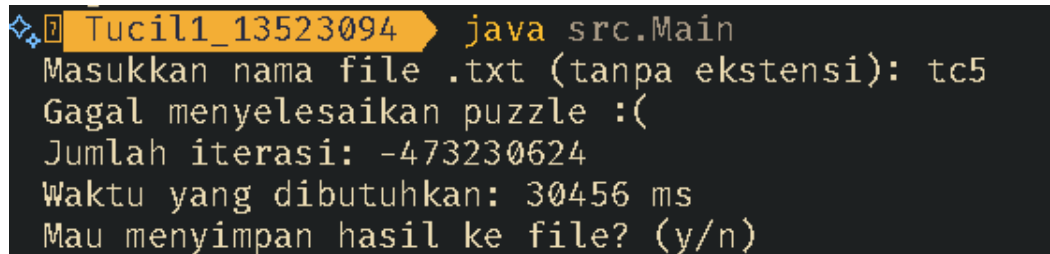
3.5 Kasus Uji 5: Tidak ada solusi

Diberikan berkas tc5.txt sebagai berikut:

```
4 6 8
DEFAULT
A
AA
B
BB
C
```

CC
D
DD
E
EE
F
FF
G
GG
H
H

Keluarannya adalah:



```

❖ Tucil1_13523094 java src.Main
Masukkan nama file .txt (tanpa ekstensi): tc5
Gagal menyelesaikan puzzle :(
Jumlah iterasi: -473230624
Waktu yang dibutuhkan: 30456 ms
Mau menyimpan hasil ke file? (y/n)
  
```

Gambar 7: Hasil kasus uji 5

Overflow pada variabel `nIterasi` nampaknya terjadi karena iterasi yang dilakukan sangatlah banyak.

3.6 Kasus Uji 6: Papan DEFAULT (1)

Diberikan berkas `tc6.txt` sebagai berikut:

Keluarannya adalah:

Gambar 8: Hasil kasus uji 6

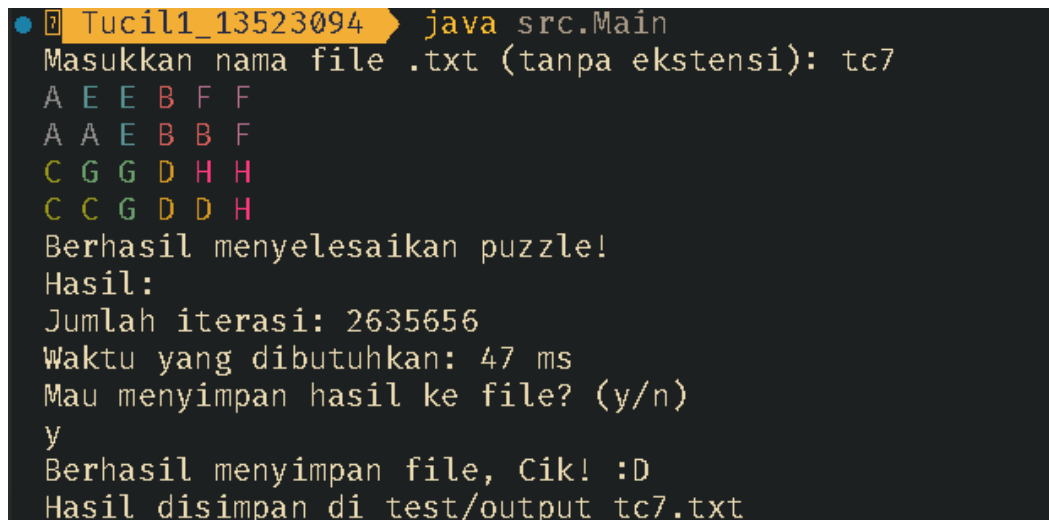
3.7 Kasus Uji 7: Papan DEFAULT (2) dan fitur menyimpan ke berkas txt baru

Diberikan berkas `tc7.txt` sebagai berikut:

```

4 6 8
DEFAULT
A
AA
B
BB
C
CC
D
DD
E
EE
F
FF
G
GG
H
HH
  
```

Keluarannya adalah:



```

Tucil1_13523094 java src.Main
Masukkan nama file .txt (tanpa ekstensi): tc7
A E E B F F
A A E B B F
C G G D H H
C C G D D H
Berhasil menyelesaikan puzzle!
Hasil:
Jumlah iterasi: 2635656
Waktu yang dibutuhkan: 47 ms
Mau menyimpan hasil ke file? (y/n)
y
Berhasil menyimpan file, Cik! :D
Hasil disimpan di test/output_tc7.txt

```

Gambar 9: Hasil kasus uji 7

Berkas output_tc7.txt yang dihasilkan adalah

```

Berhasil menyelesaikan puzzle!
Hasil:
A E E B F F
A A E B B F
C G G D H H
C C G D D H
Jumlah iterasi: 2635656
Waktu yang dibutuhkan: 47 ms

```

3.8 Kasus Uji 8: Papan DEFAULT (3)

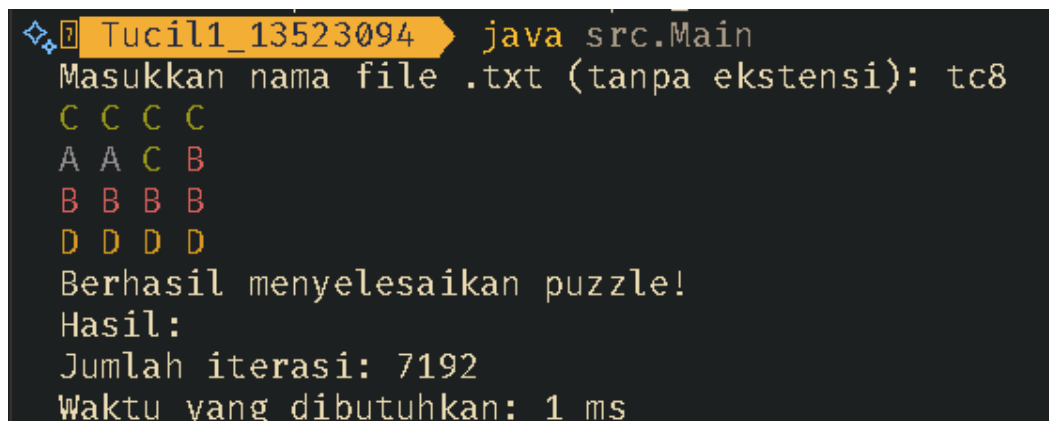
Diberikan berkas tc8.txt sebagai berikut:

```

4 4 4
DEFAULT
AA
  B
BBBB
CCCC
  C
DDDD

```

Keluarannya adalah:



```

Tucil1_13523094 java src.Main
Masukkan nama file .txt (tanpa ekstensi): tc8
C C C C
A A C B
B B B B
D D D D
Berhasil menyelesaikan puzzle!
Hasil:
Jumlah iterasi: 7192
Waktu yang dibutuhkan: 1 ms

```

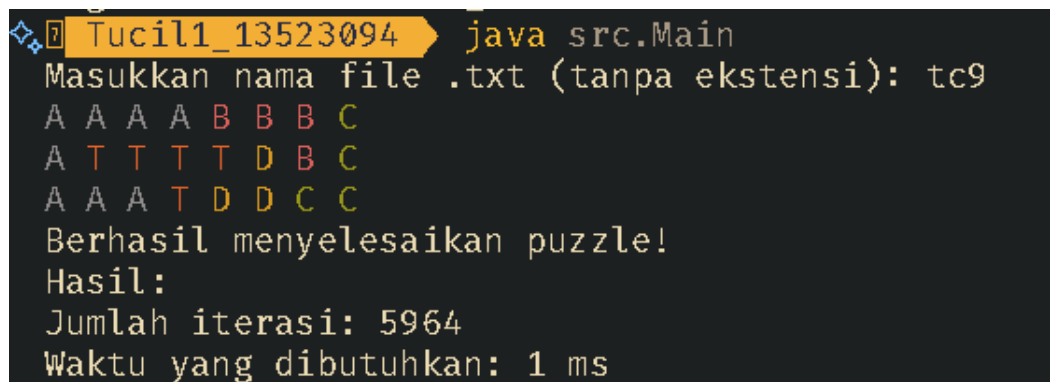
Gambar 10: Hasil kasus uji 8

3.9 Kasus Uji 9: Papan DEFAULT (4)

Diberikan berkas tc9.txt sebagai berikut:

```
3 8 5
DEFAULT
AAAA
A
AAA
BBB
B
CCC
C
D
DD
T
TTTT
```

Keluarannya adalah:



```
Tucil1_13523094 java src.Main
Masukkan nama file .txt (tanpa ekstensi): tc9
A A A A B B B C
A T T T T D B C
A A A T D D C C
Berhasil menyelesaikan puzzle!
Hasil:
Jumlah iterasi: 5964
Waktu yang dibutuhkan: 1 ms
```

Gambar 11: Hasil kasus uji 9

3.10 Kasus Uji 10: Papan DEFAULT (5)

Diberikan berkas tc10.txt sebagai berikut:

```
7 4 9
DEFAULT
ZZ
A
AAA
B
B
BB
CC
C
RR
R
R
R
E
E
E
G
GG
NN
N
I
```

Keluarannya adalah:

```
❖ Tucil1_13523094 java src.Main
Masukkan nama file .txt (tanpa ekstensi): tc10
Z Z A B
A A A B
C C B B
C R R G
E R G G
E R N N
E R N I
Berhasil menyelesaikan puzzle!
Hasil:
Jumlah iterasi: 333
Waktu yang dibutuhkan: 0 ms
Mau menyimpan hasil ke file? (y/n)
```

Gambar 12: Hasil kasus uji 10

3.11 Kasus Uji 11: Papan DEFAULT (6)

Diberikan berkas tc11.txt sebagai berikut:

```
4 6 7
DEFAULT
AA
AA
AAA
D
D
L
L
I I
I I
III
JJ
KK
WW
```

Keluarannya adalah:

```
❖ Tucil1_13523094 java src.Main
Masukkan nama file .txt (tanpa ekstensi): tc11
A A D I L I
A A D I L I
A A A I I I
J J K K W W
Berhasil menyelesaikan puzzle!
Hasil:
Jumlah iterasi: 172
Waktu yang dibutuhkan: 0 ms
Mau menyimpan hasil ke file? (y/n)
y
Berhasil menyimpan file, Cik! :D
Hasil disimpan di test/outputs/output_tc11.txt
```

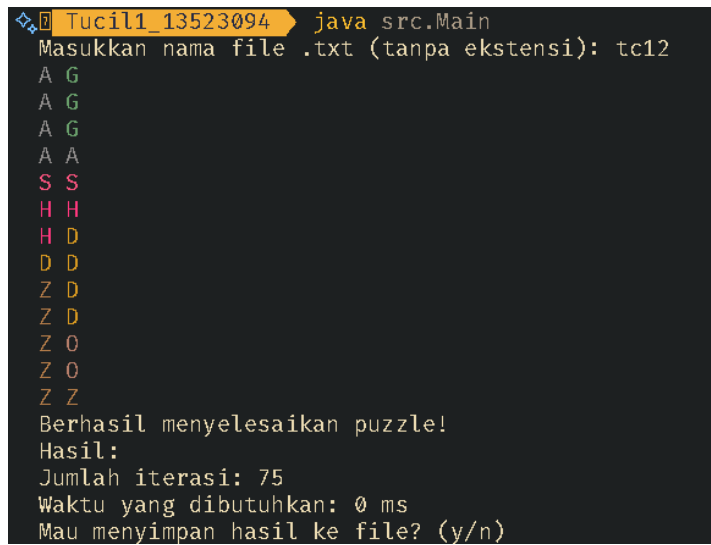
Gambar 13: Hasil kasus uji 11

3.12 Kasus Uji 11: Papan DEFAULT (7)

Diberikan berkas tc12.txt sebagai berikut:

13 2 7
DEFAULT
A
A
A
AA
G
G
G
SS
HH
H
D
DD
D
D
Z
Z
Z
Z
ZZ
O
O

Keluarannya adalah:



```
Tucil1_13523094 java src.Main
Masukkan nama file .txt (tanpa ekstensi): tc12
A G
A G
A G
A A
S S
H H
H D
D D
Z D
Z D
Z O
Z O
Z Z
Berhasil menyelesaikan puzzle!
Hasil:
Jumlah iterasi: 75
Waktu yang dibutuhkan: 0 ms
Mau menyimpan hasil ke file? (y/n)
```

Gambar 14: Hasil kasus uji 12

4 Lampiran

4.1 Pranala Repositori

Repositori yang memuat program ini dapat diakses di [sini](#).

4.2 Kode Sumber

Berikut adalah kode sumber dari program.

Kode 2: Main.java

```

1 package src;
2 import src.classes.*;
3 import src.utils.*;
4 import java.util.Scanner;
5
6
7 public class Main {
8     public static void main(String[] args) {
9         // Input
10        Scanner scanner = new Scanner(System.in);
11        System.out.print("Masukkan nama file .txt (tanpa ekstensi): ");
12        String filename = scanner.nextLine();
13
14        // Parsing txt
15        Data data = ReadTXT.readTxt("test/"+ filename + ".txt");
16
17        // Validasi (validate) input
18        if (data == null) {
19            System.out.println("Gagal membaca file T_T.");
20            scanner.close();
21            return;
22        }
23
24        // Prep
25        Papan papan = new Papan(data.getN(), data.getM());
26
27        // Solve
28        Solver solver = new Solver(papan, data);
29        String hasil = solver.solve();
30
31        // Output
32        System.out.println("Mau menyimpan hasil ke file? (y/n)");
33        String save = scanner.nextLine();
34        if (save.equalsIgnoreCase("y")) {
35            WriteTXT.writeTxt(filename, hasil);
36        }
37        scanner.close();
38    }
39 }

```

Berikut adalah kelas Keping yang digunakan untuk memodelkan tiap keping dari IQ Puzzler Pro.

Kode 3: Keping.java

```

1 package src.classes;
2
3 public class Keping {
4     // Atribut (Attributes)
5     private int panjang;
6     private int tinggi;
7     private char[][] bentuk;
8     private int nTitik;
9 }

```

```

10 // Metode (Methods)
11 /// Setters
12 public void setPanjang(int panjang) {
13     this.panjang = panjang;
14 }
15 public void setTinggi(int tinggi) {
16     this.tinggi = tinggi;
17 }
18 public void setBentuk(char[][] bentuk) {
19     this.bentuk = bentuk;
20 }
21 public void setnTitik(int nTitik) {
22     this.nTitik = nTitik;
23 }
24
25 /// Getters
26 public int getPanjang() {
27     return panjang;
28 }
29 public int getTinggi() {
30     return tinggi;
31 }
32 public char[][] getBentuk() {
33     return bentuk;
34 }
35 public int getnTitik() {
36     return nTitik;
37 }
38
39 /// Konstruktor (Constructor)
40 public Keping(char[][] bentuk) {
41     this.setBentuk(bentuk);
42     this.setTinggi(this.getBentuk().length);
43     this.setPanjang(this.getBentuk()[0].length);
44     this.setnTitik(this.countGrid());
45 }
46
47 /// Transformasi (Transformation)
48 /// Rotasi 90 derajat searah jarum jam
49 public void putar() {
50     char[][] diputar = new char[this.getPanjang()][this.getTinggi()];
51
52     for (int i = 0; i < panjang; i++) {
53         for (int j = 0; j < tinggi; j++) {
54             diputar[i][j] = bentuk[tinggi-1-j][i];
55         }
56     }
57
58     int temp = tinggi;
59     this.setTinggi(panjang);
60     this.setPanjang(temp);
61     this.setBentuk(diputar);
62 }
63
64 /// Pencerminkan (Reflection)
65 /// Pencerminkan terhadap sumbu-x
66 public void cermin() {
67     char[][] tecermin = new char[this.getTinggi()][this.getPanjang()];
68
69     for (int i = 0; i < this.getTinggi(); i++) {
70         for (int j = 0; j < this.getPanjang(); j++) {
71             tecermin[i][j] = this.bentuk[i][this.getPanjang() - 1 - j];
72         }
73     }
74 }

```

```

75         this.setBentuk(tecermin);
76     }
77
78     /// Fungsi Pembantu (Helper Function(s))
79     public int countGrid() {
80         int count = 0;
81
82         for (int i = 0; i < this.getTinggi(); i++) {
83             for (int j = 0; j < this.getPanjang(); j++) {
84                 if (Character.isLetter(this.getBentuk()[i][j])) {
85                     count++;
86                 }
87             }
88         }
89         return count;
90     }
91
92     // printKeping
93     public void printKeping() {
94         for (int i = 0; i < this.getTinggi(); i++) {
95             for (int j = 0; j < this.getPanjang(); j++) {
96                 System.out.print(this.getBentuk()[i][j]);
97             }
98             System.out.println();
99         }
100     }
101
102     public Keping copyKeping() {
103         char[][] newBentuk = new char[tinggi][panjang];
104         for (int i = 0; i < tinggi; i++) {
105             System.arraycopy(bentuk[i], 0, newBentuk[i], 0, panjang);
106         }
107         return new Keping(newBentuk);
108     }
109 }

```

Berikut adalah kode untuk kelas Papan yang digunakan untuk memodelkan papan tempat meletakkan kepingan puzzle.

Kode 4: Papan.java

```

1 package src.classes;
2
3 import java.util.HashMap;
4
5 public class Papan {
6     public int n;
7     public int m;
8     public int terisi;
9     public char[][] petak;
10    private static String RESET = "\u001B[0m";
11    private static String[] WARNA = {
12        // jujur aku gatau ini masing-masing warnanya apa
13        // Ambil dari random palette generator
14        "\u001B[30m",
15        "\u001B[31m",
16        "\u001B[32m",
17        "\u001B[33m",
18        "\u001B[34m",
19        "\u001B[35m",
20        "\u001B[36m",
21        "\u001B[38;2;255;33;111m",
22        "\u001B[38;2;77;62;199m",
23        "\u001B[38;2;31;4;108m",
24        "\u001B[38;2;93;189;204m",
25        "\u001B[38;2;114;225;38m",

```

```

26         "\u001B[38;2;227;111;223m",
27         "\u001B[38;2;105;41;179m",
28         "\u001B[38;2;138;55;24m",
29         "\u001B[38;2;253;202;196m",
30         "\u001B[38;2;103;230;193m",
31         "\u001B[38;2;249;239;147m",
32         "\u001B[38;2;240;41;93m",
33         "\u001B[38;2;229;98;41m",
34         "\u001B[38;2;35;82;41m",
35         "\u001B[38;2;26;12;24m",
36         "\u001B[38;2;118;227;124m",
37         "\u001B[38;2;238;153;170m",
38         "\u001B[38;2;44;224;115m",
39         "\u001B[38;2;182;126;74m",
40     };
41     private static HashMap<Character, String> warnaMap;
42
43     // Getters
44     public int getN() {
45         return this.n;
46     }
47     public int getM() {
48         return m;
49     }
50     public char[][] getPetak() {
51         return petak;
52     }
53
54     // Konstruktor (Constructor)
55     public Papan(int n, int m) {
56         this.n = n;
57         this.m = m;
58         this.terisi = 0;
59         this.petak = new char[n][m];
60         for (int i = 0; i < n; i++) {
61             for (int j = 0; j < m; j++) {
62                 petak[i][j] = '-';
63             }
64         }
65
66         warnaMap = new HashMap<Character, String>();
67         char huruf = 'A';
68         for (String warna : WARNA) {
69             warnaMap.put(huruf, warna);
70             huruf++;
71         }
72         warnaMap.put('-', "\u001B[0m");
73     }
74
75     public void printPapan() {
76         for (int i = 0; i < n; i++) {
77             for (int j = 0; j < m; j++) {
78                 System.out.print(warnaMap.get((char) (petak[i][j])));
79                 System.out.print(petak[i][j]);
80                 System.out.print(RESET);
81                 System.out.print(" ");
82             }
83             System.out.println();
84         }
85     }
86
87     public boolean isPenuh() {
88         return this.terisi == n * m;
89     }
90

```

```

91 public boolean isPlacementValid(Keping keping, int x, int y) {
92     char[][] bentuk = keping.getBentuk();
93
94     if (x < 0 || y < 0 || x + keping.getTinggi() > n || y + keping.getPanjang() > m) {
95         return false;
96     }
97
98     for (int i = 0; i < keping.getTinggi(); i++) {
99         for (int j = 0; j < keping.getPanjang(); j++) {
100             if (bentuk[i][j] != '-' && petak[x + i][y + j] != '-') {
101                 return false;
102             }
103         }
104     }
105     return true;
106 }
107
108 public void tambahKeping(Keping keping, int x, int y) {
109     char[][] bentuk = keping.getBentuk();
110     for (int i = 0; i < keping.getTinggi(); i++) {
111         for (int j = 0; j < keping.getPanjang(); j++) {
112             if (keping.getBentuk()[i][j] != '-') {
113                 petak[x + i][y + j] = bentuk[i][j];
114                 terisi++;
115             }
116         }
117     }
118 }
119
120 public void hapusKeping(Keping keping, int x, int y) {
121     char[][] bentuk = keping.getBentuk();
122     for (int i = 0; i < keping.getTinggi(); i++) {
123         for (int j = 0; j < keping.getPanjang(); j++) {
124             if (bentuk[i][j] != '-') {
125                 petak[x + i][y + j] = '-';
126                 terisi--;
127             }
128         }
129     }
130 }
131 }

```

Ini adalah kelas yang digunakan untuk mengoper data yang di-parsing dari berkas txt.

Kode 5: Data.java

```

1 package src.classes;
2 import java.util.HashMap;
3 import java.util.List;
4
5 public class Data {
6     // Atribut (Attributes)
7     private final int n;
8     private final int m;
9     private final int p;
10    private final HashMap<Character, Keping> Keping;
11    private final List<Character> kepingOrder;
12
13    // Metode (Methods)
14    /// Getters
15    public int getN() {
16        return n;
17    }
18    public int getM() {
19        return m;
20    }

```

```

21     public int getP() {
22         return p;
23     }
24     public HashMap<Character, Keping> getKepings() {
25         return Kepings;
26     }
27     public List<Character> getKepingOrder() {
28         return kepingOrder;
29     }
30
31
32     /// Konstruktor (Constructor)
33     public Data(int n, int m, int p, HashMap<Character, Keping> Kepings, List<Character> order)
34     {
35         this.n = n;
36         this.m = m;
37         this.p = p;
38         this.Kepings = Kepings;
39         this.kepingOrder = order;
40     }
41
42     /// Metode lain (Other methods)
43     /// printData
44     public void printData() {
45         System.out.println("n: " + n);
46         System.out.println("m: " + m);
47         System.out.println("p: " + p);
48         System.out.println("Keping:");
49         for (Character key : Kepings.keySet()) {
50             System.out.println("Keping " + key + ":");
51             Kepings.get(key).printKeping();
52         }
53     }
54
55 }

```

Ini adalah kode yang digunakan untuk membaca dan mem-*parsing* berkas txt.

Kode 6: ReadTXT.java

```

1  package src.utils;
2  import src.classes.Data;
3  import src.classes.Keping;
4
5  import java.util.ArrayList;
6  import java.util.List;
7  import java.util.HashMap;
8  import java.util.Scanner;
9  import java.io.File;
10 import java.io.FileNotFoundException;
11
12
13
14 public class ReadTXT {
15
16     public static Data readTxt(String filename) {
17         int lineNumber = 1;
18         String[] lines = null;
19         try {
20             lines = readTxtToArray(filename);
21             if (lines.length < 3) {
22                 System.err.println("Format file txt kosong atau tidak lengkap.");
23                 return null;
24             }
25         } catch (Exception e) {

```

```

26         return null;
27     }
28
29     // Baris 1: N M P
30     String[] nmp = lines[lineNumber-1].split(" ");
31
32     // Validasi (validate) input
33
34     // ID: Kasus 1: Format tidak sesuai.
35     // EN: Case 1: Format does not match.
36     if (nmp.length != 3) {
37         System.err.println("Baris " + lineNumber + "tidak lengkap, mestinya berisi 'N M P'
dengan NxM adalah dimensi papan dan P adalah banyak keping.");
38         return null;
39     }
40
41     // ID: Kasus 2: Baris pertama mengandung nilai yang bukan angka
42     // EN: Case 2: First line contains non-numerical value(s)
43     for (String string : nmp) {
44         if (!isNumeric(string) || Integer.parseInt(string) <= 0) {
45             System.err.println("Baris " + lineNumber + "harusnya hanya berisi bilangan bulat
positif dengan format `N M P` dengan NxM adalah dimensi papan dan P adalah banyak keping.")
;
46             return null;
47         }
48     }
49
50     // Assignment N, M, P
51     int n = Integer.parseInt(nmp[0]);
52     int m = Integer.parseInt(nmp[1]);
53     int p = Integer.parseInt(nmp[2]);
54
55     // Baris 2: Jenis Papan
56     lineNumber++;
57     String papanType = lines[lineNumber-1];
58
59     // Validasi (validate) input
60     // ID: Baris 2 mengandung tipe papan yang tidak valid
61     // EN: Line 2 contains invalid board type
62     if (!papanType.trim().equalsIgnoreCase("default")) {
63         System.err.println("Baris " + lineNumber + " harusnya berisi 'DEFAULT'.");
64         return null;
65     } else {
66         lineNumber++;
67     }
68
69     // Baris 3 - P+2: Keping
70
71     // Error handling
72     // ID: Tidak ada keping yang tersedia
73     // EN: No pieces available
74     if (lines.length < lineNumber + p) {
75         System.err.println("Tidak ada keping yang tersedia.");
76         return null;
77     }
78
79     HashMap<Character, Keping> kepingHashMap = new HashMap<>();
80     List<Character> kepingOrder = new ArrayList<>();
81     for (int i = 0; i < p; i++) {
82         // Error handling
83         // ID: Jumlah keping kurang dari P
84         // EN: Number of pieces is less than P
85         if (lineNumber > lines.length) {
86             System.err.println("Banyaknya keping kurang dari p = " + p + ", yakni sejumlah "
+ i + " keping.");

```

```

87         return null;
88     }
89
90     String currLine = lines[lineNumber-1];
91     char huruf = trimLeading(currLine).charAt(0); // Huruf keping
92
93     // Error handling
94     // ID: Kasus 1: Karakter tidak valid (bukan A-Z)
95     // EN: Kasus 1: Invalid character (not A-Z)
96     if (!Character.isUpperCase(huruf)) {
97         System.err.println("Baris " + lineNumber + " invalid! Suatu keping harusnya
tersusun atas huruf.");
98         return null;
99     }
100     // ID: Kasus 2: Keping dengan karakter ini sudah ada
101     // EN: Kasus 2: A piece with this character already exists
102     if (kepingHashMap.containsKey(huruf)) {
103         System.err.println("Baris " + lineNumber + " invalid! Keping dengan karakter " +
huruf + " sudah ada.");
104         return null;
105     }
106
107     int tempLineNumber = lineNumber;
108     int panjang = currLine.length();
109     while (lineNumber <= lines.length && trimLeading(lines[lineNumber-1]).charAt(0) ==
huruf) {
110         currLine = lines[lineNumber-1];
111         panjang = Math.max(panjang, currLine.length());
112         // Error handling
113         // ID: Suatu baris tersusun atas lebih dari satu jenis huruf
114         // EN: A line consists of more than one type of letter
115         if (!isLineUnique(currLine.trim())) {
116             System.err.println("Baris " + lineNumber + " invalid! Suatu keping harusnya
hanya tersusun atas SATU karakter unik (A-Z).");
117             return null;
118         }
119         lineNumber++;
120     }
121
122     int tinggi = lineNumber - tempLineNumber;
123     char[][] bentuk = new char[tinggi][panjang];
124     for (int j = 0; j < tinggi; j++) {
125         for (int k = 0; k < panjang; k++) {
126             bentuk[j][k] = k < lines[tempLineNumber+j-1].length() && lines[
tempLineNumber+j-1].charAt(k) == huruf ? huruf : '-';
127         }
128     }
129     kepingOrder.add(huruf);
130     kepingHashMap.put(huruf, new Keping(bentuk));
131 }
132
133 // Error handling
134 // ID: Jumlah keping lebih dari P
135 // EN: Number of pieces is more than P
136 if (lineNumber < lines.length) {
137     System.err.println("Banyaknya keping lebih dari p = " + p + ".");
138     return null;
139 }
140 return new Data(n, m, p, kepingHashMap, kepingOrder);
141 }
142
143 public static String[] readTxtToArray(String filename) {
144     ArrayList<String> lines = new ArrayList<>();
145
146     try (Scanner sc = new Scanner(new File(filename))) {

```



```

147         while (sc.hasNextLine()) {
148             String line = trimTrailing(sc.nextLine());
149             if (!line.isEmpty()) {
150                 lines.add(line);
151             }
152         }
153     } catch (FileNotFoundException e) {
154         System.err.println("File tidak ditemukan.");
155         return null;
156     }
157
158     return lines.toArray(new String[0]);
159 }
160
161 public static boolean isNumeric(String str) {
162     return str.matches("^\\d+$");
163 }
164
165 public static String trimLeading(String str) {
166     return str.replaceAll("^\\s+", "");
167 }
168
169 public static String trimTrailing(String str) {
170     return str.replaceAll("\\s+$", "");
171 }
172
173 public static boolean isLineUnique(String str) {
174     str = str.trim();
175     if (str.isEmpty()) {
176         return false;
177     }
178     char huruf = str.charAt(0);
179     if (!Character.isLetter(huruf)) {
180         return false;
181     }
182     str = str.replace(" ", "");
183     return str.chars().allMatch(c -> c == huruf);
184 }
185
186 public static int[] toBinaryArray(String str, char huruf) {
187     int[] row = new int[str.length()];
188     for (int i = 0; i < str.length(); i++) {
189         row[i] = str.charAt(i) == huruf ? 1 : 0;
190     }
191     return row;
192 }
193 }

```

Ini adalah kode yang digunakan untuk menyimpan solusi ke dalam berkas txt.

Kode 7: WriteTXT.java

```

1 package src.utils;
2 import java.io.FileWriter;
3 import java.io.IOException;
4
5 public class WriteTXT {
6     public static void writeTxt(String filename, String content) {
7         try {
8             FileWriter fileWriter = new FileWriter("test/outputs/output_" + filename + ".txt");
9             fileWriter.write(content);
10            fileWriter.close();
11            System.out.println("Berhasil menyimpan file, Cik! :D");
12            System.out.println("Hasil disimpan di test/outputs/output_" + filename + ".txt");
13        } catch (IOException e) {
14            System.err.println("Gagal menyimpan file T_T.");
15        }
16    }
17 }

```

```

15     }
16 }
17 }

```

Ini adalah kode yang memuat algoritma utama untuk menemukan solusi.

Kode 8: Solver.java

```

1 package src.classes;
2
3 import java.util.*;
4
5 public class Solver {
6     private Papan papan;
7     private final HashMap<Character, Keping> kepingings;
8     private final List<Character> kepingOrder;
9     private final HashMap<Character, List<Keping>> transformationsMap;
10    private int nIterasi;
11    private long waktu;
12
13    public Solver(Papan papan, Data data) {
14        this.papan = papan;
15        this.kepingings = data.getKepingings();
16        this.kepingOrder = data.getKepingOrder();
17        this.transformationsMap = precomputeTransformations();
18        this.nIterasi = 0;
19        this.waktu = 0;
20    }
21
22    private HashMap<Character, List<Keping>> precomputeTransformations() {
23        HashMap<Character, List<Keping>> map = new HashMap<>();
24        for (Character key : kepingings.keySet()) {
25            Keping original = kepingings.get(key);
26            Set<String> done = new HashSet<>();
27            List<Keping> precompute = new ArrayList<>();
28
29            for (int rot = 0; rot < 4; rot++) {
30                Keping kepingDiputar = original.copyKeping();
31                for (int r = 0; r < rot; r++) kepingDiputar.putar();
32
33                for (int flip = 0; flip < 2; flip++) {
34                    Keping kepingDicermin = kepingDiputar.copyKeping();
35                    if (flip == 1) kepingDicermin.cermin();
36
37                    String keyRep = Arrays.deepToString(kepingDicermin.getBentuk());
38                    if (!done.contains(keyRep)) {
39                        done.add(keyRep);
40                        precompute.add(kepingDicermin);
41                    }
42                }
43            }
44            map.put(key, precompute);
45        }
46        return map;
47    }
48
49    public boolean solve(int idx) {
50        if (idx >= kepingings.size()) {
51            return papan.isPenuh();
52        }
53        Character currKey = kepingOrder.get(idx);
54        for (Keping keping : transformationsMap.get(currKey)) {
55            for (int row = 0; row < papan.getN(); row++) {
56                for (int col = 0; col < papan.getM(); col++) {
57                    nIterasi++;
58                    if (papan.isPlacementValid(keping, row, col)) {

```

```

59         papan.tambahKeping(keping, row, col);
60         if (solve(idx + 1)) return true;
61         papan.hapusKeping(keping, row, col);
62     }
63 }
64 }
65 }
66 return false;
67 }
68
69 public String solve() {
70     long start = System.currentTimeMillis();
71     boolean solved = solve(0);
72     long end = System.currentTimeMillis();
73     this.waktu = end - start;
74     String hasil = "";
75
76     if (solved && papan.isPenuh()) {
77         papan.printPapan();
78         System.out.println("Berhasil menyelesaikan puzzle!");
79         hasil += "Berhasil menyelesaikan puzzle!\n";
80
81         System.out.println("Hasil:");
82         hasil += "Hasil:\n";
83
84         for (int i = 0; i < papan.getN(); i++) {
85             for (int j = 0; j < papan.getM(); j++) {
86                 hasil += papan.getPetak()[i][j];
87                 hasil += " ";
88             }
89             hasil += "\n";
90         }
91
92     } else {
93         System.out.println("Gagal menyelesaikan puzzle :(");
94     }
95
96     System.out.println("Jumlah iterasi: " + this.nIterasi);
97     hasil += "Jumlah iterasi: " + this.nIterasi + "\n";
98
99     System.out.println("Waktu yang dibutuhkan: " + this.waktu + " ms");
100    hasil += "Waktu yang dibutuhkan: " + this.waktu + " ms\n";
101    return hasil;
102 }
103 }

```

4.3 Tabel Kelengkapan

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

Tabel 1: Daftar fitur yang diimplementasikan dalam program