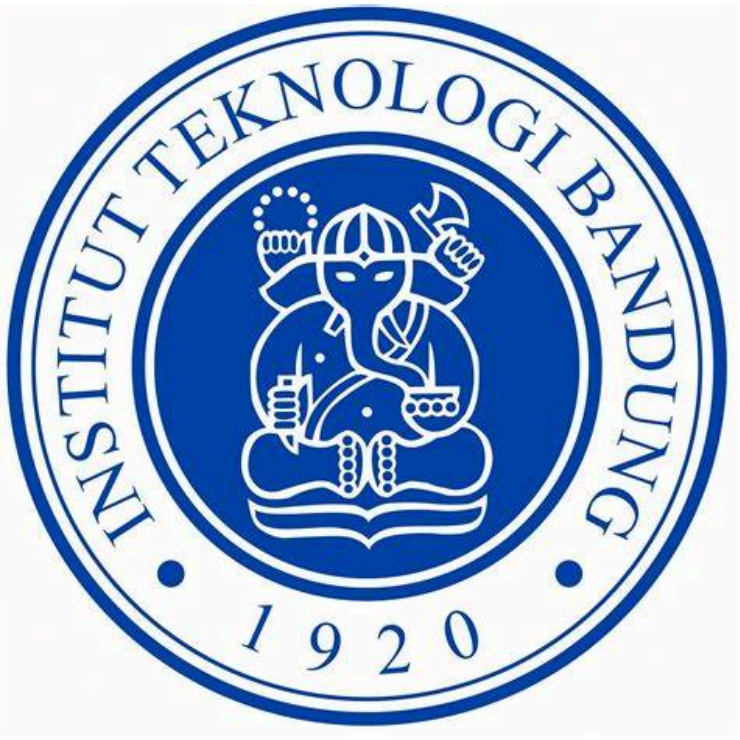
LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Penyelesaian IQ Puzzler Pro dengan Algoritma Brute Force



Disusun oleh:

Natalia Desiany Nursimin - 13523157

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

JL. GANESA 10, BANDUNG 40132

2025

DAFTAR ISI

BAB I ALGORITMA BRUTE FORCE..... 3
BAB II SOURCE CODE PROGRAM..... 4
2.1. File Main.java.....5
2.2. File Blok.java..... 5
2.3. File Papan.java..... 6
2.4. File PuzzleSolver.java.....6
2.5. File TxtHandler.java..... 7
BAB III TEST CASE.....8
3.1. Test Case 1.....8
3.2. Test Case 2.....8
3.3. Test Case 3.....9
3.4. Test Case 4.....10
3.5. Test Case 5..... 11
3.6. Test Case 6.....12
3.7. Test Case 7.....12
BAB IV LAMPIRAN.....14

BAB I ALGORITMA BRUTE FORCE

Algoritma *brute force* merupakan sebuah pendekatan yang berfungsi untuk mengeksplorasi semua kemungkinan solusi secara sistematis tanpa optimasi khusus. Dalam konteks pengimplementasian algoritma *brute force* ke dalam permainan IQ Puzzler Pro, algoritma ini mencoba menempatkan semua blok pada papan dengan mempertimbangkan setiap kombinasi posisi dan orientasi hingga akhirnya dapat menemukan solusi yang sesuai. Pendekatan ini menggunakan eksplorasi menyeluruh dan memastikan bahwa setiap kemungkinan yang dapat terjadi diuji, sehingga solusi yang ditemukan dapat bersifat valid.

Untuk mencari solusi, algoritma *brute force* mengandalkan penggunaan rekursi yang dikombinasikan dengan teknik *backtracking*. Pada konteks algoritma *brute force*, rekursi digunakan untuk menempatkan blok satu per satu ke dalam papan. *Backtracking* merupakan sebuah teknik yang digunakan untuk menyelesaikan masalah dengan mencoba semua kemungkinan solusi secara sistematis. Teknik ini memungkinkan algoritma untuk mundur jika menemui keadaan di mana tidak ada solusi yang valid. Ketika sebuah blok tidak dapat diletakkan di posisi mana pun, algoritma akan kembali ke tahap sebelumnya dan mencoba alternatif lain hingga semua kemungkinan habis. Dengan cara ini, brute force dapat memastikan bahwa seluruh ruang solusi telah diperiksa tanpa meninggalkan kemungkinan kombinasi yang belum diuji. Pendekatan *brute force* ini memungkinkan program untuk menjamin bahwa jika solusi ada, maka pasti akan ditemukan, meskipun mungkin memerlukan waktu komputasi yang cukup besar.

Berikut merupakan penjelasan tahapan-tahapan langkah penggunaan algoritma *brute force* yang digunakan dalam pembuatan program untuk penyelesaian permainan IQ Puzzler Pro:

1. Membaca Input dari File

Langkah pertama dalam program ini adalah membaca data dari file yang berisi informasi mengenai ukuran papan dan bentuk blok potongan puzzle. Proses ini dilakukan dalam kelas TxtHandler.java. Kelas ini berfungsi agar program dapat meminta pengguna memasukkan nama file input. Input ini kemudian akan dibaca dan disimpan datanya. Informasi ukuran papan diperoleh dari baris pertama file, sedangkan blok-blok puzzle

disimpan sebagai objek yang disebut blok. Jika file gagal dibaca atau kosong, program akan menampilkan pesan kesalahan.

2. Menyiapkan Papan Permainan

Setelah informasi dari file berhasil diproses, program akan kemudian membuat papan permainan yang direpresentasikan dalam kelas Papan.java. Papan ini dibuat dengan berbentuk matriks karakter yang ukurannya disesuaikan dengan input dari file. Pada awalnya, seluruh sel dalam papan diisi dengan karakter titik ('.'), yang menandakan bahwa papan masih kosong.

3. Membentuk dan Menyimpan Transformasi Blok Puzzle

Setiap blok puzzle yang dimasukkan ke dalam papan dapat mengalami rotasi dan pencerminan untuk meningkatkan kemungkinan menemukan solusi. Oleh karena itu, program menyimpan semua transformasi yang mungkin dilakukan pada setiap blok. Rotasi dilakukan dengan memindahkan elemen dari posisi i dan j pada matriks awal ke (j, rows - 1 - i) pada matriks hasil rotasi, sehingga blok dapat diputar sebesar 90°, 180°, dan 270°. Selain itu, program juga mengimplentasikan metode untuk melakukan pencerminan secara horizontal dengan membalik elemen dari sisi kanan ke kiri.

4. Mencari Posisi Kosong dan Mengecek Kecocokan Blok

Saat menjalankan algoritma, program harus mencari lokasi pertama yang masih kosong pada papan untuk meletakkan blok berikutnya. Hal ini dilakukan dalam metode `findNextEmptyCell()`, yang memeriksa setiap sel papan dari kiri atas hingga kanan bawah. Jika ditemukan sel kosong ('.'), maka koordinatnya dikembalikan untuk digunakan dalam proses penempatan blok. Jika semua sel telah terisi, berarti papan sudah penuh dan pencarian solusi bisa dihentikan. Setelah menemukan posisi kosong, metode canPlace() memastikan blok dapat ditempatkan tanpa keluar batas atau bertabrakan dengan blok lain.

5. Mengimplementasikan Algoritma Brute Force untuk Menyusun Blok di Papan

Setelah menemukan posisi yang cocok untuk blok, algoritma *brute force* akan diimplementasikan untuk mencoba menempatkan blok pertama pada papan, hingga semua blok berhasil ditempatkan. Jika semua kemungkinan telah diuji dan tidak ditemukan solusi yang valid, program akan mengembalikan status bahwa tidak ada solusi yang mungkin. Sebaliknya, jika semua blok berhasil ditempatkan dan papan terisi penuh, solusi dianggap valid, dan program akan menampilkan hasilnya kepada pengguna.

6. Menyimpan dan Menampilkan Solusi

Setelah solusi yang tepat berhasil ditemukan, program akan menampilkan papan dengan blok-blok berwarna yang telah ditempatkan kepada pengguna. Pengguna kemudian akan diberikan opsi untuk menyimpan solusi tersebut ke dalam sebuah file, sehingga hasil penyelesaian dapat disimpan dan diakses kembali jika diinginkan.

BAB II SOURCE CODE PROGRAM

Pembuatan program ini menggunakan bahasa Java. Kode yang digunakan pada program disimpan dalam folder src, yang berisi beberapa kelas untuk mengorganisir fungsionalitasnya. File-file ini terdiri dari Main, Blok, Papan, PuzzleSolver, TxtHandler. Berikut merupakan penjelasan mengenai source code setiap file.

2.1. File Main.java

File ini berisi kelas Main yang berfungsi sebagai titik masuk program. File ini menangani interaksi dengan pengguna, membaca file input, dan memulai proses penyelesaian puzzle.

Nama Method	Deskripsi
main()	Method yang bertugas untuk meminta input file dari pengguna, membaca file, dan memulai proses penyelesaian puzzle.
saveSolution(Papan papan, String filename, long executionTime, int iterationCount)	Menyimpan solusi dalam file jika pengguna menginginkannya.

2.2. File Blok.java

File ini berisikan kelas yang merepresentasikan sebuah blok puzzle. Setiap blok memiliki berbagai transformasi, seperti rotasi dan pencerminan yang disimpan dalam transformasi.

Nama Method	Deskripsi
Blok(char id, List<String> shapeLines)	Konstruktor yang menerima ID blok dan daftar string yang merepresentasikan bentuk blok.
parseShape(List<String> shapeLines)	Mengubah bentuk blok dari string menjadi array boolean
generateTransformasi(boolean[][]	Menghasilkan semua kemungkinan

baseShape)	transformasi blok
rotateRight(boolean[][] shape)	Memutar blok 90 derajat searah jarum jam
mirror(boolean[][] shape)	Mencerminkan blok secara horizontal
convertToIntArray(boolean[][] shape)	Mengubah array boolean menjadi array integer (1 untuk blok, 0 untuk kosong)

2.3. File Papan.java

File ini berisikan kelas yang merepresentasikan papan permainan, memungkinkan penempatan dan penghapusan blok.

Nama Method	Deskripsi
Papan(int N, int M)	Menginisialisasi papan dengan ukuran NxM
getRows()	Mengembalikan jumlah baris papan
getCols()	Mengembalikan jumlah kolom papan
isFull()	Mengecek apakah papan sudah penuh
findNextEmptyCell()	Mencari sel kosong berikutnya di papan
canPlace(int[][] shape, int row, int col)	Mengecek apakah suatu blok dapat ditempatkan pada posisi tertentu
placeBlock(int[][] shape, int row, int col, char id)	Menempatkan blok di papan
removeBlock(int[][] shape, int row, int col)	Menghapus blok di papan
getGrid()	Mengembalikan representasi papan dalam bentuk array karakter
printBoard()	Menampilkan papan dengan warna-warna untuk blok

2.4. File PuzzleSolver.java

File ini berisikan kelas untuk menggunakan metode brute force dengan backtracking untuk menemukan solusi.

Nama Method	Deskripsi
solve(Papan papan, List<Blok> blokList, int index)	Metode rekursif yang mencoba semua kemungkinan kombinasi blok di papan
getExecutionTime()	Mengembalikan waktu eksekusi pencarian solusi
getIterationCount()	Mengembalikan jumlah iterasi pencarian banyaknya solusi

2.5. File TxtHandler.java

File ini berisikan kelas untuk menangani pembacaan file input yang berisi konfigurasi puzzle dari Txt.

Nama Method	Deskripsi
readFile(String filename)	Membaca isi file dan mengembalikannya dalam bentuk daftar string

BAB III TEST CASE

3.1. Test Case 1

Input:
3 3 3

DEFAULT

AA

AA

B

B

CCC

Output:

```
Masukkan nama file test case: test/testcase1.txt
A A A
B B A
C C C
Waktu pencarian: 20 ms
Banyak kasus yang ditinjau: 5
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi berhasil disimpan di: C:\Users\mcn0c\Downloads\Tucil1_13523157\test\solusi_testcase1.txt
```

3.2. Test Case 2

Input:

5 5 8

DEFAULT

A

AA

B

BB

C

CC

D

DD

EE

EE

E

FF

FF

F

GGG

Output:

```
Masukkan nama file test case: test/testcase2.txt
A A A B B
C C C D B
D D E E E
F F F F E
E F G G G
Waktu pencarian: 26 ms
Banyak kasus yang ditinjau: 22
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi berhasil disimpan di: C:\Users\mcn0c\Downloads\Tucil1_13523157\test\solusi_testcase2.txt
```

3.3. Test Case 3

Input:

5 5 3

DEFAULT

A

AA

B

BB

C

CC

Output:

```
Masukkan nama file test case: test/testcase3.txt
Tidak ada solusi.
```

3.4. Test Case 4

Input:

3 3 3

DEFAULT

A

AA

B

BB

C

CC

Output:

```
Masukkan nama file test case: test/testcase4.txt
A A A
B B B
C C C
Waktu pencarian: 20 ms
Banyak kasus yang ditinjau: 6
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi berhasil disimpan di: C:\Users\mcn0c\Downloads\Tucil1_13523157\test\solusi_testcase4.txt
```

3.5. Test Case 5

Input:

5 5 8

DEFAULT

A

AA

B

BB

C

CC

D

DD

EE

EE

E

FF

FF

F

GGG

Output:

```
Masukkan nama file test case: test/testcase5.txt
A A A B B
C C C D B
D D E E E
E F F F E
F F G G G
Waktu pencarian: 22 ms
Banyak kasus yang ditinjau: 18
Apakah anda ingin menyimpan solusi? (ya/tidak): ya
Solusi berhasil disimpan di: C:\Users\mcn0c\Downloads\Tucil1_13523157\test\solusi_testcase5.txt
```

3.6. Test Case 6

Input:

4 4 6

DEFAULT

A

AA

B

BB

C

CC

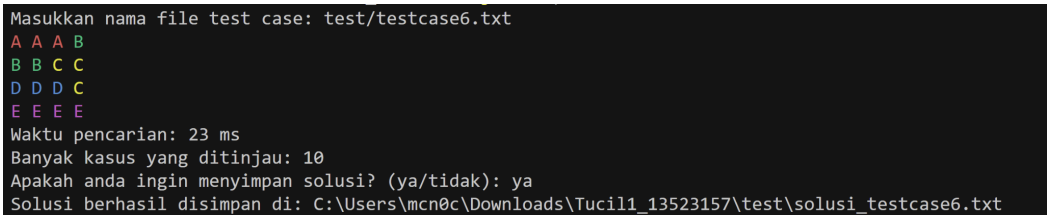
D

DD

EE

EE

Output:



3.7. Test Case 7

Input:

4 4 6

DEFAULT

A

A

B

BB

C

CC

D

D

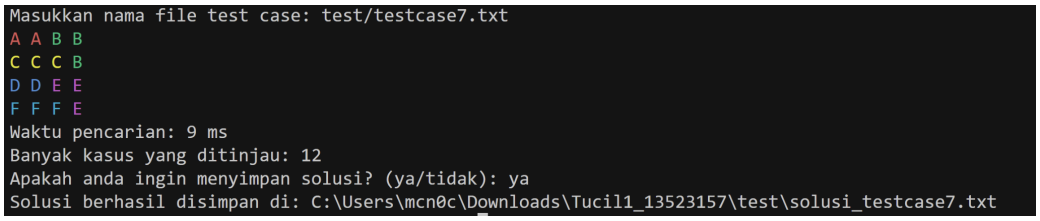
E

EE

F

FF

Output:



BAB IV LAMPIRAN

Link github: https://github.com/nataliadesiany/Tucil1_13523157.git

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 konfigurasi custom		✓
8	Program dapat menyelesaikan kasus konfigurasi piramida (3D)		✓
9	Program dibuat oleh saya sendiri	✓	