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

Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II tahun 2024/2025
Penyelesaian **IQ Puzzler Pro** dengan Algoritma Brute Force
Nadhif Radityo N. (K-1 / 13523045)
https://github.com/NadhifRadityo/Stima-Tucil1-23045

Deskripsi Tugas

Gambar 1 Permainan IQ Puzzler Pro
(Sumber: https://www.smartgamesusa.com)

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.

Komponen penting dari permainan IQ Puzzler Pro 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.

Permainan dimulai dengan **papan yang kosong**. Pemain dapat meletakkan blok puzzle sedemikian sehingga **tidak ada blok yang bertumpang tindih** (kecuali dalam kasus 3D). Setiap

blok puzzle dapat **dirotasikan** maupun **dicerminkan**. Puzzle dinyatakan **selesai** jika dan hanya jika papan **terisi penuh** dan **seluruh blok puzzle berhasil diletakkan**.

Tugas anda adalah 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.

Ilustrasi kasus
Diberikan sebuah wadah berukuran 11 x 5 serta 12 buah blok puzzle dan beberapa blok telah ditempatkan dengan bentuk sebagai berikut.

Gambar 2 Awal Permainan Game IQ Puzzler Pro
(Sumber: https://www.smartgamesusa.com)

Pemain berusaha untuk mengisi bagian papan yang kosong dengan menggunakan blok yang tersedia

Gambar 3 Pemain Mencoba Semua Kemungkinan
(Sumber: https://www.smartgamesusa.com)

Puzzle berikut dinyatakan telah selesai karena papan sudah terisi penuh dan seluruh blok telah digunakan.

Gambar 4 Pemain Menyelesaikan Permainan
(Sumber: https://www.smartgamesusa.com)

Agar lebih jelas, amati video cara bermain berikut:
https://youtube.com/shorts/MWiPAS3wfGM?feature=shared

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**.

File .txt yang akan dibaca memiliki format sebagai berikut

```
N M P
S
```

```
puzzle_1_shape
puzzle_2_shape
…
puzzle_P_shape
```

Contoh *Test case*

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

Fitur Utama

- **Branching** – Memecah proses pencarian solusi ke dalam beberapa solver untuk efisiensi.
- **Threading Support** – Memungkinkan pemrosesan paralel untuk meningkatkan performansi.
- **Penyimpanan Fast Bit Field** – Memanfaatkan penyimpanan bit-field untuk comparation cepat.
- **Kustomisasi Board** –Menentukan bentuk papan dan piece puzzle sendiri.
- **Load/Save Case File** – Mampu loading dan saving hasil solusi dalam format txt.

Persyaratan Sistem

- Java **minimal versi 21**
- Rekomendasi runtime: **GraalVM, RedHat, atau Eclipse Adoptium**

Cara Menjalankan Aplikasi

- Unduh file JAR dari GitHub Releases.
- Jalankan dengan salah satu metode berikut:
  - *Double-click* pada file JAR. (Akan menjalankan opsi default, lihat peringatan di bawah.)
  - Atau gunakan perintah terminal:

```
java -jar Stima-Tucil1-23045.jar
```

Command Line Options

- **-b, --branch <arg>**
  Menentukan jumlah branch yang digunakan dalam algoritma pencarian solusi.
  Default: 8
- **-c, --virtual-console**
  Mengaktifkan mode virtual console. Akan membuka console swing yang lebih interaktif.
- **-h, --help**
  Menampilkan pesan bantuan yang berisi daftar opsi ini.
- **-i, --input <arg>**
  Menentukan file input yang berisi konfigurasi papan dan piece puzzle yang akan diselesaikan.
- **-o, --output <arg>**
  Menentukan lokasi file output untuk menyimpan solusi yang dihasilkan oleh program.
- **-s, --solutions <arg>**
  Menentukan jumlah solusi yang akan dihasilkan. Jika ada banyak kemungkinan solusi, program akan berhenti setelah mencapai jumlah ini.
  Default: 100
- **-t, --thread <arg>**
  Menentukan jumlah *thread* yang digunakan untuk menjalankan solver secara paralel.
  Default: 2
- **-u, --no-interactive**
  Menonaktifkan mode console interaktif, program akan berjalan dengan kompatibel dengan console non-interactive.

Perhatian: Untuk memastikan komparasi yang adil dengan program lain, jalankan program ini dengan konfigurasi minimum:

```
java -jar Stima-Tucil1-23045.jar -b 1 -t 1 -s 1
```

Dengan opsi tersebut, program akan dijalankan dengan 1 branch (tidak ada branching), dengan maksimum 1 thread, dan keluaran solusi yang diinginkan hanya 1.

Threading diimplementasikan karena author gereget optimasi.

Analisis Hasil Laporan

1. Test Case #1: Basic Case

   Diberikan test case sebagai berikut:

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

Tiap-tiap piece memiliki permutasi sebagai berikut:

```
Board width: 5
Board height: 5
Board depth: 1
Board type: DEFAULT
Board shape:
Z: 0
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX

Piece A: 4 shapes, 64 compiled shapes
Z: 0
A
AA

Piece B: 4 shapes, 64 compiled shapes
Z: 0
B
BB

Piece C: 4 shapes, 64 compiled shapes
```

```
Z: 0
C
CC

Piece D: 4 shapes, 64 compiled shapes
Z: 0
D
DD

Piece E: 8 shapes, 96 compiled shapes
Z: 0
EE
EE
E

Piece F: 8 shapes, 96 compiled shapes
Z: 0
FF
FF
F

Piece G: 2 shapes, 30 compiled shapes
Z: 0
GGG
```

Dari tiap-tiap piece tersebut maka, dapat diprediksi banyak posibilitas yang dimiliki oleh board, yaitu: 64 * 64 * 64 * 64 * 96 * 96 * 30 = 4638564679680.

Salah Satu Hasil Solusi Test Case #1: Terdapat 100 alternatif solusi yang dibuat. Solver untuk solusi gambar di atas setidaknya memakan waktu 0.19 ms (dengan 141.16ms termasuk waktu scheduling thread).

2. Test Case #2: Custom Board
   Diberikan test case sebagai berikut:

```
5 7 5
CUSTOM
...X...
.XXXXX.
XXXXXXX
.XXXXX.
...X...
A
AAA
BB
BBB
CCCC
 C
D
EEE
E
```

Tiap-tiap piece memiliki permutasi sebagai berikut:

```
Board width: 7
Board height: 5
Board depth: 1
Board type: CUSTOM
Board shape:
Z: 0
   X
 XXXXX
XXXXXXX
 XXXXX
   X

Piece A: 8 shapes, 52 compiled shapes
Z: 0
A
AAA

Piece B: 8 shapes, 48 compiled shapes
Z: 0
BB
BBB
```

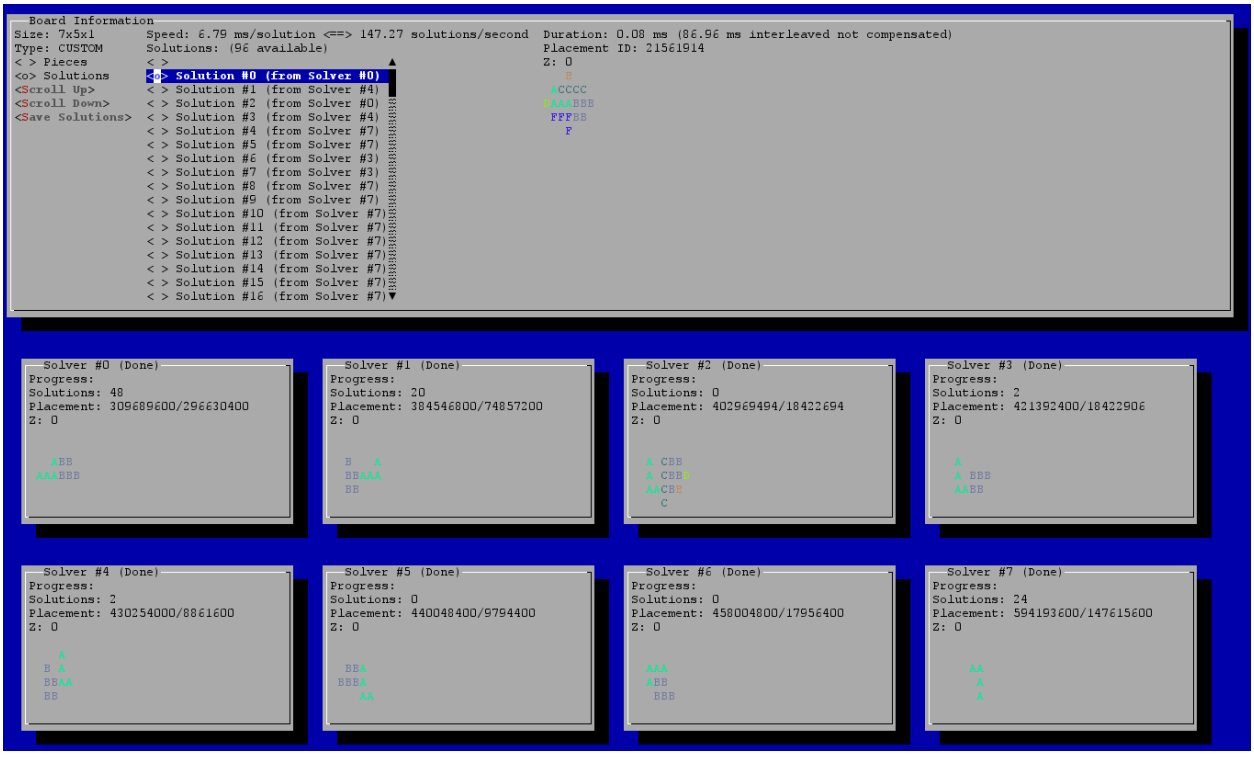
```
Piece C: 2 shapes, 10 compiled shapes
Z: 0
CCCC

Piece D: 1 shapes, 19 compiled shapes
Z: 0
D

Piece E: 1 shapes, 19 compiled shapes
Z: 0
E

Piece F: 8 shapes, 52 compiled shapes
Z: 0
FFF
F
```

Dari tiap-tiap piece tersebut maka, dapat diprediksi banyak posibilitas yang dimiliki oleh board, yaitu: 52 * 48 * 10 * 19 * 19 * 52 = 468549120.



Salah Satu Hasil Solusi Test Case #2: Terdapat 96 alternatif solusi yang dibuat. Solver untuk solusi gambar di atas setidaknya memakan waktu 0.08 ms (dengan 86.96ms termasuk waktu scheduling thread).

Analisis Performansi

Dengan menggunakan referensi dari Q&A mengenai early stopping yang ditanyakan oleh Fariz Rifqi (13523069), proyek ini dapat melompati beberapa kemungkinan bruteforce yang tidak akan menjadi solusi.

```
public void step() {
	if(this.completed) return;
	var compiledPieces = this.board.getCompiledPieces();
	var compiledPiece = compiledPieces[this.currentCompiledPieceIndex];
	var compiledShapes = compiledPiece.getCompiledShapes();
	var oldCompiledPiecePlacement =
this.board.getCompiledPiecePlacement(compiledPiece);
	var newCompiledPiecePlacement = oldCompiledPiecePlacement + 1;
	var multiplier =
this.multiplierCache[this.currentCompiledPieceIndex];
	if(newCompiledPiecePlacement >= compiledShapes.length) {
		if(this.currentCompiledPieceIndex <= 0) {
			this.completed = true;
			return;
		}
		this.board.removeCompiledPiece(compiledPiece);
		this.currentPlacement -= (oldCompiledPiecePlacement + 1) *
multiplier;
		this.currentCompiledPieceIndex--;
		return;
	}
	var intersecting = this.board.putCompiledPiece(compiledPiece,
newCompiledPiecePlacement);
	this.currentPlacement += (newCompiledPiecePlacement -
oldCompiledPiecePlacement) * multiplier;
	if(this.currentPlacement < this.minPlacement ||
this.currentPlacement >= this.maxPlacement) {
		this.completed = true;
		return;
	}
	if(intersecting)
		return;
	if(this.currentCompiledPieceIndex + 1 < compiledPieces.length)
		this.currentCompiledPieceIndex++;
	return;
}
```

Kode di atas merupakan inti dari solver. Dengan memanfaatkan suatu ID untuk setiap placement, program dapat memecah beberapa solver menjadi sub-solver lainnya dengan rata. ID placement yang unik juga memungkinkan solver untuk loncat. Meskipung besar placement ID sebanding dengan banyak permutasi yang berkembang secara pesat, dalam projek ini penggunaan placement ID dinilai memudahkan (walaupun pada awalnya ingin menggunakan prinsip counter, tapi terbuktikan dengan kompleksitas penulisan kode yang lebih rumit).

```
protected static long computeMaxPlacement(Board board) {
	long result = 1;
	for(var compiledPiece : board.getCompiledPieces())
		result *= compiledPiece.getCompiledShapes().length + 1;
	return result;
}
protected long getBoardPlacement() {
	var compiledPieces = this.board.getCompiledPieces();
	long placement = 0;
	long multiplier = 1;
	for(int i = compiledPieces.length - 1; i >= 0; i--) {
		var compiledPiece = compiledPieces[i];
		var compiledPiecePlacement =
this.board.getCompiledPiecePlacement(compiledPiece);
		placement += (compiledPiecePlacement + 1) * multiplier;
		multiplier *= (compiledPiece.getCompiledShapes().length + 1);
	}
	return placement;
}
protected void setBoardPlacement(long placement) {
	this.board.reset();
	var compiledPieces = this.board.getCompiledPieces();
	for(int i = compiledPieces.length - 1; i >= 0; i--) {
		var compiledPiece = compiledPieces[i];
		var shapesCount = compiledPiece.getCompiledShapes().length +
1;
		var compiledPiecePlacement = (int) (placement % shapesCount);
		if(compiledPiecePlacement > 0)
			this.board.putCompiledPiece(compiledPiece,
compiledPiecePlacement - 1);
		else
			this.board.removeCompiledPiece(compiledPiece);
		placement /= shapesCount;
	}
}
```

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	✓	