

Laporan Tugas Kecil I

IF2211 STRATEGI ALGORITMA

Muhammad Luqman Hakim

13523044

Institut Teknologi Bandung

2024

Daftar Isi

Daftar Isi	2
Deskripsi Masalah	3
Algoritma	3
Eksperimen	5
Eksperimen Mode Default	5
Eksperimen Mode Custom	7
Lampiran	9
Tautan Repository	9

Deskripsi Masalah

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.

Algoritma

Pengisian blok ke papan puzzle dapat digambarkan dengan sebuah pohon yang memiliki akar berupa papan kosong dan setiap anak dari sebuah simpul adalah suatu kemungkinan pengisian satu blok ke papan tersebut. Untuk mencari hanya satu solusi untuk sebuah permainan IQ Puzzler Pro, dapat digunakan algoritma Brute-Force dengan metode Depth First Search. Pada algoritma ini, setiap kasus dicoba dengan mengikuti sebuah percabangan sejauh mungkin dan melakukan *backtracking* jika jawaban tidak ditemukan di akhir percabangan.

Pada program yang dibuat, puzzle dan bloknya direpresentasikan sebagai array of array of Character. Pada awalnya seluruh elemen array ini berisi null. Kemudian program mencari seluruh kemungkinan penempatan dari satu blok. Program akan menempatkan kemungkinan pertama dan mencari lagi seluruh kemungkinan penempatan dari blok berikutnya dan mencoba kemungkinan pertamanya, begitu pula seterusnya. Jika terdapat posisi kosong dan/atau blok yang belum digunakan tetapi tidak ada kemungkinan penempatan lagi, program akan melakukan backtracking, yaitu mengurungkan penempatan sebelumnya dan mencoba kemungkinan setelahnya.

Di bawah ini adalah implementasi dari algoritma tersebut.

```
public String solve(){
    long start = System.nanoTime();
    String res = this.solveHelper();
    long end = System.nanoTime();
    this.solveTime = (end - start) / 1000000;
    return res;
}
private String solveHelper(){
    ArrayList<Integer> rows = new ArrayList<>();
    ArrayList<Integer> cols = new ArrayList<>();
    Shape head = this.shapes.remove(0);
    for (int i = 0; i < 8; i++){
        Shape currShape = head.rotation(i);
        if (i >= 4){
            currShape = currShape.reflection();
        }
        this.generatePossiblites(currShape, rows, cols);
        for (int j = 0; j < rows.size(); j++){
            this.caseVisited += 1;
            CustomPuzzle nextStep = new CustomPuzzle(
                this.row,
                this.col,
                this.p,
                this.gm,
                new ArrayList<Shape>(this.shapes),
                fillBoard(this.board, currShape, rows.get(j), cols.get(j))
            );
            if (nextStep.checkWin() && shapes.size() == 0){
                this.board = nextStep.board;
                return nextStep.boardToString();
            }
            String res = nextStep.solveHelper();
            this.caseVisited += nextStep.caseVisited;
            if (res != null){
                this.solution = res;
                this.board = nextStep.board;
                return res;
            }
        }
        rows.clear();
        cols.clear();
    }
    return null;
}
```

Eksperimen

File input disimpan di folder input dan outputnya di folder test.

Eksperimen Mode Default

1. Input valid dan bisa diselesaikan

<pre>1 1 1 DEFAULT A</pre>	<pre>java -classpath ./bin IQ.App Input File: input/testdefault1.txt Output File: test/testdefault1.txt Solving... A Waktu pencarian: 54ms Banyak kasus yang ditinjau: 1</pre>
----------------------------	--

2. Input valid dan tidak bisa diselesaikan (blok terlalu besar)

<pre>1 1 1 DEFAULT AAA AAAAA</pre>	<pre>java -classpath ./bin IQ.App Input File: input/testdefault2.txt Output File: test/testdefault2.txt Solving... No solution Waktu pencarian: 0ms Banyak kasus yang ditinjau: 0</pre>
------------------------------------	---

3. Input blok tidak valid

<pre>1 2 3 DEFAULT ABCD</pre>	<pre>Input File: input/testdefault3.txt Output File: test/testdefault3.txt Solving... Two or more different letters in the same line Line: "ABCD"</pre>
-------------------------------	---

4. Input baris, kolom, dan banyak blok tidak valid

```
0 0 0
DEFAULT
```

```
java -classpath ./bin IQ.App
Input File:
input/testdefault4.txt
Output File:
test/testdefault4.txt
Solving...
number of row/column/p must be at least 1
```

5. input baris, kolom, dan banyak blok tidak valid

```
1
DEFAULT
A
```

```
Input File:
input/testdefault5.txt
Output File:
test/testdefault5.txt
Solving...
invalid number of row/column/p
```

6. input valid dan bisa diselesaikan

```
5 5 6
DEFAULT
AAAA
BBB
B
CCC
C
DD
DD
EE
EE
FF
F
F
F
java -classpath ./bin IQ.App
Input File:
input/testdefault6.txt
Output File:
test/testdefault6.txt
Solving...
AAAC
BBBC
BDDE
FDDE
FFFFE
Waktu pencarian: 99ms
Banyak kasus yang ditinjau: 268
```

7. input valid dan bisa diselesaikan

```

5 8 8
DEFAULT
HHHHHHH
A
  C
CCC
C
  B
BBB
B
E
EE
EEE
G
G
G
G
G
  D
DD
DD
F F
FFF
F

java -classpath ./bin IQ.App
Input File:
input/testdefault7.txt
Output File:
test/testdefault7.txt
Solving...
HHHHHHHF
GGGGGFF
EDDCCBF
EEDDCBBB
EEEDCCBA
Waktu pencarian: 25815ms
Banyak kasus yang ditinjau: 4754110

```

8. input valid dan tidak bisa diselesaikan

```

2 5 3
DEFAULT
AAA
A
BB
B
DD
D

java -classpath ./bin IQ.App
Input File:
input/testdefault8.txt
Output File:
test/testdefault8.txt
Solving...
No solution
Waktu pencarian: 9ms
Banyak kasus yang ditinjau: 92

```

Eksperimen Mode Custom

1. Input valid dan dapat diselesaikan


```
testcustom1.txt
```

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

```
java -classpath ./bin IQ.App
```

```
Input File:
```

```
input/testcustom1.txt
```

```
Output File:
```

```
test/testcustom1.txt
```

```
Solving...
```

```
...A...
```

```
.BBAAA.
```

```
BBBCCCC
```

```
.EEECD.
```

```
...E...
```

```
Waktu pencarian: 85ms
```

```
Banyak kasus yang ditinjau: 593
```

2. Input papan tidak valid

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

```
java -classpath ./bin IQ.App
```

```
Input File:
```

```
input/testcustom2.txt
```

```
Output File:
```

```
test/testcustom2.txt
```

```
Solving...
```

```
Not enough cells in a row
```

3. tidak ada input papan

```
5 7 5
CUSTOM
A
AAA
BB
BBB
CCCC
C
D
EEE
E

Input File:
input/testcustom3.txt
Output File:
test/testcustom3.txt
Solving...
Not enough cells in a row
```

4. input valid dan dapat diselesaikan

```
3 3 2
CUSTOM
XXX
X.X
XXX
AAA
A
A
BB
B

Input File:
input/testcustom4.txt
Output File:
test/testcustom4.txt
Solving...
AAA
A.B
ABB
Waktu pencarian: 54ms
Banyak kasus yang ditinjau: 2
```

Lampiran

Tautan Repository

https://github.com/mlqmn/Tucil1_13523044/