

**LAPORAN TUGAS KECIL 3**  
**IF2211 STRATEGI ALGORITMA**  
**PENYELESAIAN PERSOALAN 15-PUZZLE DENGAN ALGORITMA**  
***BRANCH AND BOUND***



Oleh  
Firizky Ardiansyah  
13520095  
K02

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2022**

## BAB 1

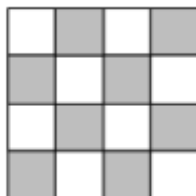
### ALGORITMA *BRANCH AND BOUND*

Algoritma *Branch and Bound* merupakan sebuah teknik penyelesaian persoalan komputasi yang digunakan dalam persoalan optimasi, yaitu meminimalkan atau memaksimalkan suatu fungsi objektif tanpa melanggar batasan (*constraints*) persoalan. *Branch and Bound* atau B&B pada implementasinya memanfaatkan struktur data graf untuk pencarian solusi. Graf yang ditelusuri pada algoritma ini berupa graf pohon dan algoritma untuk menelusuri simpul-simpul pada pohon pada dasarnya menggunakan algoritma BFS (*Breadth First Search*) yang telah dimodifikasi.

Algoritma BFS pada graf secara umum langkah pertama yang dilakukan adalah pemilihan simpul awal untuk diproses. Pada graf pohon, simpul yang dipilih merupakan simpul akar dari graf pohon. Proses traversal dimulai dengan mengunjungi semua simpul yang bertetangga dengan simpul yang dipilih. Simpul-simpul ini urutannya ditentukan berdasarkan prinsip *first in first out* (FIFO), sehingga struktur data dari algoritma BFS secara umum berupa struktur data antrean. Simpul yang bertetangga dengan simpul yang sedang diproses masuk ke dalam antrean, kemudian simpul selanjutnya yang akan diproses adalah simpul yang masuk ke dalam antrean paling awal. Simpul-simpul yang sudah dikunjungi akan diberi status, sehingga pada proses traversal selanjutnya, tidak ada simpul yang dikunjungi dua kali.

Pada algoritma B&B, berbeda dengan BFS, simpul yang diekspansi tidak memenuhi urutan dari prinsip FIFO. Simpul yang ditelusuri pada algoritma B&B didasarkan pada simpul yang memiliki *cost* paling kecil (paling besar jika fungsi objektifnya maksimasi). *Cost* sebuah simpul didefinisikan sebagai nilai taksiran lintasan termurah ke simpul status tujuan. Perlu diperhatikan bahwa pencarian nilai taksiran harus selalu dapat dibuktikan bisa membatasi simpul lainnya sehingga dapat membatasi simpul yang perlu ditelusuri. Simpul-simpul yang nilai *cost*-nya melebihi *cost* sebuah lintasan yang mampu mencapai simpul solusi akan ‘dibunuh’ sehingga tidak akan lagi diperiksa.

Pada kasus 15-Puzzle, pencarian menggunakan B&B pun tidak dijamin untuk berjalan secara optimal. Heuristik algoritma sangat perlu diperhatikan, sehingga jumlah simpul yang dibangun tidak terlalu banyak. Salah satu heuristik yang digunakan dalam algoritma pencarian solusi ini adalah dengan pengecekan awal apakah status tujuan dapat dicapai atau tidak. Definisikan Kurang( $i$ ) sebagai banyaknya ubin  $j$  sedemikian sehingga  $j < i$  dan  $POSISI(j) > POSISI(i)$ ,  $POSISI(i)$  = posisi ubin bernomor  $i$  pada susunan yang diperiksa, Definisikan juga  $X$ , bernilai 1 jika sel kosong berada pada daerah di arsir pada papan berikut.



Dengan mencari  $\Sigma_1^{16} Kurang(i) + X$ , terdapat teorema yang menyebutkan bahwa jika hasil tersebut berniali genap, status tujuan dapat dicapai dari status awal. Sebaliknya, jika hasil yang diperoleh bernilai ganjil, status tujuan tidak dapat dicapai.

Jika status tujuan dapat dicapai, proses selanjutnya adalah mencari lintasan solusi. Letak simpul solusi pada umumnya tidak diketahui, sehingga *cost* dari lintasan sebuah status merupakan sebuah taksiran. Adapun *cost* pada kasus ini didefinisikan sebagai

$$\hat{c}(i) = \hat{f}(i) + \hat{g}(i)$$

$\hat{c}(i)$  = ongkos untuk simpul  $i$ .

$\hat{f}(i)$  = ongkos mencapai simpul  $i$  dari akar, panjang lintasan dari simpul akar ke  $i$ .

$\hat{g}(i)$  = ongkos mencapai simpul tujuan dari simpul  $i$ , taksiran panjang lintasan terpendek dari  $i$  ke simpul solusi pada upapohon yang akarnya  $i$ .

Taksiran *cost* yang digunakan adalah jumlah ubin tidak kosong yang tidak berada pada tempat sesuai susunan akhir. Adapun susunan akhir yang diinginkan adalah susunan berikut.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Secara implementasi, sebuah status diinstansiasi dalam sebuah kelas berisi susunan matriks dan tetangganya yang mungkin. Pencarian ongkos dilakukan sesuai definisi. Setiap status yang diekspansi dimasukkan ke dalam *priority queue* atau *min heap* dengan *key* yang digunakan adalah *cost* dari status tersebut. *Prioirity queue* akan memasukkan status berdasarkan *cost* terkecil, sehingga urutan penelusuran akan sesuai dengan yang diharapkan dalam algoritma *B&B*. Ketika status tujuan dicapai, proses pohon akan berhenti dan setiap simpul yang belum diekspansi akan dibunuh.

## BAB 2

### TANGKAPAN LAYAR INPUT DAN OUTPUT PROGRAM

#### 2.1 Input Format tidak Sesuai

The screenshot shows the '15 Puzzle Solver' application window. On the left, the 'Input:' section contains an 'Import' button and a text area with the text 'sadhasd'. Below this, a red error message 'Input format is invalid!' is displayed. To the right of the error message are three buttons: 'Create', 'Randomize', and 'Solve'. The 'Output:' section is empty. On the right side of the window, a 4x4 grid of puzzle tiles is shown. The tiles are numbered 1 through 15, with the bottom-right tile (position 16) being empty. The tiles are arranged in a 4x4 grid, with the first three rows containing tiles 1-12 and the fourth row containing tiles 13-15 and an empty space.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

#### 2.2 Status Tujuan Tidak dapat Dicapai

Input:

Import

1 2 3 4

5 6 7 8

9 10 16 12

13 14 15 11

Create

Randomize

Reset

Solve

Output:

Goal is not reachable

See Analysis

1

2

3

4

5

6

7

8

9

10

12

13

14

15

11

Analysis

Initial Position:

1

2

3

4

5

6

7

8

9

10

12

13

14

15

11

Reachable Analysis:

i	KURANGI[i]
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	1
13	1
14	1
15	1
16	5
Sum	9
Sum + X	9

Goal is not reachable

Export

Hasil *export* output:

```
not_reachable0.txt - Notepad
File Edit Format View Help
INITIAL POSITION:
1 2 3 4
5 6 7 8
9 10 16 12
13 14 15 11

KURANGI TABLE:
KURANGI(1) = 0
KURANGI(2) = 0
KURANGI(3) = 0
KURANGI(4) = 0
KURANGI(5) = 0
KURANGI(6) = 0
KURANGI(7) = 0
KURANGI(8) = 0
KURANGI(9) = 0
KURANGI(10) = 0
KURANGI(11) = 0
KURANGI(12) = 1
KURANGI(13) = 1
KURANGI(14) = 1
KURANGI(15) = 1
KURANGI(16) = 5
SUM = 9
SUM + X = 9

SOLUTION PATH:
No Solution Path Exist|

STATUS:
Goal is not reachable
```

### 2.3 Status Tujuan Dapat Dicapai

15 Puzzle Solver

Input:

Import

1 2 3 4 |

5 11 7 16

9 10 12 6

13 8 15 14

Create

Randomize

Reset

Solve

Output:

Goal is Reachable!

Time elapsed: 2.3966 s

Number of moves: 26 moves

Generated nodes: 554542 nodes

See Analysis

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Initial Position:

1

2

3

4

5

11

7

9

10

12

6

13

8

15

14

Reachable Analysis:

i	KURANGI[i]
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	2
10	2
11	5
12	2
13	1
14	0
15	1
16	8
Sum	22
Sum + X	22

Goal is Reachable!

Time elapsed: 2.3966 s

Number of moves: 26 moves

Generated nodes: 554542 nodes

Export

Visualisasi Langkah:

<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>11</td><td>7</td><td></td></tr><tr><td>9</td><td>10</td><td>12</td><td>6</td></tr><tr><td>13</td><td>8</td><td>15</td><td>14</td></tr></table>	1	2	3	4	5	11	7		9	10	12	6	13	8	15	14	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>11</td><td>7</td><td>6</td></tr><tr><td>9</td><td>10</td><td>12</td><td></td></tr><tr><td>13</td><td>8</td><td>15</td><td>14</td></tr></table>	1	2	3	4	5	11	7	6	9	10	12		13	8	15	14	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>11</td><td>7</td><td>6</td></tr><tr><td>9</td><td>10</td><td></td><td>12</td></tr><tr><td>13</td><td>8</td><td>15</td><td>14</td></tr></table>	1	2	3	4	5	11	7	6	9	10		12	13	8	15	14	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>11</td><td>7</td><td>6</td></tr><tr><td>9</td><td>10</td><td>15</td><td>12</td></tr><tr><td>13</td><td>8</td><td></td><td>14</td></tr></table>	1	2	3	4	5	11	7	6	9	10	15	12	13	8		14
1	2	3	4																																																																
5	11	7																																																																	
9	10	12	6																																																																
13	8	15	14																																																																
1	2	3	4																																																																
5	11	7	6																																																																
9	10	12																																																																	
13	8	15	14																																																																
1	2	3	4																																																																
5	11	7	6																																																																
9	10		12																																																																
13	8	15	14																																																																
1	2	3	4																																																																
5	11	7	6																																																																
9	10	15	12																																																																
13	8		14																																																																
<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>11</td><td>7</td><td>6</td></tr><tr><td>9</td><td>10</td><td>15</td><td>12</td></tr><tr><td>13</td><td>8</td><td>14</td><td></td></tr></table>	1	2	3	4	5	11	7	6	9	10	15	12	13	8	14		<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>11</td><td>7</td><td>6</td></tr><tr><td>9</td><td>10</td><td>15</td><td></td></tr><tr><td>13</td><td>8</td><td>14</td><td>12</td></tr></table>	1	2	3	4	5	11	7	6	9	10	15		13	8	14	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>11</td><td>7</td><td></td></tr><tr><td>9</td><td>10</td><td>15</td><td>6</td></tr><tr><td>13</td><td>8</td><td>14</td><td>12</td></tr></table>	1	2	3	4	5	11	7		9	10	15	6	13	8	14	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>11</td><td></td><td>7</td></tr><tr><td>9</td><td>10</td><td>15</td><td>6</td></tr><tr><td>13</td><td>8</td><td>14</td><td>12</td></tr></table>	1	2	3	4	5	11		7	9	10	15	6	13	8	14	12
1	2	3	4																																																																
5	11	7	6																																																																
9	10	15	12																																																																
13	8	14																																																																	
1	2	3	4																																																																
5	11	7	6																																																																
9	10	15																																																																	
13	8	14	12																																																																
1	2	3	4																																																																
5	11	7																																																																	
9	10	15	6																																																																
13	8	14	12																																																																
1	2	3	4																																																																
5	11		7																																																																
9	10	15	6																																																																
13	8	14	12																																																																
<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td></td><td>11</td><td>7</td></tr><tr><td>9</td><td>10</td><td>15</td><td>6</td></tr><tr><td>13</td><td>8</td><td>14</td><td>12</td></tr></table>	1	2	3	4	5		11	7	9	10	15	6	13	8	14	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>10</td><td>11</td><td>7</td></tr><tr><td>9</td><td></td><td>15</td><td>6</td></tr><tr><td>13</td><td>8</td><td>14</td><td>12</td></tr></table>	1	2	3	4	5	10	11	7	9		15	6	13	8	14	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>10</td><td>11</td><td>7</td></tr><tr><td>9</td><td>8</td><td>15</td><td>6</td></tr><tr><td>13</td><td></td><td>14</td><td>12</td></tr></table>	1	2	3	4	5	10	11	7	9	8	15	6	13		14	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>10</td><td>11</td><td>7</td></tr><tr><td>9</td><td>8</td><td>15</td><td>6</td></tr><tr><td>13</td><td>14</td><td></td><td>12</td></tr></table>	1	2	3	4	5	10	11	7	9	8	15	6	13	14		12
1	2	3	4																																																																
5		11	7																																																																
9	10	15	6																																																																
13	8	14	12																																																																
1	2	3	4																																																																
5	10	11	7																																																																
9		15	6																																																																
13	8	14	12																																																																
1	2	3	4																																																																
5	10	11	7																																																																
9	8	15	6																																																																
13		14	12																																																																
1	2	3	4																																																																
5	10	11	7																																																																
9	8	15	6																																																																
13	14		12																																																																
<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>10</td><td>11</td><td>7</td></tr><tr><td>9</td><td>8</td><td></td><td>6</td></tr><tr><td>13</td><td>14</td><td>15</td><td>12</td></tr></table>	1	2	3	4	5	10	11	7	9	8		6	13	14	15	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>10</td><td>11</td><td>7</td></tr><tr><td>9</td><td>8</td><td>6</td><td></td></tr><tr><td>13</td><td>14</td><td>15</td><td>12</td></tr></table>	1	2	3	4	5	10	11	7	9	8	6		13	14	15	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>10</td><td>11</td><td></td></tr><tr><td>9</td><td>8</td><td>6</td><td>7</td></tr><tr><td>13</td><td>14</td><td>15</td><td>12</td></tr></table>	1	2	3	4	5	10	11		9	8	6	7	13	14	15	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>10</td><td></td><td>11</td></tr><tr><td>9</td><td>8</td><td>6</td><td>7</td></tr><tr><td>13</td><td>14</td><td>15</td><td>12</td></tr></table>	1	2	3	4	5	10		11	9	8	6	7	13	14	15	12
1	2	3	4																																																																
5	10	11	7																																																																
9	8		6																																																																
13	14	15	12																																																																
1	2	3	4																																																																
5	10	11	7																																																																
9	8	6																																																																	
13	14	15	12																																																																
1	2	3	4																																																																
5	10	11																																																																	
9	8	6	7																																																																
13	14	15	12																																																																
1	2	3	4																																																																
5	10		11																																																																
9	8	6	7																																																																
13	14	15	12																																																																
<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>10</td><td>6</td><td>11</td></tr><tr><td>9</td><td>8</td><td></td><td>7</td></tr><tr><td>13</td><td>14</td><td>15</td><td>12</td></tr></table>	1	2	3	4	5	10	6	11	9	8		7	13	14	15	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>10</td><td>6</td><td>11</td></tr><tr><td>9</td><td></td><td>8</td><td>7</td></tr><tr><td>13</td><td>14</td><td>15</td><td>12</td></tr></table>	1	2	3	4	5	10	6	11	9		8	7	13	14	15	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td></td><td>6</td><td>11</td></tr><tr><td>9</td><td>10</td><td>8</td><td>7</td></tr><tr><td>13</td><td>14</td><td>15</td><td>12</td></tr></table>	1	2	3	4	5		6	11	9	10	8	7	13	14	15	12	<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>5</td><td>6</td><td></td><td>11</td></tr><tr><td>9</td><td>10</td><td>8</td><td>7</td></tr><tr><td>13</td><td>14</td><td>15</td><td>12</td></tr></table>	1	2	3	4	5	6		11	9	10	8	7	13	14	15	12
1	2	3	4																																																																
5	10	6	11																																																																
9	8		7																																																																
13	14	15	12																																																																
1	2	3	4																																																																
5	10	6	11																																																																
9		8	7																																																																
13	14	15	12																																																																
1	2	3	4																																																																
5		6	11																																																																
9	10	8	7																																																																
13	14	15	12																																																																
1	2	3	4																																																																
5	6		11																																																																
9	10	8	7																																																																
13	14	15	12																																																																



<div>1234</div> <div>5611</div> <div>91087</div> <div>13141512</div>	<div>1234</div> <div>56117</div> <div>9108</div> <div>13141512</div>	<div>1234</div> <div>56117</div> <div>910</div> <div>13141512</div>	<div>1234</div> <div>56</div> <div>910118</div> <div>13141512</div>
<div>1234</div> <div>567</div> <div>910118</div> <div>13141512</div>	<div>1234</div> <div>5678</div> <div>9101112</div> <div>131415</div>		

## Hasil *export* pencarian:

<p>INITIAL POSITION: 1 2 3 4 5 11 7 16 9 10 12 6 13 8 15 14</p> <p>KURANGI TABLE: KURANGI(1) = 0 KURANGI(2) = 0 KURANGI(3) = 0 KURANGI(4) = 0 KURANGI(5) = 0 KURANGI(6) = 0 KURANGI(7) = 1 KURANGI(8) = 0 KURANGI(9) = 2 KURANGI(10) = 2 KURANGI(11) = 5 KURANGI(12) = 2 KURANGI(13) = 1 KURANGI(14) = 0 KURANGI(15) = 1 KURANGI(16) = 8 SUM = 22 SUM + X = 22</p> <p>SOLUTION PATH: MOVE: 0 DIRECTION: - 1 2 3 4 5 11 7 16 9 10 12 6 13 8 15 14</p> <p>MOVE: 1 DIRECTION: DOWN 1 2 3 4 5 11 7 6 9 10 12 16 13 8 15 14</p>	<p>MOVE: 2 DIRECTION: LEFT 1 2 3 4 5 11 7 6 9 10 16 12 13 8 15 14</p> <p>MOVE: 3 DIRECTION: DOWN 1 2 3 4 5 11 7 6 9 10 15 12 13 8 16 14</p> <p>MOVE: 4 DIRECTION: RIGHT 1 2 3 4 5 11 7 6 9 10 15 12 13 8 14 16</p> <p>MOVE: 5 DIRECTION: UP 1 2 3 4 5 11 7 6 9 10 15 16 13 8 14 12</p> <p>MOVE: 6 DIRECTION: UP 1 2 3 4 5 11 7 16 9 10 15 6 13 8 14 12</p> <p>MOVE: 7 DIRECTION: LEFT</p>	<p>MOVE: 7 DIRECTION: LEFT 1 2 3 4 5 11 16 7 9 10 15 6 13 8 14 12</p> <p>MOVE: 8 DIRECTION: LEFT 1 2 3 4 5 16 11 7 9 10 15 6 13 8 14 12</p> <p>MOVE: 9 DIRECTION: DOWN 1 2 3 4 5 10 11 7 9 16 15 6 13 8 14 12</p> <p>MOVE: 10 DIRECTION: DOWN 1 2 3 4 5 10 11 7 9 8 15 6 13 16 14 12</p> <p>MOVE: 11 DIRECTION: RIGHT 1 2 3 4 5 10 11 7 9 8 15 6 13 14 16 12</p> <p>MOVE: 12 DIRECTION: UP</p>	<p>MOVE: 12 DIRECTION: UP 1 2 3 4 5 10 11 7 9 8 16 6 13 14 15 12</p> <p>MOVE: 13 DIRECTION: RIGHT 1 2 3 4 5 10 11 7 9 8 6 16 13 14 15 12</p> <p>MOVE: 14 DIRECTION: UP 1 2 3 4 5 10 11 16 9 8 6 7 13 14 15 12</p> <p>MOVE: 15 DIRECTION: LEFT 1 2 3 4 5 10 16 11 9 8 6 7 13 14 15 12</p> <p>MOVE: 16 DIRECTION: DOWN 1 2 3 4 5 10 6 11 9 8 16 7 13 14 15 12</p> <p>MOVE: 17 DIRECTION: LEFT</p>	<p>MOVE: 17 DIRECTION: LEFT 1 2 3 4 5 10 6 11 9 16 8 7 13 14 15 12</p> <p>MOVE: 18 DIRECTION: UP 1 2 3 4 5 16 6 11 9 10 8 7 13 14 15 12</p> <p>MOVE: 19 DIRECTION: RIGHT 1 2 3 4 5 6 16 11 9 10 8 7 13 14 15 12</p> <p>MOVE: 20 DIRECTION: RIGHT 1 2 3 4 5 6 11 16 9 10 8 7 13 14 15 12</p> <p>MOVE: 21 DIRECTION: DOWN 1 2 3 4 5 6 11 7 9 10 8 16 13 14 15 12</p> <p>MOVE: 22 DIRECTION: LEFT</p>	<p>MOVE: 22 DIRECTION: LEFT 1 2 3 4 5 6 11 7 9 10 16 8 13 14 15 12</p> <p>MOVE: 23 DIRECTION: UP 1 2 3 4 5 6 16 7 9 10 11 8 13 14 15 12</p> <p>MOVE: 24 DIRECTION: RIGHT 1 2 3 4 5 6 7 16 9 10 11 8 13 14 15 12</p> <p>MOVE: 25 DIRECTION: DOWN 1 2 3 4 5 6 7 8 9 10 11 16 13 14 15 12</p> <p>MOVE: 26 DIRECTION: DOWN 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</p> <p>STATUS: Goal is Reachable! Time elapsed: 2.3966 s Number of moves: 26 moves Generated nodes: 954542 nodes</p>
--	---	--	---	--	--

### BAB 3

#### ***TABEL CHECK LIST***

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil <i>running</i>	√	
3. Program dapat menerima input dan menuliskan output	√	
4. Luaran sudah benar untuk semua data uji	√	
5. Bonus dibuat	√	

## BAB 4

### KODE PROGRAM

#### 6.1 Kelas Main

Kelas Main hanya berisi pemanggilan GUI.

```
import gui.MainWindow;

public class Main
{
    public static void main(String[] args) {

        MainWindow gui = new MainWindow();
        gui.setVisible(true);

    }
}
```

#### 6.2 Kelas Node

Cetak biru untuk objek simpul. Objek simpul berisi status yang nanti pada graf akan diproses.

```
package tree;

import enums.Direction;

import java.util.*;

public class Node {
    private int[][] board; // state matrix
    private int r, c; // empty tile position
    private List<Direction> adj; // possible direction from the state

    public int depth; // depth of the state from root
    public Node parent; // parent of state in the status tree
    public String direction; // direction taken from parent to this state

    public Node(boolean randomize){
        // root state constructor
        List<Integer> a = new ArrayList();
        for(int i=1; i<=16; i++){
            a.add(i);
        }
    }
}
```

```

        if(randomize){
            Collections.shuffle(a);
        }

        this.board = new int[4][4];
        for(int i=0; i<4; i++){
            for(int j=0; j<4; j++){
                this.board[i][j] = a.get(i*4 + j);
                if(this.board[i][j]==16){
                    this.r = i;
                    this.c = j;
                }
            }
        }

        // find all safe direction
        this.adj = new ArrayList<>();
        if(this.r != 0){
            this.adj.add(Direction.UP);
        }
        if(this.r != 3){
            this.adj.add(Direction.DOWN);
        }
        if(this.c != 0){
            this.adj.add(Direction.LEFT);
        }
        if(this.c != 3){
            this.adj.add(Direction.RIGHT);
        }

        this.depth = 0;
        this.parent = null;
        this.direction = "-";
    }

    public Node(String[][] board){
        // root state constructot from user input
        this.board = new int[4][4];
        for(int i=0; i<4; i++){
            for(int j=0; j<4; j++){
                int ref;
                try{
                    ref = Integer.parseInt(board[i][j]);
                    if(!(1<=ref&&ref<=15)){
                        throw new NumberFormatException();
                    }
                } catch(NumberFormatException e){
                    this.r = i;

```

```

        this.c = j;
        ref = 16;
    }
    this.board[i][j] = ref;
}

this.adj = new ArrayList<>();
if(this.r != 0){
    this.adj.add(Direction.UP);
}
if(this.r != 3){
    this.adj.add(Direction.DOWN);
}
if(this.c != 0){
    this.adj.add(Direction.LEFT);
}
if(this.c != 3){
    this.adj.add(Direction.RIGHT);
}

this.depth = 0;
this.parent = null;
this.direction = "-";
}

private Node(int[][] board, int r, int c, int depth, Direction d, Node
n){
    // non-root state transition. the node is being generated by its
parent n taken direction d
    this.r = r;
    this.c = c;
    this.board = board;
    this.adj = new ArrayList<>();

    // find all possible direction with previous direction taken into
account
    // we have to avoid the tree to go back to its previous state
    if(this.r != 0 && !d.equals(Direction.DOWN)){
        this.adj.add(Direction.UP);
    }
    if(this.r != 3 && !d.equals(Direction.UP)){
        this.adj.add(Direction.DOWN);
    }
    if(this.c != 0 && !d.equals(Direction.RIGHT)){
        this.adj.add(Direction.LEFT);
    }
    if(this.c != 3 && !d.equals(Direction.LEFT)){

```

```

        this.adj.add(Direction.RIGHT);
    }
    this.depth = depth;
    this.parent = n;
    this.direction = d.label;
}

public List<Node> getAdj(){
    // generate state from all possible direction this state has
    List<Node> res = new ArrayList<>();
    for(Direction d: this.adj){
        int r_new = this.r + d.dr;
        int c_new = this.c + d.dc;
        int[][] board_new = this.drag(r_new, c_new);
        int depth_new = this.depth + 1;
        res.add(new Node(board_new, r_new, c_new, depth_new, d, this));
    }
    return res;
}

private int[][] drag(int r_new, int c_new){
    // swapping two tiles in the state matrix
    int[][] board_new = this.board.clone();
    for(int i=0; i<4; i++){
        board_new[i] = board_new[i].clone();
    }
    int tmp = board_new[this.r][this.c];
    board_new[this.r][this.c] = board_new[r_new][c_new];
    board_new[r_new][c_new] = tmp;
    return board_new;
}

// getter
public int getBoard(int i, int j){
    return this.board[i][j];
}

public int[][] getBoard(){
    return this.board;
}

public int getCost(){
    // total cost of the state
    int cost = this.depth;
    int g = 0;
    int x = 1;
    for(int i=0; i<4; i++){
        for(int j=0; j<4; j++){

```

```

        if(this.board[i][j]!=x&&x!=16){
            g++;
        }
        x++;
    }
}
return cost+g;
}

public int getR(){
    return this.r;
}

public int getC(){
    return this.c;
}

// predicate
public boolean isGoal(){
    // return true if the state is a goal state
    boolean res = true;
    int x = 1;
    for(int i=0; i<4&&res; i++){
        for(int j=0; j<4&&res; j++){
            if(this.board[i][j]!=x){
                res = false;
            }
            x++;
        }
    }
    return res;
}

// helper
public Node clone(){
    Node res = new Node(false);
    res.r = this.r;
    res.c = this.c;
    res.board = this.board.clone();
    for(int i=0; i<4; i++){
        res.board[i] = res.board[i].clone();
    }
    res.adj = new ArrayList<>();
    res.adj.addAll(this.adj);
    res.depth = this.depth;
    res.direction = this.direction;
}

```

```

        return res;
    }

    public void print(){
        for(int i=0; i<4; i++){
            for(int j=0; j<4; j++){
                System.out.print(this.board[i][j] + " ");
            }
            System.out.println();
        }
    }
}

```

### 6.3 Kelas Graph

Kelas untuk objek graf secara umum digunakan untuk pencarian solusi.

```

package tree;

import java.util.*;

public class Graph {
    private Node root; // root state of the graph
    private List<Node> solutionPath; // solution path
    private int cntNode; // number of nodes generated
    private String status; // status of processed graph
    private List<Integer> KURANGI; // KURANGI List

    public Graph(boolean randomize){
        // Random Graph Constructor
        this.root = new Node(randomize);
        this.solutionPath = new ArrayList<>();
        this.cntNode = 1;
        this.KURANGI = new ArrayList<>();
        for(int i=0; i<17; i++){
            this.KURANGI.add(0);
        }
        for(int i=0; i<4; i++){
            for(int j=0; j<4; j++){
                int ref = root.getBoard(i, j);
                int cur = 0;
                for(int k=i; k<4; k++){
                    for(int l=0; l<4; l++){
                        if(k==i && l==j){
                            continue;
                        }
                        if(root.getBoard(k, l)<ref){
                            cur++;
                        }
                    }
                }
            }
        }
    }
}

```



```

        }
    }
    }
    this.KURANGI.set(ref, cur);
}
}

public Graph(String[][] root_raw){
    // Input-based graph constructor
    this.root = new Node(root_raw);
    this.solutionPath = new ArrayList<>();
    this.cntNode = 1;
    this.KURANGI = new ArrayList<>();
    for(int i=0; i<17; i++){
        this.KURANGI.add(0);
    }
    for(int i=0; i<4; i++){
        for(int j=0; j<4; j++){
            int ref = root.getBoard(i, j);
            int cur = 0;
            for(int k=i; k<4; k++){
                for(int l=0; l<4; l++){
                    if(k==i && l==j){
                        continue;
                    }
                    if(root.getBoard(k, l)<ref){
                        cur++;
                    }
                }
            }
            this.KURANGI.set(ref, cur);
        }
    }
}

public void process(){
    // B&B process
    long time = System.nanoTime();
    if(!isReachableGoal()){
        this.status = "Goal is not reachable";
        return;
    }

    PriorityQueue<Node> pq = new
PriorityQueue<>(Comparator.comparingInt(Node::getCost));
    HashMap<String, Boolean> visited = new HashMap<>();
    Node to = null;

```

```

        pq.add(root);
        visited.put(Arrays.deepToString(root.getBoard()), Boolean.TRUE);
        while(!pq.isEmpty()){
            Node p = pq.remove();
            if(p.isGoal()){
                to = p;
                break;
            }
            for(Node u: p.getAdj()){
                if(visited.containsKey(Arrays.deepToString(u.getBoard()))){
                    continue;
                }
                this.cntNode++;
                visited.put(Arrays.deepToString(u.getBoard()),
Boolean.TRUE);
                pq.add(u);
            }
        }
        for(Node u = to; u.parent!=null; u=u.parent){
            this.solutionPath.add(0, u);
        }
        this.solutionPath.add(0, root);

        Double elapsed = (System.nanoTime() - time) / 1e9;
        this.status = String.format("<html>Goal is Reachable!<br/>Time
elapsed: %.4f s<br/>Number of moves: %d moves<br/>Generated nodes: %d
nodes</html>", elapsed, solutionPath.size()-1, cntNode);
    }

    // getter
    public List<Node> getSolutionPath() {
        return this.solutionPath;
    }

    public List<Integer> getKURANGI(){
        return this.KURANGI;
    }
    public int getSumKURANGI(){
        // sum
        return this.KURANGI.stream().reduce(0, (x, y)->x+y);
    }

    public int getReachableCost(){
        // sum + x
        return this.getSumKURANGI()+(root.getR()+root.getC())%2;
    }

    public Object[][] getRowContent(){

```

```

        // content of KURANGI list
        Object[][] data = new Object[18][2];
        for(int i=1; i<=16; i++){
            data[i-1][0] = Integer.toString(i);
            data[i-1][1] = this.KURANGI.get(i);
        }
        data[16][0] = "Sum";
        data[16][1] = getSumKURANGI();
        data[17][0] = "Sum + X";
        data[17][1] = getReachableCost();
        return data;
    }

    public Node getRoot(){
        return this.root;
    }

    // predicate
    private boolean isReachableGoal(){
        return (this.getReachableCost()%2==0);
    }

    // helper
    public void print(){
        for(int i=0; i<this.solutionPath.size(); i++){
            this.solutionPath.get(i).print();
            System.out.println();
        }
    }
}
}

```

## 6.4 Kelas Direction

Enumerasi arah yang mungkin ditempuh kotak kosong.

```

package enums;

public enum Direction {
    // Direction enumeration and its displacement
    UP(-1, 0, "UP"),
    DOWN(1, 0, "DOWN"),
    RIGHT(0, 1, "RIGHT"),
    LEFT(0, -1, "LEFT");

    public final int dr;
    public final int dc;
    public final String label;
}

```

```

    Direction(int dr, int dc, String label){
        this.dr = dr;
        this.dc = dc;
        this.label = label;
    }

    public boolean equals(Direction d){
        return this.dr==d.dr && this.dc==d.dc && this.label.equals(d.label);
    }
}

```

## 6.5 Kelas MainWindow

Kelas untuk objek tampilan utama program.

```

package gui;

import tree.*;

import javax.swing.*.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.text.BadLocationException;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Arrays;
import java.util.Scanner;

public class MainWindow extends JFrame {
    protected JButton[] tiles;
    protected JButton solveButton, randomButton, importButton, createButton,
analysisButton, resetButton;
    protected JLabel outStatus, createStatus, input, output;
    protected JFileChooser inputWindows;
    protected JPanel panelOut, panelNav;
    protected JTextField pathIn;
    protected JTextArea inputText;
    protected Graph tree;

    private void createButton(JButton button){
        button.setPreferredSize(new Dimension(100, 35));
        button.setFocusable(false);
        button.setBackground(Color.decode("#4DB4D7"));
    }
}

```

```

        button.setForeground(Color.WHITE);
        button.setFont(new Font("Gill Sans", Font.PLAIN, 12));
    }

    public MainWindow(){
        // GUI Main Window
        this.tiles = new JButton[16];
        this.setResizable(false);
        this.setLayout(new GridBagLayout());

        // panelOut and its component
        this.panelOut = new JPanel();
        this.panelOut.setBackground(Color.decode("#fafafa"));
        this.panelOut.setLayout(new GridBagLayout());
        this.panelOut.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,
10));
        GridBagConstraints outConstraint = new GridBagConstraints();
        for(int i=0; i<4; i++){
            for(int j=0; j<4; j++){
                if(i!=3||j!=3){
                    this.tiles[i*4+j] = new
JButton(Integer.toString(i*4+j+1));
                }
                else{
                    this.tiles[i*4+j] = new JButton("KOSONG");
                    this.tiles[i*4+j].setVisible(false);
                }
                this.tiles[i*4+j].setFont(new Font("Roboto", 0, 25));
                this.tiles[i*4+j].setBackground(Color.lightGray);
                this.tiles[i*4+j].setFocusable(false);
                outConstraint.gridy = i;
                outConstraint.gridx = j;
                outConstraint.weightx = 1;
                outConstraint.weighty = 1;
                outConstraint.insets = new Insets(5, 5, 5, 5);
                outConstraint.fill = GridBagConstraints.BOTH;
                this.panelOut.add(this.tiles[i*4+j], outConstraint);
            }
        }

        // panelNav and its component
        this.panelNav = new JPanel();
        this.panelNav.setBackground(Color.decode("#fafafa"));
        this.panelNav.setLayout(new GridBagLayout());
        this.panelNav.setPreferredSize(new Dimension(300, 700));
    }

```

```

this.panelNav.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,
10));

// component initialization
// button
this.solveButton = new JButton("Solve");
createButton(solveButton);

this.randomButton = new JButton("Randomize");
createButton(randomButton);

this.importButton = new JButton("Import");
createButton(importButton);

this.createButton = new JButton("Create");
createButton(createButton);

this.analysisButton = new JButton("See Analysis");
createButton(analysisButton);
this.analysisButton.setPreferredSize(new Dimension(160, 35));
this.analysisButton.setVisible(false);

this.resetButton = new JButton("Reset");
createButton(resetButton);
this.resetButton.setVisible(false);

// text
this.pathIn = new JTextField();
this.pathIn.setPreferredSize(new Dimension(200, 35));
this.pathIn.setFont(new Font("Gill Sans", Font.PLAIN, 16));

this.inputText = new JTextArea();
this.inputText.setBorder(BorderFactory.createCompoundBorder(BorderFa
ctory.createLineBorder(Color.BLACK),
BorderFactory.createEmptyBorder(1,1,1,1)));
this.inputText.setPreferredSize(new Dimension(250, 300));
this.inputText.setFont(new Font("Gill Sans", Font.PLAIN, 30));

// label
this.createStatus = new JLabel();
this.createStatus.setForeground(Color.RED);
this.outStatus = new JLabel();

this.input = new JLabel("Input: ");
input.setFont(new Font("Gill Sans", Font.BOLD, 14));

this.output = new JLabel("Output: ");

```

```
output.setFont(new Font("Gill Sans", Font.BOLD, 14));

// Component placement

GridBagConstraints navConstraint = new GridBagConstraints();
navConstraint.insets = new Insets(5, 5, 5, 5);

navConstraint.gridx = 0;
navConstraint.gridy = 0;
navConstraint.gridwidth = 2;
navConstraint.anchor = GridBagConstraints.LINE_START;
this.panelNav.add(input, navConstraint);

navConstraint.gridx = 0;
navConstraint.gridy = 1;
navConstraint.gridwidth = 1;
this.panelNav.add(importButton, navConstraint);

navConstraint.gridx = 1;
navConstraint.gridy = 1;
navConstraint.gridwidth = 2;
navConstraint.anchor = GridBagConstraints.LINE_END;
this.panelNav.add(pathIn, navConstraint);

navConstraint.gridx = 0;
navConstraint.gridy = 2;
navConstraint.gridwidth = 3;
navConstraint.fill = GridBagConstraints.HORIZONTAL;
this.panelNav.add(inputText, navConstraint);

navConstraint.gridx = 0;
navConstraint.gridy = 3;
navConstraint.gridwidth = 1;
navConstraint.fill = GridBagConstraints.NONE;
navConstraint.anchor = GridBagConstraints.LINE_START;
this.panelNav.add(createStatus, navConstraint);

navConstraint.gridx = 2;
navConstraint.gridy = 3;
navConstraint.anchor = GridBagConstraints.LINE_END;
this.panelNav.add(createButton, navConstraint);

navConstraint.gridx = 0;
navConstraint.gridy = 4;
navConstraint.anchor = GridBagConstraints.LINE_START;
this.panelNav.add(randomButton, navConstraint);
```

```

navConstraint.gridx = 2;
navConstraint.gridy = 4;
navConstraint.anchor = GridBagConstraints.LINE_END;
this.panelNav.add(solveButton, navConstraint);

navConstraint.gridx = 1;
navConstraint.gridy = 4;
navConstraint.anchor = GridBagConstraints.CENTER;
navConstraint.gridheight = 1;
this.panelNav.add(resetButton, navConstraint);

navConstraint.gridx = 0;
navConstraint.gridy = 5;
navConstraint.fill = GridBagConstraints.HORIZONTAL;
navConstraint.gridwidth = 3;
this.panelNav.add(output, navConstraint);

navConstraint.gridy = 6;
navConstraint.fill = GridBagConstraints.HORIZONTAL;
navConstraint.gridheight = 2;
this.panelNav.add(outStatus, navConstraint);

navConstraint.gridy = 8;
navConstraint.gridwidth = 2;
this.panelNav.add(analysisButton, navConstraint);

// Panel Placement
GridBagConstraints containerConstraint = new GridBagConstraints();
containerConstraint.gridx = 1;
containerConstraint.gridy = 0;
containerConstraint.weightx = 7;
containerConstraint.weighty = 1;
containerConstraint.fill = GridBagConstraints.BOTH;
this.add(panelOut, containerConstraint);

containerConstraint.gridx = 0;
containerConstraint.weightx = 3;
this.add(panelNav, containerConstraint);

this.setSize(1000, 700);
this.setTitle("15 Puzzle Solver");
this.setDefaultCloseOperation(EXIT_ON_CLOSE);

// Button Action
// random button: to generate a matrix with random manner
this.randomButton.addActionListener(new ActionListener() {

```



```

        @Override
        public void actionPerformed(ActionEvent e) {
            reset();
            tree = new Graph(true);
            MainWindow.this.drawBoard(tree.getRoot());
        }
    });

    // solve button: to solve current matrix
    this.solveButton.addActionListener(new ActionListener() {
        public int idx;
        @Override
        public void actionPerformed(ActionEvent e) {
            idx = 0;
            reset();
            currentTree();
            if(tree.getSolutionPath().size()==0){
                tree.process();
            }
            new Timer(500, new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    if(idx >= tree.getSolutionPath().size()){
                        outStatus.setText(tree.status);
                        analysisButton.setVisible(true);
                        resetButton.setVisible(true);
                        ((Timer)e.getSource()).stop();
                        return;
                    }
                    MainWindow.this.drawBoard(tree.getSolutionPath().get
(idx));
                    idx++;
                }
            }).start();
        }
    });

    // import button: to read input from file text
    this.importButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            inputWindows = new JFileChooser();
            inputWindows.setFileFilter(new FileNameExtensionFilter("Text
File", "txt", "text"));
            inputWindows.setCurrentDirectory(new File("."));
            inputWindows.showOpenDialog(null);
            if(inputWindows.getSelectedFile()!=null) {

```

```

        pathIn.setText(inputWindows.getSelectedFile().getAbsolutePath());

        File file = new File(pathIn.getText());
        inputText.setText("");
        try{
            Scanner inp = new Scanner(file);
            while(inp.hasNextLine()){
                inputText.append(inp.nextLine() + "\n");
            }
        } catch (FileNotFoundException fileNotFoundException) {
            createStatus.setText("File not found!");
            fileNotFoundException.printStackTrace();
        }
    }
});

// create button: to read input from current text box to actual
boards
this.createButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        reset();
        try{
            parseInput();
            drawBoard(tree.getRoot());
        } catch (BadLocationException | IndexOutOfBoundsException
e1){
            createStatus.setText("Input format is invalid!");
        }
    }
});

// analysis button: to open up analysis window
this.analysisButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        AnalysisWindow analysisWindow = new AnalysisWindow(tree);
        analysisWindow.setVisible(true);
    }
});

// reset button: to reset solved state to its root
this.resetButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        reset();
        if(tree == null){

```

```

        tree = new Graph(false);
    }
    drawBoard(tree.getRoot());
}
});

}

public void drawBoard(Node n){
    // draw current GUI board with state n
    for(int i=0; i<4; i++){
        for(int j=0; j<4; j++){
            if(n.getBoard(i, j)==16){
                this.tiles[i*4+j].setText("KOSONG");
                this.tiles[i*4+j].setVisible(false);
            }
            else{
                this.tiles[i*4+j].setText(String.valueOf(n.getBoard(i,
j)));
                this.tiles[i*4+j].setVisible(true);
            }
        }
    }
    this.panelOut.repaint();
    this.panelOut.revalidate();
}

public void currentTree(){
    // converting GUI board into node object
    String[][] raw = new String[4][4];
    for(int i=0; i<4; i++){
        for(int j=0; j<4; j++){
            raw[i][j] = tiles[i*4+j].getText();
            if(raw[i][j]=="KOSONG"){
                raw[i][j] = "16";
            }
        }
    }
    if(tree!=null&&Arrays.deepToString(tree.getRoot().getBoard()).equals
(Arrays.deepToString(raw))){
        return;
    }
    tree = new Graph(raw);
}

public void parseInput() throws BadLocationException {
    // reading input and place it into a text box to soon be processed
    int lineCount = inputText.getLineCount();

```

```

        if(lineCount<4){
            throw new ArrayIndexOutOfBoundsException();
        }
        else{
            String[][] raw_inp = new String[4][4];
            String[] lines = new String[4];
            for(int i=0; i<4; i++){
                int offset = inputText.getLineStartOffset(i);
                int offset2 = inputText.getLineEndOffset(i);
                lines[i] = inputText.getText().substring(offset, offset2);
                while(lines[i].length() > 1){
                    char c = lines[i].charAt(lines[i].length() - 1);
                    if (!(c == ' ' || c == '\n')) break;
                    lines[i] = lines[i].substring(0, lines[i].length()-1);
                }
                if(lines[i].split(" ").length!=4){
                    throw new ArrayIndexOutOfBoundsException();
                }
                else{
                    String[] raw_line = lines[i].split(" ");
                    raw_inp[i] = raw_line;
                }
            }
            this.tree = new Graph(raw_inp);
            int[][] raw_boards = tree.getRoot().getBoard();
            Graph dummy_tree = new Graph(false);
            int[][] dummy_boards = dummy_tree.getRoot().getBoard();
            java.util.List<Integer> a = new ArrayList<>();
            java.util.List<Integer> b = new ArrayList<>();
            for(int i=0; i<4; i++){
                for(int j=0; j<4; j++){
                    a.add(raw_boards[i][j]);
                    b.add(dummy_boards[i][j]);
                }
            }
            Collections.sort(a);
            if(!Objects.equals(a.toString(), b.toString())){
                throw new ArrayIndexOutOfBoundsException();
            }
        }
    }

    public void reset(){
        // revert all component that is just temporary
        analysisButton.setVisible(false);
        resetButton.setVisible(false);
        outStatus.setText("");
    }

```

```

        createStatus.setText("");
    }
}

```

## 6.6 Kelas AnalysisWindow

*Frame* khusus yang menampilkan analisis heuristik dari status awal.

```

package gui;

import tree.Graph;

import javax.swing.*;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;

public class AnalysisWindow extends JFrame {
    protected JPanel tablePanel, rootPanel, tablePane;
    protected JButton exportButton;
    protected JFileChooser exportWindows;
    protected JLabel[] tiles;
    protected JLabel root, table, status;
    protected JTable analysisTable;
    protected Graph g;

    private void createButton(JButton button){
        button.setPreferredSize(new Dimension(100, 35));
        button.setFocusable(false);
        button.setBackground(Color.decode("#4DB4D7"));
        button.setForeground(Color.WHITE);
        button.setFont(new Font("Gill Sans", Font.PLAIN, 12));
    }

    public AnalysisWindow(Graph g){
        // it is a windows showing heuristics analysis after the tree has
        been processed
        this.tiles = new JLabel[16];
        this.setResizable(false);
        this.setLayout(new GridBagLayout());
    }
}

```

```

this.g = g;

// root panel and its component
this.rootPanel = new JPanel();
this.rootPanel.setBackground(Color.decode("#fafafa"));
this.rootPanel.setLayout(new GridBagLayout());
this.rootPanel.setPreferredSize(new Dimension(350, 390));
this.rootPanel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5,
5));

this.root = new JLabel("Initial Position: ");
this.root.setPreferredSize(new Dimension(350, 35));

GridBagConstraints rootConstraint = new GridBagConstraints();
rootConstraint.insets = new Insets(5, 5, 5, 5);

rootConstraint.gridx = 0;
rootConstraint.gridy = 0;
rootConstraint.gridwidth = 4;
rootConstraint.anchor = GridBagConstraints.LINE_START;
this.rootPanel.add(root, rootConstraint);
rootConstraint.gridwidth = 1;

for(int i=0; i<4; i++){
    for(int j=0; j<4; j++){
        this.tiles[i*4+j] = new JLabel();
        this.tiles[i*4+j].setOpaque(true);
        if(g.getRoot().getBoard(i, j)==16){
            this.tiles[i*4+j].setText("KOSONG");
            this.tiles[i*4+j].setVisible(false);
        }
        else{
            this.tiles[i*4+j].setText(String.valueOf(g.getRoot().get
Board(i, j)));
            this.tiles[i*4+j].setVisible(true);
        }
        this.tiles[i*4+j].setFont(new Font("Roboto", 0, 25));
        this.tiles[i*4+j].setBackground(Color.lightGray);
        this.tiles[i*4+j].setHorizontalAlignment(SwingConstants.CENT
ER);
        this.tiles[i*4+j].setVerticalAlignment(SwingConstants.CENTER
);

        this.tiles[i*4+j].setFocusable(false);
        rootConstraint.anchor = GridBagConstraints.CENTER;
        rootConstraint.gridy = i+1;
        rootConstraint.gridx = j;
        rootConstraint.weightx = 1;
        rootConstraint.weighty = 1;

```

```

        rootConstraint.fill = GridBagConstraints.BOTH;
        this.rootPanel.add(this.tiles[i*4+j], rootConstraint);
    }
}

// table panel and its component
this.tablePanel = new JPanel();
this.tablePanel.setBackground(Color.decode("#fafafa"));
this.tablePanel.setLayout(new GridBagLayout());
this.tablePanel.setPreferredSize(new Dimension(200, 600));

this.table = new JLabel("Reachable Analysis: ");
this.table.setPreferredSize(new Dimension(150, 35));

this.status = new JLabel(g.status);
this.status.setPreferredSize(new Dimension(150, 35));

this.exportButton = new JButton("Export");
createButton(exportButton);

// tablepane and its component
this.tablePane = new JPanel();
this.tablePane.setLayout(new BorderLayout());
this.tablePane.setPreferredSize(new Dimension(200, 320));

String[] column = {"i", "KURANGI[i]"};
this.analysisTable = new JTable(g.getRowContent(), column);

this.tablePane.add(analysisTable, BorderLayout.CENTER);
this.tablePane.add(analysisTable.getTableHeader(),
BorderLayout.NORTH);

GridBagConstraints tableConstraint = new GridBagConstraints();

tableConstraint.gridx = 0;
tableConstraint.gridy = 0;
tableConstraint.fill = GridBagConstraints.BOTH;

this.tablePanel.add(table, tableConstraint);

tableConstraint.gridy = 1;
tableConstraint.ipady = 20;
this.tablePanel.add(tablePane, tableConstraint);

tableConstraint.gridy = 2;
tableConstraint.ipady = 30;

```

```

this.tablePanel.add(status, tableConstraint);

tableConstraint.gridy = 4;
tableConstraint.weighty = 0.5; //request any extra vertical space
tableConstraint.fill = GridBagConstraints.HORIZONTAL;
tableConstraint.ipady = 0; //reset to default
tableConstraint.anchor = GridBagConstraints.CENTER;
tableConstraint.insets = new Insets(10,0,0,0);
this.tablePanel.add(exportButton, tableConstraint);


// panel placement
GridBagConstraints containerConstraint = new GridBagConstraints();
containerConstraint.gridx = 0;
containerConstraint.gridy = 0;
containerConstraint.weightx = 7;
containerConstraint.weighty = 1;
containerConstraint.fill = GridBagConstraints.BOTH;
this.add(rootPanel, containerConstraint);

containerConstraint.gridx = 1;
containerConstraint.weightx = 3;
this.add(tablePanel, containerConstraint);

this.setSize(800, 600);
this.setTitle("Analysis");
this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);


// button action

// export button: to write output into text file
this.exportButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        exportWindows = new JFileChooser();
        exportWindows.setApproveButtonText("Export");
        exportWindows.setCurrentDirectory(new File("."));
        exportWindows.showOpenDialog(null);
        if(exportWindows.getSelectedFile()!=null) {
            String path =
exportWindows.getSelectedFile().getAbsolutePath();
            exportOutput(path);
        }
    }
});
}

```



```

public void exportOutput(String path){
    // writing output
    try {
        FileWriter fstream = new FileWriter(path);
        BufferedWriter out = new BufferedWriter(fstream);
        out.write("INITIAL POSITION:");
        out.newLine();
        for(int i=0; i<4; i++){
            for(int j=0; j<4; j++){
                out.write(g.getRoot().getBoard(i, j) + " ");
            }
            out.newLine();
        }
        out.newLine();

        out.write("KURANGI TABLE: ");
        out.newLine();
        for(int i=1; i<=16; i++){
            out.write("KURANGI("+i+") = " + g.getKURANGI().get(i));
            out.newLine();
        }

        out.write("SUM = " + g.getSumKURANGI());
        out.newLine();
        out.write("SUM + X = " + g.getReachableCost());
        out.newLine();
        out.newLine();

        out.write("SOLUTION PATH:");
        out.newLine();
        for(int i=0; i<g.getSolutionPath().size(); i++){
            out.write("MOVE: "+i);
            out.newLine();
            out.write("DIRECTION:
"+g.getSolutionPath().get(i).direction);
            out.newLine();
            for(int j=0; j<4; j++){
                for(int k=0; k<4; k++){
                    out.write(g.getSolutionPath().get(i).getBoard(j, k)
+" ");
                }
                out.newLine();
            }
            out.newLine();
            out.newLine();
        }
        if(g.getSolutionPath().size()==0){

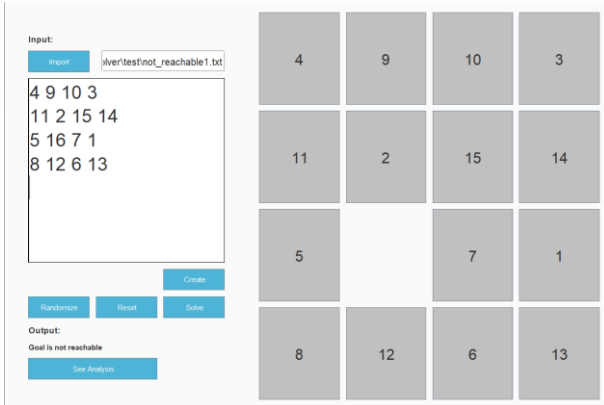
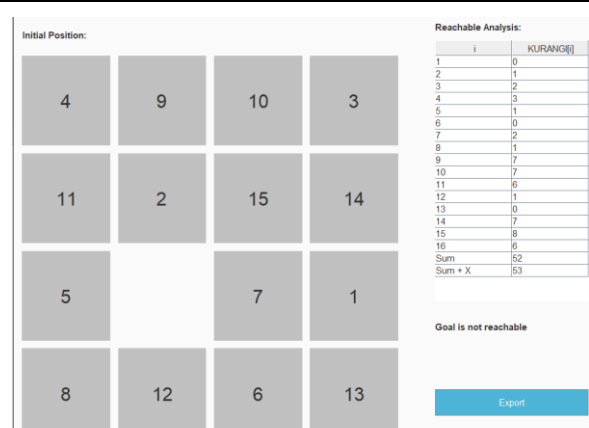
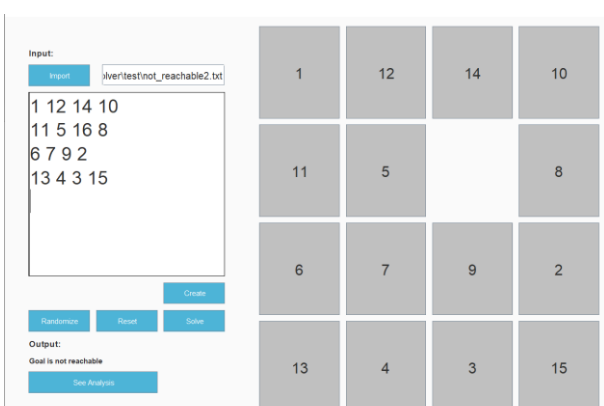
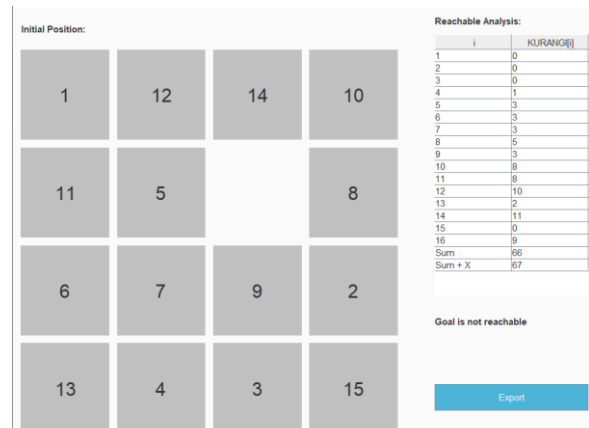
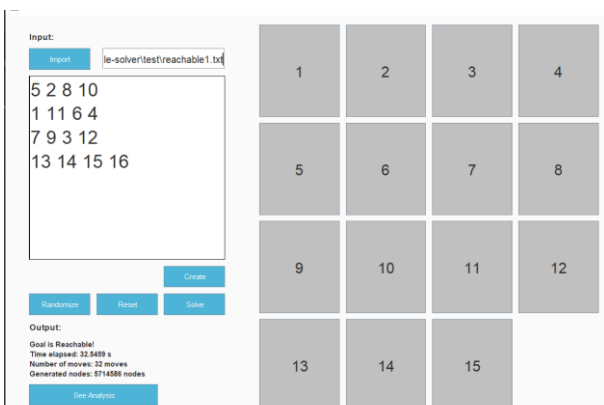
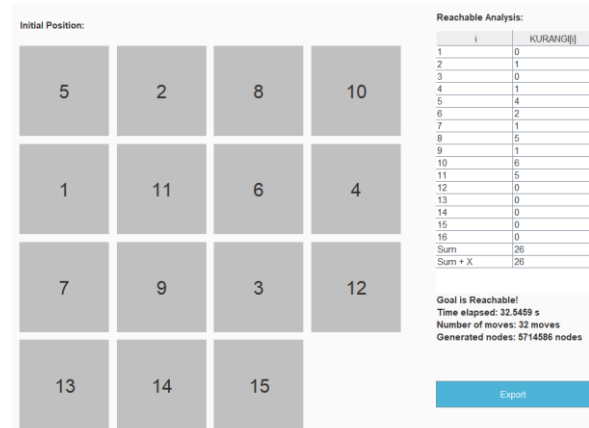
```

```
        out.write("No Solution Path Exist");
        out.newLine();
    }
    out.newLine();

    out.write("STATUS: ");
    out.newLine();
    String s = g.status;
    s = s.replaceAll("<html>", "");
    s = s.replaceAll("</html>", "");
    s = s.replaceAll("<br/>", "\n");
    out.write(s);
    out.close();
} catch (IOException ex) {
    ex.printStackTrace();
}
}
```

## BAB 5

### INSTANSIASI KASUS UJI

No.	Input	Output
1.	<p>Not Reachable Goal:</p> 	 <p>Detail solusi: <a href="https://github.com/firizky29/15-puzzle-solver/blob/main/test/not_reachable1_out.txt">https://github.com/firizky29/15-puzzle-solver/blob/main/test/not_reachable1_out.txt</a></p>
2.	<p>Not Reachable Goal:</p> 	 <p>Detail solusi: <a href="https://github.com/firizky29/15-puzzle-solver/blob/main/test/not_reachable2_out.txt">https://github.com/firizky29/15-puzzle-solver/blob/main/test/not_reachable2_out.txt</a></p>
3.	<p>Reachable Goal (32 Moves):</p> 	

		<p>Detail solusi: <a href="https://github.com/firizky29/15-puzzle-solver/blob/main/test/reachable1_out.txt">https://github.com/firizky29/15-puzzle-solver/blob/main/test/reachable1_out.txt</a></p>
4.	<p>Reachable Goal (29 Moves):</p> 	 <p>Detail solusi: <a href="https://github.com/firizky29/15-puzzle-solver/blob/main/test/reachable2_out.txt">https://github.com/firizky29/15-puzzle-solver/blob/main/test/reachable2_out.txt</a></p>
5.	<p>Reachable Goal (24 Moves):</p> 	 <p>Detail solusi: <a href="https://github.com/firizky29/15-puzzle-solver/blob/main/test/reachable3_out.txt">https://github.com/firizky29/15-puzzle-solver/blob/main/test/reachable3_out.txt</a></p>

## **BAB 6**

### **DOKUMENTASI**

Link Github: <https://github.com/firizky29/15-puzzle-solver>.