

Laporan Tugas Kecil 3 IF2211 Strategi Algoritma

15-Puzzle

Bryan Amirul Husna / 13520146 / K02

A. Cara kerja Program *Branch and Bound* untuk Pemecahan 15-Puzzle

Algoritma *Branch and Bound* mencari solusi optimum dengan mengevaluasi dahulu simpul yang memiliki total bobot minimum. Total bobot dihitung sebagai jumlah biaya akar ke simpul saat ini ditambah *estimasi* biaya dari simpul ini ke solusi. Berbeda dengan *greedy*, branch and bound perlu untuk menyimpan state yang tidak optimum lokal, karena bisa jadi simpul itu memberikan solusi optimum global nantinya. Secara umum, langkah-langkah algoritma branch and bound adalah sebagai berikut.

1. Evaluasi simpul pertama, jika sudah solusi berhenti, jika tidak lanjutkan ke langkah 2
2. Bangkitkan seluruh simpul anaknya yang tidak melanggar batasan (*constraint*), hitung biaya total = biaya saat ini + estimasi biaya ke solusi
3. Pilih dari keseluruhan simpul yang telah yang memiliki biaya total terendah. Periksa apakah simpul ini sudah solusi, jika ya berhenti (atau jika dicari lebih dari satu solusi pangkas simpul pada node yang bobotnya lebih besar daripada simpul terbaik ini kemudian ulangi langkah 2), jika tidak ulangi langkah 2 dengan simpul terpilih ini sebagai simpul yang akan membangkitkan anak

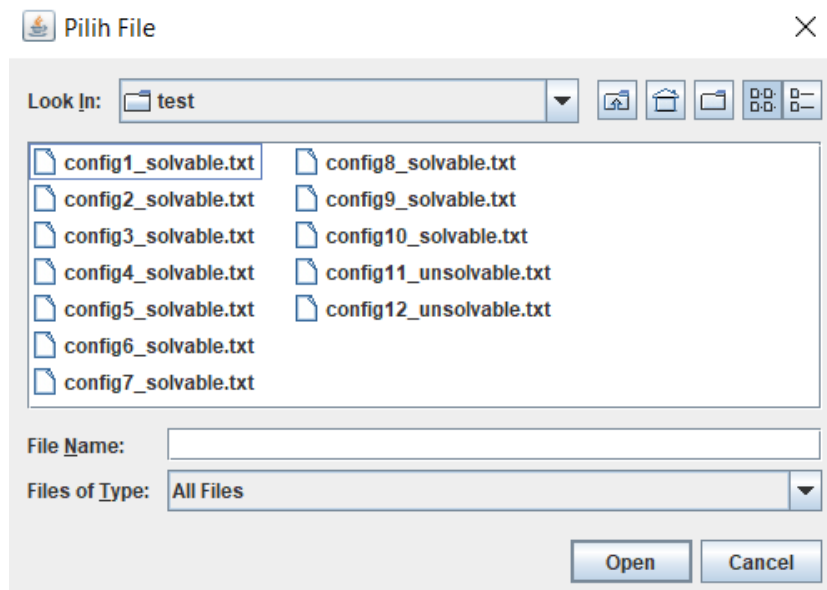
Pada pemecahan 15-puzzle, terdapat dua metode yang umum digunakan untuk penghitungan estimasi biaya, yaitu ubin tak kosong berposisi tidak tepat dan jarak manhattan. Posisi-tidak-tepat menghitung banyak ubin tak kosong yang tidak berada pada tempatnya ($i \neq \text{posisi}(i)$), sedangkan metode kedua menghitung jarak manhattan dari i ke posisi(i) sebagai estimasi. Dalam program ini, keduanya diimplementasikan dan pengguna dapat memilih metode yang ingin digunakan. Dalam hal struktur data, digunakan PriorityQueue yang menyimpan simpul-simpul pencarian agar pemilihan simpul berbobot minimum lebih mudah.

B. Screenshot Input-Output

```
Selamat datang di 15-Puzzle Solver
Dibuat oleh: Bryan Amirul Husna / 13520146

Asal Puzzle:
1. Randomly generated
2. File
Pilihan: 2
```

Gambar B.1 Pesan pembuka dan pilihan asal susunan puzzle



Gambar B.2 Jika memilih masukan dari file, akan muncul GUI File Chooser agar tidak repot menulis nama file dan menentukan path program relatif terhadap mana

Metode:

1. B&B posisi tidak tepat
 2. B&B jarak manhattan
 3. Cara kreasi sendiri, solusi tidak selalu optimal tetapi perhitungan selalu cepat
- Pilihan: 1

Gambar B.3 Pemilihan metode heuristik

Board Initial:

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

```
Kurang(1)= 0
Kurang(2)= 0
Kurang(3)= 1
Kurang(4)= 1
Kurang(5)= 0
Kurang(6)= 0
Kurang(7)= 1
Kurang(8)= 0
Kurang(9)= 0
Kurang(10)= 0
Kurang(11)= 3
Kurang(12)= 6
Kurang(13)= 0
Kurang(14)= 4
Kurang(15)= 11
Kurang(16)= 10
X + Sigma Kurang(i) = 37
```

Karena $X + \text{Sigma Kurang}(i)$ ganjil, puzzle tidak dapat diselesaikan

Gambar B.4 Contoh jika puzzle tidak dapat diselesaikan

```

X + Sigma Kurang(i) = 26

Sedang mencari solusi...
Solusi:
UP
 1  8  3  4
   5  6 10
 9  2  7 11
13 14 15 12

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

.....
.....

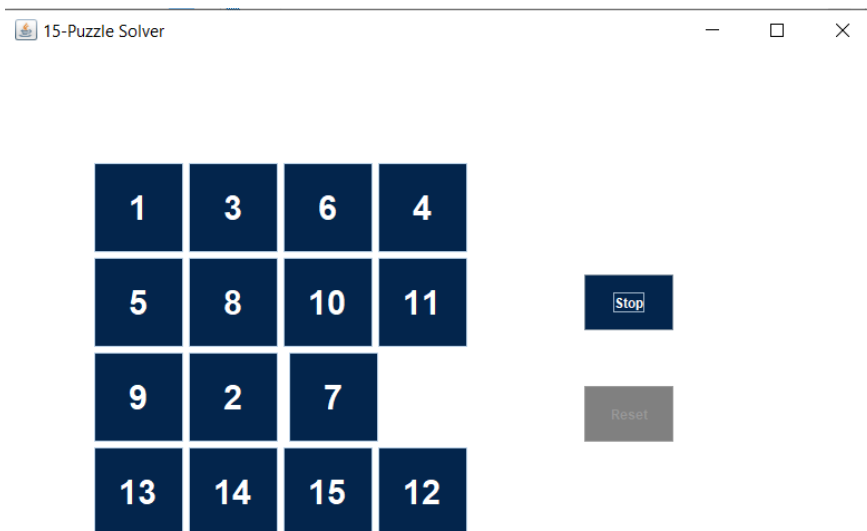
DOWN
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15

Jumlah langkah solusi: 22
Jumlah simpul dibangkitkan: 478246
Jumlah simpul diperiksa: 154621
Waktu eksekusi: 1606 ms

GUI animasi akan segera muncul pada jendela baru...

```

Gambar B.5 Jika ada solusi, memulai pencarian dan menampilkan solusinya



Gambar B.6 Tampilan GUI animasi solusi 15-puzzle

C. Checklist

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	√	

D. Source Code dalam bahasa Java

- **Solver.java**, berisi interface untuk kelas yang dapat menyelesaikan permasalahan 15-puzzle

```

7 // Interface untuk menyatukan Solver dan UltraSolver
8 // keduanya berbeda cukup jauh, tetapi memiliki interface yang sama
9 public interface Solver {
10     public StatedGameBoard getGameBoardInitial();
11     public void startSolving();
12     public void displaySolution();
13     public List<Direction> getSolutionSteps();
14 }
15

```

- **BnBSolver.java**, berisi algoritma utama pencarian algoritma pencarian solusi dengan Branch and Bound

```

46 @Override
47 public void startSolving() {
48     this.simpulDibangkitkan = 0;
49     this.simpulDiperiksa = 0;
50
51     StatedGameBoard sgb;
52     if(this.gameBoardInitial instanceof ManhattanBoard) {
53         sgb = new ManhattanBoard((ManhattanBoard)this.gameBoardInitial);
54     } else {
55         sgb = new OutPositionBoard((OutPositionBoard)this.gameBoardInitial);
56     }
57     stateQueue.add(sgb);
58
59     this.simpulDiperiksa = 0;
60     while(!stateQueue.isEmpty() && !isFound) {
61         this.simpulDiperiksa++;
62         sgb = stateQueue.poll();
63         if(sgb.isSolution()) {
64             this.isFound = true;
65             this.solutionSteps = sgb.getSteps();
66         } else {
67             if(sgb.canMoveLeft()) {
68                 StatedGameBoard temp;
69                 if(this.gameBoardInitial instanceof ManhattanBoard) {
70                     temp = new ManhattanBoard((ManhattanBoard)sgb);
71                 } else {
72                     temp = new OutPositionBoard((OutPositionBoard)sgb);
73                 }
74                 temp.moveLeft();
75                 stateQueue.add(temp);
76                 this.simpulDibangkitkan++;
77             }
78         }
79     }
80 }

```

```

78         if(sgb.canMoveUp()) {
79             StatedGameBoard temp;
80             if(this.gameBoardInitial instanceof ManhattanBoard) {
81                 temp = new ManhattanBoard((ManhattanBoard)sgb);
82             } else {
83                 temp = new OutPositionBoard((OutPositionBoard)sgb);
84             }
85             temp.moveUp();
86             stateQueue.add(temp);
87             this.simpulDibangkitkan++;
88         }
89         if(sgb.canMoveRight()) {
90             StatedGameBoard temp;
91             if(this.gameBoardInitial instanceof ManhattanBoard) {
92                 temp = new ManhattanBoard((ManhattanBoard)sgb);
93             } else {
94                 temp = new OutPositionBoard((OutPositionBoard)sgb);
95             }
96             temp.moveRight();
97             stateQueue.add(temp);
98             this.simpulDibangkitkan++;
99         }
100        if(sgb.canMoveDown()) {
101            StatedGameBoard temp;
102            if(this.gameBoardInitial instanceof ManhattanBoard) {
103                temp = new ManhattanBoard((ManhattanBoard)sgb);
104            } else {
105                temp = new OutPositionBoard((OutPositionBoard)sgb);
106            }
107            temp.moveDown();
108            stateQueue.add(temp);
109            this.simpulDibangkitkan++;
110        }
111    }
112 }
113 }

```

- **GameBoard.java**, berisi struktur data untuk merepresentasikan papan dan fungsi-fungsi untuk mengakses dan mengolahnya

```

11 public class GameBoard {
12     protected int[][] arr;
13     protected int emptyCellRow;
14     protected int emptyCellCol;
15
16     public GameBoard() {}
17
18     /* Constructor pembuatan GameBoard dari pembacaan file */
19     public GameBoard(String configPath) throws Exception {}
20
21     public GameBoard(GameBoard gb) {}
22
23     /* Pengecekan status awal */
24     public boolean isSolvable() {}
25
26     public int sigmaKurangAndX() {}
27
28     public int kurang(int i) {}
29
30     /* Mengecek apakah dari state sekarang bisa bergeser ke arah tertentu */
31     public boolean canMoveRight() {}
32     public boolean canMoveLeft() {}
33     public boolean canMoveDown() {}
34     public boolean canMoveUp() {}
35
36     /* Aksi move, asumsi bisa (sudah dicek dengan canMove<direction>) */
37     public void moveRight() {}
38     public void moveLeft() {}
39     public void moveDown() {}
40     public void moveUp() {}
41
42     public int getElement(int i, int j) {}
43     public int getElement(int pos) {}

```

```

149  /* isThe<Dir>Is(int num, int numadj)
150  * mencari apakah sebelah Dir dari num adalah numadj */
151  public boolean isTheRightIs(int num, int numadj) {}
179  public boolean isTheBelowIs(int num, int numadj) {}
207  public boolean isTheRightBelowIs(int num, int numadj) {}
235  public boolean isTheCoordinate(int num, int i, int j) {}
239  public boolean isThePos(int num, int pos) {}
243
244  /* Mencetak papan */
245  public void printBoard() {}
257
258  /* Fungsi-fungsi utilitas */
259  public int calculateManhattanDistance(int srcNum, int destPos) {}
272  public int findPos(int num) {}
282
283  public static int convertCoorToNumber(int i, int j) {}
286 }

```

- **StatedGameBoard.java**, berisi struktur data untuk merepresentasikan papan pada simpul pohon pencarian dengan biaya

```

7  // Kelas yang menyimpan papan dengan perhitungan biaya dan sejarah perpindahannya
8  // digunakan untuk node pada pohon di PriorityQueue
9  public class StatedGameBoard extends GameBoard implements Comparable<StatedGameBoard> {
10  public enum Direction {
11      LEFT, UP, RIGHT, DOWN
12  }
13
14  protected int currentCost; // cost dari root sampai node ini (terrealisasi)
15  protected int estimatedCost; // estimasi node ini sampai tujuan
16  public List<Direction> steps;
17
18  public StatedGameBoard() {
19      super();
20      this.currentCost = 0;
21      this.estimatedCost = 0;
22      this.steps = new ArrayList<>();
23  }
24
25  public StatedGameBoard(String configPath) throws Exception {
26      super(configPath);
27      this.currentCost = 0;
28      this.estimatedCost = 0;
29      this.steps = new ArrayList<>();
30  }
31
32  public StatedGameBoard(StatedGameBoard sgb) {
33      super(sgb);
34      this.currentCost = sgb.currentCost;
35      this.estimatedCost = sgb.estimatedCost;
36      this.steps = new ArrayList<>(sgb.steps);
37  }

```

```

39  /* Implementasi compareTo untuk PriorityQueue */
40  @Override
41  public int compareTo(StatedGameBoard o) {
42      if(this.currentCost + this.estimatedCost < o.currentCost + o.estimatedCost) {
43          return -1;
44      } else if(this.currentCost + this.estimatedCost > o.currentCost + o.estimatedCost) {
45          return 1;
46      } else {
47          return 0;
48      }
49  }
50  }
51  /* Update dan perhitungan harga, serta aksesornya */
52  public void updateEstimatedCost() {
53      this.estimatedCost = this.calculateEstimatedCost();
54  }
55
56  public int calculateEstimatedCost() {
57      return 0;
58  }
59
60  public int getTotalCost() {
61      return this.currentCost + this.estimatedCost;
62  }
63
64  public boolean isSolution() {
65      return this.estimatedCost == 0;
66  }
67
68  public List<Direction> getSteps() {
69      return this.steps;
70  }
71  }

```

- **OutPositionBoard.java**, turunan dari StatedGameBoard yang mengoverride calculateEstimatedCost() untuk perhitungan estimasi posisi yang tidak tepat

```

3  // Kelas yang perhitungannya dengan estimasi tile yang tidak berada pada tempatnya
4  public class OutPositionBoard extends StatedGameBoard {
5      public OutPositionBoard() {
6          super();
7      }
8      public OutPositionBoard(String configPath) throws Exception{
9          super(configPath);
10     }
11     public OutPositionBoard(OutPositionBoard opb) {
12         super(opb);
13     }
14
15     @Override
16     public int calculateEstimatedCost() {
17         int estimated = 0;
18         for(int i = 0; i < 4; i++) {
19             for(int j = 0; j < 4; j++) {
20                 if(this.arr[i][j] == 16) {
21                     continue;
22                 } else {
23                     estimated += arr[i][j] == (4*i + j + 1) ? 0 : 1;
24                 }
25             }
26         }
27         return estimated;
28     }
29 }
30

```

- **ManhattanBoard.java**, turunan dari StatedGameBoard yang mengoverride calculateEstimatedCost() untuk perhitungan estimasi jarak manhattan

```

3 // Perhitungan dengan estimasi jarak Manhattan
4 public class ManhattanBoard extends StatedGameBoard {
5     public ManhattanBoard() {
6         super();
7     }
8     public ManhattanBoard(String configPath) throws Exception {
9         super(configPath);
10    }
11    public ManhattanBoard(ManhattanBoard mb) {
12        super(mb);
13    }
14
15    @Override
16    public int calculateEstimatedCost() {
17        int estimated = 0;
18        for(int i = 0; i < 4; i++) {
19            for(int j = 0; j < 4; j++) {
20                if(this.arr[i][j] == 16) {
21                    continue;
22                } else {
23                    int inum = (arr[i][j]-1) / 4;
24                    int jnum = (arr[i][j]-1) % 4;
25                    estimated += Math.abs(inum - i) + Math.abs(jnum - j);
26                }
27            }
28        }
29        return estimated;
30    }
31 }
32

```

- **GraphicalBoard.java**, turunan dari StatedGameBoard yang memiliki fungsi-fungsi untuk menampilkan animasi langkah-langkah solusi puzzle

```

18 // Kelas untuk menampilkan animasi puzzle
19 // Dibuat dengan Java Swing
20 public class GraphicalBoard extends StatedGameBoard implements ActionListener {
21     private StatedGameBoard initial;
22     public JFrame jf;
23     private JButton startStop, resetButton;
24     private Cell[][] cells = new Cell[4][4];
25     private int k = 0;
26     private int i, j;
27     private int nextX, nextY;
28     private int deltaX = 5, deltaY = 5;
29     private boolean curElFinish = true;
30     private boolean started = false;
31
32     Timer tm = new Timer(10, this);
33
34     @Override
35     public void actionPerformed(ActionEvent e) {
36         if(curElFinish) {
37             Direction dir = this.steps.get(this.k);
38             if(dir == Direction.LEFT) {
39                 this.moveLeft();
40                 Cell cur = this.cells[i][j];
41                 deltaX = 5;
42                 deltaY = 0;
43                 nextX = cur.getBounds().x + 85;
44                 nextY = cur.getBounds().y;
45             } else if(dir == Direction.UP) {
46                 this.moveUp();
47                 Cell cur = this.cells[i][j];
48                 deltaX = 0;
49                 deltaY = 5;
50                 nextX = cur.getBounds().x;
51                 nextY = cur.getBounds().y + 85;

```


- **UltraHeuristicBoard.java**, turunan dari StatedGameBoard untuk penyelesaian dengan cara kreasi sendiri

```

7 public class UltraHeuristicBoard extends StatedGameBoard{
8     private boolean[] locked; // locked[i] berarti posisi i sudah benar sehingga tidak boleh dipindah
9     private NextTarget nextTarget;
10
11* public UltraHeuristicBoard() {}
19
20* public UltraHeuristicBoard(String configPath) throws Exception {}
28
29* public UltraHeuristicBoard(UltraHeuristicBoard uhb) {}
37
38* public void setTarget(NextTarget nt) {}
41
43* public void updateEstimatedCost() {}
47* public int calculateEstimatedCost() {}
80
81* public boolean isTargetFulfilled() {}
84
85* public void lockPosition(int pos) {}
88* public void unlockPosition(int pos) {}
91* public boolean isLocked(int pos) {}
94
95
96 /* Override canMove<dir> agar mengakomodasi lock */
98* public boolean canMoveRight() {}
103* public boolean canMoveDown() {}
108* public boolean canMoveLeft() {}
113* public boolean canMoveUp() {}
117 }

```

- **UltraSolver.java**, berisi penyelesaian dengan cara kreasi sendiri

```

13 public class UltraSolver implements Solver{
14     private UltraHeuristicBoard gameBoardInitial;
15     public boolean isFound = false;
16     Queue<NextTarget> nextTargetQueue;
17     private List<Direction> solutionSteps;
18     private int simpulDibangkitkan = 0;
19     private int simpulDiperiksa = 0;
20
21
22* public static void main(String[] args) {}
33
34* public UltraSolver() {}
39
40* public UltraSolver(String configPath) throws Exception {}
45
46* public UltraSolver(UltraHeuristicBoard gameBoard) {}
50
51* public void initTargetQueue() {}
101
103* public void startSolving() {}
142
143* public UltraHeuristicBoard solveNext(UltraHeuristicBoard uhb, NextTarget nextTarget) {}
193
195* public void displaySolution() {}
204
206* public UltraHeuristicBoard getGameBoardInitial() {}
209
211* public List<Direction> getSolutionSteps() {}
214 }

```

- **Main.java**, berisi program utama

```

5 import javax.swing.JDialog;
6 import javax.swing.JFileChooser;
7
8 import bryan.fifteenpuzzle.BnBSolver;
9 import bryan.fifteenpuzzle.GraphicalBoard;
10 import bryan.fifteenpuzzle.Solver;
11 import bryan.fifteenpuzzle.UltraSolver;
12
13 // Program Utama
14 // Menerima masukan pengguna, kemudian memanggil berbagai kelas Solver, lalu menampilkannya di kelas GUI
15 public class Main {
16     public static void main(String[] args) {
17         int src, method; // menyimpan pilihan sumber dan metode
18         String configPath = null;
19         Scanner scanner = new Scanner(System.in);
20
21         System.out.println("Selamat datang di 15-Puzzle Solver");
22         System.out.println("Dibuat oleh: Bryan Amirul Husna / 13520146");
23         System.out.println();
24
25         // Pemilihan sumber puzzle
26         System.out.println("Asal Puzzle:");
27         System.out.println("1. Randomly generated");
28         System.out.println("2. File");
29         System.out.print("Pilihan: ");
30         src = scanner.nextInt();
31         System.out.println();
32
33         if(src < 1 || src > 2) {
34             System.out.println("Masukkan tidak valid!");
35             scanner.close();
36             return;
37         }
38
39         .....
40
41         .....
42
43         System.out.println("Board Initial:");
44         solver.getGameBoardInitial().printBoard();
45         System.out.println();
46         for(int i = 1; i <= 16; i++) {
47             System.out.printf("Kurang(%d)= %d\n", i, solver.getGameBoardInitial().kurang(i));
48         }
49         System.out.println("X + Sigma Kurang(i) = " + solver.getGameBoardInitial().sigmaKurangAndX());
50         System.out.println();
51
52         if(solver.getGameBoardInitial().isSolvable()) {
53             System.out.println("Sedang mencari solusi...");
54             long startTime = System.nanoTime();
55
56             solver.startSolving(); // memulai pemecahan
57
58             long duration = System.nanoTime() - startTime;
59             duration /= 1000000; // ubah ke milidetik
60
61             System.out.println("Solusi:");
62             solver.displaySolution();
63             System.out.printf("Waktu eksekusi: %d ms\n", duration);
64
65             // Menampilkan GUI, tunggu 2,5 detik dahulu
66             System.out.println("\nGUI animasi akan segera muncul pada jendela baru...");
67             Thread.sleep(2500);
68             GraphicalBoard gb = new GraphicalBoard(solver.getGameBoardInitial(), solver.getSolutionSteps());
69         } else {
70             System.out.println("Karena X + Sigma Kurang(i) ganjil, puzzle tidak dapat diselesaikan");
71         }
72     } catch(Exception e) {
73         e.printStackTrace();
74     }
75 }
76
77 }

```

E. Instansiasi Persoalan

Pada file konfigurasi yang dibaca program, ubin kosong ditulis sebagai angka 16. Nantinya pada pencetakan/visualisasi angka 16 ini direpresentasikan dengan tampilan ubin kosong. Pada subbab ini dituliskan dalam bentuk tampilan ubin kosong (bukan angka 16).

- config1_solvable.txt, dapat diselesaikan

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

```
Jumlah langkah solusi: 3
Jumlah simpul dibangkitkan: 11
Jumlah simpul diperiksa: 4
Waktu eksekusi: 2 ms
```

```
Jumlah langkah solusi: 3
Jumlah simpul dibangkitkan: 11
Jumlah simpul diperiksa: 4
Waktu eksekusi: 2 ms
```

Gambar E.1 Hasil eksekusi program untuk config 1, (kiri) metode posisi tidak tepat, (kanan) metode jarak manhattan

- config2_solvable.txt, dapat diselesaikan

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

```
Jumlah langkah solusi: 19
Jumlah simpul dibangkitkan: 14336
Jumlah simpul diperiksa: 4786
Waktu eksekusi: 64 ms
```

```
Jumlah langkah solusi: 19
Jumlah simpul dibangkitkan: 387
Jumlah simpul diperiksa: 125
Waktu eksekusi: 7 ms
```

Gambar E.2 Hasil eksekusi program untuk config 2, (kiri) metode posisi tidak tepat, (kanan) metode jarak manhattan

- config3_solvable.txt, dapat diselesaikan

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

```
Jumlah langkah solusi: 22
Jumlah simpul dibangkitkan: 212753
Jumlah simpul diperiksa: 70535
Waktu eksekusi: 636 ms
```

```
Jumlah langkah solusi: 22
Jumlah simpul dibangkitkan: 1111
Jumlah simpul diperiksa: 367
Waktu eksekusi: 12 ms
```

Gambar E.3 Hasil eksekusi program untuk config 3, (kiri) metode posisi tidak tepat, (kanan) metode jarak manhattan

- config4_unsolvable.txt, tidak memiliki solusi

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

Board Initial:

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

```
Kurang(1)= 0
Kurang(2)= 0
Kurang(3)= 1
Kurang(4)= 1
Kurang(5)= 0
Kurang(6)= 0
Kurang(7)= 1
Kurang(8)= 0
Kurang(9)= 0
Kurang(10)= 0
Kurang(11)= 3
Kurang(12)= 6
Kurang(13)= 0
Kurang(14)= 4
Kurang(15)= 11
Kurang(16)= 10
X + Sigma Kurang(i) = 37
```

Karena $X + \text{Sigma Kurang}(i)$ ganjil, puzzle tidak dapat diselesaikan

Gambar E.4 Hasil eksekusi untuk config 4, tidak memiliki solusi

- config5_unsolvable.txt, tidak memiliki solusi

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

Board Initial:

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

```
Kurang(1)= 0
Kurang(2)= 0
Kurang(3)= 0
Kurang(4)= 0
Kurang(5)= 1
Kurang(6)= 3
Kurang(7)= 5
Kurang(8)= 4
Kurang(9)= 4
Kurang(10)= 6
Kurang(11)= 8
Kurang(12)= 8
Kurang(13)= 11
Kurang(14)= 2
Kurang(15)= 5
Kurang(16)= 0
X + Sigma Kurang(i) = 57
```

Karena $X + \text{Sigma Kurang}(i)$ ganjil, puzzle tidak dapat diselesaikan

Gambar E.5 Hasil eksekusi untuk config 5, tidak memiliki solusi

F. Alamat Kode Sumber Program

<https://github.com/bryanahusna/Stima-Fifteen-Puzzle>