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, hitung biaya total = biaya saat ini + estimasi biaya
3. Pilih dari simpul-simpul anak itu yang memiliki biaya total terendah. Periksa apakah simpul ini sudah solusi, jika ya berhenti, jika tidak ulangi langkah 2 dengan simpul terpilih ini sebagai simpul yang akan membangkitkan anak

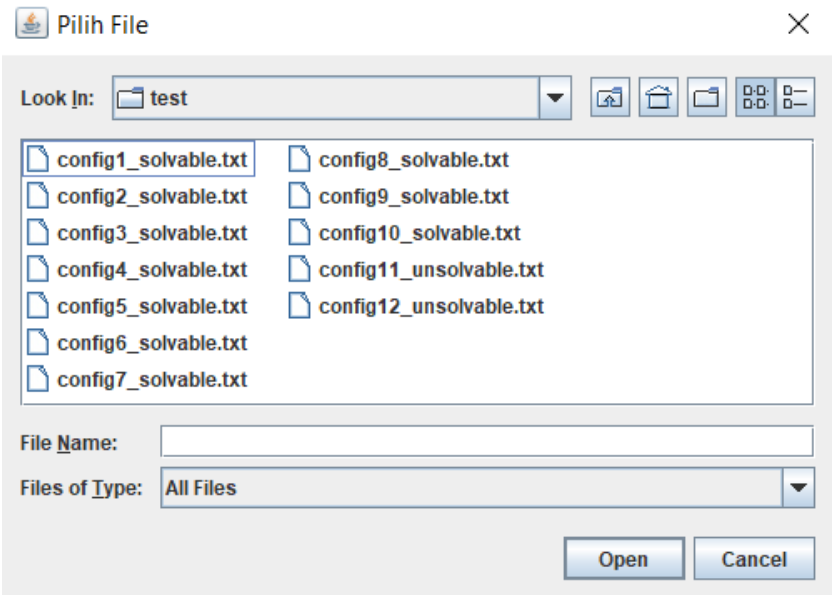
Pada pemecahan 15-puzzle, terdapat dua metode yang umum digunakan untuk penghitungan estimasi biaya, yaitu posisi tidak tepat dan jarak manhattan. Posisi-tidak-tepat menghitung banyak tile yang tidak berada pada tempatnya ($i \neq posisi(i)$), sedangkan metode kedua menghitung jarak manhattan dari i ke posisi(i) sebagai estimasi. Dalam program ini, keduanya diimplementasikan dan pengguna dapat memilih metode yang ingin digunakan.

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 + 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
```

.....

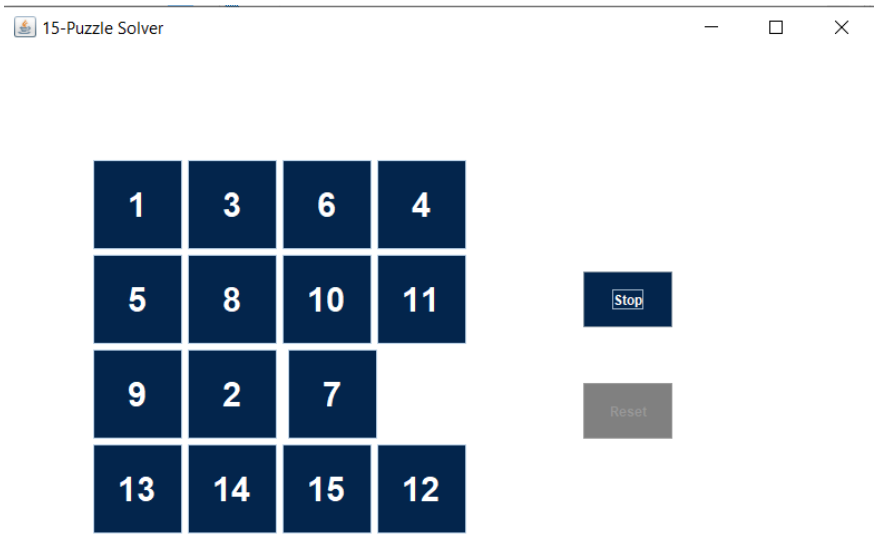
.....

```
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 *running*	√	
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

- **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                }
```

```
                if(sgb.canMoveUp()) {
                    StatedGameBoard temp;
                    if(this.gameBoardInitial instanceof ManhattanBoard) {
                        temp = new ManhattanBoard((ManhattanBoard)sgb);
                    } else {
                        temp = new OutPositionBoard((OutPositionBoard)sgb);
                    }
                    temp.moveUp();
                    stateQueue.add(temp);
                    this.simpulDibangkitkan++;
                }
                if(sgb.canMoveRight()) {
                    StatedGameBoard temp;
                    if(this.gameBoardInitial instanceof ManhattanBoard) {
                        temp = new ManhattanBoard((ManhattanBoard)sgb);
                    } else {
                        temp = new OutPositionBoard((OutPositionBoard)sgb);
                    }
                    temp.moveRight();
                    stateQueue.add(temp);
                    this.simpulDibangkitkan++;
                }
                if(sgb.canMoveDown()) {
                    StatedGameBoard temp;
                    if(this.gameBoardInitial instanceof ManhattanBoard) {
                        temp = new ManhattanBoard((ManhattanBoard)sgb);
                    } else {
                        temp = new OutPositionBoard((OutPositionBoard)sgb);
                    }
                    temp.moveDown();
                    stateQueue.add(temp);
                    this.simpulDibangkitkan++;
                }
            }
        }
    }
```

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

```
// Kelas yang menyimpan papan dengan perhitungan biaya dan sejarah perpindahannya
// digunakan untuk node pada pohon di PriorityQueue
public class StatedGameBoard extends GameBoard implements Comparable<StatedGameBoard> {
    public enum Direction {
        LEFT, UP, RIGHT, DOWN
    }

    protected int currentCost;  // cost dari root sampai node ini (terealisasi)
    protected int estimatedCost;    // estimasi node ini sampai tujuan
    public List<Direction> steps;

    public StatedGameBoard() {
        super();
        this.currentCost = 0;
        this.estimatedCost = 0;
        this.steps = new ArrayList<>();
    }

    public StatedGameBoard(String configPath) throws Exception {
        super(configPath);
        this.currentCost = 0;
        this.estimatedCost = 0;
        this.steps = new ArrayList<>();
    }

    public StatedGameBoard(StatedGameBoard sgb) {
        super(sgb);
        this.currentCost = sgb.currentCost;
        this.estimatedCost = sgb.estimatedCost;
        this.steps = new ArrayList<>(sgb.steps);
    }
```

```
    /* Implementasi compareTo untuk PriorityQueue */
    @Override
    public int compareTo(StatedGameBoard o) {
        if(this.currentCost + this.estimatedCost < o.currentCost + o.estimatedCost) {
            return -1;
        } else if(this.currentCost + this.estimatedCost > o.currentCost + o.estimatedCost) {
            return 1;
        } else {
            return 0;
        }
    }

    /* Update dan perhitungan harga, serta aksesornya */
    public void updateEstimatedCost() {
        this.estimatedCost = this.calculateEstimatedCost();
    }

    public int calculateEstimatedCost() {
        return 0;
    }

    public int getTotalCost() {
        return this.currentCost + this.estimatedCost;
    }

    public boolean isSolution() {
        return this.estimatedCost == 0;
    }

    public List<Direction> getSteps() {
        return this.steps;
    }
```

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

```
// Kelas yang perhitungannya dengan estimasi tile yang tidak berada pada tempatnya
public class OutPositionBoard extends StatedGameBoard {
    public OutPositionBoard() {
        super();
    }
    public OutPositionBoard(String configPath) throws Exception{
        super(configPath);
    }
    public OutPositionBoard(OutPositionBoard opb) {
        super(opb);
    }

    @Override
    public int calculateEstimatedCost() {
        int estimated = 0;
        for(int i = 0; i < 4; i++) {
            for(int j = 0; j < 4; j++) {
                if(this.arr[i][j] == 16) {
                    continue;
                } else {
                    estimated += arr[i][j] == (4*i + j + 1) ? 0 : 1;
                }
            }
        }
        return estimated;
    }
}
```

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

```java
// Perhitungan dengan estimasi jarak Manhattan
public class ManhattanBoard extends StatedGameBoard {
    public ManhattanBoard() {
        super();
    }
    public ManhattanBoard(String configPath) throws Exception{
        super(configPath);
    }
    public ManhattanBoard(ManhattanBoard mb) {
        super(mb);
    }

    @Override
    public int calculateEstimatedCost() {
        int estimated = 0;
        for(int i = 0; i < 4; i++) {
            for(int j = 0; j < 4; j++) {
                if(this.arr[i][j] == 16) {
                    continue;
                } else {
                    int inum = (arr[i][j]-1) / 4;
                    int jnum = (arr[i][j]-1) % 4;
                    estimated += Math.abs(inum - i) + Math.abs(jnum - j);
                }
            }
        }
        return estimated;
    }
}
```

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

```java
// Kelas untuk menampilkan animasi puzzle
// Dibuat dengan Java Swing
public class GraphicalBoard extends StatedGameBoard implements ActionListener {
    private StatedGameBoard initial;
    public JFrame jf;
    private JButton startStop, resetButton;
    private Cell[][] cells = new Cell[4][4];
    private int k = 0;
    private int i, j;
    private int nextX, nextY;
    private int deltaX = 5, deltaY = 5;
    private boolean curElFinish = true;
    private boolean started = false;

    Timer tm = new Timer(10, this);

    @Override
    public void actionPerformed(ActionEvent e) {
        if(curElFinish) {
            Direction dir = this.steps.get(this.k);
            if(dir == Direction.LEFT) {
                this.moveLeft();
                Cell cur = this.cells[i][j];
                deltaX = 5;
                deltaY = 0;
                nextX = cur.getBounds().x + 85;
                nextY = cur.getBounds().y;
            } else if(dir == Direction.UP) {
                this.moveUp();
                Cell cur = this.cells[i][j];
                deltaX = 0;
                deltaY = 5;
                nextX = cur.getBounds().x;
                nextY = cur.getBounds().y + 85;
```

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

```java
import javax.swing.JDialog;
import javax.swing.JFileChooser;

import bryan.fifteenpuzzle.BnBSolver;
import bryan.fifteenpuzzle.GraphicalBoard;
import bryan.fifteenpuzzle.Solver;
import bryan.fifteenpuzzle.UltraSolver;

// Program Utama
// Menerima masukan pengguna, kemudian memanggil berbagai kelas Solver, lalu menampilkannya di kelas GUI
public class Main {
    public static void main(String[] args) {
        int src, method;    // menyimpan pilihan sumber dan metode
        String configPath = null;
        Scanner scanner = new Scanner(System.in);

        System.out.println("Selamat datang di 15-Puzzle Solver");
        System.out.println("Dibuat oleh: Bryan Amirul Husna / 13520146");
        System.out.println();

        // Pemilihan sumber puzzle
        System.out.println("Asal Puzzle:");
        System.out.println("1. Randomly generated");
        System.out.println("2. File");
        System.out.print("Pilihan: ");
        src = scanner.nextInt();
        System.out.println();

        if(src < 1 || src > 2) {
            System.out.println("Masukkan tidak valid!");
            scanner.close();
            return;
        }

```

-

E. Instansiasi Persoalan

- config1_solvable.txt, dapat diselesaikan, tingkat kesulitan mudah

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

- config2_solvable.txt
- config3_solvable.txt
- config4_unsolvable.txt, tidak memiliki solusi

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

- config5_unsolvable.txt, tidak memiliki solusi

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

F. Alamat Drive Kode Program

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