Laporan Tugas Kecil 3

Penyelesaian Persoalan 15-Puzzle dengan Algoritma *Branch and Bound*

Mata Kuliah IF2211 - Strategi Algoritma

Oleh :

Nama : Roby Purnomo

NIM : 13520106

Kelas : 01

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021/2022

Daftar Isi

Bab I : Cara Kerja (Algoritma)

1. Pada penyelesaian 15 puzzle dengan Branch and Bound dibutuhkan sebuah priority queue yang digunakan untuk menampung puzzle yang diotak-atik dengan key untuk comparatornya merupakan cost dari puzzle state tersebut.
2. Sebelumnya, perlu dicek apakah puzzle dapat diselesaikan atau tidak, yakni dengan rumus :

   $$\sum_{i=1}^{16} KURANG(i) + X$$

   Jika bernilai genap maka puzzle dapat diselesaikan, dan jika bernilai ganjil maka puzzle tidak bisa diselesaikan sehingga langsung mengeluarkan message bahwa puzzle tidak dapat diselesaikan.
3. Jika puzzle dapat diselesaikan lanjut ke langkah selanjutnya, masukkan puzzle awal kedalam priority queue, lalu lakukan loop while hingga priority queue kosong atau ditemukan solusi, dengan program didalam whilenya adalah :
   a. Melakukan pop pada queue yang mengembalikan sebuah puzzle.
   b. Jika puzzle merupakan solusi maka hentikan loop dan mengembalikan semua informasi yang dibutuhkan.
   c. Jika tidak, lakukan command up, down, left, right untuk block kosong pada puzzle yang dapat ditempuh dengan syarat command sebelumnya bukan merupakan kebalikan dari command saat ini. Misal : sebelumnya telah dilakukan command up maka selanjutnya tidak perlu melakukan command down.
   d. Untuk setiap command yang dilakukan, cost pada puzzle yang baru dihitung yaitu dengan rumus :

      $c(P) = f(P) + g(P)$

      $f(P) = panjang\ lintasan\ dari\ simpul\ akar\ ke\ P$

      $g(P) = jumlah\ ubin\ yang\ tidak\ kosong\ yang\ tidak\ terdapat\ pada\ susunan\ akhir$

      Cost digunakan sebagai key pada priority queue, untuk cost yang lebih kecil maka akan didahulukan untuk dicek.

Bab II : Source Program (Bahasa C)

Dapat dilihat secara full pada alamat drive di bawah ini.

https://github.com/robypurnomo/tucil_3_stima

Modularity Program

```
├──doc
├──result
├──test
│   ├──1.txt
│   ├──2.txt
│   ├──3.txt
│   ├──4.txt
│   └──5.txt
└──main.c
    ├──BranchNBound.java
    ├──Puzzle.java
    ├──ReadFile.java
    └──Main.java
```

Main.java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args){
        int choice = -1;
        while(!(choice == 1 || choice == 2)) {
            System.out.println("Pilih salah satu : ");
            System.out.println("1. Puzzle automatis");
            System.out.println("2. Input File");
            System.out.println();
            System.out.print(">> ");
            Scanner input = new Scanner(System.in);
            choice = input.nextInt();
            Puzzle puzzle;
            if (choice == 1) {
                System.out.println();
                puzzle = new Puzzle().shufflePuzzle();
            } else {
                System.out.println();
                System.out.print("Path File >> ");
```

```
                input = new Scanner(System.in);
                String path = input.next();
                System.out.println();
                ReadFile readFile = new ReadFile();
                if (readFile.read(path) == null) {
                    break;
                } else {
                    puzzle = readFile.read(path);
                }
            }
            System.out.println("Puzzle awal :");
            System.out.println();
            puzzle.Show();
            BranchNBound solver = new BranchNBound(puzzle);
            solver.solve();
        }
    }
}
```

BranchNBound.java

```
import java.util.PriorityQueue;
import java.time.LocalTime;
import java.time.Duration;

public class BranchNBound {

    PriorityQueue<Puzzle> queue;
    int simpulHidup;

    BranchNBound(Puzzle puzzle) {
        this.queue = new PriorityQueue<>(new PuzzleComparator());
        this.queue.add(puzzle);
        this.simpulHidup = 1;
    }

    public void solve() {
        Puzzle awal = this.queue.poll();
        Puzzle puzzle = awal;
        int kurang = puzzle.Kurang(true);
        int X = puzzle.X();
        LocalTime start = LocalTime.now();
        if ((kurang+X)%2 == 0) {
            this.queue.add(puzzle);
            while (!this.queue.isEmpty())
            {
                puzzle = this.queue.poll();
```

```
                int cost = puzzle.calculateCost();
                System.out.println("Simpul hidup = " + this.simpulHidup +
"\ng(x)         = " + puzzle.calculateCost());
                System.out.print("\33[1A\33[2K");
                System.out.print("\33[1A\33[2K");
                if (cost == 0)
                {
                    break;
                }
                else {
                    Puzzle newPuzzle;
                    if (puzzle.Down() != null) {
                        newPuzzle = puzzle.Down();
                        if (puzzle.lastmove != 0) {
                            this.queue.add(newPuzzle);
                            this.simpulHidup++;
                        }
                    }
                    if (puzzle.Right() != null) {
                        newPuzzle = puzzle.Right();
                        if (puzzle.lastmove != 2) {
                            this.queue.add(newPuzzle);
                            this.simpulHidup++;
                        }
                    }
                    if (puzzle.Up() != null) {
                        newPuzzle = puzzle.Up();
                        if (puzzle.lastmove != 1) {
                            this.queue.add(newPuzzle);
                            this.simpulHidup++;
                        }
                    }
                    if (puzzle.Left() != null) {
                        newPuzzle = puzzle.Left();
                        if (puzzle.lastmove != 3) {
                            this.queue.add(newPuzzle);
                            this.simpulHidup++;
                        }
                    }
                }
            }
            LocalTime end = LocalTime.now();
            Duration duration = Duration.between(start, end);
            System.out.println();
            System.out.println("Result : ");
            System.out.println();
            for (int i=0; i<puzzle.movesHistory.size(); i++) {
                if (puzzle.movesHistory.get(i) == 0) {
```

```
                    awal = awal.Up();
                } else if (puzzle.movesHistory.get(i) == 1) {
                    awal = awal.Down();
                } else if (puzzle.movesHistory.get(i) == 2) {
                    awal = awal.Left();
                } else {
                    awal = awal.Right();
                }
                awal.Show();
                System.out.println();
            }
            puzzle.Show();
            System.out.println();
            System.out.println("Puzzle moves  = " + puzzle.moves);
            System.out.println("Kurang(i)+X  = " + (kurang + X));
            System.out.println("Simpul Hidup = " + this.simpulHidup);
            System.out.println("Duration     = " + duration.toMillis() + "
ms");
        } else {
            System.out.println();
            System.out.println("Result : ");
            System.out.println();
            System.out.println("Tidak ada solusi dengan nilai Kurang(i)+X =
" + (kurang + X));
        }
    }
}
```

Puzzle.java

```
import java.util.*;

public class Puzzle {
    private int mat[][];
    final int length = 4;
    public int lastmove;
    public List<Integer> movesHistory;
    public int moves;
    public int cost;

    Puzzle () {
        this.mat = new int[length][length];
        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                this.mat[i][j] = (length*i)+j+1;
            }
        }
```

```
        this.lastmove = -1;
        this.movesHistory = new ArrayList<>();
        this.moves = 0;
        this.cost = this.calculateCost() + this.moves;
    }

    Puzzle (int[][] mat) {
        this.mat = new int[length][length];
        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                this.mat[i][j] = mat[i][j];
            }
        }
        this.lastmove = -1;
        this.movesHistory = new ArrayList<>();
        this.moves = 0;
        this.cost = this.calculateCost() + this.moves;
    }

    public boolean isSolved() {
        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                if (this.mat[i][j] != (length*i)+j+1) {
                    return false;
                }
            }
        }
        return true;
    }

    public boolean isSame(Puzzle p) {
        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                if (this.mat[i][j] != p.mat[i][j]) {
                    return false;
                }
            }
        }
        return true;
    }

    public int[] getIndex(int x) {
        int[] idx = new int[2];
        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                if (this.mat[i][j] == x) {
                    idx[0] = i;
                    idx[1] = j;
```

```
                    break;
                }
            }
        }
        return idx;
    }

    public int Kurang(boolean print) {
        int count = 0;
        int countTemp;
        if (print) {
            System.out.println();
            System.out.println("Nilai dari Kurang(i) : ");
            System.out.println();
        }
        for (int x = 1; x <= 16; x++) {
            int idx[] = this.getIndex(x);
            int idxi = idx[0]; int idxj = idx[1];
            countTemp = 0;
            for (int i = 0; i < length; i++) {
                for (int j = 0; j < length; j++) {
                    if ((this.mat[i][j] < x) && ((length*i + j + 1) >
(length*idxi + idxj + 1))) {
                        countTemp++;
                    }
                }
            }
            count += countTemp;
            if (print) {
                if (x > 9) {
                    System.out.println(x + " : " + countTemp);
                } else {
                    System.out.println(x + "  : " + countTemp);
                }
            }
        }
        return count;
    }

    public int X() {
        int idx[] = this.getIndex(16);
        int idxi = idx[0]; int idxj = idx[1];
        if ((idxi%2 == 0 && idxj%2 == 1) || (idxi%2 == 1 && idxj%2 == 0)) {
            return 1;
        }
        return 0;
    }
```

```
    public Puzzle Up() {
        Puzzle newPuzzle = new Puzzle(this.mat);
        int idx[] = this.getIndex(16);
        int i = idx[0]; int j = idx[1];
        if (i > 0) {
            int temp = newPuzzle.mat[i][j];
            newPuzzle.mat[i][j] = newPuzzle.mat[i-1][j];
            newPuzzle.mat[i-1][j] = temp;
            newPuzzle.lastmove = 0;
            newPuzzle.moves = this.moves + 1;
            newPuzzle.cost = newPuzzle.calculateCost() + newPuzzle.moves;
            for (int k = 0; k < this.movesHistory.size(); k++) {
                newPuzzle.movesHistory.add(this.movesHistory.get(k));
            }
            newPuzzle.movesHistory.add(0);
            return newPuzzle;
        } else {
            return null;
        }
    }

    public Puzzle Down() {
        Puzzle newPuzzle = new Puzzle(this.mat);
        int idx[] = this.getIndex(16);
        int i = idx[0]; int j = idx[1];
        if (i < 3) {
            int temp = newPuzzle.mat[i][j];
            newPuzzle.mat[i][j] = newPuzzle.mat[i+1][j];
            newPuzzle.mat[i+1][j] = temp;
            newPuzzle.lastmove = 1;
            newPuzzle.cost = newPuzzle.calculateCost() + newPuzzle.moves;
            newPuzzle.moves = this.moves + 1;
            for (int k = 0; k < this.movesHistory.size(); k++) {
                newPuzzle.movesHistory.add(this.movesHistory.get(k));
            }
            newPuzzle.movesHistory.add(1);
            return newPuzzle;
        } else {
            return null;
        }
    }

    public Puzzle Left() {
        Puzzle newPuzzle = new Puzzle(this.mat);
        int idx[] = this.getIndex(16);
        int i = idx[0]; int j = idx[1];
        if (j > 0) {
            int temp = newPuzzle.mat[i][j];
```

```
            newPuzzle.mat[i][j] = newPuzzle.mat[i][j-1];
            newPuzzle.mat[i][j-1] = temp;
            newPuzzle.lastmove = 2;
            newPuzzle.cost = newPuzzle.calculateCost() + newPuzzle.moves;
            newPuzzle.moves = this.moves + 1;
            for (int k = 0; k < this.movesHistory.size(); k++) {
                newPuzzle.movesHistory.add(this.movesHistory.get(k));
            }
            newPuzzle.movesHistory.add(2);
            return newPuzzle;
        } else {
            return null;
        }
    }

    public Puzzle Right() {
        Puzzle newPuzzle = new Puzzle(this.mat);
        int idx[] = this.getIndex(16);
        int i = idx[0]; int j = idx[1];
        if (j < 3) {
            int temp = newPuzzle.mat[i][j];
            newPuzzle.mat[i][j] = newPuzzle.mat[i][j+1];
            newPuzzle.mat[i][j+1] = temp;
            newPuzzle.lastmove = 3;
            newPuzzle.cost = newPuzzle.calculateCost() + newPuzzle.moves;
            newPuzzle.moves = this.moves + 1;
            for (int k = 0; k < this.movesHistory.size(); k++) {
                newPuzzle.movesHistory.add(this.movesHistory.get(k));
            }
            newPuzzle.movesHistory.add(3);
            return newPuzzle;
        } else {
            return null;
        }
    }

    public int calculateCost()
    {
        int count = 0;
        for (int i = 0; i < length; i++)
            for (int j = 0; j < length; j++)
                if (this.mat[i][j] != (length*i + j+1) && this.mat[i][j] !=
16)
                    count++;
        return count;
    }

    public void Show() {
```

```
        for (int i = 0; i < length; i++) {
            // Loop through all elements of current row
            for (int j = 0; j < length; j++) {
                if (j != length-1) {
                    if (mat[i][j] == 16) {
                        System.out.print("- ");
                    } else {
                        System.out.print(mat[i][j]);
                    }
                    if (mat[i][j] < 10) {
                        System.out.print(" ");
                    }
                    System.out.print("  ");
                } else {
                    if (mat[i][j] == 16) {
                        System.out.println("- ");
                    } else {
                        System.out.println(mat[i][j]);
                    }
                }
            }
        }
    }

    public Puzzle shufflePuzzle() {
        Puzzle newPuzzle = this;
        int move = 60;
        for (int i = 0; i<move; i++) {
            int rand = new Random().nextInt(4);
            if (rand == 0) {
                if (newPuzzle.Up() != null) {
                    newPuzzle = newPuzzle.Up();
                }
            } else if (rand == 1) {
                if (newPuzzle.Down() != null) {
                    newPuzzle = newPuzzle.Down();
                }
            } else if (rand == 2) {
                if (newPuzzle.Left() != null) {
                    newPuzzle = newPuzzle.Left();
                }
            } else {
                if (newPuzzle.Right() != null) {
                    newPuzzle = newPuzzle.Right();
                }
            }
        }
        newPuzzle.moves = 0;
```

```
        newPuzzle.cost = newPuzzle.calculateCost() + newPuzzle.moves;
        newPuzzle.lastmove = -1;
        newPuzzle.movesHistory = new ArrayList<>();
        return newPuzzle;
    }
}

class PuzzleComparator implements Comparator<Puzzle>{
    public int compare(Puzzle p1, Puzzle p2) {
        if (p1.cost > p2.cost)
            return 1;
        else if (p1.cost < p2.cost)
            return -1;
        else
            if (p1.Kurang(false) > p2.Kurang(false)) {
                return 1;
            } else if (p1.Kurang(false) < p2.Kurang(false)) {
                return -1;
            } else {
                return 0;
            }
    }
}
```

Bab III : Testing

1.txt

```
\Semester 4\Tucil\Strategi Algoritma\tucil_3_stima'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '--enable-preview'
'-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Asus\AppData\Roaming\Code\User\workspaceStorage\263aa30406f02106
6fb71e1bc1ad1789\redhat.java\jdt_ws\tucil_3_stima_2785487\bin' 'Main' ucil_3_stima>
Pilih salah satu :
1. Puzzle automatis
2. Input File

>> 2

Path File >> test/1.txt

Puzzle awal :

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

Nilai dari Kurang(i) :

1  : 0
2  : 0
3  : 0
4  : 0
5  : 0
6  : 0
7  : 0
8  : 1
9  : 2
10 : 4
11 : 2
12 : 0
13 : 2
14 : 2
15 : 7
16 : 11

Result :

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Puzzle moves  = 21
Kurang(i)+X  = 32
Simpul Hidup = 456911
Duration     = 3013 ms
```

2.txt

```
Pilih salah satu :
1. Puzzle automatis
2. Input File

>> 2

Path File >> test/2.txt

Puzzle awal :

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

Nilai dari Kurang(i) :

1  : 0
2  : 0
3  : 0
4  : 0
5  : 1
6  : 4
7  : 0
8  : 2
9  : 1
10 : 1
11 : 3
12 : 1
13 : 4
14 : 1
15 : 2
16 : 3

Result :

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Puzzle moves  = 23
Kurang(i)+X  = 24
Simpul Hidup = 622851
Duration     = 4586 ms
```

3.txt

```
Pilih salah satu :
1. Puzzle automatis
2. Input File

>> 2

Path File >> test/3.txt

Puzzle awal :

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

Nilai dari Kurang(i) :

1  : 0
2  : 1
3  : 1
4  : 0
5  : 3
6  : 0
7  : 0
8  : 4
9  : 1
10 : 1
11 : 3
12 : 1
13 : 1
14 : 1
15 : 0
16 : 4

Result :

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

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

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

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

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

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

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

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

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

```
Puzzle moves  = 21
Kurang(i)+X  = 22
Simpul Hidup = 279699
Duration     = 1715 ms
```

4.txt

```
Pilih salah satu :
1. Puzzle automatis
2. Input File

>> 2

Path File >> test/4.txt

Puzzle awal :

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

Nilai dari Kurang(i) :

1  : 0
2  : 0
3  : 0
4  : 0
5  : 1
6  : 0
7  : 0
8  : 0
9  : 5
10 : 5
11 : 1
12 : 0
13 : 2
14 : 11
15 : 4
16 : 2

Result :

Tidak ada solusi dengan nilai Kurang(i)+X = 31
```

5.txt

```
Pilih salah satu :
1. Puzzle automatis
2. Input File

>> 2

Path File >> test/5.txt

Puzzle awal :

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

Nilai dari Kurang(i) :

1  : 0
2  : 0
3  : 2
4  : 2
5  : 0
6  : 2
7  : 6
8  : 2
9  : 5
10 : 2
11 : 0
12 : 0
13 : 1
14 : 1
15 : 1
16 : 15

Result :

Tidak ada solusi dengan nilai Kurang(i)+X = 39
```

Bab V : Kesimpulan dan Saran

Kesimpulan

Saya berhasil membuat program **"Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound"** dan sejauh ini belum ada solusi dari program yang terlewat dan/atau tidak tepat.

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		✓

Saran

Karena pada program ini hanya berbasis command line, mungkin kedepannya bisa dalam bentuk GUI sehingga lebih ramah untuk user. Untuk fungsionalitasnya mungkin kedepannya program ini bisa diupgrade sehingga dapat memasukkan input puzzle yang tidak hanya 4x4 saja. Selain itu, untuk algoritma Branch and Bound mungkin terdapat algoritma yang lebih efisien sehingga dapat mengurangi waktu penyelesaiannya.