

# **LAPORAN TUGAS KECIL II [IF2211] STRATEGI ALGORITMA**

***LONELY ISLAND***



Dibuat oleh :

Leonardo

13517048

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2019**

## I. Algoritma yang Digunakan

*Decrease and Conquer* : metode desain algoritma dengan mereduksi persoalan menjadi beberapa sub-persoalan yang lebih kecil, tetapi selanjutnya hanya memproses **satu** sub-persoalan saja.

Berbeda dengan *divide and conquer* yang memproses **semua** sub-persoalan dan menggabungkan semua solusi untuk setiap sub-persoalan.<sup>[1]</sup>

Implementasi *Decrease and Conquer* yang saya gunakan adalah algoritma pendekatan DFS (*Depth First Search*). Berikut langkah-langkahnya.

1. Dibentuk suatu *Incidence List* yang berupa list jembatan yang terdapat pada peta.  
Bentuk : Seperti pair, dimana *first* adalah pulau asal, dan *second* adalah pulau tujuan.
2. Program memanggil fungsi DFS yang dimulai dari pulau pertama.
3. Melakukan Iterasi pada *Incidence List*, apabila pulau awal sama dengan *first* dari elemen list[i], maka terpanggil fungsi DFS yang dimulai dari *second* dari element list[i] apabila *second* dari elemen list[i] belum pernah dikunjungi karena pulau yang telah dikunjungi tidak dapat dikunjungi kembali. Apabila tidak ada pulau awal yang sama dengan *first* dari elemen list[i], maka pulau tersebut dinyatakan sebagai *Lonely Island*. Jika ditemukan *lonely island*, akan dicetak ke layar jalan dari pulau asal menuju *lonely island* tersebut, lalu program melakukan *backtrack* untuk mencari semua solusi.
4. Saat masuk ke dalam DFS, pulau yang awalnya dikunjungi dikurangi agar tidak dapat dikunjungi kembali. Dengan cara memakai *list of passed islands*, setiap fungsi dipanggil, *list of passed islands* akan ditambahkan dengan pulau asal. Sehingga pulau tersebut tidak dapat dikunjungi lagi apabila terdapat siklik.
5. Ulangi langkah 3 – 4 hingga seluruh rekursif sudah kembali dan iterasi berhenti.
6. Mencetak ke layar pulau-pulau yang dinyatakan *lonely island*.

## II. Source code Program

Terdapat dua buah file kode sumber.

1. Edge.java : implementasi kelas Edge berbentuk *pair of vertex*.

```
/* Tugas Kecil 2 IF2211 Strategi Algoritma "Lone Island"
NIM / Nama   : 13517048 / Leonardo
Nama File    : Edge.java
Deskripsi    : Kelas Edge (berupa pair of template) untuk dipakai pada
LonelyIsland.java
*/
import java.util.*;

public class Edge<T> { //tidak pasti memakai integer agar dapat dipakai pada
    proyek-proyek lain
    //Terdapat First dan Second, Edge penghubung dari vertex first ke vertex
    second

    private T first;
    private T second;

    //ctor
    public Edge(T _first, T _second){
        this.first = _first;
        this.second = _second;
    }

    //getter
    public T getFirst(){
        return this.first;
    }

    public T getSecond(){
        return this.second;
    }

    //setter
    public void setFirst(T _first){
        this.first = _first;
    }

    public void setSecond(T _second){
        this.second = _second;
    }
}
```

## 2. LonelyIsland.java : program utama berisi algoritma Solve, dan main.

```
/* Tugas Kecil 2 IF2211 Strategi Algoritma "Lone Island"
NIM / Nama : 13517048 / Leonardo
Nama File : LonelyIsland.java
Deskripsi : Main program untuk mencari pulau dead-end dari suatu graf berarah.
Memanggil Edge.java untuk class Edge
*/

import java.util.*;
import java.io.*;

public class LonelyIsland {
    public static void Solve(int vertex, int bridges, int firstvert,
List<Edge<Integer>> listOfBridge){
        List<Integer> passed = new ArrayList<Integer>();
        boolean stuck[] = new boolean[vertex+1]; //memakai boolean agar hasil unik
        for (int i = 0; i <= vertex; i++){ // inisialisasi
            stuck[i] = false;
        }
        System.out.println("Stuck Routes :");
        DFS(bridges, firstvert, listOfBridge, passed, stuck); //memanggil
        algoritma DFS
        System.out.println("\nStuck Islands :");
        for (int j = 1; j <= vertex; j++){ // print terurut
            if (stuck[j]){
                System.out.println(j);
            }
        }
    }

    public static void DFS(int bridges, int firstvert, List<Edge<Integer>>
listOfBridge, List<Integer> passed, boolean stuck[]){
        // penyelesaian dengan algoritma Decrease and Conquer - Depth First
        Search (DFS)
        passed.add(firstvert);
        boolean skt = false;
        for (int i = 0; i < bridges; i++){
            if (listOfBridge.get(i).getFirst() == firstvert){
                if (!isPassed(listOfBridge.get(i).getSecond(), passed)){
                    DFS(bridges, listOfBridge.get(i).getSecond(), listOfBridge,
passed, stuck);
                    skt = true;
                }
            }
        }
    }
}
```

```

        //backtrack, hilangkan elemen terakhir
        passed.remove(passed.size()-1);
    } // apabila pulau sudah dilewati, pulau tidak dapat dilewati
kembali
    }
}

if (skt == false){//tidak ada yang jembatan ke pulau lain
    stuck[firstvert] = true;
    for (int i = 0; i < passed.size(); i++){
        System.out.print(passed.get(i));// menampilkan rute yang stuck
        if (i < passed.size() - 1){
            System.out.print(" -> ");
        } else {
            System.out.println();
        }
    }
}

}

public static boolean isPassed(int vert, List<Integer> li){
    //fungsi pencarian dengan implementasi boolean
    boolean sem = false;
    int i = 0;
    while ((i < li.size()) && (!sem)){
        if (vert == li.get(i)){
            sem = true;
        } else {
            i += 1;
        }
    }
    return sem;
}

public static void main(String[] args){
    // main program
    System.out.print("Input file name(end with .txt) : ");
    Scanner scan = new Scanner(System.in);
    String namafile = scan.nextLine();
    try {
        List<Edge<Integer>> listOfBridge = new ArrayList<Edge<Integer>>();
        File input = new File(namafile); //deklarasi file agar data di
dalamnya dapat diambil
        Scanner scaninp = new Scanner(input); //mengambil input dari file

```

```

        int vertex = scaninp.nextInt();
        int bridges = scaninp.nextInt();
        int firstvert = scaninp.nextInt();

        for (int i = 0; i < bridges; i++){
            int fst = scaninp.nextInt();
            int scd = scaninp.nextInt();

            Edge<Integer> temp = new Edge<Integer>(fst, scd);
            listOfBridge.add(temp);
        }

        long start = System.nanoTime();
        Solve(vertex, bridges, firstvert, listOfBridge);
        long end = System.nanoTime();

        System.out.printf("Execution Time : %.2f milliseconds\n",
(double) ((end-start)/1000000));
        scaninp.close();
    } catch (Exception e) {
        System.out.println("File not found! Program stopped.");
    }
    scan.close();
}
}

```

### III. Input – Output

Penjelasan : n = banyak pulau, m = banyak jembatan

Contoh 1. n = 5, m = 7

The screenshot shows two windows. The left window is Windows PowerShell (x86) with the following text:

```

PS C:\Java\Tucil2Stima\src> java LonelyIsland
Input file name(end with .txt) : input.txt
Stuck Routes :
1 -> 2 -> 4
1 -> 2 -> 5
1 -> 3 -> 4
1 -> 4
1 -> 5

Stuck Islands :
4
5
Execution Time : 6.00 milliseconds
PS C:\Java\Tucil2Stima\src>

```

The right window is Notepad titled 'input.txt - Notepad' with the following text:

```

File Edit Format View Help
5 7 1
1 2
1 3
1 4
1 5
2 4
2 5
3 4

```

Contoh 2.  $n = 16, m = 25$

```
Windows PowerShell (x86)
PS C:\Java\Tucil2Stima\src> javac LonelyIsland.java
PS C:\Java\Tucil2Stima\src> java LonelyIsland
Input file name(end with .txt) : input.txt
Stuck Routes :
1 -> 2 -> 3 -> 5 -> 7
1 -> 2 -> 3 -> 5 -> 6 -> 10 -> 8 -> 9
1 -> 2 -> 3 -> 5 -> 6 -> 10 -> 8 -> 13 -> 12 -> 14 -> 15
1 -> 2 -> 3 -> 5 -> 6 -> 10 -> 8 -> 13 -> 12 -> 16
1 -> 2 -> 3 -> 5 -> 6 -> 10 -> 8 -> 13 -> 14 -> 15
1 -> 2 -> 3 -> 5 -> 6 -> 10 -> 8 -> 13 -> 16 -> 12 -> 14 -> 15
1 -> 2 -> 4 -> 7
1 -> 2 -> 15 -> 14
Stuck Islands :
7
9
14
15
16
Execution Time : 106.00 milliseconds
PS C:\Java\Tucil2Stima\src>
```

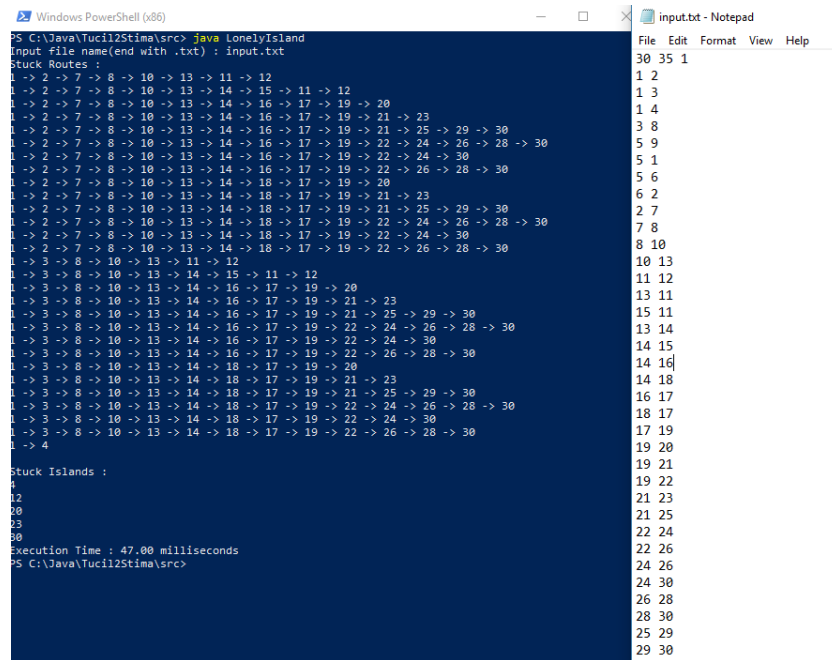
```
input.txt - Notepad
File Edit Format View Help
16 25 1
1 2
2 3
2 4
3 5
5 7
4 7
5 6
6 2
6 10
10 1
10 8
8 9
9 9
16 12
11 13
13 12
13 14
14 15
2 15
15 15
15 14
8 13
12 14
12 16
13 16
16 15
```

Contoh 3.  $n = 10, m = 10$

```
Windows PowerShell (x86)
PS C:\Java\Tucil2Stima\src> java LonelyIsland
Input file name(end with .txt) : input.txt
Stuck Routes :
1 -> 3 -> 2 -> 4 -> 7 -> 5 -> 9 -> 10
1 -> 6 -> 8 -> 9 -> 10
Stuck Islands :
10
Execution Time : 2.00 milliseconds
PS C:\Java\Tucil2Stima\src>
```

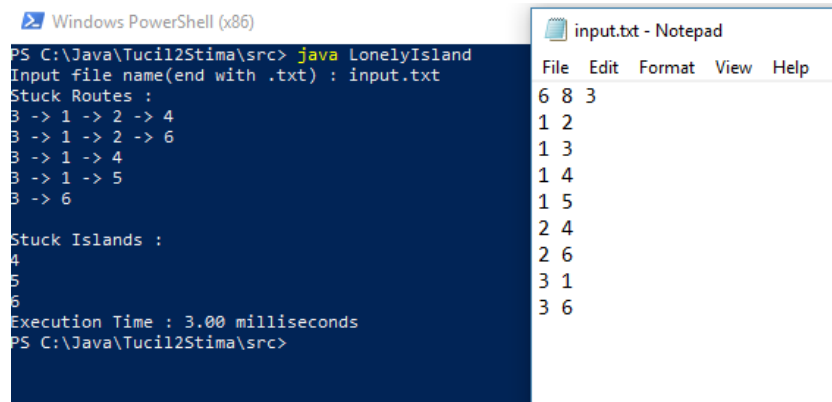
```
input.txt - Notepad
File Edit Format View Help
10 10 1
1 3
2 4
3 2
4 7
7 5
1 6
6 8
8 9
5 9
9 10
```

#### Contoh 4. $n = 30$ , $m = 35$



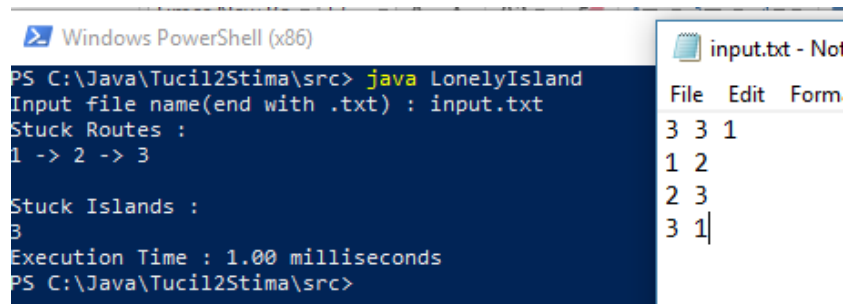
The screenshot shows a Windows PowerShell window and a Notepad window. The PowerShell window displays the output of the Java program 'LonelyIsland'. It reads 'input.txt' and prints 'Stuck Routes :'. The routes are listed as sequences of numbers connected by arrows, representing paths between islands. The routes are: 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 11 -> 12; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 15 -> 11 -> 12; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 20; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 21 -> 23; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 21 -> 25 -> 29 -> 30; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 22 -> 24 -> 26 -> 28 -> 30; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 22 -> 24 -> 30; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 22 -> 26 -> 28 -> 30; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 20; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 21 -> 23; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 21 -> 25 -> 29 -> 30; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 22 -> 24 -> 26 -> 28 -> 30; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 22 -> 24 -> 30; 1 -> 2 -> 7 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 22 -> 26 -> 28 -> 30; 1 -> 3 -> 8 -> 10 -> 13 -> 11 -> 12; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 15 -> 11 -> 12; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 20; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 21 -> 23; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 21 -> 25 -> 29 -> 30; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 22 -> 24 -> 26 -> 28 -> 30; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 16 -> 17 -> 19 -> 22 -> 24 -> 30; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 20; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 21 -> 23; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 21 -> 25 -> 29 -> 30; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 22 -> 24 -> 26 -> 28 -> 30; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 22 -> 24 -> 30; 1 -> 3 -> 8 -> 10 -> 13 -> 14 -> 18 -> 17 -> 19 -> 22 -> 26 -> 28 -> 30; 1 -> 4. The PowerShell window also shows 'Stuck Islands : 4, 12, 30, 23, 30' and 'Execution Time : 47.00 milliseconds'. The Notepad window shows the content of 'input.txt': 30 35 1, 1 2, 1 3, 1 4, 3 8, 5 9, 5 1, 5 6, 6 2, 2 7, 7 8, 8 10, 10 13, 11 12, 13 11, 15 11, 13 14, 14 15, 14 16, 14 18, 16 17, 18 17, 17 19, 19 20, 19 21, 19 22, 21 23, 21 25, 22 24, 22 26, 24 26, 24 30, 26 28, 28 30, 25 29, 29 30.

#### Contoh 5. Pulau mulai bukan 1



The screenshot shows a Windows PowerShell window and a Notepad window. The PowerShell window displays the output of the Java program 'LonelyIsland'. It reads 'input.txt' and prints 'Stuck Routes :'. The routes are: 3 -> 1 -> 2 -> 4; 3 -> 1 -> 2 -> 6; 3 -> 1 -> 4; 3 -> 1 -> 5; 3 -> 6. The PowerShell window also shows 'Stuck Islands : 4, 5, 6' and 'Execution Time : 3.00 milliseconds'. The Notepad window shows the content of 'input.txt': 6 8 3, 1 2, 1 3, 1 4, 1 5, 2 4, 2 6, 3 1, 3 6.

#### Contoh 6. Ada siklik



The screenshot shows a Windows PowerShell window and a Notepad window. The PowerShell window displays the output of the Java program 'LonelyIsland'. It reads 'input.txt' and prints 'Stuck Routes :'. The routes are: 1 -> 2 -> 3. The PowerShell window also shows 'Stuck Islands : 3' and 'Execution Time : 1.00 milliseconds'. The Notepad window shows the content of 'input.txt': 3 3 1, 1 2, 2 3, 3 1.



#### IV. Tabel Poin

Poin	Ya	Tidak
1. Program berhasil dikompilasi	✓	
2. Program berhasil dieksekusi	✓	
3. Program dapat menerima input dan menuliskan output	✓	
4. Luaran sudah benar untuk semua $n$	✓	

#### V. Referensi

1. [http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Decrease-and-Conquer-\(2018\).pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Decrease-and-Conquer-(2018).pdf)