LAPORAN TUGAS KECIL-3 STRATEGI ALGORITMA

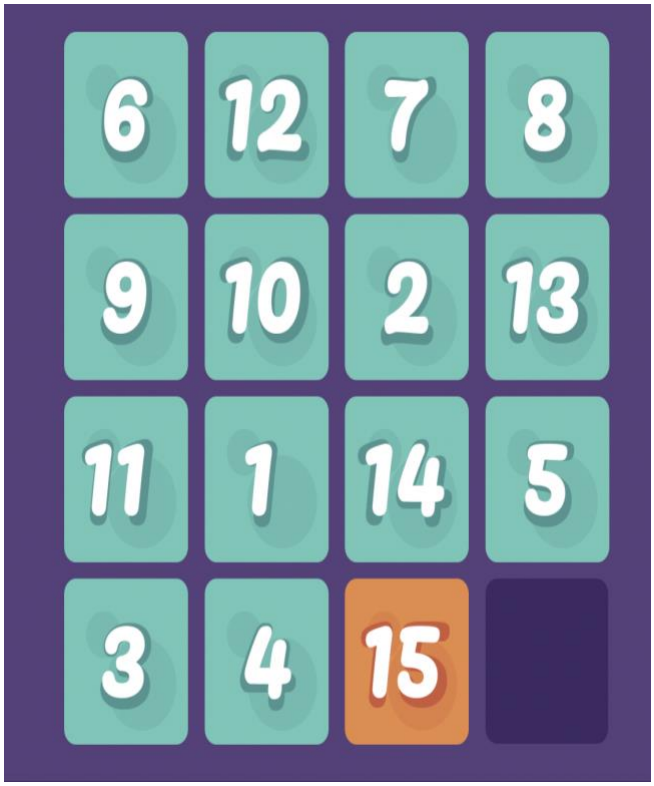
Program penyelesaian 15-puzzle dengan Algoritma Branch and Bound

Disusun Oleh:

Muhammad Gerald Akbar Giffera 13520143

Institut Teknologi Bandung
2022

1. Algoritma *Branch and Bound*



Algoritma *branch and bound,* adalah algoritma digunakan untuk menyelesaikan persoalan optimalisasi (minimalisasi/maksimalisasi). Algoritma ini, mirip dengan algoritma *BFS,* karena sama-sama memanfaatkan struktur data *queue.* Sama seperti halnya pada BFS setiap simpul yang dibangkitkan yang akan dilakukan analisis lebih lanjut akan disimpan ke dalam queue, akan tetapi terdapat sedikit perbedaan pada *branch and bound,* yaitu pada proses mengkases queue, karena pada *branch and bound* setiap simpul memiliki nilai/*cost* tertentu yang akan menyatakan urutan aksesnya, sehingga bisa dibilang bahwa lebih tepatnya algoritma branch and bound memanfaatkan struktur data *priority queue.*

Secara garis besar, terdapat langkah-langkah yang dilakukan dalam algoritma *branch and bound,* yaitu sebagai berikut:

a. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (goal node), maka solusi telah ditemukan. Jika hanya satu solusi yang diinginkan, maka stop.
b. Jika Q kosong, Stop.
c. Jika Q tidak kosong, pilih dari antrian Q simpul i yang mempunyai nilai ‘cost’ ĉ(i) paling kecil. Jika terdapat beberapa simpul i yang memenuhi, pilih satu secara sembarang.
d. Jika simpul i adalah simpul solusi, berarti solusi sudah ditemukan. Jika satu solusi yang diinginkan, maka stop. Pada persoalan optimasi dengan pendekatan least cost search, periksa cost semua simpul hidup. Jika cost nya lebih besar dari cost simpul solusi, maka matikan simpul tersebut.
e. Jika simpul i bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.
f. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak-anak tersebut ke dalam Q.
g. Kembali ke langkah 2

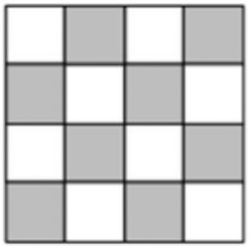
Selain itu ada beberapa heuristik yang diterapkan untuk meminimalisasi Langkah-langkah yang dilalui saat proses penyelesaian puzzle.

a. Menentukan apakah puzzle dapat diselesaikan
   Karena terdapat 16! Kemungkinan penempatan tile pada puzzle, dan hanya setengah dari jumlah tersebut yang mempunyai solusi sehingga, dilakukan kalkulasi nilai goal yang menandakan apakah puzzle tersebut mempunyai solusi.

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

   - KURANG(i)
     Banyaknya ubin bernomor j sedemikian sehingga j < i dan POSISI(j) > POSISI(i). POSISI(i) = posisi ubin bernomor i pada susunan yang diperiksa

   - X



     X bernilai 1 jika sel kosong pada posisi awal berada pada tile yang diarsir.

b. Menghitung cost simpul (ĉ(i))

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

   - f(i) → jarak dari simpul akar ke simpul i
   - g(i) → jumlah tile yang tidak sesuai dengan simpul tujuan (target)

2. Kode Program

Implementasi program ini memanfaatkan Bahasa pemrograman Java, sehingga dibentuk beberapa kelas untuk memudahkan representasi simpul, puzzle, pembacaan input dan algoritma penyelesaian puzzle menggunakan *branch and bound.*

Kelas Puzzle (Puzzle.java)

```
import java.util.*;

public class Puzzle{

    Integer cost;
    ArrayList<Integer> puzzle;

    // constructor
    Puzzle(){
        this.puzzle = new ArrayList<Integer>();
    }
    Puzzle(Puzzle p){
        this.cost = p.cost;
        this.puzzle = new ArrayList<Integer>(p.puzzle);
    }

    // getter & setter
    public void setPuzzle(ArrayList<Integer> a){
        this.puzzle = new ArrayList<Integer>(a);
    }
    public void setCost(Integer cost) {
        this.cost = cost;
    }
    public Integer getCost(){
        return cost;
    }
    public Boolean targetReached() {
        return this.g().equals(0);
    }

    // menampilkan puzzle
    public void displayPuzzle() {
        for (int i = 0; i < 4; i ++){
```

```java
// menampilkan nilai KURANG(i)
public void displayKurangValue() {
    for (int i = 1; i <= this.puzzle.size(); i++){
        System.out.print("Kurang(" + i + ") = " + kurang(this.puzzle, i) + "\n");
    }
}

// kalkulasi nilai g(i)
public Integer g() {
    Integer res = 0;
    for (int i = 0; i < this.puzzle.size(); i ++){
        if (!(this.puzzle.get(i).equals(i+1)) && !(this.puzzle.get(i).equals(16))){
            res++;
        }
    }
    return res;
}

public Boolean isSolvable(){
    return ((reachable(this.puzzle)%2) == 0);
}

// memeriksa nilai sigma(kurang(i)) + X
public static int reachable(ArrayList<Integer> puzzle){
    Integer reachable = 0;
    for (int i = 0; i < puzzle.size(); i++){
        if (!puzzle.get(i).equals(0)){
            reachable += kurang(puzzle, i+1);
        }
    }
    return reachable + getX(puzzle);
}

// kalkulasi nilai KURANG(i)
public static int kurang(ArrayList<Integer> puzzle, Integer i){
    Integer idx = puzzle.indexOf(i);
    Integer res = 0;
    for (int j = idx; j < puzzle.size(); j++){
        if (puzzle.get(j) < i){
```

```
    Collections.addAll(arsir,1,3,4,6,9,11,12,14);
    Integer emptyidx = puzzle.indexOf(16);

    if (arsir.contains(emptyidx)){
        return 1;
    }
    else{
        return 0;
    }
}
```

```
// menentukan apakah move dapat dilakukan dan tidak redundan
public Boolean isMoveSafe(String direction, String lastMove){
    Integer emptyidx = this.puzzle.indexOf(16);
    if (direction.equals("down")){
        return emptyidx <= 11 && lastMove != "up";
    }
    else if (direction.equals("up")){
        return emptyidx >= 4 && lastMove != "down";
    }
    else if (direction.equals("left")){
        return (emptyidx%4) != 0 && lastMove != "right";
    }
    else{
        return ((emptyidx+1)%4) != 0 && lastMove != "left";
    }
}

// mengembalikan puzzle yang sudah digerakkan tile-nya
public ArrayList<Integer> move(String direction){
    ArrayList<Integer> temp;

    if (direction.equals("down")){
        temp = new ArrayList<Integer>(this.down());
        return temp;
    }
    else if (direction.equals("up")){
        temp = new ArrayList<Integer>(this.up());
```

```
}

// FUNGSI PERGERAKAN
public ArrayList<Integer> down() {
   Integer emptyidx = this.puzzle.indexOf(16);
   ArrayList<Integer> temp = new ArrayList<>(this.puzzle);
   Collections.swap(temp, emptyidx, emptyidx+4);
   return temp;
}
public ArrayList<Integer> up(){
    Integer emptyidx = this.puzzle.indexOf(16);
    ArrayList<Integer> temp = new ArrayList<>(this.puzzle);
    Collections.swap(temp, emptyidx, emptyidx-4);
    return temp;
}
public ArrayList<Integer> left(){
    Integer emptyidx = this.puzzle.indexOf(16);
    ArrayList<Integer> temp = new ArrayList<>(this.puzzle);
    Collections.swap(temp, emptyidx, emptyidx-1);
    return temp;
}
```

Kelas Node (Node.java)

```
public class Node implements Comparable<Node>{

    Node parent;
    Puzzle puzzle;
    Integer level;
    String lastMove;

    //ctor dan cctor
    Node (Puzzle p,Node parent, Integer level, String lastMove){
        this.puzzle = p;
        this.parent = parent;
        this.level = level;
        this.lastMove = lastMove;
    }
    Node (Node n){
        this.puzzle = n.puzzle;
        this.parent = n.parent;
        this.level = n.level;
        this.lastMove = n.lastMove;
    }

    // getter
    public Puzzle getPuzzle(){
        return this.puzzle;
    }

    // menampilkan path solusi
    public static void printPath(Node root){
        if (root == null){
            return;
        }
        printPath(root.parent);
        System.out.println("------------");
        root.puzzle.displayPuzzle();
        System.out.println("------------");
```

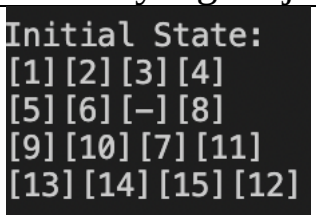
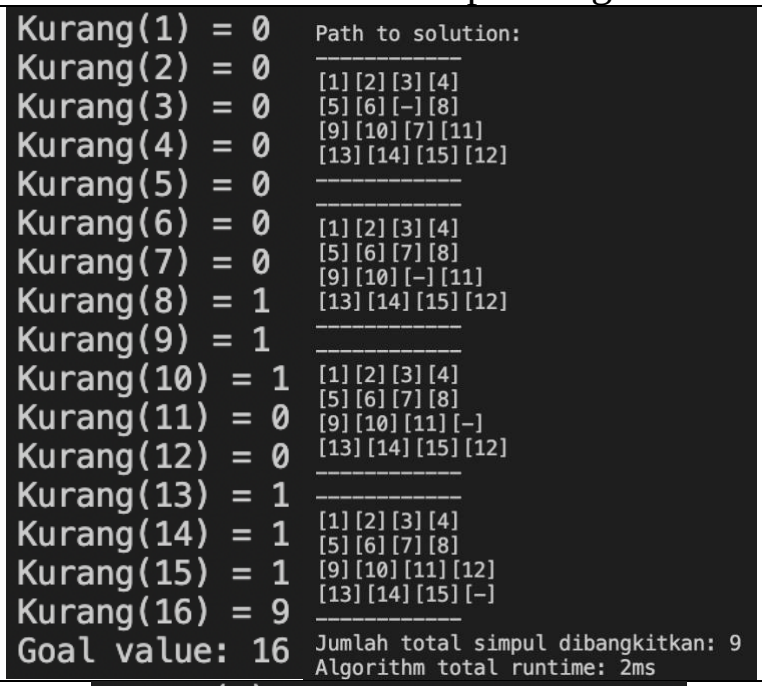
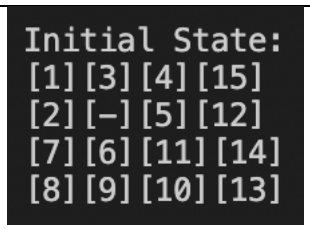
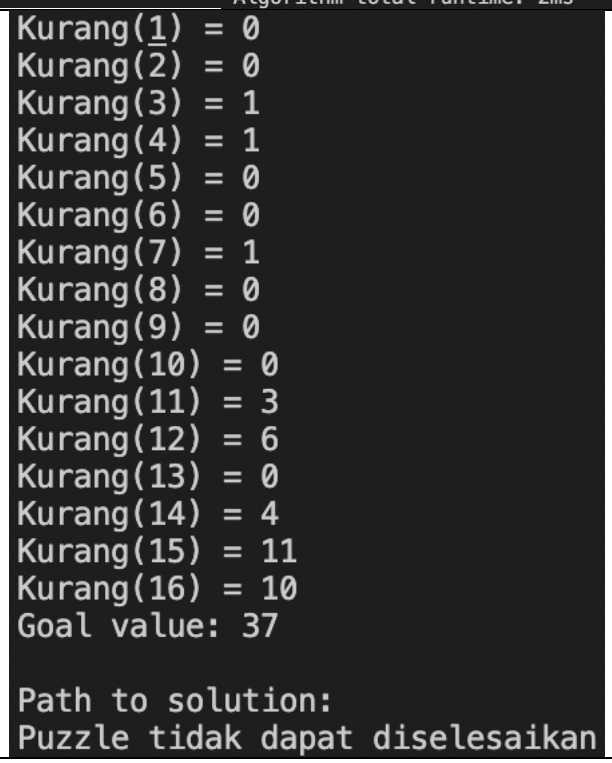
Main Program

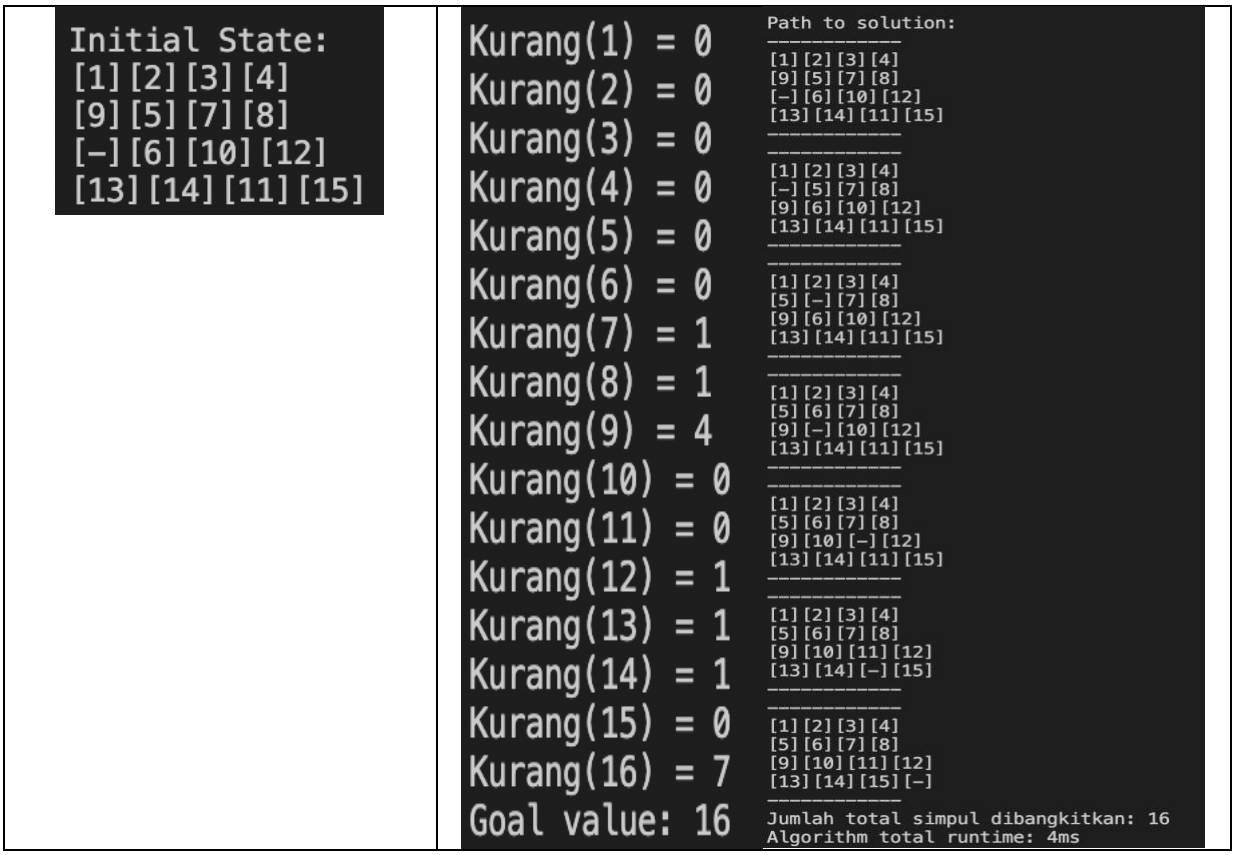
```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.PriorityQueue;
import java.util.Queue;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;
```

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

        Integer simpulDibangkitkan = 0;
        Queue<Node> pq = new PriorityQueue<>();

        Puzzle puzzle = new Puzzle();
        Puzzle target = new Puzzle();

        ReadText fileReader1 = new ReadText();
        ReadText fileReader2 = new ReadText();

        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter file name with puzzle: ");
        String filename = scanner.nextLine();

        fileReader1.readPuzzleFile(filename);
        fileReader2.readPuzzleFile("target.txt");

        puzzle.setPuzzle(fileReader1.getHasil());
        target.setPuzzle(fileReader2.getHasil());

        puzzle.setCost(puzzle.g());

        Node root = new Node(puzzle, null, 0,"None");
```

```
if (puzzle.isSolvable()){
    // handle exception out of memory
    // terjadi kalo puzzle nya rumit
    try{
        Long startTime = System.nanoTime();
      while (pq.size() > 0){
            Node minNode = new Node(pq.remove());

            // goal state tercapai
            if (minNode.getPuzzle().targetReached()){
                Node.printPath(minNode);
                Long endTime = System.nanoTime();
                Long totalTime = endTime-startTime;
                System.out.println("Jumlah total simpul dibangkitkan: " +
      simpulDibangkitkan);
                System.out.println("Algorithm total runtime: " + TimeUnit.NANO  SECONDS.toMillis(totalTime) + "ms");
                return;
            }

            String moves[] = {"up","down","right","left"};
            for (int i = 0; i < 4; i++){
                // memeriksa apakah move valid dan tidak redundan
                // cth. left-right
                if (minNode.getPuzzle().isMoveSafe(moves[i],minNode.lastMove)){
                    Puzzle tempPuzzle = new Puzzle(minNode.getPuzzle());

                    tempPuzzle.setPuzzle(minNode.getPuzzle().move(moves[i]));
                    // minNode.level + 1 menandakan jarak dari root ke node

                    tempPuzzle.setCost(tempPuzzle.g() + minNode.level + 1);
                    Node child = new Node(tempPuzzle,minNode,minNode.level + 1,moves[i]);
                    simpulDibangkitkan++;
                    pq.add(child);
                }
            }
        }


    }
    catch(OutOfMemoryError e){
```

3. Hasil pengujian Program

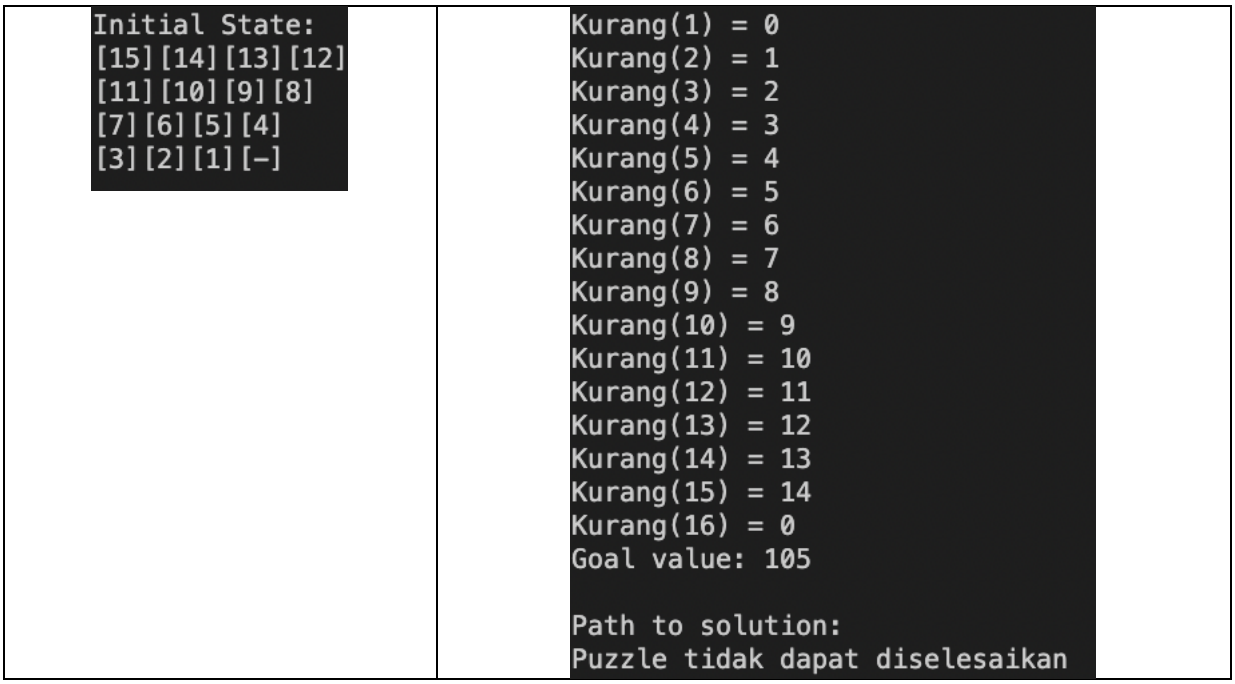
Puzzle yang diuji	Bukti Screenshot Output Program
	
	


Initial State:
[1][2][3][4]
[9][5][7][8]
[-][6][10][12]
[13][14][11][15]
Kurang(1) = 0
Kurang(2) = 0
Kurang(3) = 0
Kurang(4) = 0
Kurang(5) = 0
Kurang(6) = 0
Kurang(7) = 1
Kurang(8) = 1
Kurang(9) = 4
Kurang(10) = 0
Kurang(11) = 0
Kurang(12) = 1
Kurang(13) = 1
Kurang(14) = 1
Kurang(15) = 0
Kurang(16) = 7
Goal value: 16
Path to solution:
[1][2][3][4]
[9][5][7][8]
[-][6][10][12]
[13][14][11][15]
[1][2][3][4]
[-][5][7][8]
[9][6][10][12]
[13][14][11][15]
[1][2][3][4]
[5][-][7][8]
[9][6][10][12]
[13][14][11][15]
[1][2][3][4]
[5][6][7][8]
[9][-][10][12]
[13][14][11][15]
[1][2][3][4]
[5][6][7][8]
[9][10][-][12]
[13][14][11][15]
[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][-]
Jumlah total simpul dibangkitkan: 16
Algorithm total runtime: 4ms

```
Initial State:
[1][3][7][4]
[5][2][10][8]
[9][6][12][-]
[13][14][11][15]
```

```
Kurang(1) = 0
Kurang(2) = 0
Kurang(3) = 1
Kurang(4) = 1
Kurang(5) = 1
Kurang(6) = 0
Kurang(7) = 4
Kurang(8) = 1
Kurang(9) = 1
Kurang(10) = 3
Kurang(11) = 0
Kurang(12) = 1
Kurang(13) = 1
Kurang(14) = 1
Kurang(15) = 0
Kurang(16) = 4
Goal value: 20
```

```
Path to solution:
------------
[1][3][7][4]
[5][2][10][8]
[9][6][12][-]
[13][14][11][15]
------------
------------
[1][3][7][4]
[5][2][10][8]
[9][6][-][12]
[13][14][11][15]
------------
------------
[1][3][7][4]
[5][2][-][8]
[9][6][10][12]
[13][14][11][15]
------------
------------
[1][3][-][4]
[5][2][7][8]
[9][6][10][12]
[13][14][11][15]
------------
------------
[1][-][3][4]
[5][2][7][8]
[9][6][10][12]
[13][14][11][15]
------------
------------
[1][2][3][4]
[5][-][7][8]
[9][6][10][12]
[13][14][11][15]
------------
------------
[1][2][3][4]
[5][6][7][8]
[9][-][10][12]
[13][14][11][15]
------------
------------
[1][2][3][4]
[5][6][7][8]
[9][10][-][12]
[13][14][11][15]
------------
------------
[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][-]
------------
Jumlah total simpul dibangkitkan: 31
Algorithm total runtime: 8ms
```



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

Link Repository:
https://github.com/geraldakbar/Tucil3_13520143