Penyelesaian Persoalan 15-Puzzle dengan Algoritma Branch and Bound

Averrous Saloom | 13520100
Tugas Kecil III, IF2211 Strategi Algoritma
Program Studi Teknik Informatika, Institut Teknologi Bandung

Penjelasan Algoritma

Algoritma Branch and Bound adalah algoritma yang menambahkan BFS (Breadth First Search) dengan perhitungan ongkos. Perhitungan ongkos ini akan menjadi urutan pengaksesan suatu simpul status dalam antrian BFS. Penambahan perhitungan ongkos ini bermanfaat untuk optimalisasi solusi.

Penjelasan umum

Pada permasalahan 15-Puzzle, perlu ditemukan rangkaian perintah yang paling efisien untuk menemukan solusi. Algoritma BnB adalah salah satu metode yang tepat, karena perilaku BFS-nya yang memungkinan pemeriksaan ongkos dilakukan secara menyeluruh.

Struktur data

Struktur data yang tepat untuk merepresentasikan puzzle menjadi kunci dalam performa BnB yang dibuat. Saya memilih untuk merepresentasikan puzzle dalam bentuk hashtable yang diimplementasikan dengan list. Hashtable memiliki *key* angka serta *value* posisi yang berbentuk list dua elemen. Struktur data ini memungkinkan pencarian posisi yang cepat, dan membuat operasi yang mengandung pencarian yang banyak terdapat di BnB menjadi lebih murah.

Pemeriksaan "Reachable"

Permasalahan 15-Puzzle memiliki sebuah teorema yang dapat menunjukkan bahwa masalah ini dapat diselesaikan atau tidak. Teorema tersebut berbentuk:

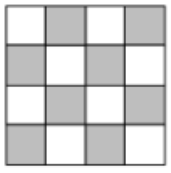
$$\sum_{i=1}^{16} Kurang(i) + X$$

Dengan,

$$Kurang(i): Banyaknya\ ubin\ bernomor\ j\ sedemikian\ sehingga\ j < i\ dan\ POSISI(j) > POSISI(i).\ POSISI(k)\ adalah\ posisi\ ubin\ bernomor\ k\ pada\ susunan\ yang\ diperiksa$$

dan,

$$X = 1\ jika\ POSISI(k)\ berada\ pada\ posisi\ di\ arsir\ dan\ X = 0\ jika\ tidak\ diarsir$$



Puzzle disebut "reachable" atau dapat diselesaikan, jika teorema bernilai bilangan genap.

Ongkos

Ongkos diperhitungkan dengan taksiran:

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

$\hat{c}(i)$: *Ongkos untuk simpul* i
$\hat{f}(i)$: *Ongkos mencapai simpul* i *dari akar*
$\hat{g}(i)$: *Ongkos mencapai simput tujuan dari simpul* i

Pada persoalan ini, ongkos mencapai simpul tujuan tidak dapat ditentukan karena tidak dapat diketahui secara trivial jalan yang harus dilalui untuk mencapai simpul tujuan. Oleh karena itu, diperlukan taksiran nilai $\hat{g}(i)$:

$\hat{g}(i)$: *jumlah ubin tidak kosong yang tidak terdapat pada susunan akhir*
$*$ 3 *ditambah dengan* 2
$*$ *aproksimasi jarak terdekat dari seluruh posisi ubin ke posisi benar.*

Operasi
Pencarian dilakukan dengan mengantrikan status awal pada antrian. Status awal ini lalu diakses untuk diperiksa apakah sesuai dengan tujuan atau tidak. Jika sesuai tujuan, program selesai. Jika tidak, akan dibangkitkan status baru yakni anak daripada status awal tersebut. Anak daripada status dihasilkan dengan melakukan operasi "UP", "DOWN", "RIGHT", atau "LEFT". Operasi yang dijalankan hanya operasi yang rasional, yakni:

- Tidak berlawanan dengan operasi yang sebelumnya dilakukan, karena jika berlawanan, operasi hanya akan mengembalikan ke dua status sebelumnya
- Tidak menghasilkan status yang berbeda. Seperti memberikan operasi "UP" ketika elemen sudah berada di baris paling atas.

Anak status yang dibangkitkan ini akan diantrikan sesuai dengan ongkos yang dimiliki status tersebut. Semakin kecil ongkos yang dimiliki, semakin depan posisi antrian status berada. Antrian yang dimodifikasi inilah yang akan memberikan dampak optimisasi pada penyelesaian masalah.

Pengaksesan akan dilanjutkan dengan mengambil antrian terdepan. Proses akan diulangi sampai tujuan ditemukan. Tujuan sudah pasti ditemukan karena proses pencarian hanya akan berjalan jika Puzzle bersifat "reachable".

Kode Program

- **Program Utama**

```java
import IO.*;
import Puzzle.BnBPuzzle;
You, 11 hours ago | 1 author (You)
public class Program {
    public static void welcome(){
        System.out.println("Welcome to 15-Puzzle Solver");
    }

    public static int[][] input(){
        System.out.println("Input your puzzle down here");
        int[][] c = new int[][]{};
        try {
            c = IOManagement.read();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
        return c;
    }
    Run | Debug
    public static void main(String[] args) {
        welcome();
        BnBPuzzle solver = new BnBPuzzle(input());
        solver.run();
    }
}
```

- **Kelas BnBPuzzle**

```
public class BnBPuzzle {
    private BnBQueue queue;
    public Puzzle problem;

    public BnBPuzzle(int[][] p) {
        problem = new Puzzle(p);
    }

    private void enqueueAllChild(StateTreeNode<Puzzle> n) {
        for (int i = 0; i < n.childs.size(); i++) {
            queue.enqueue(n.childs.get(i));
        }
    }

    Run | Debug
    public static void main(String[] args) {
        BnBPuzzle 1 = new BnBPuzzle(IOManagement.read());
        1.run();
    }

    public void printInitialStatus(StateTreeNode<Puzzle> initialState) {
        System.out.println("INITIAL STATE");
        initialState.val.print();
        System.out.println();
        System.out.println("'KURANG' VALUE");
        for (int i = 1; i <= 16; i++) {
            System.out.print(i + ":" + DifferenceTheorem.difference(i, initialState.val.getContent()) + "; ");
        }
        System.out.println();
        System.out.println();
        System.out.println(
                "'REACHABLE GOAL' THEOREM VALUE: " + DifferenceTheorem.differeceTheorem(initialState.val.getContent()));
    }
```

```
    public void run() {
        StateTreeNode<Puzzle> access;
        StateTreeNode<Puzzle> initialState = new StateTreeNode<Puzzle>(problem, 0, "", new ArrayList<>());
        int generatedState = 1;
        queue = new BnBQueue();
        System.out.println();
        printInitialStatus(initialState);
        // some time passes
        if (DifferenceTheorem.differeceTheorem(initialState.val.getContent()) % 2 == 0) {
            System.out.println("Solving...");
            queue.enqueue(initialState);
            boolean found = false;
            long start = System.nanoTime();
            while (!queue.isEmpty() && !found) {
                access = queue.dequeue();
                if (access.val.isGoal()) {      You, 11 hours ago • bulk commit …
                    found = true;
                    long end = System.nanoTime();
                    access.printPath(initialState.val);
                    long elapsedTime = end - start;
                    System.out.println("ELAPSED TIME: " + elapsedTime +" Nanoseconds");
                    System.out.println("GENERATED STATE: " + generatedState +" States");
                } else {
                    generatedState += access.expand();
                    enqueueAllChild(access);
                }
            }
        } else {
            System.out.println();
            System.out.println("SOLUTION IS UNREACHABLE");
        }

    }
```

- **Kelas Puzzle**

```java
public class Puzzle  {
    /*
    content berbentuk 16x2 dengan 16 angka yang menyimpan 2 buah nilai posisi
    pendekatan ini dipilih karena memudahkan pencarian dan pertukaran posisi
    */
    private int[][] content;
    public Puzzle(int[][] c){
        content = c;
    }

    public void copy(Puzzle a){
        // a has been initialize
        a.content = content;
    }

    public Puzzle copy(){
        Puzzle b = new Puzzle(content.clone());
        return b;
    }

    public int[][] getContent() {
        return content;
    }

    public void print(){
        int tmp;
        for (int i = 0; i< 4; i++){
            for (int j = 0; j < 4; j++){
                tmp = occupiedBy(new int[]{i, j});
                if (tmp == 16) {
                    System.out.print("? ");
                } else {
                    System.out.print(tmp +" ");
                }
            }
            System.out.println();
        }
    }
    /*
    Untuk melakukan print, perlu dilakukan pencarian mana angka yang harus diprint duluan
    */
    public void printPositionStatus(){
        for (int i = 0; i < content.length; i++) {
            System.out.print("Position of " + (i+1) + " is in position of " );
            System.out.print(correctNumberOf(content[i]));
            System.out.println();
        }
    }
```

```java
public void setPosition(int num, int[] position) {
    content[num - 1] = position;
}


public int[] getPosition(int num){
    return content[num-1];
}

public boolean isGoal(){
    boolean v = true;
    int i = 0;
    while (i < content.length && v) {
        v = v && isCorrectPosition(i+1, content[i]);
        i++;
    }

    return v;
}

public static boolean isCorrectPosition(int num, int[] pos){
    return (pos[0]*4 + pos[1] + 1) == num;
}

public static int correctNumberOf(int[] pos) {
    return (pos[0]*4 + pos[1] + 1);
}

public static int[] correctPositionOf(int num) {
    int row = num / 4;
    int col = num % 4;
    return new int[]{row, col};
}

public static int distanceFromCorrectPosition(int num, int[] pos) {
    return Math.abs((pos[0]*4 + pos[1] + 1) - num);
}

public static int shortestDistanceFromCorrectPosition(int num, int[] pos) {
    int[] a = correctPositionOf(num);
    int difX = Math.abs(pos[0] - a[0]);
    int difY = Math.abs(pos[1] - a[1]);
    return (difX+difY);       You, 4 minutes ago • bulk commit …
}
```

```java
  private static boolean isArrEqual(int[] a, int[] b){
      if (a.length == b.length){
          int i = 0;
          boolean same = true;
          while (i < a.length && same){
              same = a[i] == b[i];
              i++;
          }
          return same;
      } else {
          return false;
      }
  }

  public int occupiedBy(int[] pos) {
      int i = 0;
      while (i < content.length && !isArrEqual(pos, content[i])){
          i++;
      }

      if (i < content.length && isArrEqual(pos, content[i])) {
          return i + 1;
      } else {
          return -1;
      }
  }

}
```

- **Kelas StateTreeNode**

```java
public class StateTreeNode<T> extends TreeNode<T> {
    public ArrayList<StateTreeNode<T>> childs;
    public int pathCount;
    public String commandCreator;
    public ArrayList<String> commandCreators;
    ArrayList<Command<Puzzle>> commands;

    @SuppressWarnings("unchecked")
    public StateTreeNode(T p, int pathCount, String co, ArrayList<String> before) {
        super(p);
        this.pathCount = pathCount;
        commandCreator = co;
        commandCreators = (ArrayList<String>) before.clone();
        if (!commandCreator.equals("")) {
            commandCreators.add(co);
        }
        this.childs = new ArrayList<StateTreeNode<T>>();
        this.commands = new ArrayList<Command<Puzzle>>();
        commands.add(new UpCommand());
        commands.add(new DownCommand());
        commands.add(new RightCommand());
        commands.add(new LeftCommand());
    }

    private static <U> U runCommand(Command<U> commands, U v) {
        return commands.nextState(v);
    }

    public int cost() {
        if (val instanceof Puzzle) {
            Puzzle a = (Puzzle) val;
            return  g(a.getContent()) + pathCount ;
        } else {
            return 0;
        }
    }
```

```
    private int g(int[][] p) {
        int count = 0;
        for (int i = 0; i < p.length; i++) {
            count += Puzzle.shortestDistanceFromCorrectPosition(i+1, p[i]);
            if (!Puzzle.isCorrectPosition(i + 1, p[i]) && (i+1) != p.length) {
                count += 3;       You, 5 minutes ago • bulk commit …
            }
        }
        return count;
    }

    public void printPath(Puzzle initial){
        initial.print();
        for (int i = 0; i < commandCreators.size(); i++) {
            int j = 0;
            boolean found = false;
            while (j < commands.size() && !found) {
                if (commands.get(j).toString().equals(commandCreators.get(i))) {
                    Puzzle next = StateTreeNode.runCommand(commands.get(j), initial);
                    System.out.println("COMMAND: " + commands.get(j).toString());
                    System.out.println();
                    next.print();
                    initial = next;
                    found = true;
                }
                j++;
            }
        }
        System.out.println("GOAL REACHED");
        System.out.println();
    }

    @Override
    @SuppressWarnings("unchecked")
    public int expand() {
        if (val instanceof Puzzle) {
            int n = 0;
            Puzzle t = (Puzzle) val;
            childs.clear();
            for (int i = 0; i < commands.size(); i++) {
                if (commands.get(i).isAllowed(t) && !commands.get(i).opposite().equals(commandCreator)) {
                    childs.add((StateTreeNode<T>) new StateTreeNode<Puzzle>(
                            StateTreeNode.runCommand(commands.get(i), t), pathCount + 1,
                            commands.get(i).toString(), commandCreators));
                    n++;
                }
            }
            return n;
        }
        return 0;
    }

}
```

- **Kelas Queue**

```
You, 6 minutes ago | 1 author (You)
public class BnBQueue extends ArrayList<StateTreeNode<Puzzle>> {
    public BnBQueue(){

    }
    public void enqueue(StateTreeNode<Puzzle> elmt){
        int i = 0;
        int c = elmt.cost();
        while (i < size() && get(i).cost() <= c) {
            i++;
        }
        if (i == size()){
            add(elmt);       You, 11 hours ago • bulk commit …
        } else {
            add(i, elmt);
        }
    }

    public StateTreeNode<Puzzle> dequeue(){
        return remove(0);
    }
}
```

- **Kelas Command**

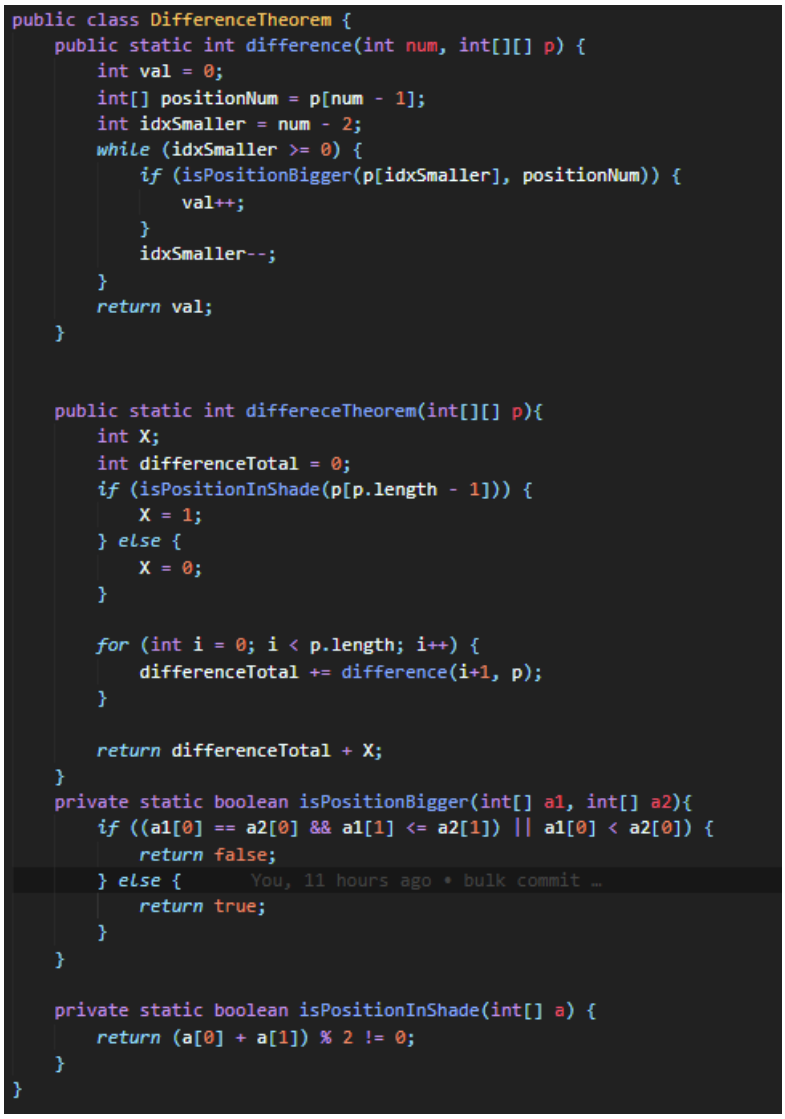
```java
public interface Command<T> {
    public T nextState(T previousState);
    public boolean isAllowed(T previousState);
    public String opposite();
}
```

- **Kelas IOManagement**

```java
public class IOManagement {
    Run | Debug
    public static void main(String[] args) {
        read();
    }

    public static int[][] readFile(String p) {
        int[][] puzzle = new int[16][2];
        try {
            File file = new File(p);
            BufferedReader b = new BufferedReader(new FileReader(file));
            String st;
            int row = 0;
            while ((st = b.readLine()) != null) {
                int[] l = readLine(st);
                for (int i = 0; i < l.length; i++) {
                    puzzle[l[i] - 1] = new int[] { row, i };
                }
                row++;
            }
            b.close();
        } catch (Exception e) {

        }
        return puzzle;

    }
```

```java
    private static int[] readLine(String l) {
        int i = 0;
        String tmp = "";
        int[] val = new int[4];
        int eff = 0;
        String t;

        while (i < l.length() && (t = l.substring(i, i + 1)) != null) {
            if (t.equals(" ") && (tmp.equals(" ") || tmp.equals("?"))) {
                val[eff] = 16;
                tmp = "";
                eff++;
            } else if (t.equals(" ") && tmp.strip().length() != 0) {
                val[eff] = (Integer.parseInt(tmp.strip()));
                tmp = "";
                eff++;
            } else {
                tmp += t;
            }
            if (i == l.length() - 1 && (tmp.equals(" ") || tmp.equals("?"))) {
                val[eff] = 16;       You, 11 hours ago • bulk commit …
            } else if (i == l.length() - 1 && tmp.strip().length() != 0) {
                val[eff] = (Integer.parseInt(tmp.strip()));
            }
            i++;
        }

        return val;
    }

    public static int[][] read() {
        Scanner s = new Scanner(System.in);
        int[][] content = new int[16][2];
        String line;
        int[] rowContent;

        for (int row = 0; row < 4; row++) {
            line = s.nextLine();
            rowContent = readLine(line);
            for (int i = 0; i < rowContent.length; i++) {
                content[rowContent[i] - 1] = new int[] { row, i };
            }
        }

        s.close();
        return content;
    }
}
```

- **Kelas DifferenceTheorem**



```java
public class DifferenceTheorem {
    public static int difference(int num, int[][] p) {
        int val = 0;
        int[] positionNum = p[num - 1];
        int idxSmaller = num - 2;
        while (idxSmaller >= 0) {
            if (isPositionBigger(p[idxSmaller], positionNum)) {
                val++;
            }
            idxSmaller--;
        }
        return val;
    }


    public static int differeceTheorem(int[][] p){
        int X;
        int differenceTotal = 0;
        if (isPositionInShade(p[p.length - 1])) {
            X = 1;
        } else {
            X = 0;
        }

        for (int i = 0; i < p.length; i++) {
            differenceTotal += difference(i+1, p);
        }

        return differenceTotal + X;
    }
    private static boolean isPositionBigger(int[] a1, int[] a2){
        if ((a1[0] == a2[0] && a1[1] <= a2[1]) || a1[0] < a2[0]) {
            return false;
        } else {       You, 11 hours ago • bulk commit …
            return true;
        }
    }

    private static boolean isPositionInShade(int[] a) {
        return (a[0] + a[1]) % 2 != 0;
    }
}
```

Input-Output Program dan Uji Program

Untuk memasukkan puzzle, masukkan sebagai input ketikan ke terminal. Nilai kosong dapat direpresentasikan oleh spasi atau tanda tanya. Pastikan tidak ada spasi yang tidak perlu.

1. **Puzzle *unreachable* 1**

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

```
Welcome to 15-Puzzle Solver
Input your puzzle down here
1 3 4 15
2 ? 5 12
7 6 11 14
8 9 10 13

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

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

'REACHABLE GOAL' THEOREM VALUE: 37

SOLUTION IS UNREACHABLE
```

2. **Puzzle *unreachable* 2**

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

```
Welcome to 15-Puzzle Solver
Input your puzzle down here
2 3 4 ?
1 7 6 5
10 9 15 8
13 12 14 11

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

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

'REACHABLE GOAL' THEOREM VALUE: 31

SOLUTION IS UNREACHABLE
```

3. **Puzzle *reachable* 1**

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

```
Welcome to 15-Puzzle Solver
Input your puzzle down here
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

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

'REACHABLE GOAL' THEOREM VALUE: 16
Solving...
1 2 3 4
5 6 ? 8
9 10 7 11
13 14 15 12
COMMAND: DOWN

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

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

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

ELAPSED TIME: 530000 Nanoseconds
GENERATED STATE: 10 States
```

4. **Puzzle *reachable* 2**

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

```
Welcome to 15-Puzzle Solver
Input your puzzle down here
1 2 3 4
? 6 7 8
5 9 11 12
13 10 14 15

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

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

'REACHABLE GOAL' THEOREM VALUE: 18
Solving...
1 2 3 4
? 6 7 8
5 9 11 12
13 10 14 15
COMMAND: DOWN

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

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

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

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

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

ELAPSED TIME: 603000 Nanoseconds
GENERATED STATE: 13 States
```

5. **Puzzle *reachable* 3**

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

```
Welcome to 15-Puzzle Solver
Input your puzzle down here
1 2 ? 4
6 7 3 8
5 9 11 12
13 10 14 15

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

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

'REACHABLE GOAL' THEOREM VALUE: 22
Solving...
1 2 ? 4
6 7 3 8
5 9 11 12
13 10 14 15
COMMAND: DOWN

1 2 3 4
6 7 ? 8
5 9 11 12
13 10 14 15
COMMAND: LEFT

1 2 3 4
6 ? 7 8
5 9 11 12
13 10 14 15
COMMAND: LEFT

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

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

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

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

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

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


ELAPSED TIME: 613100 Nanoseconds
GENERATED STATE: 21 States
```

Source Code

https://github.com/averrous-s/Tucil3_13520100

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		✓