

Tugas Kecil 3 IF2211 Strategi Algoritma

Semester II tahun 2021/2022

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

Disusun oleh:

Rayhan Kinan Muhannad

13520065



PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2022

1. Penjelasan Algoritma *Branch and Bound*

Strategi algoritma *branch and bound* merupakan suatu metode penyelesaian persoalan optimisasi kombinatorial dengan mengikuti beberapa fungsi heuristik (*best first rule*) untuk mendapatkan solusi optimum global. Umumnya, jika persoalan optimisasi kombinatorial diselesaikan menggunakan strategi *brute force*, solusi tersebut memiliki kompleksitas waktu eksponensial serta diharuskan untuk mengeksplorasi setiap kemungkinan dari ruang solusi persoalan. Hal yang sama mungkin terjadi pada solusi persoalan yang dicari menggunakan strategi *backtracking*. Pada *worst case scenario*, strategi algoritma *backtracking* tidak mengalami *bound* pada iterasi setiap kombinasi solusi. Oleh karena itu, dibutuhkan strategi agar subsolusi yang terpilih pada setiap iterasinya mengarah atau semakin mendekati solusi. Hal tersebut dapat diatasi dengan menggunakan strategi algoritma *branch and bound*, lebih spesifiknya adalah algoritma *informed search*. Terdapat dua algoritma yang dikategorikan sebagai algoritma *informed search*, yaitu *best-first search* serta *A* search*. Kedua algoritma tersebut sama-sama menggunakan fungsi heuristik dalam perhitungan *cost* dari setiap subsolusi yang diambil, tetapi pada *A* search algorithm* terdapat fungsi perhitungan *cost* yang diambil dari *root node* hingga *node* subsolusi tersebut. Fungsi heuristik yang digunakan pada *informed search* haruslah terbukti secara matematis bahwa solusi yang dihasilkan oleh algoritma *informed search* optimum global untuk setiap persoalan yang diberikan. Oleh karena itu, umumnya untuk setiap persoalan sudah terdapat fungsi heuristik yang paling optimum yang telah didefinisikan terlebih dahulu.

Terdapat beberapa keuntungan dan kerugian dari masing-masing algoritma *best-first search* maupun *A* search*. Pada *best-first search*, kompleksitas waktu serta kompleksitas ruang yang dibutuhkan oleh algoritma untuk mencari solusi paling optimum relatif lebih baik dibandingkan menggunakan *A* search*, tetapi solusi yang dihasilkannya tersebut tidak selalu dipastikan solusi paling optimum. Pada *A* search*, meskipun kompleksitas waktu serta kompleksitas ruang yang dibutuhkan oleh algoritma untuk mencari solusi paling optimum relatif lebih buruk dibandingkan menggunakan *best-first search*, tetapi dengan menggunakan fungsi heuristik yang tepat, algoritma *A* search* dapat menghasilkan solusi paling optimum untuk setiap persoalan.

Berikut merupakan langkah-langkah algoritma *A* search* dengan menggunakan strategi *branch and bound*:

1. Masukkan *node* akar ke dalam antrian *Q*. Jika *node* akar sama dengan *node* solusi, maka solusi telah ditemukan dan stop algoritma *branch and bound*.

2. Jika Q kosong, maka solusi tidak ditemukan dan stop algoritma *branch and bound*.
3. Jika Q tidak kosong, pilih salah satu *node* dari antrian Q dengan indeks i , dimana *node* dengan indeks i memiliki nilai *cost* $c(\text{node}[i])$ paling kecil. Jika terdapat beberapa *node* dengan indeks i yang memenuhi, pilih salah satu secara sembarang.
4. Jika *node* dengan indeks i sama dengan *node* solusi, maka solusi telah ditemukan dan stop algoritma *branch and bound*. Jika *node* dengan indeks i bukan merupakan *node* solusi, maka bangkitkan semua *node* anak-anaknya.
5. Jika *node* dengan indeks i tidak mempunyai *node* anak, maka kembali ke langkah 2. Jika *node* dengan indeks i mempunyai *node* anak, maka untuk setiap *node* anak hitung nilai *cost* $c(\text{node}_{\text{anak}[i]})$ dan masukkan semua *node* anak tersebut ke dalam antrian Q .
6. Kembali ke langkah 2.

Catatan

Nilai *cost* pada A^* search dapat dihitung dengan menggunakan taksiran:

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

- Fungsi c merupakan fungsi *cost* untuk *node* P
- Fungsi f merupakan fungsi panjang lintasan dari *node*
- Fungsi g merupakan fungsi heuristik taksiran panjang P ke *node* solusi

Untuk menyelesaikan persoalan 15-puzzle menggunakan algoritma *branch and bound*, terlebih dahulu penulis mendefinisikan *state* dari setiap *node* yang akan dievaluasi pada *search tree*. *State* yang terdefinisi pada persoalan 15-puzzle dapat diinferensikan dari lokasi setiap ubin yang terdapat pada ubin puzzle. Kemudian, penulis juga harus mendefinisikan aksi yang dapat dilakukan pada ubin, yaitu UP, RIGHT, DOWN, dan LEFT. Aksi-aksi tersebut bekerja dengan menggerakkan *tile* kosong sesuai dengan arah yang ditentukan. Terakhir, penulis mendefinisikan *state* tujuan yang dijadikan *node* solusi pada persoalan. *State* tujuan dari persoalan 15-puzzle ini adalah *tile* terurut dari 1 – 15 dari atas kiri hingga kanan bawah.

Agar persoalan 15-puzzle dapat diselesaikan, maka terlebih dahulu dilakukan penginferensian mengenai apakah *state* yang dimiliki oleh puzzle dapat mencapai *node* solusi (*reachable goal*). Hal tersebut dapat dicari dengan cara berikut:

$$\left(\sum_{i=1}^{16} KURANG(i) + X \right) \bmod 2 = \begin{cases} 0, & \text{status tujuan dapat tercapai} \\ 1, & \text{status tujuan tidak dapat tercapai} \end{cases}$$

- $KURANG(i)$: banyaknya ubin bernomor j sedemikian sehingga $j < i$ dan $POSISI(j) > POSISI(i)$
- $POSISI(i)$: posisi ubin bernomor i pada susunan yang diperiksa
- $X = (POSISI(i).x + POSISI(i).y) \bmod 2$

Selain *reachable goal*, penulis juga harus mendefinisikan fungsi heuristik yang digunakan untuk menghitung *cost* dari *node* hidup. Terdapat beberapa fungsi heuristik yang dapat digunakan oleh penulis, tetapi dikarenakan terdapat pada spek tugas, penulis memiliki fungsi heuristik “jumlah ubin tidak kosong yang tidak berada pada tempat sesuai susunan akhir (*goal state*)”. Didefinisikan fungsi *cost* pada persoalan 15-puzzle sebagai berikut:

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

- Fungsi c merupakan fungsi *cost* untuk *node* M pada persoalan 15-puzzle
- Fungsi f merupakan fungsi panjang lintasan dari *node* M yang telah dilalui pada persoalan 15-puzzle
- Fungsi g merupakan fungsi heuristik “jumlah ubin tidak kosong pada *node* M yang tidak berada pada tempat sesuai susunan akhir M_0 ”.

Dalam implementasinya, penulis membuat beberapa *class* sebagai representasi *node* serta *search tree* yang terbentuk saat melakukan *branch and bound*. Berikut merupakan nama-nama *class* yang didefinisikan oleh penulis beserta penjelasan *attribute* beserta *method* yang terkandung:

- *Class* PositionMatrix:
 - *nRow*: *constant attribute* yang berisi jumlah *row* yang terdapat pada *grid*.
 - *nCol*: *constant attribute* yang berisi jumlah *column* yang terdapat pada *grid*.
 - *moveDir*: *constant attribute* yang berisi aksi – aksi yang terdefinisi pada persoalan beserta artinya.
 - *matrix*: berisi data *state* dari puzzle.
 - *prevPosition*: berisi *parent node* dari *node*.
 - *currentCost*: berisi *cost* dari *node*.
 - *currentLength*: berisi *length* yang telah ditempuh oleh *node*.

- indexKosong: berisi indeks dari ubin kosong.
 - visitedNodes: *static attribute* yang berisi *node – node* yang telah dikunjungi ketika *branch and bound*.
 - getEmptyMatrix: *method* yang menghasilkan *node* dengan data kosong.
 - fromFile: *method* yang dapat membaca *binary string* dan membentuk *node* baru.
 - Overloading constructor: *method* yang berfungsi menginisialisasi *attribute*.
 - getKurang: *method* yang berfungsi untuk mendapatkan nilai $KURANG(i)$.
 - getSumKurang: *method* yang berfungsi untuk mendapatkan nilai $\sum_{i=1}^{16} KURANG(i)$.
 - getIndexKosong: *method* yang berfungsi untuk mendapatkan indeks dari *grid* kosong.
 - getPerbedaanUbin: *method* yang berfungsi untuk mendapatkan perbedaan ubin *node* terhadap *node* target.
 - getX: *method* yang berfungsi untuk mendapatkan nilai X .
 - getTotalCost: *method* yang berfungsi untuk mendapatkan nilai total *cost*.
 - isReachable: *method* yang berfungsi untuk mencari tahu apakah *node* tersebut dapat menghasilkan *node* tujuan.
 - Overloading operator= : *method* yang berfungsi untuk menyamakan dua *node*.
 - Overloading operator< : *method* yang berfungsi untuk membandingkan dua *node*.
 - Overloading operator<< : *method* yang berfungsi untuk menambahkan *child node* dengan memasukkan aksi.
- *Class PositionTree*:
 - move: *constant attribute* yang berisi aksi – aksi yang terdefinisi pada persoalan.
 - targetPosition: *constant attribute* yang berisi *node* tujuan.
 - first: *root node* dari *search tree*.

- Overloading constructor: *method* yang berfungsi menginisialisasi *attribute*.
- branchAndBound: *method* yang berfungsi untuk melakukan algoritma *branch and bound* pada *search tree*.
- calculate: *method* yang berfungsi untuk menghitung seluruh *output* pada program.

2. Source Code

a. File puzzle.py (Program Utama)

```
from queue import PriorityQueue
from time import time

import numpy as np

class PositionMatrix:
    # CONSTANT ATTRIBUTE
    nRow = 4
    nCol = 4
    moveDir = {"UP" : (-1, 0), "RIGHT" : (0, 1), "DOWN" : (1, 0), "LEFT" : (0, -1)}

    # STATIC ATTRIBUTE
    visitedNodes = {}

    # STATIC METHOD
    def getEmptyMatrix():
        matrix = np.array([[None for j in range(PositionMatrix.nCol)] for i in range(PositionMatrix.nRow)])

        return matrix

    def fromFile(rawString):
        # INITIALIZE legalElement
        legalElement = [str(i) for i in range(1, PositionMatrix.nRow * PositionMatrix.nCol + 1)]

        # INITIALIZE matrix
        matrix = np.empty((PositionMatrix.nRow, PositionMatrix.nCol), int)

        listOfRow = rawString.split("\r\n")

        if len(listOfRow) != PositionMatrix.nRow:
            raise Exception(f"Jumlah baris pada file txt harus berjumlah 4! Jumlah baris pada file adalah {len(listOfRow)}.")

        else:
            for i in range(len(listOfRow)):
                listOfElement = listOfRow[i].split(" ")

                if len(listOfElement) != PositionMatrix.nCol:
                    raise Exception(f"Jumlah kolom pada file txt harus berjumlah 4! Terdapat kolom pada file dengan jumlah {len(listOfElement)}.")

                else:
                    for j in range(len(listOfElement)):
```

```

        if listOfElement[j] not in legalElement:
            raise Exception(f"Terdapat elemen ilegal pada file
txt! Elemen ilegal tersebut adalah {listOfElement[j]}.")

        else:
            legalElement.remove(listOfElement[j])
            matrix[i, j] = int(listOfElement[j])

    PM = PositionMatrix(matrix)

    if not PM.isReachable():
        raise Exception(f"Puzzle tidak dapat diselesaikan! (SUM KURANG
+ X = {PM.getSumKurang() + PM.getX()})")

    else:
        return PM

# CONSTRUCTOR
def __init__(self, data):

    # INITIALIZE matrix
    self.matrix = data

    # INITIALIZE prevPosition
    self.prevPosition = None

    # INITIALIZE currentCost
    self.currentCost = 0

    # INITIALIZE currentLength
    self.currentLength = 0

    # INITIALIZE indexKosong
    self.indexKosong = (-1, -1)

# OPERATION
def getKurang(self, N):
    nilaiKurang = -1

    for i in range(PositionMatrix.nRow):
        for j in range(PositionMatrix.nCol):
            if self.matrix[i, j] == N:
                nilaiKurang = 0
            else:
                nilaiKurang = nilaiKurang + 1 if nilaiKurang != -1 and
self.matrix[i, j] < N else nilaiKurang

    if nilaiKurang == -1:

```



```

        if N != PositionMatrix.nRow * PositionMatrix.nCol:
            raise Exception(f"Tidak terdapat elemen {N} pada matrix!")

        else:
            raise Exception("Tidak terdapat elemen kosong pada matrix!")

    else:
        return nilaiKurang

def getSumKurang(self):
    nilaiSumKurang = 0

    for i in range(PositionMatrix.nRow):
        for j in range(PositionMatrix.nCol):
            nilaiSumKurang += self.getKurang(PositionMatrix.nCol * i + j +
1)

    return nilaiSumKurang

def getIndexKosong(self):
    for i in range(PositionMatrix.nRow):
        for j in range(PositionMatrix.nCol):
            if self.matrix[i, j] == PositionMatrix.nRow *
PositionMatrix.nCol:
                return (i, j)

            raise Exception("Tidak terdapat elemen kosong pada matrix!")

def getPerbedaanUbin(self):
    N = 0

    for i in range(PositionMatrix.nRow):
        for j in range(PositionMatrix.nCol):
            if self.matrix[i, j] != PositionMatrix.nCol * i + j + 1 and
self.matrix[i, j] != PositionMatrix.nRow * PositionMatrix.nCol:
                N += 1

    return N

def getX(self):
    i, j = self.getIndexKosong()

    return (i + j) % 2

def getTotalCost(self):
    return self.currentCost + self.currentLength

def isReachable(self):

```

```

        return (self.getSumKurang() + self.getX()) % 2 == 0

# OPERATOR OVERLOADING
def __eq__(self, other):
    return self.matrix.tobytes() == other.matrix.tobytes()

def __lt__(self, other):
    return self.getTotalCost() <= other.getTotalCost()

def __lshift__(self, move):
    i, j = self.indexKosong
    deltaX, deltaY = PositionMatrix.moveDir[move]

    if i + deltaX < 0 or i + deltaX >= PositionMatrix.nRow or j + deltaY <
0 or j + deltaY >= PositionMatrix.nCol:
        raise IndexError("Invalid move.")

    else:
        # ADD matrix
        other = PositionMatrix(self.matrix.copy())

        isPrevValid = other.matrix[i + deltaX, j + deltaY] ==
PositionMatrix.nRow * (i + deltaX) + (j + deltaY) + 1
        other.matrix[i, j], other.matrix[i + deltaX, j + deltaY] =
other.matrix[i + deltaX, j + deltaY], other.matrix[i, j]
        isFollowingValid = other.matrix[i, j] == PositionMatrix.nCol * i +
j + 1

        if other.matrix.tobytes() in PositionMatrix.visitedNodes:
            return None

        else:
            # ADD prevPosition
            other.prevPosition = self

            # ADD currentCost
            if isPrevValid and not isFollowingValid:
                other.currentCost = self.currentCost + 1
            elif not isPrevValid and isFollowingValid:
                other.currentCost = self.currentCost - 1
            else:
                other.currentCost = self.currentCost

            # ADD currentLength
            other.currentLength = self.currentLength + 1

            # ADD indexKosong
            other.indexKosong = (i + deltaX, j + deltaY)

```

```

        # ADD visitedNodes
        PositionMatrix.visitedNodes[other.matrix.tobytes()] = other

        return other

class PositionTree:
    # CONSTANT ATTRIBUTE
    move = ["UP", "RIGHT", "DOWN", "LEFT"]
    targetPosition = PositionMatrix(np.array([[PositionMatrix.nCol * i + j + 1
for j in range(PositionMatrix.nCol)] for i in range(PositionMatrix.nRow)]))

    # CONSTRUCTOR
    def __init__(self, first):
        if not first.isReachable():
            raise Exception(f"Puzzle tidak dapat diselesaikan! (SUM KURANG + X
= {first.getSumKurang() + first.getX()}")

        else:
            self.first = first
            self.first.currentCost = self.first.getPerbedaanUbin()
            self.first.indexKosong = self.first.getIndexKosong()

    # OPERATION
    def branchAndBound(self):
        rootNode = self.first

        Q = PriorityQueue()

        Q.put(rootNode)
        PositionMatrix.visitedNodes[rootNode.matrix.tobytes()] = rootNode

        currentNode = None

        while not Q.empty():
            currentNode = Q.get()

            if currentNode == PositionTree.targetPosition:
                Q.queue.clear()

            else:
                for move in PositionTree.move:
                    try:
                        childNode = currentNode << move

                        if childNode is not None:
                            Q.put(childNode)

```

```

        except IndexError:
            pass

    result = []
    while currentNode is not None:
        result.insert(0, currentNode.matrix.tolist())
        currentNode = currentNode.prevPosition

    return result

def calculate(self):
    PositionMatrix.visitedNodes = {}

    sumKurangPlusX = self.first.getSumKurang() + self.first.getX()

    startTime = time()
    pathOfMatrix = self.branchAndBound()
    endTime = time()

    numOfNodes = len(PositionMatrix.visitedNodes)
    PositionMatrix.visitedNodes = {}

    executionTime = round(endTime - startTime, 2)

    return (sumKurangPlusX, pathOfMatrix, numOfNodes, executionTime)

```

b. File app.py (GUI)

```

from flask import Flask, render_template, request, jsonify
from puzzle import PositionMatrix, PositionTree
from pyfladesk import init_gui

import json
import secrets
import os
import numpy as np

import sys

app = Flask(__name__)

app.config["SECRET_KEY"] = secrets.token_urlsafe()

uploaded_matrix = json.dumps(PositionTree.targetPosition.matrix.tolist())

# FRONTEND
@app.route("/", methods = ["GET"])
def main_page():
    return render_template("index.html")

```

```

@app.route("/view", methods = ["GET"])
def view_page():
    return render_template("view.html")

# BACKEND
@app.route("/display", methods = ["GET"])
def display_matrix():
    global uploaded_matrix

    try:
        return jsonify(matrix =
PositionMatrix(np.array(json.loads(uploaded_matrix))).matrix.tolist(), nRow =
PositionMatrix.nRow, nCol = PositionMatrix.nCol)

    except KeyError:
        return jsonify(matrix = PositionTree.targetPosition.matrix.tolist(),
nRow = PositionMatrix.nRow, nCol = PositionMatrix.nCol)

@app.route("/upload_txt", methods = ["POST"])
def upload_txt():
    global uploaded_matrix

    try:
        file = request.files["file"]

        uploaded_matrix =
json.dumps(PositionMatrix.fromFile(file.stream.read().decode("ASCII")).matrix.
tolist())

        return "Created", 201

    except Exception as e:
        return str(e), 400

@app.route("/calculate", methods = ["GET"])
def calculate_matrix():
    global uploaded_matrix

    try:
        PT =
PositionTree(PositionMatrix(np.array(json.loads(uploaded_matrix))))
        sumKurangPlusX, pathOfMatrix, numOfNodes, executionTime =
PT.calculate()

        return jsonify(sumKurangPlusX = sumKurangPlusX, pathOfMatrix =
pathOfMatrix, numOfNodes = numOfNodes, executionTime = executionTime, nRow =
PositionMatrix.nRow, nCol = PositionMatrix.nCol)

```

```

except Exception as e:
    return str(e), 400

@app.route("/upload_json", methods = ["POST"])
def process_matrix():
    global uploaded_matrix

    try:
        data = request.get_json()

        uploaded_matrix = json.dumps(data)

        return "Created", 201

    except Exception as e:
        return str(e), 400

if __name__ == "__main__":
    icondir = os.path.join(os.path.dirname(__file__),
"static/images/logo.png")

    init_gui(app, port = 3000, window_title = "15 Puzzle Solver", icon =
icondir)

```

c. File index.html (GUI)

```

<!DOCTYPE html>
<html lang="id">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>15 Puzzle Solver</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles/stylessheet.css') }}">
    <link rel="icon" type="image/png" href="{{ url_for('static',
filename='images/logo.png') }}">
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scrip
t>
    <script type="text/javascript" src="{{ url_for('static',
filename='js/index.js') }}"></script>
  </head>
  <body>
    <header>
      <h1 class="header-title">15 Puzzle Solver</h1>
    </header>
    <div class="container">
      <div class="board"></div>

```

```

        <form class="submission-form" enctype="multipart/form-data">
            <input type="file" name="file" class="file-input-button"
accept=".txt" required>
            <button type="submit" class="file-submit-
button">Upload</button>
        </form>
        <button type="button" class="calculate-button">Calculate</button>
    </div>
    <footer>
        <p class="footer-label">Made by 13520065 Rayhan Kinan Muhannad</p>
    </footer>
</body>
</html>

```

d. File view.html (GUI)

```

<!DOCTYPE html>
<html lang="id">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>15 Puzzle Solver</title>
        <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='styles/stylesheet.css') }}">
        <link rel="icon" type="image/png" href="{{ url_for('static',
filename='images/logo.png') }}">
        <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scrip
t>
        <script type="text/javascript" src="{{ url_for('static',
filename='js/view.js') }}"></script>
    </head>
    <body>
        <header>
            <h1 class="header-title">15 Puzzle Solver</h1>
        </header>
        <div class="container">
            <div class="board">
                
            </div>
            <p class="nilai-kurang-x-label"></p>
            <p class="jumlah-node-label"></p>
            <p class="waktu-eksekusi-label"></p>
        </div>
        <footer>
            <p class="footer-label">Made by 13520065 Rayhan Kinan Muhannad</p>
        </footer>
    </body>

```

```
</html>
```

e. File index.js (GUI)

```
let board, zx, zy

const getPossibles = (nRow, nCol) => {
  let ii, jj, cx = [-1, 0, 1, 0], cy = [0, -1, 0, 1]
  possibles = []

  for (let i = 0; i < 4; i++) {
    ii = zx + cx[i]
    jj = zy + cy[i]

    if (ii >= 0 && ii < nRow && jj >= 0 && jj < nCol) {
      possibles.push({ x: ii, y: jj })
    }
  }

  return possibles
}

const updateBtns = (nRow, nCol) => {
  for (let i = 0; i < nRow; i++) {
    for (let j = 0; j < nCol; j++) {
      const id = `btn${i * nCol + j}`

      if (board[i][j] < nRow * nCol) {
        $(`#${id}`).html(board[i][j])
        $(`#${id}`).attr("class", "button")
      } else {
        $(`#${id}`).html("")
        $(`#${id}`).attr("class", "empty")
      }
    }
  }
}

const btnHandle = (event) => {
  let p = -1

  const c = event.target.i, r = event.target.j, nRow = event.target.nRow,
  nCol = event.target.nCol
  const possibles = getPossibles(nRow, nCol)

  for (let i = 0; i < possibles.length; i++) {
    if (possibles[i].x == c && possibles[i].y == r) {
      p = i
    }
  }
}
```



```

        break
    }
}

if (p > -1) {
    const t = possibles[p]

    board[zx][zy] = board[t.x][t.y]
    zx = t.x
    zy = t.y
    board[zx][zy] = nRow * nCol

    updateBtns(nRow, nCol)
}
}

const createBoard = (newBoard, nRow, nCol) => {
    board = new Array(nRow)

    for (let i = 0; i < nRow; i++) {
        board[i] = new Array(nCol)
        for (let j = 0; j < nCol; j++) {
            if (newBoard[i][j] == nRow * nCol) {
                zx = i
                zy = j
            }

            board[i][j] = newBoard[i][j]
        }
    }
}

const createBtns = (nRow, nCol) => {
    for (let i = 0; i < nRow; i++) {
        for (let j = 0; j < nCol; j++) {
            const b = document.createElement("button")

            b.id = `btn${i * nCol + j}`
            b.i = i
            b.j = j
            b.nRow = nRow
            b.nCol = nCol

            $(b).click((event) => {
                btnHandle(event)
            })

            $(b).appendTo(".board")
        }
    }
}

```

```

    }
  }
}

$(document).ready(() => {
  $.ajax({
    type: "GET",
    url: "/display",
    contentType: "application/json; charset=utf-8",
    async: true,
    error: (jqXHR) => {
      alert(jqXHR.responseText)
    },
    success: (data) => {
      createBtns(data.nRow, data.nCol)
      createBoard(data.matrix, data.nRow, data.nCol)
      updateBtns(data.nRow, data.nCol)
    }
  })

  $(".submission-form").submit((event) => {
    const formData = new FormData(event.target)

    $.ajax({
      data: formData,
      type: "POST",
      url: "/upload_txt",
      contentType: false,
      processData: false,
      async: true,
      error: (jqXHR) => {
        alert(jqXHR.responseText)
      },
      success: () => {
        window.location.reload()
      }
    })
  })

  $(".calculate-button").click(() => {
    $.ajax({
      type: "POST",
      url: "/upload_json",
      async: true,
      data: JSON.stringify(board),
      contentType: "application/json",
      error: (jqXHR) => {
        alert(jqXHR.responseText)
      }
    })
  })
})

```

```

    },
    success: () => {
        window.location.replace("/view")
    }
  })
})
})
})

```

f. File view.js (GUI)

```

let board, zx, zy

const getPossibles = (nRow, nCol) => {
  let ii, jj, cx = [-1, 0, 1, 0], cy = [0, -1, 0, 1]
  possibles = []

  for (let i = 0; i < 4; i++) {
    ii = zx + cx[i]
    jj = zy + cy[i]

    if (ii >= 0 && ii < nRow && jj >= 0 && jj < nCol) {
      possibles.push({ x: ii, y: jj })
    }
  }

  return possibles
}

const updateBtns = (nRow, nCol) => {
  for (let i = 0; i < nRow; i++) {
    for (let j = 0; j < nCol; j++) {
      const id = `btn${i * nCol + j}`

      if (board[i][j] < nRow * nCol) {
        $(`#${id}`).html(board[i][j])
        $(`#${id}`).attr("class", "button")
      } else {
        $(`#${id}`).html("")
        $(`#${id}`).attr("class", "empty")
      }
    }
  }
}

const btnHandle = (event) => {
  let p = -1

```

```

    const c = event.target.i, r = event.target.j, nRow = event.target.nRow,
nCol = event.target.nCol
    const possibles = getPossibles(nRow, nCol)

    for (let i = 0; i < possibles.length; i++) {
        if (possibles[i].x == c && possibles[i].y == r) {
            p = i
            break
        }
    }

    if (p > -1) {
        const t = possibles[p]

        board[zx][zy] = board[t.x][t.y]
        zx = t.x
        zy = t.y
        board[zx][zy] = nRow * nCol

        updateBtns(nRow, nCol)
    }
}

const createBoard = (newBoard, nRow, nCol) => {
    board = new Array(nRow)

    for (let i = 0; i < nRow; i++) {
        board[i] = new Array(nCol)
        for (let j = 0; j < nCol; j++) {
            if (newBoard[i][j] == nRow * nCol) {
                zx = i
                zy = j
            }

            board[i][j] = newBoard[i][j]
        }
    }
}

const createBtns = (nRow, nCol) => {
    for (let i = 0; i < nRow; i++) {
        for (let j = 0; j < nCol; j++) {
            const b = document.createElement("button")

            b.id = `btn${i * nCol + j}`
            b.i = i
            b.j = j
            b.nRow = nRow

```

```

        b.nCol = nCol

        $(b).click((event) => {
            btnHandle(event)
        })

        $(b).appendTo(".board")
    }
}

const updateAttribute = (sumKurangPlusX, numOfNodes, executionTime) => {
    $(".nilai-kurang-x-label").html(`Nilai dari KURANG(i) + X :
    ${sumKurangPlusX}`)
    $(".jumlah-node-label").html(`Jumlah simpul yang dibangkitkan :
    ${numOfNodes}`)
    $(".waktu-eksekusi-label").html(`Waktu eksekusi : ${executionTime} s`)
}

$(document).ready(() => {
    $.ajax({
        type: "GET",
        url: "/calculate",
        contentType: "application/json; charset=utf-8",
        async: true,
        error: (jqXHR) => {
            alert(jqXHR.responseText)
        },
        success: (data) => {
            // DELETE LOADING GIF
            $(".loading-gif").remove()

            // FIRST ITERATION
            createBtns(data.nRow, data.nCol)
            createBoard(data.pathOfMatrix[0], data.nRow, data.nCol)
            updateBtns(data.nRow, data.nCol)

            // OTHER ITERATION
            let index = 1
            var interval = setInterval(() => {
                if (index == data.pathOfMatrix.length) {
                    updateAttribute(data.sumKurangPlusX, data.numOfNodes,
data.executionTime)
                    clearInterval(interval)
                } else {
                    createBoard(data.pathOfMatrix[index], data.nRow,
data.nCol)

```

```

        updateBtns(data.nRow, data.nCol)
        index++
    }
    }, 500)
}
})
})

```

g. File stylesheet.css (GUI)

```

/* GLOBAL VARIABLE */
html, body {
    padding: 0;
    margin: 0;
    padding-top: 8vh;
    background: #222;
    color: #feffff;
    font-size: 100%;
    font-family: 'Helvetica', sans-serif;
    letter-spacing: 1px;
}

button {
    cursor: pointer;
}

/* HEADER */
header {
    position: fixed;
    height: 60px;
    top: 0;
    width: 100%;
    background-color: #111;
    border-top: 5px solid #111;
    border-bottom: 5px solid #111;
    display: flex;
    flex-direction: row;
}

.header-title {
    font-size: 20px;
    margin-left: 10px;
    margin-top: 20px;
    font-weight: 600;
}

/* BOARD */

```

```

.txt {
  color: #feffff;
  text-align: center;
  font-size: 5vh;
}

.board {
  padding: 0;
  margin: auto;
  width: 33vh;
  height: 33vh;
}

.button, .empty {
  border: 0;
  font-size: 3.5vh;
  margin: 0.5vh;
  padding: 0;
  height: 6vh;
  width: 7.25vh;
  line-height: 5vh;
  vertical-align: middle;
  background: #feffff;
  text-align: center;
  border-radius: 3px;
  cursor: pointer;
  float: left}

.empty {
  background: #333;
  border: 1px solid #111
}

/* CONTAINER */
.container {
  padding: 0;
  margin: 0 auto;
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
}

.file-submit-button, .calculate-button {
  width: 120px;
  height: 45px;
  margin-top: 6px;
  border-radius: 30px;
}

```

```

font-weight: 600;
background-color: #ffffff;
color: #111;
cursor: pointer;
outline: none;
border: 2px solid #ffffff;
font-size: 14px;
}

.file-submit-button:hover, .calculate-button:hover {
background-color: #111;
color: #ffffff;
transition: all 0.25s ease;
}

.file-input-button::file-selector-button {
cursor: pointer;
}

.loading-gif {
display: block;
margin-left: auto;
margin-right: auto;
width: 50%;
height: 50%;
}

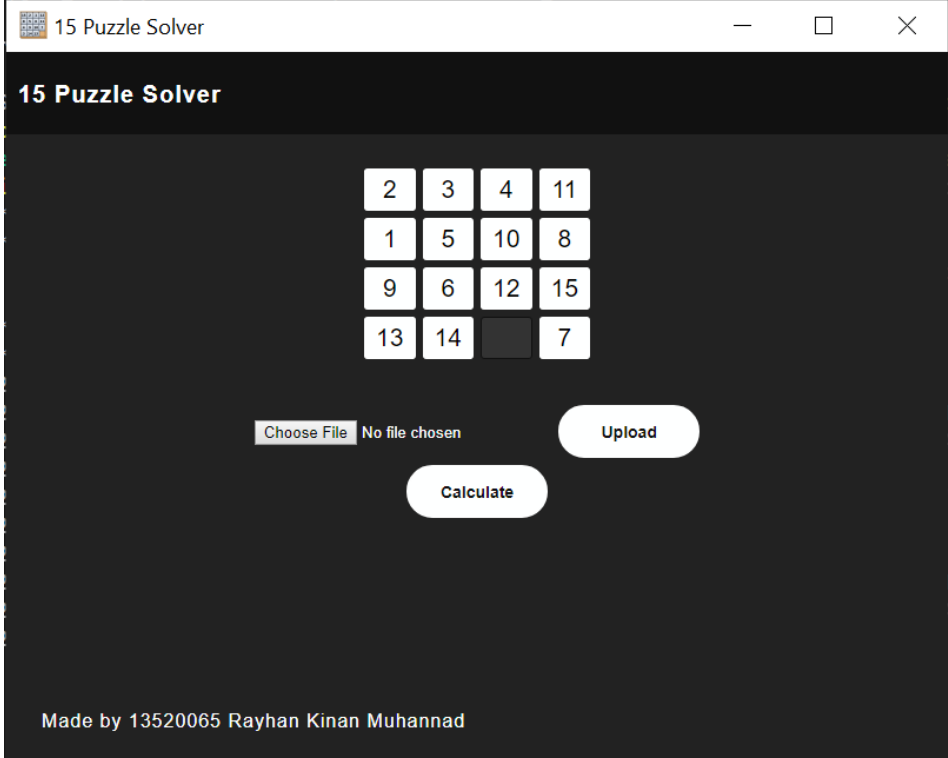
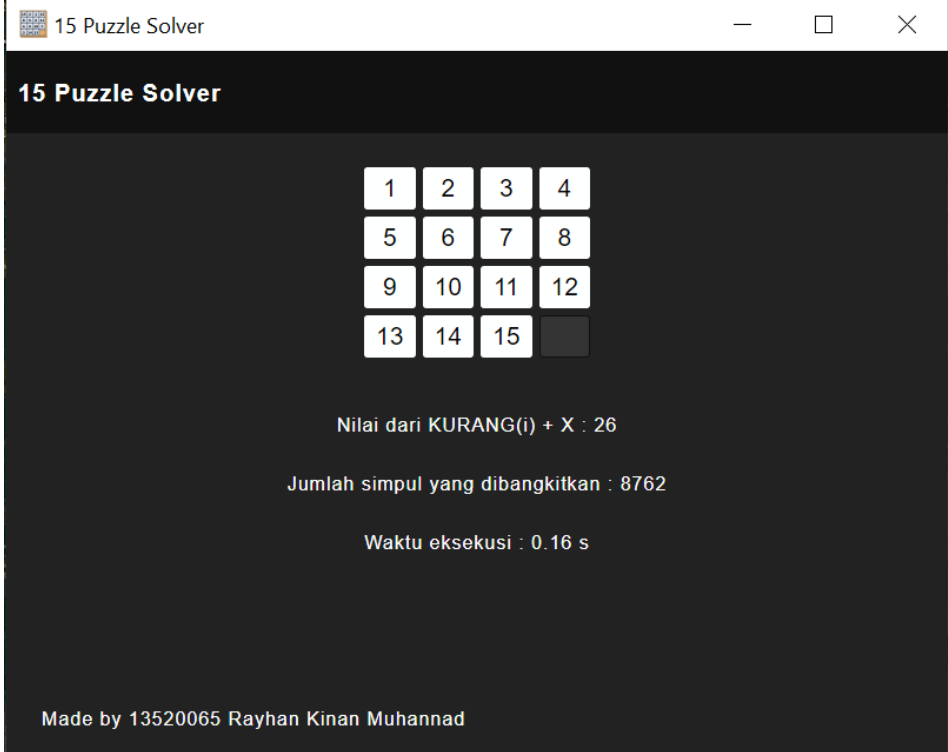
/* FOOTER */
footer {
position: fixed;
padding: 10px 10px 0px 10px;
bottom: 0;
width: 100%;
height: 40px;
}

.footer-label {
margin: 0 20px;
}

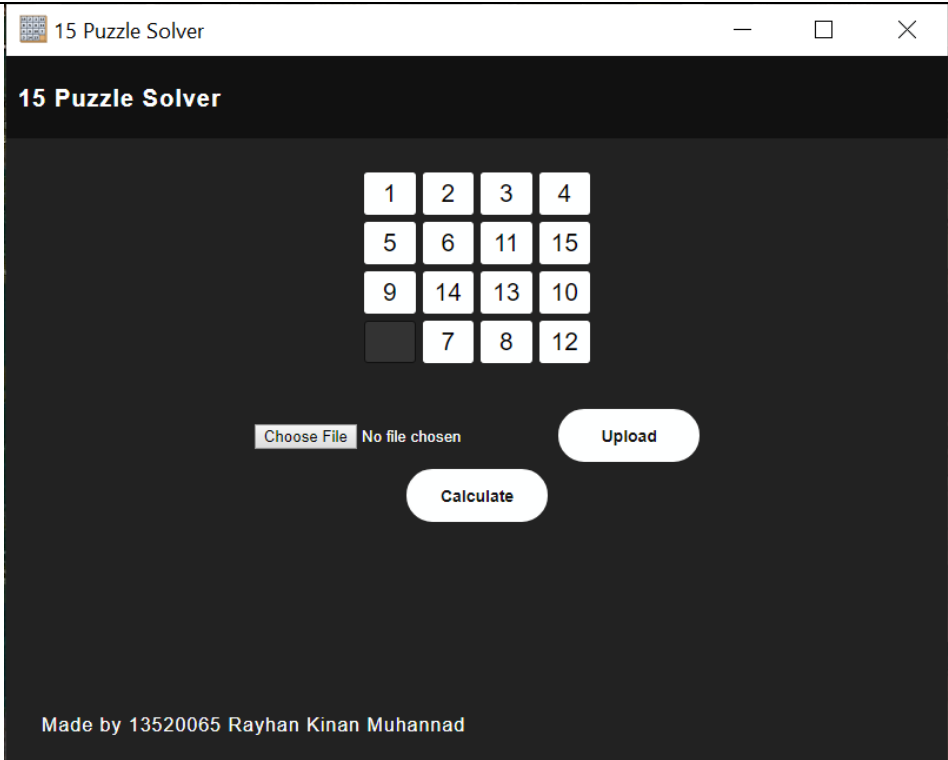
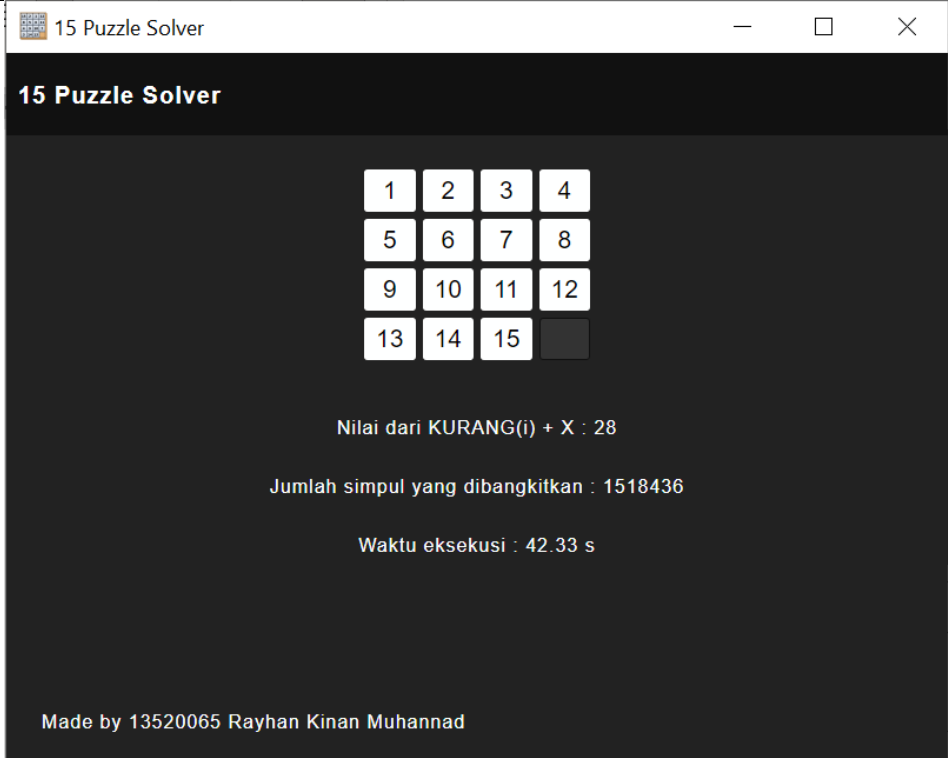
```


3. Screenshot dari *Input* dan *Output* Program

a. File bisa1.txt

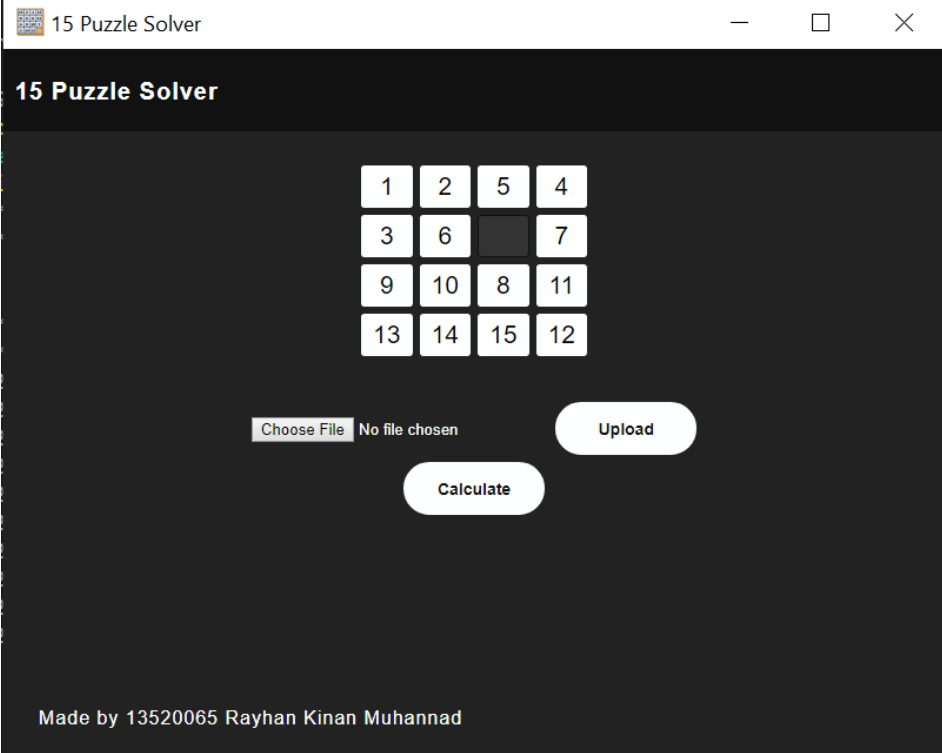
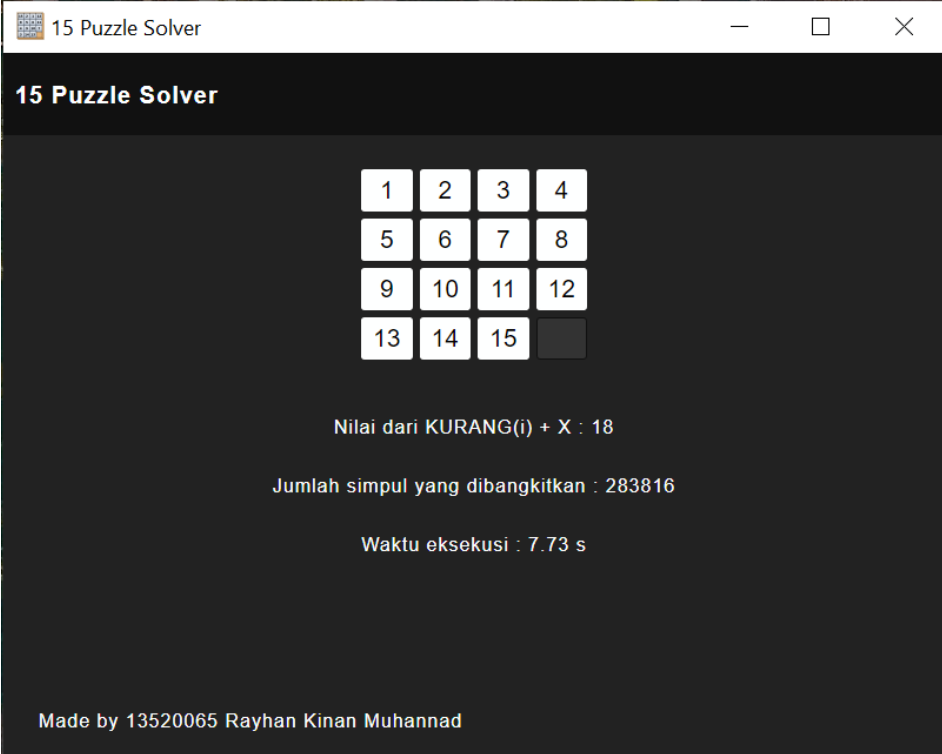
	Screenshot
Input	
Output	

b. File bisa2.txt

	Screenshot
Input	
Output	

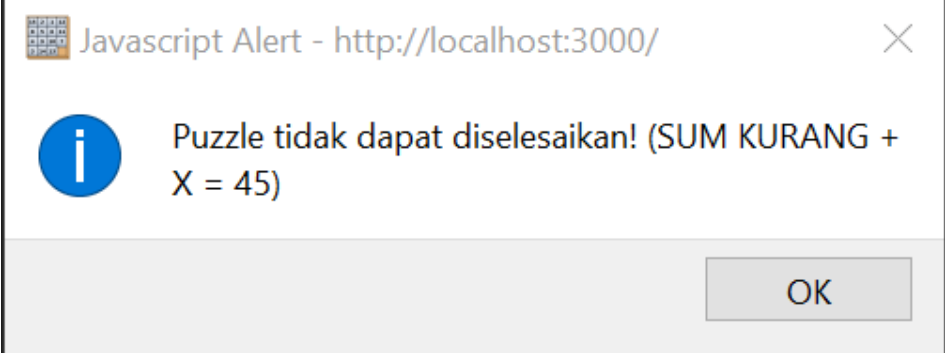
c. File bisa3.txt

	Screenshot
--	------------

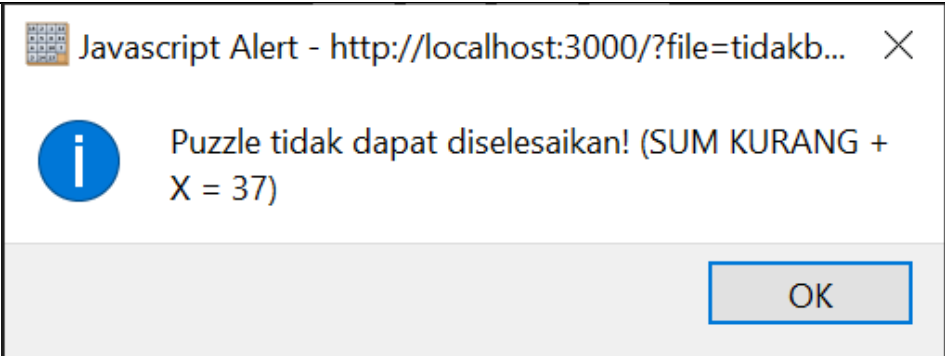
Input	
Output	

d. File tidakbisa1.txt

	Screenshot
--	------------

Input	
Output	-

e. File tidakbisa2.txt

	<i>Screenshot</i>
Input	
Output	-

4. Contoh Instansiasi 5 buah Persoalan 15-Puzzle

a. File bisa1.txt (Bisa Diselesaikan)

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

b. File bisa2.txt (Bisa Diselesaikan)

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

c. File bisa3.txt (Bisa Diselesaikan)

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

d. File tidakbisa1.txt (Tidak Bisa Diselesaikan)

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

e. File tidakbisa2.txt (Tidak Bisa Diselesaikan)

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

5. Link Google Drive dan Repository GitHub

a. Repository GitHub

<https://github.com/rayhankinan/15-puzzle-solver>

6. Checklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi.	√	
2. Program berhasil <i>running</i> .	√	
3. Program dapat menerima <i>input</i> dan menuliskan <i>output</i> .	√	
4. Luaran sudah benar untuk semua data uji.	√	
5. Bonus: Program dibuat secara interaktif dengan graphical user interface (GUI), dapat menampilkan gambar 15-puzzle grafis yang menarik, dan dapat menampilkan pergeseran ubin.	√	

7. Referensi

- https://rosettacode.org/wiki/15_puzzle_game
- <https://www.tutorialspoint.com/flask/index.htm>
- <https://www.geeksforgeeks.org/check-instance-15-puzzle-solvable/>
- <https://www.geeksforgeeks.org/8-puzzle-problem-using-branch-and-bound/>
- <https://medium.com/@prestonbjensen/solving-the-15-puzzle-e7e60a3d9782>
- <https://www.geeksforgeeks.org/branch-and-bound-algorithm/>
- <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf>
- <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf>