**TUGAS KECIL**

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

**LAPORAN**

**Diajukan sebagai salah satu tugas mata kuliah IF2211 Strategi Algoritma pada**

**Semester II**

**Tahun Akademik 2021-2022**

**oleh**

**Owen Christian Wijaya 13520124**

**A picture containing text

Description automatically generated**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2022**

DAFTAR ISI

[BAB I. ALGORITMA *DIVIDE AND CONQUER* 3](#_Toc96887029)

[BAB II. *SOURCE PROGRAM* DALAM BAHASA PYTHON 4](#_Toc96887030)

[2.1 process.py 4](#_Toc96887031)

[2.2 utils.py 7](#_Toc96887032)

[2.3 main.py 10](#_Toc96887033)

[BAB III. PENGUJIAN 13](#_Toc96887034)

[3.1 Sepal Length vs Sepal Width 13](#_Toc96887035)

[3.2 Petal Length vs Petal Width 13](#_Toc96887036)

[3.3 Color Intensity vs Hue 14](#_Toc96887037)

[3.4 Ash vs Alcalinity of Ash 14](#_Toc96887038)

[3.5 Alcohol vs Malic Acid 14](#_Toc96887039)

[3.6 OD280/OD315 of Diluted Wines vs Proline 15](#_Toc96887040)

[3.7 Mean Smoothness vs Mean Compactness 15](#_Toc96887041)

[3.8 Worst Concavity vs Worst Concave Points 15](#_Toc96887042)

[3.9 pixel\_0\_2 vs pixel\_0\_3 16](#_Toc96887043)

[3.10 pixel\_6\_3 vs pixel\_6\_4 16](#_Toc96887044)

[BAB IV. *REPOSITORY* 16](#_Toc96887045)

# BAB I. ALGORITMA *BRANCH AND BOUND*

Dalam tugas kecil ini, algoritma *branch and bound* digunakan untuk menentukan langkah-langkah optimal untuk menyelesaikan sebuah permasalahan 15-puzzle. Dalam permainan 15-puzzle, pemain harus menyelesaikan sebuah puzzle untuk mencapai posisi akhir yang diinginkan dengan empat buah gerakan. Algoritma *branch and bound* akan melakukan pencarian terhadap semua kemungkinan gerakan, dan menentukan urutan yang diperlukan untuk menyelesaikan permasalahan tersebut.

Saat melakukan eksekusi, program akan menerima input puzzle dari pengguna baik secara langsung atau dari plainteks dan melakukan validasi. Apabila puzzle tersebut sudah mencapai posisi yang diinginkan, maka program tidak akan melakukan pencarian. Sebaliknya, program akan memulai algoritma dengan melakukan perhitungan nilai “kurang” per ubin. Nilai “kurang” menandakan jumlah ubin yang berada pada posisi di atas ubin tertentu namun mempunyai nilai lebih kecil. Setelah itu, nilai jumlah ubin ini akan dijumlahkan. Posisi *empty space* (dirujuk sebagai ES lebih lanjut di dokumen ini) akan diperhitungkan juga untuk menentukan apakah puzzle tersebut dapat diselesaikan. Apabila jumlah nilai “kurang” dan nilai posisi ES puzzle bernilai ganjil, maka program tidak akan menyelesaikan puzzle. Sebaliknya, program akan melakukan inisialisasi algoritma.

Program akan menginisialisasikan sebuah *priority queue* untuk menampung *state* puzzle dan *cost* yang dibutuhkan untuk mencapai *state* tersebut. *Cost* diperhitungkan dari kedalaman *state* tersebut dan jumlah ubin yang terletak di posisi yang salah pada puzzle. Setelah itu, *state* tersebut akan di-*enqueue* ke dalam *priority queue* beserta *cost* dan informasi tentang arah yang diambil. *Priority queue* melakukan *enqueue* berdasarkan prioritas *cost. Cost* yang lebih kecil akan diurutkan di posisi yang lebih depan. Setelah semua selesai diperiksa, program akan melakukan *dequeue* untuk melakukan pengecekan terhadap *state* tersebut. Program akan melakukan *enqueue* terhadap *state* yang menggunakan semua kemungkinan arah kecuali arah yang diambil sebelumnya. Misalkan, apabila *state* yang sedang diperiksa diperoleh dari arah UP, maka program tidak akan meng-*enqueue* state DOWN karena akan kembali ke posisi awal. Proses ini akan terus dilakukan sampai antara *priority queue* kosong atau ditemukan *state* selesai.

Setelah proses pencarian selesai, program akan menghubungkan *state-state* yang menghasilkan jawaban dengan melihat nilai ID dari *state* yang ada. Setelah itu, program akan menampilkan hasil urutan, jumlah simpul yang dibangkitkan, dan waktu eksekusi. Untuk GUI, program akan menerima *array* berisi *states* yang kemudian di-*update* per 1 detik untuk menampilkan animasi pergerakan.

# BAB II. *SOURCE PROGRAM* DALAM BAHASA PYTHON

## 2.1 main.py

File CLI untuk menggunakan program via *command line*.

import fparser as fp

import puzzle as pc

import algo

def main():

    while (True):

        print("\n\nSelect your desired input method: ")

        print("[1] Text file")

        print("[2] Input by user")

        print("[0] Exit")

        try:

            option = int(input("| >> "))

            if (option == 1):

                print("\n[SELECTED] Text file")

                print("Input your filename (without .txt)|")

                print("[IMPORTANT] File must be included in the test folder!")

                fname = input("| >> ")

                buffer = fp.parseText(fname)

            elif (option == 2):

                buffer = fp.parseInput()

            elif (option == 0):

                print("Exiting program...")

                break

        except Exception as e:

            print(e)

            continue

        p = pc.Puzzle(buffer)

        \_, res, outputMessage = algo.solve(p)

        for i in range(len(res)):

            res[i][0].show()

            print("Step {} | Command: {} \n".format(i, res[i][1]))

        print(outputMessage)

main()

2.2 gui.py  
File berisi GUI untuk digunakan dalam bentuk *desktop app* (dibangun menggunakan TKinter)

from tkinter import \*

from tkinter import messagebox

from algo import solve

from puzzle import \*

from fparser import \*

import time

puzzle\_arr = []

def solveClick():

    global puzzle\_arr

    filepath = fname\_entry.get()

    try:

        if (len(filepath) != 0):

            p = Puzzle(parseText(filepath))

        else:

            p = Puzzle(parseGUI(layout.getBuf()))

        kurangMsg, res, outputMsg = solve(p)

        ans\_text.configure(text = outputMsg)

        kurang\_label.configure(text = kurangMsg)

        layout.renderAll(res)

    except Exception as e:

        messagebox.showerror("[ERROR]", e)

class GUIPuzzle:

    def \_\_init\_\_(self):

        for i in range(4):

            for j in range(4):

                self.e = Entry(frame, width = 4, font = ('Arial', 20))

                self.e.grid(row = i, column = j)

                self.e.insert(END, "")

    def getBuf(self):

        buffer = ""

        for i in range(4):

            for j in range(4):

                buffer += frame.grid\_slaves(row = i, column = j)[0].get() + " "

        return buffer

    def clear(self):

        for i in range(4):

            for j in range(4):

                frame.grid\_slaves(row = i, column = j)[0].config({"background": "white"})

                frame.grid\_slaves(row = i, column = j)[0].delete(0, END)

    def render(self, puzzle):

        for i in range(4):

            for j in range(4):

                self.e = Entry(frame, width = 4, font = ('Arial', 20))

                self.e.grid(row = i, column = j)

                if (puzzle.buffer[i][j] == "ES"):

                    self.e.insert(END, "")

                    self.e.config({"background": "gray"})

                else:

                    self.e.insert(END, puzzle.buffer[i][j])

    def renderAll(self, puzzle\_arr):

        global frame

        for i in range(len(puzzle\_arr)):

            self.render(puzzle\_arr[i][0])

            steps\_label.configure(text = "Step " + str(i + 1) + ": " + puzzle\_arr[i][1])

            time.sleep(1)

            window.update()

begin\_coord = 150

window = Tk()

window.geometry("500x400")

window.minsize(500, 400)

window.maxsize(500, 400)

window.title("15-Puzzle Solver")

frame = Frame(window)

frame.pack(fill= BOTH, expand= True, padx= 20, pady=20)

fname\_entry = Entry(frame, text = "Input file name (without \*.txt)", font = ('Arial', 10), width = 36)

fname\_entry.place(x = 0, y = begin\_coord + 40)

layout = GUIPuzzle()

steps\_label = Label(frame, font = ("Arial", 8))

steps\_label.place(x = 0, y = begin\_coord)

fname\_label = Label(frame, font = ("Arial, 8"), text = "Input file name (without \*.txt)")

fname\_label.place(x = 0, y = begin\_coord + 20)

kurang\_label = Label(frame, font = ("Arial, 8"), wraplength = 150)

kurang\_label.place(x = 300, y = 0)

solve\_button = Button(frame, text = "Solve", width = 16, command = solveClick)

solve\_button.place(x = 0, y = begin\_coord + 70)

clear\_button = Button(frame, text = "Clear", width = 16, command = layout.clear)

clear\_button.place(x = 128, y = begin\_coord + 70)

ans\_text = Label(frame, font = ("Arial", 8), text = "Waiting for search to begin...")

ans\_text.place(x = 25, y = begin\_coord + 100)

window.mainloop()

## 2.3 Puzzle.py

## File ini berisi implementasi class Puzzle dan PuzzleItem, item yang dipakai untuk melakukan *enqueue*

from dataclasses import dataclass, field

import string

from typing import Any

@dataclass(order=True)

class PuzzleItem:

    priority: int

    item: Any=field(compare=False)

    direction: string

@dataclass(order=True)

class InvItem:

    priority: int

    value: Any=field(compare=False)

class Puzzle:

    # data members

    ROW\_SIZE = 4

    COL\_SIZE = 4

    NULL\_I = 0

    NULL\_J = 0

    curr\_depth = 0

    id = 0

    '''

    Constructor for the puzzle matrix

    '''

    def \_\_init\_\_(self, puzzle\_string):

        self.buffer = [[0 for \_ in range(self.COL\_SIZE)] for \_ in range(self.ROW\_SIZE)]

        for i in range(self.ROW\_SIZE):

            for j in range(self.COL\_SIZE):

                elmt = puzzle\_string[i \* self.COL\_SIZE + j]

                self.buffer[i][j] = elmt

                if (elmt == "ES"):

                    self.NULL\_I = i

                    self.NULL\_J = j

    '''

    Prints the puzzle matrix in a readable format

    '''

    def show(self):

        for i in range(self.ROW\_SIZE):

            for j in range(self.COL\_SIZE):

                print(self.buffer[i][j], end = " ")

            print()

    def \_\_equals\_\_(self, other):

        for i in range(self.ROW\_SIZE):

            for j in range(self.COL\_SIZE):

                if (self.buffer[i][j] != other.buffer[i][j]):

                    return False

        return True

    '''

    Checks possible movement directions for the current position

    '''

    def checkDir(self, direction):

        i = self.NULL\_I

        j = self.NULL\_J

        if (direction == "LEFT"):

            return j != 0 and self.buffer[i][j - 1] != "ES"

        elif (direction == "RIGHT"):

            return (j != self.COL\_SIZE - 1) and self.buffer[i][j + 1] != "ES"

        elif (direction == "UP"):

            return i != 0 and self.buffer[i - 1][j] != "ES"

        elif (direction == "DOWN"):

            return (i != self.ROW\_SIZE - 1) and self.buffer[i + 1][j] != "ES"

    '''

    shifts element of the puzzle matrix

    '''

    def shift(self, direction):

        i = self.NULL\_I

        j = self.NULL\_J

        if (self.checkDir(direction)):

            if (direction == "LEFT"): # NULL goes left

                self.buffer[i][j] = self.buffer[i][j - 1]

                self.buffer[i][j - 1] = "ES"

                self.NULL\_J -= 1

            elif (direction == "RIGHT"): # NULL goes right

                self.buffer[i][j] = self.buffer[i][j + 1]

                self.buffer[i][j + 1] = "ES"

                self.NULL\_J += 1

            elif (direction == "UP"): # NULL goes up

                self.buffer[i][j] = self.buffer[i - 1][j]

                self.buffer[i - 1][j] = "ES"

                self.NULL\_I -= 1

            elif (direction == "DOWN"): # NULL goes down

                self.buffer[i][j] = self.buffer[i + 1][j]

                self.buffer[i + 1][j] = "ES"

                self.NULL\_I += 1

    '''

    Checks if the puzzle is solved

    '''

    def isSolved(self):

        # return False if last element is not NULL

        if (self.buffer[self.ROW\_SIZE - 1][self.COL\_SIZE - 1] != "ES"):

            return False

        # else, check if all elements are in correct order, except for last element

        flattened\_buffer = [x for arr in self.buffer for x in arr]

        for i in range(1, len(flattened\_buffer) - 1):

            # return False if any element is not in correct order

            if (int(flattened\_buffer[i]) != int(flattened\_buffer[i - 1]) + 1):

                return False

        # return solved if all is sorted

        return True

    '''

    Returns 1 if:

    - odd row and even column

    - even row and odd column

    '''

    def nullPos(self):

        return 1 if (self.NULL\_I % 2 != self.NULL\_J % 2) else 0

    '''

    Counts the appearance of invalid position where

    element with less value than current element appears on a higher position

    '''

    def invalidPos(self, idx):

        count = 0

        flattened\_buffer = [x for arr in self.buffer for x in arr]

        if (flattened\_buffer[idx] == "ES"):

            count = self.COL\_SIZE \* self.ROW\_SIZE - idx - 1

        for i in range(idx, len(flattened\_buffer)):

            if (flattened\_buffer[i] != "ES" and flattened\_buffer[idx] != "ES"):

                if (int(flattened\_buffer[i]) < int(flattened\_buffer[idx]) and i > idx):

                    count += 1

        return count

    '''

    Returns the sum of invalid position

    '''

    def sumOfInvalidPos(self):

        sum = 0

        for i in range(0, self.ROW\_SIZE \* self.COL\_SIZE):

            sum += self.invalidPos(i)

        return sum

    '''

    Counts the appearance of invalid position where

    tile position doesn't match the value of the tile

    '''

    def nonMatchingTile(self):

        count = 0

        flattened\_buffer = [x for arr in self.buffer for x in arr]

        for i in range(0, len(flattened\_buffer)):

            if (flattened\_buffer[i] != "ES" and (int(flattened\_buffer[i]) != (i + 1))):

                count += 1

        return count;

    '''

    Checks whether current state of the puzzle has existed before

    '''

    def stateExisted(self, state\_dict):

        state = "|".join([x for arr in self.buffer for x in arr])

        return True if state in state\_dict else False

## 2.4 algo.py

File berisi konfirugrasi algoritma untuk melakukan pencarian

from queue import PriorityQueue

from puzzle import PuzzleItem, InvItem, Puzzle

import timeit as time

def solve(p):

    # displays initial puzzle

    print()

    print("Initial Puzzle:")

    p.show()

    print()

    outputMessage = ""

    res = []

    kurangMessage = ""

    # displays initial invalid values (kurang[i])

    if (not p.isSolved()):

        invalid\_pq = PriorityQueue()

        for i in range(16):

            if (p.buffer[int(i/4)][int(i%4)] == "ES"):

                invalid\_pq.put(InvItem(16, p.invalidPos(i)))

            else:

                currentValue = int(p.buffer[int(i/4)][int(i%4)])

                invalid\_pq.put(InvItem(currentValue, p.invalidPos(i)))

        print("List of Invalid Values: ")

        kurangMessage += "List of Invalid Values: \n"

        while (not invalid\_pq.empty()):

            temp = invalid\_pq.get()

            if (temp.priority == 16):

                print("Kurang[ES] = {}".format(temp.value))

                kurangMessage += "Kurang[ES] = {}\n".format(temp.value)

            else:

                print("Kurang[{}] = {}".format(temp.priority, temp.value))

                kurangMessage += "Kurang[{}] = {}\n".format(temp.priority, temp.value)

        print("Sum of invalid values:", p.sumOfInvalidPos())

        print("Sum of invalid values and whether empty space in determined position:",

              p.sumOfInvalidPos() + p.nullPos())

        kurangMessage+= "Sum of invalid values: {}\n".format(p.sumOfInvalidPos())

        kurangMessage += "Sum of invalid values and whether empty space in determined position: {}\n".format(

            p.sumOfInvalidPos() + p.nullPos())

        if ((p.sumOfInvalidPos() + p.nullPos()) % 2 != 0):

            outputMessage = "This puzzle cannot be solved!\n"

            print("This puzzle cannot be solved!\n")

        else:

            print("\nSolving puzzle ... ")

            prioqueue = PriorityQueue()

            # initializes initial puzzle as first item in queue

            prioqueue.put(PuzzleItem(0, p, "NONE"))

            state\_dict = {}

            puzzle\_arr = []

            curr\_id = 0

            # starts searching process

            start\_time = time.default\_timer()

            deq\_count = 0

            while (prioqueue.qsize() != 0):

                # dequeues item with lowest priority

                puzzleItem = prioqueue.get()

                temp = puzzleItem.item

                prev\_direction = puzzleItem.direction

                deq\_count += 1

                # adds that item to the list of solved puzzles

                state\_dict["|".join([x for arr in puzzleItem.item.buffer for x in arr])] = True

                puzzle\_arr.append([puzzleItem.item, puzzleItem.direction])

                # puzzleItem.item.show()

                curr\_id += 1

                #stop loop if current puzzle is already solved

                #else, keep inserting

                if puzzleItem.item.isSolved():

                    res.append([puzzleItem.item, puzzleItem.direction])

                    break

                else:

                    # if (not puzzleItem.item.insertingIntoQueue(prioqueue, state\_dict, curr\_id, puzzleItem.direction)):

                    #     break

                    if (p.isSolved()):

                        break

                    if (temp.checkDir("UP") and prev\_direction != "DOWN"):

                        puzzle = Puzzle([x for arr in temp.buffer for x in arr])

                        puzzle.shift("UP")

                        puzzle.curr\_depth = temp.curr\_depth + 1

                        puzzle.id = curr\_id

                        currCost = puzzle.curr\_depth + puzzle.nonMatchingTile()

                        if (puzzle.isSolved()):

                            prioqueue.put(PuzzleItem(0, puzzle, "UP"))

                            break

                        elif (not puzzle.stateExisted(state\_dict)):

                            prioqueue.put(PuzzleItem(currCost, puzzle, "UP"))

                    if (temp.checkDir("RIGHT") and prev\_direction != "LEFT"):

                        puzzle = Puzzle([x for arr in temp.buffer for x in arr])

                        puzzle.shift("RIGHT")

                        puzzle.curr\_depth = temp.curr\_depth + 1

                        puzzle.id = curr\_id

                        currCost = puzzle.curr\_depth + puzzle.nonMatchingTile()

                        if (puzzle.isSolved()):

                            prioqueue.put(PuzzleItem(0, puzzle, "RIGHT"))

                            break

                        elif (not puzzle.stateExisted(state\_dict)):

                            prioqueue.put(PuzzleItem(currCost, puzzle, "RIGHT"))

                    if (temp.checkDir("DOWN") and prev\_direction != "UP"):

                        puzzle = Puzzle([x for arr in temp.buffer for x in arr])

                        puzzle.shift("DOWN")

                        puzzle.curr\_depth = temp.curr\_depth + 1

                        puzzle.id = curr\_id

                        currCost = puzzle.curr\_depth + puzzle.nonMatchingTile()

                        if (puzzle.isSolved()):

                            prioqueue.put(PuzzleItem(0, puzzle, "DOWN"))

                            break

                        elif (not puzzle.stateExisted(state\_dict)):

                            prioqueue.put(PuzzleItem(currCost, puzzle, "DOWN"))

                    if (temp.checkDir("LEFT") and prev\_direction != "RIGHT"):

                        puzzle = Puzzle([x for arr in temp.buffer for x in arr])

                        puzzle.shift("LEFT")

                        puzzle.curr\_depth = temp.curr\_depth + 1

                        puzzle.id = curr\_id

                        currCost = puzzle.curr\_depth + puzzle.nonMatchingTile()

                        if (puzzle.isSolved()):

                            prioqueue.put(PuzzleItem(0, puzzle, "LEFT"))

                            break

                        elif (not puzzle.stateExisted(state\_dict)):

                            prioqueue.put(PuzzleItem(currCost, puzzle, "LEFT"))

            stop\_time = time.default\_timer()

            # backtracks parent id to get the path

            # of solutions

            answer = prioqueue.get()

            res = [[answer.item, answer.direction]]

            puzzle\_elmt = puzzle\_arr[-1]

            while (puzzle\_elmt[0].id != 0):

                res = [puzzle\_elmt] + res

                puzzle\_elmt = puzzle\_arr[puzzle\_elmt[0].id - 1]

            res = [[p, "NONE"]] + res

            # outputs process information

            outputMessage += "\nPuzzle solved successfully!"

            outputMessage += "\nElapsed time: " + str("%.11f" % (stop\_time - start\_time)) + " seconds"

            outputMessage += "\nRaised nodes: " + str(deq\_count+ len(res))

            outputMessage += "\nSteps taken : " + str(len(res))

            return kurangMessage, res, outputMessage

    else:

        outputMessage = "This puzzle is already solved! >:(\n"

        print("This puzzle is already solved! >:(\n")

## 2.5 fparser.py

File berisi konfigurasi *parsing* untuk penerimaan input

# Parser script for testcases

import os

def checkValid(arr):

    temp = [x for x in (arr)]

    temp.remove("ES")

    temp.append("16")

    temp = [int(x) for x in temp]

    temp.sort()

    for i in range(len(temp)):

        if (int(temp[i]) != i + 1):

            raise Exception("[INVALID] Input is not valid!")

    return True

def parseText(fname):

    dirname = os.path.dirname(\_\_file\_\_)

    path = os.path.join(dirname, '../test/')

    if (os.path.exists(path + fname + ".txt")):

        file = open(path + fname + ".txt", "r")

        arr = file.read().replace("-", "ES").replace("\n", " ").split(" ")

        # temp = [x for x in (arr)]

        # temp.remove("ES")

        # temp.append("16")

        # temp = [int(x) for x in temp]

        # temp.sort()

        # for i in range(len(temp)):

        #     if (int(temp[i]) != i + 1):

        #         raise Exception("[INVALID] Input is not valid!")

        return arr if checkValid(arr) else None

    else:

        raise Exception("[INVALID] File doesn't exist! Make sure it is stored in the 'test' folder and the filename is correct! (without .txt)")

def parseInput():

    print("\n[SELECTED] Input by user")

    print("Input the desired matrix in a 4 x 4 grid style!")

    print("Fill the empty space character with '-'!")

    buffer = [[0 for \_ in range(4)] for \_ in range(4)]

    for i in range(4):

        print("[ROW {}] | >> ".format(i + 1), end = " ")

        buffer[i] = list(map(str, input().split()))

    flattened\_buffer = ' '.join([x for arr in buffer for x in arr])

    arr = flattened\_buffer.replace("-", "ES").replace("\n", " ").split(" ")

    # temp = [x for x in (arr)]

    # temp.remove("ES")

    # temp.append("16")

    # temp = [int(x) for x in temp]

    # temp.sort()

    # for i in range(len(temp)):

    #     if (int(temp[i]) != i + 1):

    #         raise Exception("[INVALID] Input is not valid!")

    return arr if checkValid(arr) else None

def parseGUI(buffer):

    print(buffer)

    arr = buffer.rstrip().replace("-", "ES").replace("\n", " ").split(" ")

    # print(arr)

    # temp = [x for x in (arr)]

    # temp.remove("ES")

    # temp.append("16")

    # print(temp)

    # temp = [int(x) for x in temp]

    # temp.sort()

    # for i in range(len(temp)):

    #     if (int(temp[i]) != i + 1):

    #         raise Exception("[INVALID] Input is not valid!")

    return arr if checkValid(arr) else None

# BAB III. PENGUJIAN

Pengujian dilakukan menggunakan dataset pada *library* sklearn dan dijalankan dengan file main.py. File main.py dijalankan di IDE *Visual Studio Code / PyCharm Community Edition.* Hasil yang dilampirkan pada pengujian adalah scatter plot yang ditunjukkan oleh fungsiplt.show() *(*pastikan pustaka matplotlib sudah terpasang sebelum pengujian, informasi lebih lanjut pada file README.md*).*

## 3.1 Sepal Length vs Sepal Width

|  |  |
| --- | --- |
| Opsi dataset pada main = 1 (iris) , opsi kolom = 0 | |
|  |  |
| Gambar 3.1.1 Pengujian menggunakan Pustaka myConvexHull | Gambar 3.1.2 Pengujian menggunakan Pustaka SciPy |

## 3.2 Petal Length vs Petal Width

|  |  |
| --- | --- |
| Opsi dataset pada main = 1 (iris), opsi kolom = 2 | |
|  |  |
| Gambar 3.2.1 Pengujian menggunakan Pustaka myConvexHull | Gambar 3.2.2 Pengujian menggunakan Pustaka SciPy |

## 3.3 Color Intensity vs Hue

|  |  |
| --- | --- |
| Opsi dataset pada main = 2 (wine), opsi kolom = 9 | |
|  |  |
| Gambar 3.3.1 Pengujian menggunakan Pustaka myConvexHull | Gambar 3.3.2 Pengujian menggunakan Pustaka SciPy |

## 3.4 Ash vs Alcalinity of Ash

|  |  |
| --- | --- |
| Opsi dataset pada main = 2 (wine), opsi kolom = 2 | |
|  |  |
| Gambar 3.4.1 Pengujian menggunakan Pustaka myConvexHull | Gambar 3.4.2 Pengujian menggunakan Pustaka SciPy |

## 3.5 Alcohol vs Malic Acid

|  |  |
| --- | --- |
| Opsi dataset pada main = 2 (wine), opsi kolom = 0 | |
| Chart, scatter chart  Description automatically generated | Chart  Description automatically generated |
| Gambar 3.5.1 Pengujian menggunakan Pustaka myConvexHull | Gambar 3.5.2 Pengujian menggunakan Pustaka SciPy |

## 3.6 OD280/OD315 of Diluted Wines vs Proline

|  |  |
| --- | --- |
| Opsi dataset pada main = 2 (breast\_cancer), opsi kolom = 11 | |
| Chart, radar chart  Description automatically generated | Chart, radar chart  Description automatically generated |
| Gambar 3.6.1 Pengujian menggunakan Pustaka myConvexHull | Gambar 3.6.2 Pengujian menggunakan Pustaka SciPy |

## 3.7 Mean Smoothness vs Mean Compactness

|  |  |
| --- | --- |
| Opsi dataset pada main = 4 (breast\_cancer), opsi kolom = 4 | |
|  |  |
| Gambar 3.7.1 Pengujian menggunakan Pustaka myConvexHull | Gambar 3.7.2 Pengujian menggunakan Pustaka SciPy |

## 3.8 Worst Concavity vs Worst Concave Points

|  |  |
| --- | --- |
| Opsi dataset pada main = 4 (breast\_cancer), opsi kolom = 26 | |
|  |  |
| Gambar 3.8.1 Pengujian menggunakan Pustaka myConvexHull | Gambar 3.8.2 Pengujian menggunakan Pustaka SciPy |

## 3.9 pixel\_0\_2 vs pixel\_0\_3

|  |  |
| --- | --- |
| Opsi dataset pada main = 3 (digits), opsi kolom = 2 | |
|  |  |
| Gambar 3.9.1 Pengujian menggunakan Pustaka myConvexHull | Gambar 3.9.2 Pengujian menggunakan Pustaka SciPy |

## 3.10 pixel\_6\_3 vs pixel\_6\_4

|  |  |
| --- | --- |
| Opsi dataset pada main = 3 (digits), opsi kolom = 51 | |
|  |  |
| Gambar 3.10.1 Pengujian menggunakan Pustaka myConvexHull | Gambar 3.10.2 Pengujian menggunakan Pustaka SciPy |

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Pustaka *myConvexHull* berhasil dibuat dan tidak ada kesalahan | ✓ |  |
| 2. *Convex Hull* yang dihasilkan sudah benar | ✓ |  |
| 3. Pustaka *myConvexHull* dapat digunakan untuk menampilkan *convex hull* setiap label dengan warna yang berbeda. | ✓ |  |
| 4. Program dapat menerima input dan menuliskan output untuk dataset lainnya. | ✓ |  |

# BAB IV. *REPOSITORY*

*Repository* dapat diakses via https://github.com/clumsyyyy/TucilStima2