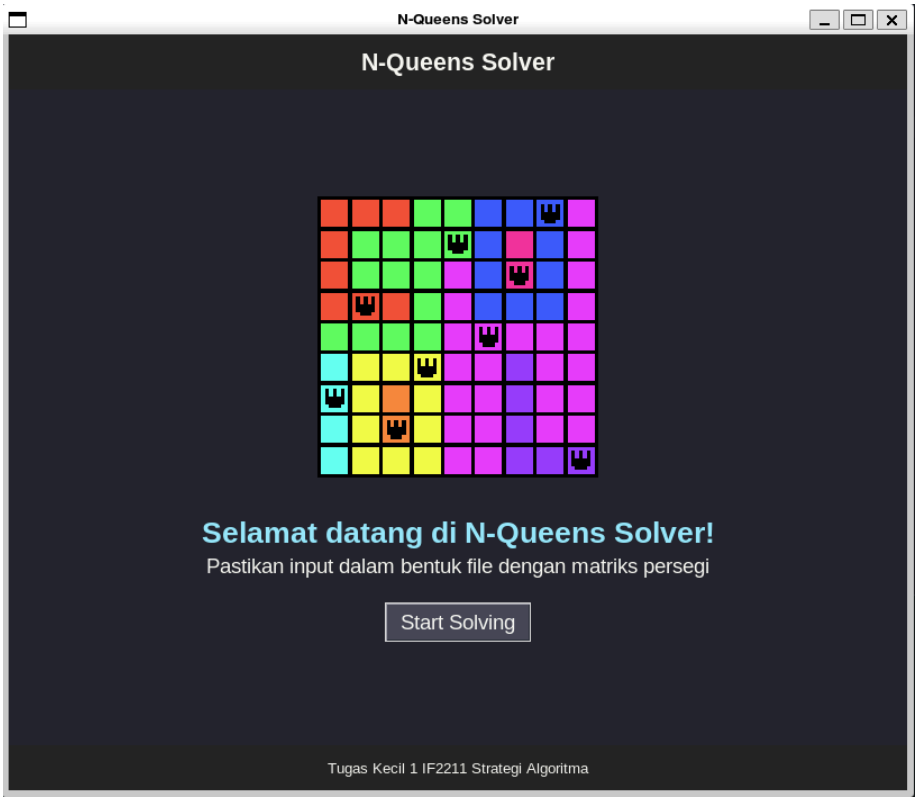
LAPORAN TUGAS KECIL I
IF2211 Strategi Algoritma
Semester II tahun 2025/2026



Penyelesaian Permainan *Queens* Linkedin

Renuno Yuqa Frinardi
13524080
Teknik Informatika

PERNYATAAN TIDAK MELAKUKAN KECURANGAN

Tugas ini disusun sepenuhnya tanpa bantuan kecerdasan buatan (Generative AI), melainkan hasil pemikiran dan analisis mandiri.

Renuno Yuqa Frinardi

I. Pendahuluan

Queens merupakan sebuah game puzzle. Game puzzle ini memiliki tujuan untuk memposisikan ratu dalam sebuah papan persegi agar dua atau lebih ratu tidak berada dalam satu baris atau kolom dan tidak ditaruh secara bersebelahan, baik diagonal maupun vertikal-horizontal. Solusi dari permainan ini dapat diselesaikan menggunakan beberapa jenis algoritma. Salah satunya adalah dengan pendekatan *brute force*.

II. Penjelasan Algoritma

Program ini ditulis menggunakan algoritma *brute force* yang mengecek seluruh kemungkinan posisi ratu. Algoritma ini ditulis sebagai berikut:

```
fungsi generateCombinations(tile_options: array of array of point,
current_combination: array of point, color_index : integer) → boolean
{ Menghasilkan semua kombinasi posisi ratu secara rekursif }

ALGORITMA
     if color_index = panjang(tile_options) then
          if isValid(current_combination) then
               i traversal [0..length(current_combination)-1]
                    queen_positions[i] <- current_combination[i]
            solved <- True
            return True
     else
          return False

     for tile in tile_options[color_index]
          case <- case + 1
          current_combination.append(tile)


          if case mod 1000000 = 0 then
               displayProgress(current_combination)

          if generateCombinations(tile_options,
current_combination, color_index + 1) then
               return True
          current_combination.hapusTerakhir()
      return False
```

```
fungsi isValid(positions : array of point) → boolean
{ Mengecek apakah posisi ratu valid }

ALGORITMA
     i traversal [0..length(positions)-1] then
          j traversal [i+1..length(positions)] then
               if positions[i][0] = positions[j][0] or
                  positions[i][1] = positions[j][1] or
                  abs(positions[i][0] - positions[j][0]) =
```

```
                    abs(positions[i][1] - positions[j][1]) = 1 then
                      return False
    return True
```

Komponen utama dari algoritma ini terdapat pada dua fungsi, yaitu fungsi rekursif untuk mengecek tiap kombinasi dan fungsi untuk mengecek apakah kombinasi valid.

Alur kerja dari algoritma ini bermulai dari pemanggilan fungsi **`generateCombinations`** di mana fungsi akan menerima input sebuah array berisi list koordinat yang dikelompokan berdasarkan warna, kombinasi posisi ratu sejauh ini, dan index warna.

Basis dari fungsi ini adalah ketika panjang index warna sudah berada pada index terakhir. Ketika sudah ada di posisi ini maka akan dicek apakah kombinasi tersebut sudah sesuai dengan aturan. Dilakukan pemanggilan fungsi **`isValid`** yang akan melakukan pengecekkan pada tiap pasangan dalam kombinasi apakah ada yang terdapat dalam satu baris, kolom, atau bersebelahan (secara horizontal, vertikal, atau diagonal). Apabila bernilai benar, maka fungsi akan mengembalikan nilai true dan apabila salah akan bernilai false

Rekurens dari fungsi ini bekerja ketika index warna belum sampai di akhir, sehingga akan dilakukan penambahan posisi ratu pada array kombinasi posisi, penambahan jumlah pengecekkan kasus, juga print out untuk progress pencetakan ke layar. Kemudian fungsi akan memanggil kembali dirinya sendiri dengan kombinasi ratu terbaru juga index warna yang sudah ditambah satu. Apabila pemanggilan fungsi mengembalikan nilai true, maka fungsi akan berlanjut dengan mengembalikan nilai true. Apabila tidak maka fungsi akan menghilangkan posisi kotak terakhir dan mengecek opsi tile selanjutnya pada kelompok warna. Jika tidak ada posisi yang sesuai, maka fungsi akan mengembalikan nilai false.

III. *Source Code* Program

Berikut adalah keseluruhan *source program* yang dibuat dalam bahasa pemrograman python. Program dibagi menjadi 5 file sebagai berikut:

[main.py]

```python
# Entry utama untuk program

import gui.gui as App

class MainProgram:
    def __init__(self):
        App.Window()
```

```
if __name__ == "__main__":
    MainProgram()
```

[matrix.py]

```
# Kelas Matriks untuk menyimpan data

class Matrix:
    # Method untuk inisiasi kelas
    def __init__(self, filename):
        file = open(filename, 'r')
        lines = file.readlines()
        file.close()

        self.valid = False
        self.error_message = ""

        # Mengecek apakah file kosong
        if len(lines) == 0:
            self.error_message = "File kosong"
            self.row = 0
            self.col = 0
            self.matrix = []
            return

        # Mengecek apakah keseluruhan file berisi matriks
        lines = [line for line in lines if line.strip()]
        if len(lines) == 0:
            self.error_message = "File tidak mengandung data matrix"
            self.row = 0
            self.col = 0
            self.matrix = []
            return
        self.row = len(lines)

        # Mengecek apakah ada nilai kosong pada matriks
        first_row_length = len(lines[0].strip())
        for _, line in enumerate(lines):
            if len(line.strip()) != first_row_length:
```

```
                self.error_message = "Matrix harus persegi"
                self.row = 0
                self.col = 0
                self.matrix = []
                return

        # Mengecek apakah matriks persegi
        if self.row != first_row_length:
            self.error_message = "Matrix tidak persegi"
            self.row = 0
            self.col = 0
            self.matrix = []
            return

        self.col = self.row
        self.matrix = [['0' for _ in range(self.row)] for _ in
range(self.row)]
        self.input(lines)
        self.valid = True

    # Menerima input
    def input(self, string_input):
        for i in range(self.row):
            line = list(string_input[i].strip())
            for j in range(self.col):
                if j < len(line):
                    self.matrix[i][j] = line[j]
```

[algorithm.py]

```
# Kelas untuk algoritma brute force dalam penyelesaian

import time

class Algorithm:
    def __init__(self, matrix, window=None):
        # Deklarasi variabel yang diperlukan
        self.matrix = matrix
        self.window = window
```

```
        self.queen_positions = []
        self.color_dict = {}
        self.color_list = []
        self.solved = False
        self.case = 0
        self.modulo = 1000

        # Pemrosesan input
        self.saveColor()
        self.start = time.time()
        self.computeBruteForce()
        self.end = time.time()
        self.displayResult()

    # Menyimpan banyak warna pada papan
    def saveColor(self):
        for i in range(self.matrix.row):
            for j in range(self.matrix.col):
                color = self.matrix.matrix[i][j]
                if color not in ['\n', ' ']:
                    if color not in self.color_dict:
                        self.color_dict[color] = []
                    self.color_dict[color].append((i, j))
        self.color_list = list(self.color_dict.keys())

    # Fungsi inisasi bruteforce
    def computeBruteForce(self):
        tile_options = [self.color_dict[color] for color in
self.color_list]
        self.generateCombinations(tile_options, [], 0)

    # Fungsi proses rekursif pada brute force
    def generateCombinations(self, tile_options, current_combination,
color_index):
        # Basis apabila sudah berada di warna terakhir
        if color_index == len(tile_options):
            if self.isValid(current_combination):
                for i in range(len(current_combination)):
                    self.queen_positions.append(current_combination[i])
```

```
                self.solved = True
                return True
            return False

        # Rekurens mengecek seluruh kombinasi warna
        for tile in tile_options[color_index]:
            self.case += 1
            current_combination.append(tile)

            # Melakukan penyimpanan kombinasi
            if self.case % 1000000 == 0:
                self.displayProgress(current_combination)

            if self.generateCombinations(tile_options,
current_combination, color_index + 1):
                return True
            current_combination.pop()
        return False

    # Mengecek apakah susunan ratu valid
    def isValid(self, positions):
        for i in range(len(positions)):
            for j in range(i + 1, len(positions)):
                pos1, pos2 = positions[i], positions[j]
                if pos1[0] == pos2[0] or pos1[1] == pos2[1] or
abs(pos1[0] - pos2[0]) == abs(pos1[1] - pos2[1]) == 1:
                    return False
        return True

    # Menampilkan progress
    def displayProgress(self, positions=None):
        matrix = []
        if positions is not None:
            for row in range(self.matrix.row):
                row_now = []
                for col in range(self.matrix.col):
                    placed = False
                    for position in positions:
                        if row == position[0] and col == position[1]:
```

```
                            row_now.append('#')
                            placed = True
                            break
                    if not placed:
                        row_now.append(self.matrix.matrix[row][col])
                matrix.append(row_now)
            if self.window:
                self.window.initProg(matrix, self.matrix.row,
self.matrix.col)
        else:
            for row in range(self.matrix.row):
                row_now = []
                for col in range(self.matrix.col):
                    placed = False
                    for position in self.queen_positions:
                        if row == position[0] and col == position[1]:
                            row_now.append('#')
                            placed = True
                            break
                    if not placed:
                        row_now.append(self.matrix.matrix[row][col])
                matrix.append(row_now)

    # Menampilkan hasil akhir
    def displayResult(self):
        if self.solved:
            self.displayProgress()
            self.window.initResult(self)
        else:
            self.window.initResult(self)
```

[drawimg.py]

```
# Kelas untuk menggambar solusi

from PIL import Image, ImageDraw

class Drawer:
    # Insiasi variable dan skema warna
```

```
    def __init__(self, algoritma):
        self.color_dict = {
                'A': '#FF5733', 'B': '#33FF57', 'C': '#3357FF', 'D':
'#F333FF',
                'E': '#FF33A1', 'F': '#33FFF3', 'G': '#F3FF33', 'H':
'#A133FF',
                'I': '#FF8C33', 'J': '#33FF8C', 'K': '#8C33FF', 'L':
'#FF3333',
                'M': '#33FF33', 'N': '#3333FF', 'O': '#FFFF33', 'P':
'#FF33FF',
                'Q': '#33FFFF', 'R': '#A52A2A', 'S': '#800080', 'T':
'#008080',
                'U': '#FFD700', 'V': '#C0C0C0', 'W': '#808080', 'X':
'#FFA500',
                'Y': '#000080', 'Z': '#FF1493'
            }
        self.height = algoritma.matrix.row * 8 + 1
        self.width = self.height
        self.solution = self.drawQueen(self.drawGrid(algoritma.matrix),
algoritma.queen_positions)

    # Menggambar grid terlbih dahulu
    def drawGrid(self, matrix):
        image = Image.new("RGB", (self.height, self.width), "white")
        draw = ImageDraw.Draw(image)
        for i in range(matrix.row):
            for j in range(matrix.col):
                draw.rectangle([(j*8, i*8), (j*8+8, i*8+8)],
fill=self.color_dict.get(matrix.matrix[i][j]), outline="black")
        return image

    # Menggambar ratu pada papan
    def drawQueen(self, image, position):
        draw = ImageDraw.Draw(image)
        for pos in position:
            draw.rectangle([(pos[1]*8+2, pos[0]*8+2), (pos[1]*8+2,
pos[0]*8+5)], fill="black")
            draw.rectangle([(pos[1]*8+3, pos[0]*8+4), (pos[1]*8+5,
pos[0]*8+6)], fill="black")
```

```
            draw.rectangle([(pos[1]*8+4, pos[0]*8+2), (pos[1]*8+4,
pos[0]*8+3)], fill="black")
            draw.rectangle([(pos[1]*8+6, pos[0]*8+2), (pos[1]*8+6,
pos[0]*8+5)], fill="black")
        upscaled_image = image.resize((250, 250),
resample=Image.NEAREST)
        return upscaled_image
```

[gui.py]

```
# Kelas untuk membuat GUI dari program

import tkinter as Tk
from tkinter import filedialog as Fdg
from tkinter import messagebox as Msg
import matrix as Mat
import algorithm as Algo
import drawimg as Drw
import math as Math

class Window:
    def __init__(self, algorithm=None):
        # Setup initializer window
        self.root = Tk.Tk()
        self.algorithm = algorithm
        self.root.geometry("800x600")
        self.root.title("N-Queens Solver")
        self.root.configure(bg="#21222c")

        # Setup layout dasar aplikasi
        self.header = Tk.Frame(self.root, bg="#222222", height=50)
        self.content = Tk.Frame(self.root, bg="#21222c")
        self.footer = Tk.Frame(self.root, bg="#222222", height=50)

        self.header.pack(fill="both")
        self.content.pack(fill="both", expand=True)
        self.footer.pack(fill="both")

        # Setup header
```

```
        self.header_label = Tk.Label(self.header, text="N-Queens
Solver", font=("Arial", 16, "bold"), bg="#222222", fg="#f8f8f2")
        self.header_label.pack(pady=10)

        # Setup footer
        self.footer_label = Tk.Label(self.footer, text="Tugas Kecil 1
IF2211 Strategi Algoritma", font=("Arial", 10), bg="#222222",
fg="#f8f8f2")
        self.footer_label.pack(pady=10)

        # Setup Loop
        self.solution = None
        self.problem = None
        self.initHome()
        self.root.mainloop()

    # Fungsi inisiasi home page aplikasi
    def initHome(self):
        if self.solution:
            del self.solution
        if self.problem:
            del self.problem
        self.clearLayout()

        # Setup container
        main_content = Tk.Frame(self.content, bg="#21222c")
        main_content.pack(fill="both")
        main_content.place(relx=0.5, rely=0.5, anchor="center")

        # Setup gambar
        img = Tk.PhotoImage(file="src/assets/image.png")
        label_img = Tk.Label(main_content, image=img, bg="#21222c")
        label_img.image = img
        label_img.pack(pady=(0, 30))

        # Setup text
        label1 = Tk.Label(main_content, text="Selamat datang di N-Queens
Solver!", font=("Arial", 20, "bold"), bg="#21222c", fg="#8be9fd")
        label2 = Tk.Label(main_content, text="Pastikan input dalam
```

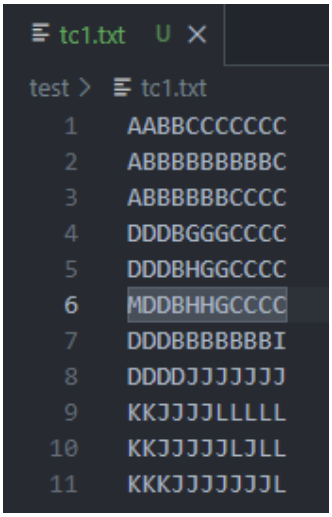
```
bentuk file dengan matriks persegi", font=("Arial", 14), bg="#21222c",
fg="#f8f8f2")
        label1.pack()
        label2.pack(pady=(0, 20))

        # Setup button
        btn = Tk.Button(main_content, text="Start Solving",
font=("Arial", 14), bg="#44475a", fg="#f8f8f2", command=self.fileTxt)
        btn.pack()

    # Fungsi inisiasi page progress
    def initProg(self, array_mat, row_i, col_i):
        self.clearLayout()
        grid_frame = Tk.Frame(self.content, bg="#21222c")
        grid_frame.place(relx=0.5, rely=0.5, anchor="center")

        # Print matriks proses
        for r in range(row_i):
            for c in range(col_i):
                label = Tk.Label(grid_frame, text=array_mat[r][c],
relief="solid", width=4, height=2, bg="#44475a", fg="#f8f8f2",
font=("Courier", 10, "bold"))
                label.grid(row=r, column=c, padx=2, pady=2)

        self.root.update()

    # Fungsi inisiasi page hasil
    def initResult(self, algorithm):
        self.clearLayout()
        self.algorithm = algorithm

        main_content = Tk.Frame(self.content, bg="#21222c")
        main_content.pack(fill="both")
        main_content.place(relx=0.5, rely=0.5, anchor="center")

        # Kasus apabila tdk ditemukan solusi
        if not self.algorithm.solved:

            img = Tk.PhotoImage(file="src/assets/failed.png")
```

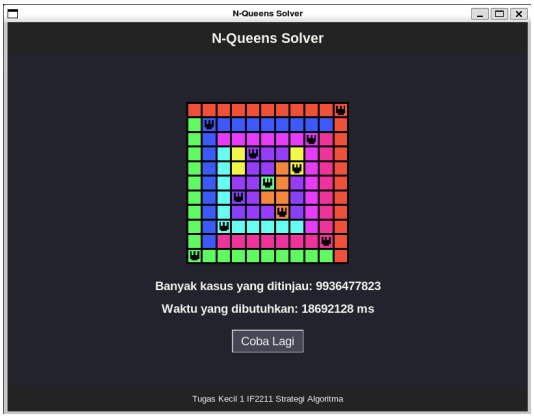
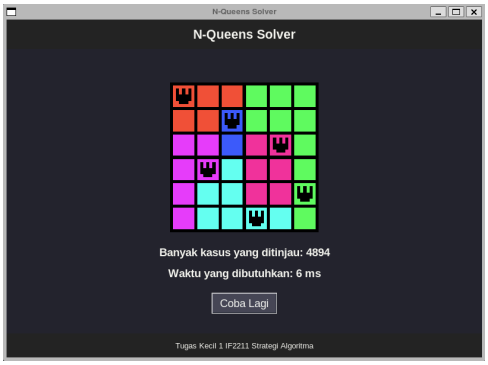
```
            label_img = Tk.Label(main_content, image=img, bg="#21222c")
            label_img.image = img
            label_img.pack(pady=20)

            label = Tk.Label(main_content, text="Tidak ada solusi yang
ditemukan.", font=("Arial", 16, "bold"), bg="#21222c", fg="#f8f8f2")
            label.pack(pady=20)

        # Kasus apabila ditemukan solusi
        else:
            image = Drw.Drawer(self.algorithm)

            image.solution.save("src/output/solution.png")
            img = Tk.PhotoImage(file="src/output/solution.png")

            label_img = Tk.Label(main_content, image=img, bg="#21222c")
            label_img.image = img
            label_img.pack(pady=20)

            label1 = Tk.Label(main_content, text=f"Banyak kasus yang
ditinjau: {self.algorithm.case}", font=("Arial", 14, "bold"),
bg="#21222c", fg="#f8f8f2")
            label2 = Tk.Label(main_content, text=f"Waktu yang
dibutuhkan: {Math.ceil((self.algorithm.end - self.algorithm.start) *
1000)} ms", font=("Arial", 14, "bold"), bg="#21222c", fg="#f8f8f2")
            label1.pack(pady=(0, 10))
            label2.pack(pady=(0, 20))

        btn = Tk.Button(main_content, text="Coba Lagi", font=("Arial",
14), bg="#44475a", fg="#f8f8f2", command=self.initHome)
        btn.pack()

    # Fungsi untuk dialog box file
    def fileTxt(self):
        self.my_txt = Fdg.askopenfilename(initialdir="", title="Pilih
sebuah file txt", filetypes=[("Text files", "*.txt"), ("All files",
"*")])
        if not self.my_txt:
            return
```

```
        self.problem = Mat.Matrix(self.my_txt)

        if not self.problem.valid:
            Msg.showerror("Validation Error",
self.problem.error_message)
            return

        self.solution = Algo.Algorithm(self.problem, self)

    # Fungsi membersihkan bagian content pada aplikasi
    def clearLayout(self):
        for widget in self.content.winfo_children():
            widget.destroy()
```

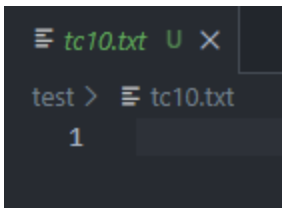
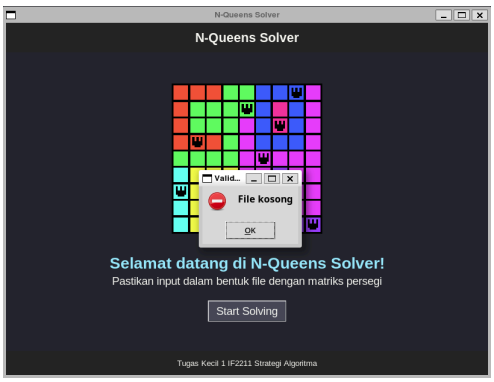
IV. Pengujian Program

Berikut adalah beberapa kasus uji yang diberikan dalam

Input	Yang Diharapkan	Hasil
	Ditemukan sebuah solusi	

tc2.txt test > tc2.txt 1 AAAAAAAAAAA 2 BCCCCCCCCCA 3 BCDDDDDDDEA 4 BCFGHHHGDEA 5 BCFGHHIGDEA 6 BCFHHJIHDEA 7 BCFKHIIKDEA 8 BCFKHHIKDEA 9 BCFFFFFFDEA 10 BCEEEEEEEEA 11 BBBBBBBBBBA	Ditemukan sebuah solusi	
tc3.txt test > tc3.txt 1 AAABB 2 ABBBC 3 ABDDD 4 CCCDD 5 EEEEE	Ditemukan sebuah solusi	
tc4.txt test > tc4.txt 1 AAABBB 2 AACBBB 3 DDCEEB 4 DDFEEB 5 DFFEEB 6 DFFFFB	Ditemukan sebuah solusi	
tc5.txt test > tc5.txt 1 AABCCC 2 ADBCCC 3 DDBBEE 4 DFFBEE 5 FFBBEE 6 FFBBEE	Ditemukan sebuah solusi	

tc6.txt U test > tc6.txt 1 AAAAA 2 BBCCD 3 BBCCE 4 FFFFE 5 GGGGG	Tidak ditemukan solusi	
tc7.txt U test > tc7.txt 1 AAAAA 2 BBCCD 3 4 FFFFE 5 GGGGG	Error karena matriks bukan persegi	
tc8.txt U test > tc8.txt 1 AAAAA 2 BBCCD 3 BBCCE 4 FFFFE	Error karena matriks bukan persegi	
tc9.txt U test > tc9.txt 1 AAAAA 2 BBCCD 3 BBCCE 4 FFFE 5 GGGGG	Error karena matriks bukan persegi	

	Error karena tidak ada matriks	

V. Lampiran

Repository GitHub:

- https://github.com/renuno-frinardi/Tucil1_13524080

Tabel keterangan tugas:

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	✔	
2	Program berhasil dijalankan	✔	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	✔	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	✔	
5	Program memiliki Graphical User Interface (GUI)	✔	
6	Program dapat menyimpan solusi dalam bentuk file gambar	✔	