LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Penyelesaian Permainan *Queens* Linkedin

Disusun Oleh :

Geraldo Artemius / 13524005

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

JL. GANESA 10, BANDUNG 40132

2026

DAFTAR ISI

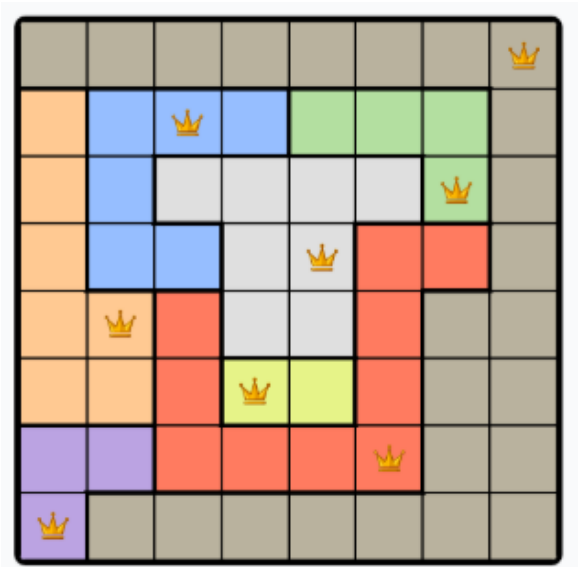
BAB I

DESKRIPSI PERSOALAN DAN ALGORITMA

1.1 Algoritma Brute Force

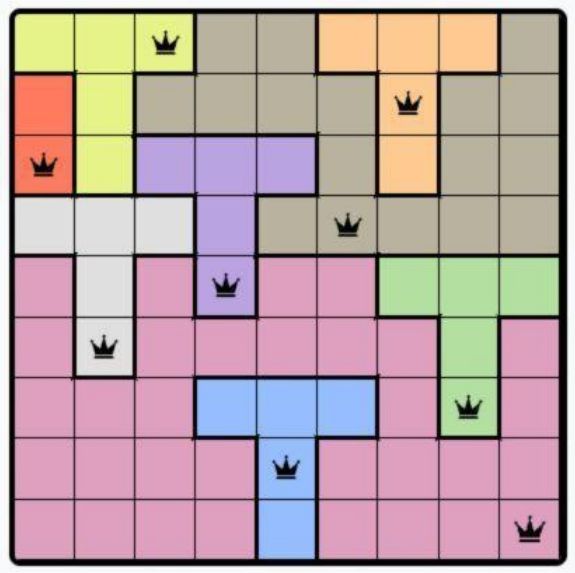
Algoritma *Brute Force* adalah metode yang digunakan untuk menyelesaikan persoalan secara *straightforward*. *Brute Force* ini bukan artinya ada spesifikasi khusus untuk menyelesaikan persoalan tertentu, tetapi umumnya mengandung lima tahap. Tahap pertama, inisialisasi yang menentukan semua parameter dan variabel yang diperlukan. Tahap kedua, iterasi melalui semua kemungkinan secara berurutan. Tahap ketiga, pengujian untuk menentukan solusi benar atau tidak. Tahap keempat, jika solusi yang ditemukan sudah benar maka algoritma akan menghentikan iterasi. Tahap kelima, algoritma akan menghasilkan solusi yang benar setelah mengecek semua kemungkinan. Algoritma ini menyelesaikan persoalan secara sederhana, langsung, dan langkah-langkahnya mudah untuk dipahami. Metode ini sering digunakan ketika tidak ada cara yang lebih efisien untuk menyelesaikan masalah tersebut.

1.2 Permasalahan Permainan *Queen* di LinkedIn



Queens adalah gim logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari gim ini adalah menempatkan queen pada sebuah papan persegi berwarna sehingga terdapat hanya satu queen pada tiap baris, kolom, dan daerah warna. Selain itu, satu queen tidak dapat ditempatkan bersebelahan dengan queen lainnya, termasuk secara diagonal.

Berikut merupakan salah satu solusi permainan *queens* yang valid :



1.3 Penyelesaian Permasalahan *Queens* dengan Algoritma *Brute Force*

TERMINAL :

1. Program meminta user untuk memasukkan nama file .txt yang berada pada folder input. Setelah nama file dimasukkan, program akan membentuk path menuju file tersebut. Jika file tidak ditemukan, program akan meminta user memasukkan ulang nama file.
2. Program membaca file menggunakan fungsi `readBoard()`. Fungsi ini akan membaca setiap baris file, lalu menyimpan setiap baris sebagai dalam list. Hasil akhirnya adalah list string yang merepresentasikan board permainan.
3. Setelah itu, program akan memeriksa apakah board tersebut valid dengan menggunakan fungsi `isBoardValid()`. Board dianggap valid jika board dengan ukuran N x N, N tidak bernilai nol.
4. Lalu program akan mengelompokkan semua warna, serta letak-letaknya dalam board. Ibarat seperti *mapping*.
5. Program kemudian mengurutkan warna berdasarkan jumlah paling sedikit karena dengan mencoba kemungkinan posisi paling sedikit maka akan mudah dalam menentukan posisi *queen* selanjutnya. Jika posisi *queen* sudah cocok maka dapat melanjutkan untuk meletakkan *queen* selanjutnya. Sebaliknya, *queen* dengan posisi yang belum cocok maka akan mencoba posisi lain hingga valid. Tetapi jika pada suatu tahap tidak ada posisi yang valid untuk suatu warna maka terpaksa menerapkan *backtrack* yaitu kembali ke warna sebelumnya untuk diganti posisi *queen*, namun tetap memperhatikan aturan.
6. Validasi posisi *queen* dilakukan dengan fungsi `isQueenValid()`.
7. Setiap pemilihan posisi *queen*, program akan selalu menghitung jumlah kasus yang diuji dengan variabel `count`.

8. Selain itu, program akan mencatat waktu dalam mencari solusi permainan *queens* dengan menggunakan library `time`.
9. Jika solusi ditemukan maka program akan menampilkan solusi berupa board yang sudah meletakkan semua *queens* di posisi nya masing-masing, serta dengan waktu dan banyaknya kasus uji yang ditinjau.
10. Pada akhirnya, user akan ditawarkan opsi untuk menyimpan solusi (*save*). Penyimpanan solusi akan diletakkan di folder bernama test/. Jika belum ada maka akan dibuat foldernya terlebih dahulu.

Graphical User Interface (GUI) :

1. Program akan dimulai saat *user* menjalankan command `python gui.py` yang akan memulaikan GUI untuk berjalan. Akan ditampilkan dua tombol yaitu *input manual* atau *input file text*.
2. Setelah *user* menginput *board* (manual atau file .txt), maka *board* akan di proses oleh fungsi `processBoard()`. Fungsi tersebut akan memeriksa apakah papan tersebut valid dan akan melakukan pengelompokkan warna seperti apa yang dilakukan saat di terminal. Jika *board* sudah valid maka akan dicarikan solusi untuk permasalahan *queens* tersebut.
3. Jika solusinya ditemukan, maka akan divisualisasikan dengan fungsi `drawBoard()` dengan bantuan fungsi `self.canvas.create_rectangle()` dengan bantuan warna-warna dari fungsi `getColour()`. Posisi *queen* dengan gambarnya akan ditampilkan dengan fungsi `self.canvas.create_image()`.
4. Pada akhirnya, *user* memiliki opsi untuk save solusi dengan fungsi `saveSolution()`. *User* akan diberi opsi untuk memilih format *save* (antara .png atau .txt).

BAB II
SOURCE CODE PROGRAM

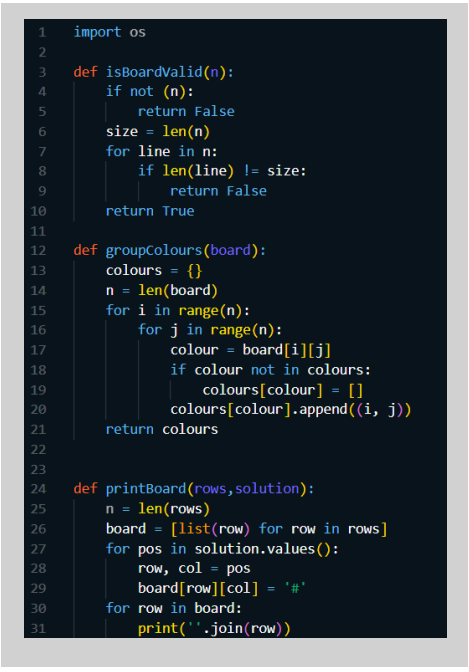
2.1 Penjelasan Struktur Program

Program ini menggunakan bahasa pemrograman Python karena sintaks yang sederhana, memiliki banyak library bawaan yang memudahkan pembuatan program, dan juga mudahnya pembuatan *Graphical User Interface* (GUI). Struktur folder program saya dapat dilihat pada tulisan di bawah ini.

```
├── assets/
├── bin/
├── doc/
├── input/
├── src/
│   └── board.py
│   └── file_processing.py
│   └── gui.py
│   └── main.py
│   └── queens.py
└── test/
    └── GUI/
    └── terminal/
```

2.2 *Source Code* Program

2.2.1 File board.py



```python
import os

def isBoardValid(n):
    if not (n):
        return False
    size = len(n)
    for line in n:
        if len(line) != size:
            return False
    return True

def groupColours(board):
    colours = {}
    n = len(board)
    for i in range(n):
        for j in range(n):
            colour = board[i][j]
            if colour not in colours:
                colours[colour] = []
            colours[colour].append((i, j))
    return colours


def printBoard(rows,solution):
    n = len(rows)
    board = [list(row) for row in rows]
    for pos in solution.values():
        row, col = pos
        board[row][col] = '#'
    for row in board:
        print(''.join(row))
```

2.2.2 File file_processing.py

```python
import os
from PIL import Image, ImageDraw

base = os.path.abspath(os.path.join(os.path.dirname(__file__), os.pardir))
palette = [
    "#C6E0B4", "#FFE699", "#A9D18E", "#F4B6B6", "#9DC3E6",
    "#B4A7D6", "#D996B6", "#F9CB9C", "#A2C4C9", "#D9D9D9",
    "#93C47D", "#FFD966", "#6FA8DC", "#E06666", "#8E7CC3",
    "#C27BA0", "#F6B26B", "#76A5AF", "#B7B7B7", "#B6D7A8",
]
queenImg = os.path.join(base, "assets", "crown.png")

def readBoard(filePath):
    file = open(filePath, "r")
    lines = []

    for line in file:
        line = line.strip()
        if line != '':
            lines.append(line)
    file.close()
    return lines

def saveSolution(path, oriBoard, solution, time, count):
    board = [list(row) for row in oriBoard]

    for pos in solution.values():
        row, col = pos
        board[row][col] = '#'

    f = open(path,'w')
    for row in board:
        f.write(''.join(row) + '\n')

    f.write(f"\nWaktu eksekusi: {time} ms\n")
    f.write(f"Banyak kasus yang ditinjau: {count} kasus\n")
```

```python
    f.close()


def saveImage(board, solution, path):
    crown = Image.open(queenImg).convert("RGBA")

    n = len(board)
    cell = max(20, min(48, 600 // max(1, n)))
    size = n * cell

    img = Image.new("RGB", (size, size), "white")
    draw = ImageDraw.Draw(img)

    colourMap = {}
    queens = set(solution.values())

    def color(ch):
        key = ch.upper()
        if key not in colourMap:
            colourMap[key] = palette[len(colourMap) % len(palette)]
        return colourMap[key]

    sizeCrown = int(cell * 0.7)
    crownResize = crown.resize((sizeCrown, sizeCrown), Image.LANCZOS)

    for r in range(n):
        for c in range(n):
            x1, y1 = c * cell, r * cell
            x2, y2 = x1 + cell, y1 + cell
            fill = color(board[r][c])
            draw.rectangle([x1, y1, x2, y2], fill=fill, outline="#222", width=2)
            if (r, c) in queens:
                if crownResize:
                    cx = int(x1 + (cell - crownResize.width) / 2)
```

```
                    cy = int(y1 + (cell - crownResize.height) / 2)
                    img.paste(crownResize, (cx, cy), crownResize)
                else:
                    draw.text((x1 + cell * 0.35, y1 + cell * 0.2), "#", fill="#111")

    img.save(path)
    return True

```

2.2.3 File main.py

```
import os
import time
from board import isBoardValid, groupColours, printBoard
from file_processing import readBoard, saveSolution
from queens import solveQueenPositions

def main():
    base = os.path.abspath(os.path.join(os.path.dirname(__file__), os.pardir))
    while True:
        file = input("Masukkan nama file (.txt) dari folder input: ")
        filePath = os.path.join(base,"input", file)

        if not os.path.exists(filePath):
            print("File tidak ditemukan!\n")
            continue

        board = readBoard(filePath)
        if not isBoardValid(board):
            print("Ukuran board queen tidak valid! (N x N)\n")
            continue

        colours = groupColours(board)

        print("\n\nMencari solusi untuk board pada file:", file,"\n")

        start = time.perf_counter()
        found, solution, count = solveQueenPositions(colours)
        end = time.perf_counter()
```

```
        elapseTime = (end - start) * 1000

        if found :
            print("Solusi ditemukan!\nMenampilkan solusi : \n")
            printBoard(board, solution)

            print(f"\n\nWaktu pencarian: {round(elapseTime)} ms\n")
            print(f"Banyak kasus yang ditinjau: {count} kasus\n")
            choice = input("Apakah Anda ingin menyimpan solusi? (Y/N): ")

            if choice.lower() == 'y':
                fmt = input("Pilih format (1=txt, 2=png): ").strip()
                if fmt == '2':
                    img_dir = os.path.join(base, "test", "image")
                    os.makedirs(img_dir, exist_ok=True)
                    pngPath = os.path.join(img_dir, os.path.splitext(file)[0] + ".png")
                    if saveImage(board, solution, pngPath):
                        print(f"Solusi PNG disimpan di {pngPath}\n")
                else:
                    txt_dir = os.path.join(base, "test", "text")
                    os.makedirs(txt_dir, exist_ok=True)
                    savePath = os.path.join(txt_dir, file)
                    saveSolution(savePath, board, solution, round(elapseTime), count)
                    print("Solusi TXT disimpan dalam folder test/text\n")
        else :
            print("Solusi tidak ditemukan!\n")
```

```
49
50
51  def printBanner():
52      print("-------------------- Selamat datang di program QUEENS LinkedIn SOLVER --------------------")
53      print("  ___   __ __    ___   ___ ____   _____      _____ ___   _      __ __    ___  ____   ")
54      print(" /   \\ |  |  |  /  _]  /  _]|    \\ / ___/     / ___/ /   \\ | |    |  |  |  /  _]|    \\ ")
55      print("|     ||  |  | /  [_  /  [_ |  _  (   \\_     (   \\_ |     || |    |  |  | /  [_ |  D  )")
56      print("|  Q  ||  |  ||    _]|    _]|  |  |\\__  |     \\__  ||  O  || |___ |  |  ||    _]|    / ")
57      print("|     ||  :  ||   [_ |   [_ |  |  |/  \\ |     /  \\ ||     ||     ||  :  ||   [_ |    \\ ")
58      print("|     ||     ||     ||     ||  |  |\\    |     \\    ||     ||     | \\   / |     ||  .  \\")
59      print(" \\__,_| \\__,_||_____||_____||__|__| \\___|      \\___| \\___/ |_____|  \\_/  |_____||__|\\_|")
60      print("\n\n")
61
62
63  if __name__ == "__main__":
64      printBanner()
65      main()
```

2.2.4 File queens.py

```
1   import itertools
2
3   def isQueenValid(pos):
4       n = len(pos)
5
6       for i in range(n):
7           row1, col1 = pos[i]
8
9           for j in range(i+1, n):
10              row2, col2 = pos[j]
11              if row1 == row2:
12                  return False
13
14              if col1 == col2:
15                  return False
16
17              if abs(row1 - row2) <= 1 and abs(col1 - col2) <= 1: # Ini buat cek diagonal bersebelahan
18                  return False
19
20      return True
21
22  def backtrack(index,colours, positions, queen):
23      if index == len(colours):
24          return True,0
25
26      count = 0
27
28      for pos in positions[index]:
29          count += 1
30
31          if isQueenValid(queen + [pos]):
32              queen.append(pos)
33
34              found, count1 = backtrack(index + 1, colours, positions, queen)
35              count += count1
```

```
                if found:
                    return True, count

                queen.pop()

        return False,count

    def solveQueenPositions(area):
        orderedItems = list(area.items())
        orderedItems.sort(key=lambda x: len(x[1]))
        colours = [items[0] for items in orderedItems]
        positions = [list(items[1]) for items in orderedItems]
        queen = []
        found, count = backtrack(0, colours, positions, queen)
        if found :
            solution = {
                colours[i]: queen[i] for i in range (len(colours))
            }
            return True, solution, count
        return False, None, count
```

2.2.5 File gui.py

```
import os
import time
import tkinter as tk
import customtkinter as ctk
from tkinter import filedialog, messagebox
from PIL import Image, ImageDraw, ImageTk

from board import isBoardValid, groupColours
from file_processing import readBoard, saveSolution, saveImage
from queens import solveQueenPositions

ctk.set_appearance_mode("light")
ctk.set_default_color_theme("blue")

basePath = os.path.abspath(os.path.join(os.path.dirname(__file__), os.pardir))
crownPath = os.path.join(basePath, "assets", "crown.png")

colorPalette = [
    "#C6E0B4", "#FFE699", "#A9D18E", "#F4B6B6", "#9DC3E6",
    "#B4A7D6", "#D996B6", "#F9CB9C", "#A2C4C9", "#D9D9D9",
    "#93C47D", "#FFD966", "#6FA8DC", "#E06666", "#8E7CC3",
    "#C27BA0", "#F6B26B", "#76A5AF", "#B7B7B7", "#B6D7A8"
]

def getColour(colourMap, key):
    key = key.upper()
    if key not in colourMap:
        colourMap[key] = colorPalette[len(colourMap) % len(colorPalette)]
    return colourMap[key]

class QueensGUI(ctk.CTk):
    def __init__(self):
        super().__init__()

        self.title("Queens Solver Pro")
        self.geometry("600x250")
```

```
        self.configure(fg_color="#FFFFFF")
        self.resizable(False, False)

        self.curBoard = None
        self.curSol = None
        self.curInput = "solution"

        self.crownTk = None
        self.crownBase = Image.open(crownPath).convert("RGBA")

        self.headerFrame = ctk.CTkFrame(self, fg_color="transparent")
        self.headerFrame.pack(pady=(30, 10), fill="x")

        self.titleLabel = ctk.CTkLabel(
            self.headerFrame,
            text="Queens Solver",
            font=ctk.CTkFont(family="Segoe UI", size=32, weight="bold")
        )
        self.titleLabel.pack()

        self.buttonFrame = ctk.CTkFrame(self, fg_color="transparent")
        self.buttonFrame.pack(pady=10)

        self.btnManual = ctk.CTkButton(
            self.buttonFrame, text="Input Manual",
            command=self.inputManual, corner_radius=12, font=("Segoe UI", 14, "bold"),
            height=40, width=160
        )
        self.btnManual.grid(row=0, column=0, padx=15)

        self.btnFile = ctk.CTkButton(
            self.buttonFrame, text="Input File TXT",
            command=self.inputFile, corner_radius=12, font=("Segoe UI", 14, "bold"),
            height=40, width=160
```

```
        )
        self.btnFile.grid(row=0, column=1, padx=15)

        self.mainContainer = ctk.CTkFrame(self, fg_color="#F0F2F5", corner_radius=20)

        self.canvas = tk.Canvas(
            self.mainContainer,
            bg="#F0F2F5",
            highlightthickness=0,
            bd=0
        )
        self.canvas.pack(pady=20, padx=20)

        self.saveButton = ctk.CTkButton(
            self, text="Save Solution Image",
            command=self.saveSolution,
            fg_color="#28a745", hover_color="#218838",
            corner_radius=12, font=("Segoe UI", 14, "bold"),
            height=40
        )

    def inputFile(self):
        filePath = filedialog.askopenfilename(
            initialdir=os.path.join(basePath, "input"),
            filetypes=[("Text Files", "*.txt")]
        )
        if filePath:
            self.curInput = os.path.splitext(os.path.basename(filePath))[0]
            boardData = readBoard(filePath)
            self.processBoard(boardData)

```

```python
    def showDialog(self, n):
        result = {"data": None}

        dialog = ctk.CTkToplevel(self)
        dialog.title("Input Board Configuration")
        dialog.geometry("400x450")
        dialog.grab_set()
        dialog.resizable(False, False)

        label = ctk.CTkLabel(dialog, text=f"Masukkan konfigurasi board {n}x{n}:", font=("Segoe UI", 13, "bold"))
        label.pack(pady=(15, 5))

        textbox = ctk.CTkTextbox(dialog, width=350, height=250, border_width=2)
        textbox.pack(padx=20, pady=10)

        def on_submit():
            result["data"] = textbox.get("1.0", "end-1c")
            dialog.destroy()

        def on_cancel():
            dialog.destroy()

        btn_frame = ctk.CTkFrame(dialog, fg_color="transparent")
        btn_frame.pack(pady=15)

        ctk.CTkButton(btn_frame, text="Submit", width=100, command=on_submit).grid(row=0, column=0, padx=10)
        ctk.CTkButton(btn_frame, text="Cancel", width=100, fg_color="#aaaaaa", hover_color="#888888", command=on_cancel).grid(row=0, column=1, padx=10)

        self.wait_window(dialog)
        return result["data"]

```

```python
    def inputManual(self):
        inputSize = ctk.CTkInputDialog(text="Masukkan ukuran N:", title="Board Size").get_input()

        if not inputSize or not inputSize.isdigit():
            return

        boardSize = int(inputSize)
        inputBoard = self.showDialog(boardSize)

        if inputBoard:
            board = inputBoard.strip().split("\n")
            if len(board) != boardSize:
                messagebox.showerror("Error", f"Jumlah baris harus {boardSize}.")
                return

            self.curInput = f"manual_{boardSize}x{boardSize}"
            self.processBoard(board)

    def processBoard(self, boardData):
        if not isBoardValid(boardData):
            messagebox.showerror("Error", "Board tidak valid!")
            return

        colourGroups = groupColours(boardData)
        startTime = time.perf_counter()
        isFound, solutionData, caseCount = solveQueenPositions(colourGroups)
        endTime = time.perf_counter()
        elapsedTime = (endTime - startTime) * 1000

        if not isFound:
            messagebox.showinfo("Result", "Solusi tidak ditemukan.")
            return

        self.curBoard = boardData
        self.curSol = solutionData
```

```
        self.lastElapsed = round(elapsedTime)
        self.lastCaseCount = caseCount

        self.drawBoard()
        self.mainContainer.pack(pady=20, padx=40)
        self.saveButton.pack(pady=(10, 30))

        self.update_idletasks()
        newHeight = self.winfo_reqheight()
        self.geometry(f"600x{newHeight}")

        messagebox.showinfo("Success", f"Selesai dalam {round(elapsedTime)} ms\nKasus: {caseCount}")

    def drawBoard(self):
        self.canvas.delete("all")
        nSize = len(self.curBoard)

        maxCellSize = 400 // nSize
        cellSize = min(50, maxCellSize)
        totalPixelSize = nSize * cellSize

        self.canvas.config(width=totalPixelSize, height=totalPixelSize)

        colourMap = {}
        queenPositions = set(self.curSol.values())
        pilCrown = Image.open(crownPath).convert("RGBA")
        resizedCrown = pilCrown.resize((int(cellSize * 0.7), int(cellSize * 0.7)), Image.LANCZOS)
        self.crownTk = ImageTk.PhotoImage(resizedCrown)
        self.crownBase = self.crownBase

        for r in range(nSize):
            for c in range(nSize):
                x1, y1 = c * cellSize, r * cellSize
                x2, y2 = x1 + cellSize, y1 + cellSize
```

```

                cellFill = getColour(colourMap, self.curBoard[r][c])

                self.canvas.create_rectangle(
                    x1, y1, x2, y2,
                    fill=cellFill,
                    outline="#FFFFFF",
                    width=2
                )

                if (r, c) in queenPositions:
                    self.canvas.create_image(
                        (x1 + x2) / 2, (y1 + y2) / 2,
                        image=self.crownTk,
                    )


    def saveSolution(self):
        if not self.curBoard or not self.curSol:
            return
        imgDir = os.path.join(basePath, "test", "image")
        txtDir = os.path.join(basePath, "test", "text")
        os.makedirs(imgDir, exist_ok=True)
        os.makedirs(txtDir, exist_ok=True)
        defaultStem = self.curInput

        choice = {"val": None}
        dlg = ctk.CTkToplevel(self)
        dlg.title("Simpan sebagai")
        dlg.resizable(False, False)
        dlg.grab_set()

```

```
        def choosePNG():
            choice["val"] = "png"
            dlg.destroy()

        def chooseTXT():
            choice["val"] = "txt"
            dlg.destroy()

        def chooseCancel():
            choice["val"] = None
            dlg.destroy()

        ctk.CTkLabel(dlg, text="Pilih format penyimpanan:", font=("Segoe UI", 14, "bold")).pack(padx=20, pady=(15, 10))
        btn_frame = ctk.CTkFrame(dlg, fg_color="transparent")
        btn_frame.pack(pady=(0, 15))

        ctk.CTkButton(btn_frame, text="PNG", width=90,
                      command=choosePNG).grid(row=0, column=0, padx=6)
        ctk.CTkButton(btn_frame, text="TXT", width=90,
                      command=chooseTXT).grid(row=0, column=1, padx=6)
        ctk.CTkButton(btn_frame, text="Cancel", width=90, fg_color="#aaaaaa", hover_color="#888888",
                      command=lambda: (choice.update(val=None), dlg.destroy())).grid(row=0, column=2, padx=6)

        dlg.wait_window()
        if choice["val"] is None:
            return

        if choice["val"] == "png":
            targetPath = os.path.join(imgDir, defaultStem + ".png")
            saveImage(self.curBoard, self.curSol, targetPath)
            messagebox.showinfo("Saved", f"Berhasil disimpan ke {targetPath}!")
        else:
            targetPath = os.path.join(txtDir, defaultStem + ".txt")
            saveSolution(targetPath, self.curBoard, self.curSol, self.lastElapsed, self.lastCaseCount)
```

```
                messagebox.showinfo("Saved", f"Berhasil disimpan ke {targetPath}!")

if __name__ == "__main__":
    app = QueensGUI()
    app.mainloop()
```

BAB III

MASUKAN DAN LUARAN PROGRAM

3.1 Test Case 1

<table>
<tr><td>test1.txt
Kasus dari spesifikasi laporan tucil1</td><td>

```
AAABBCCCD
ABBBBCECD
ABBBDCECD
AAABDCCCD
BBBBDDDDD
FGGGDDHDD
FGIGDDHDD
FGIGDDHDD
FGGGDDHHH
```

</td></tr>
<tr><td colspan="2">TERMINAL</td></tr>
<tr><td>Tampilan awal dan input file</td><td>

```
------------------- Selamat datang di program QUEENS LinkedIn SOLVER --------------------
 ___   __ __    ___    ___  ____   _____      _____  ___   _      __ __    ___  ____
/   \ |  |  |  /  _]  /  _]|    \ / ___/     / ___/ /   \ | |    |  |  |  /  _]|    \
|     ||  |  | /  [_  /  [_ |  _  (   \_     (   \_ |     || |    |  |  | /  [_ |  D  )
|  Q  ||  |  ||    _]|    _]|  |  |\__  |     \__  ||  O  || |___ |  |  ||    _]|    /
|     ||  :  ||   [_ |   [_ |  |  |/  \ |     /  \ ||     ||     ||  :  ||   [_ |    \
|     ||     ||     ||     ||  |  |\    |     \    ||     ||     | \   / |     ||  .  \
 \___,_| \__,_||_____||_____||__|__| \___|      \___| \___/ |_____|  \_/  |_____||__|\_|


Masukkan nama file (.txt) dari folder input: test1.txt
```

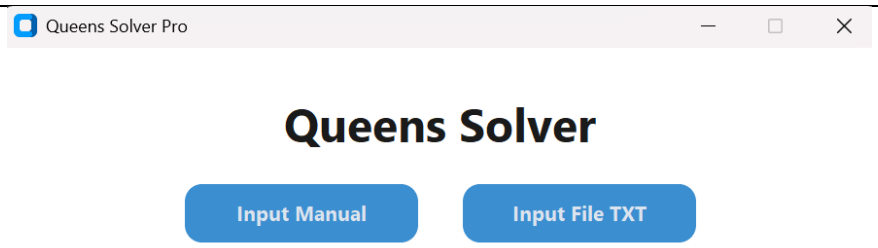
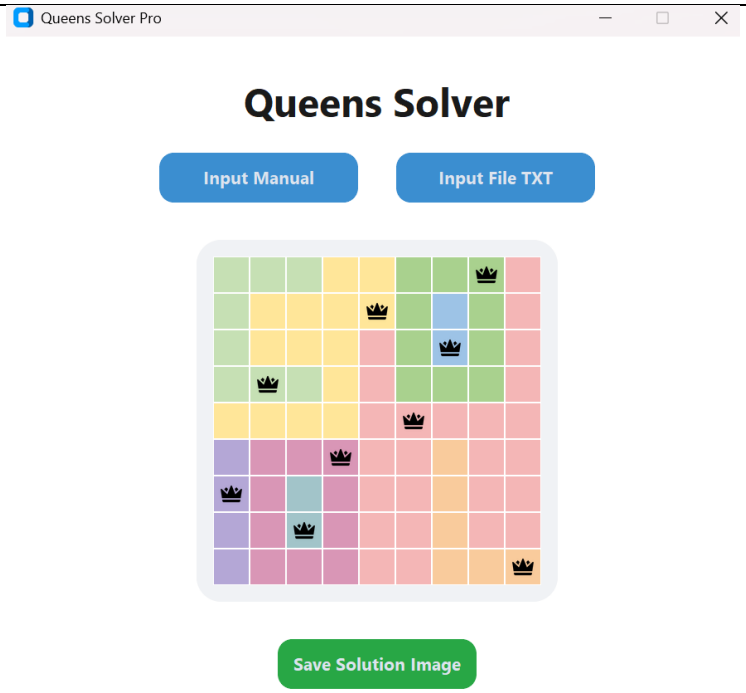
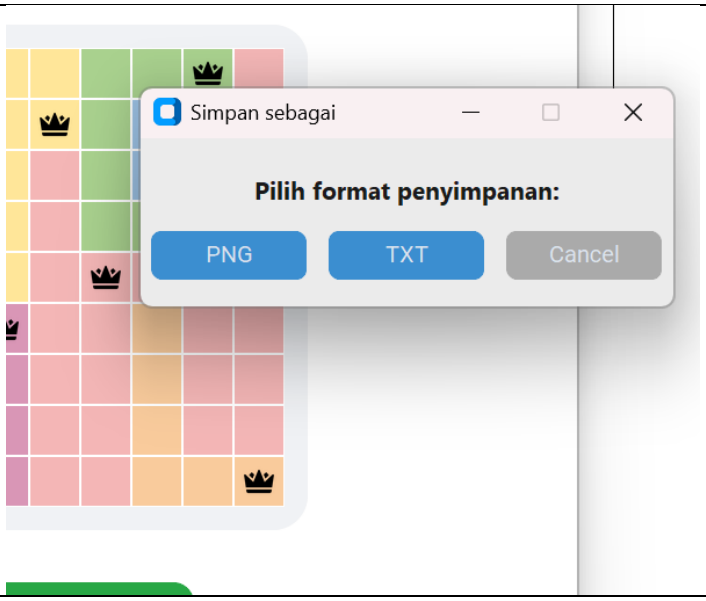
</td></tr>
<tr><td>Pencarian solusi dan keluarannya serta opsi save</td><td>

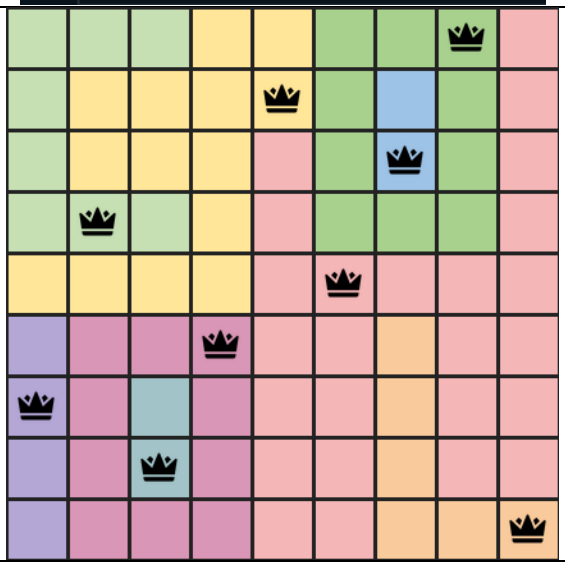
```
Mencari solusi untuk board pada file: test1.txt

Solusi ditemukan!
Menampilkan solusi :

AAABBCC#D
ABBB#CECD
ABBBDC#CD
A#ABDCCCD
BBBBD#DDD
FGG#DDHDD
#GIGDDHDD
FG#GDDHDD
FGGGDDHH#


Waktu pencarian: 3 ms

Banyak kasus yang ditinjau: 843 kasus

Apakah Anda ingin menyimpan solusi? (Y/N):
```

</td></tr>
<tr><td>Pemilihan format save</td><td>

```
Apakah Anda ingin menyimpan solusi? (Y/N): Y
Pilih format (1=txt, 2=png): 1
Solusi TXT disimpan dalam folder test/text
```

</td></tr>
<tr><td colspan="2">GUI (GRAPHICAL USER INTERFACE)</td></tr>
</table>

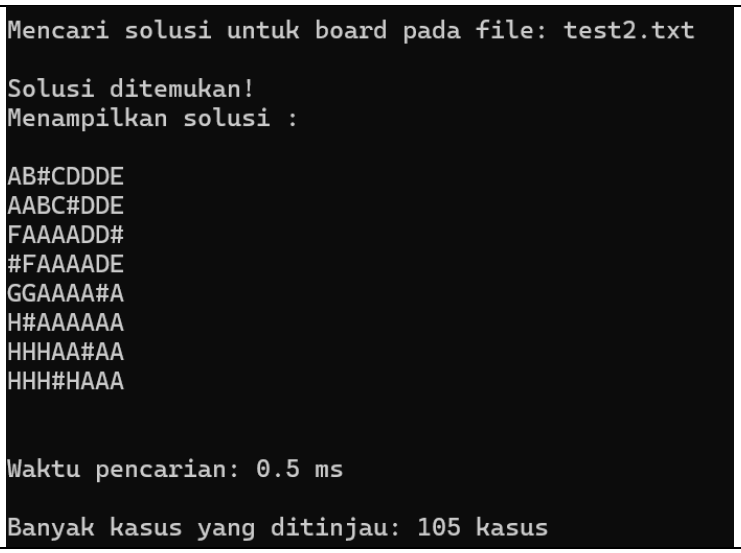
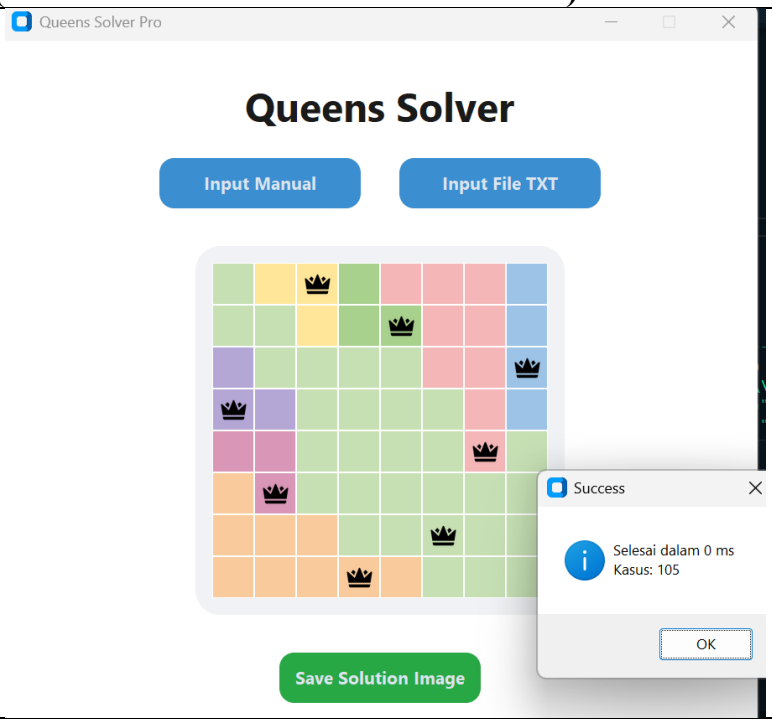
Tampilan awal dan input file (.txt) atau manual. Namun, untuk menghindari duplikasi dalam laporan ini maka akan digunakan input file	
Pencarian solusi dan keluarannya serta opsi *save*	
Pemilihan format *save* (.png atau .txt)	
Hasil Penyimpanan Solusi note : untuk format save dari terminal dan GUI sama	

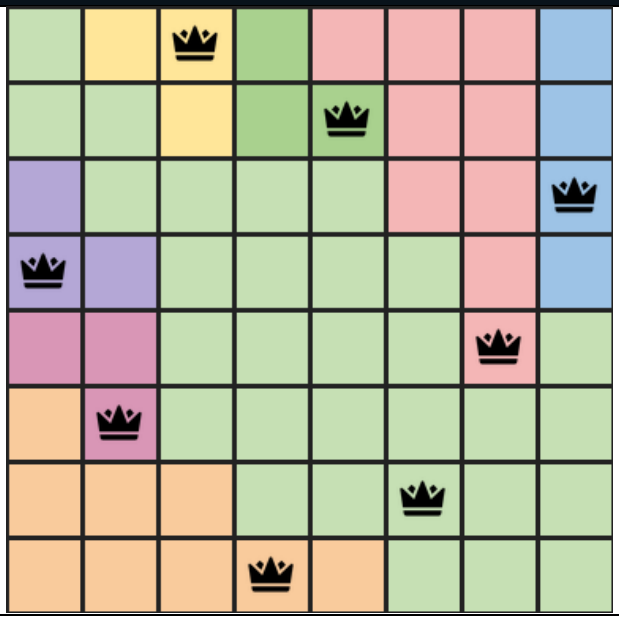
Hasil save format teks (.txt)	AAABBCC#D ABBB#CECD ABBBDC#CD A#ABDCCCD BBBBD#DDD FGG#DDHDD #GIGDDHDD FG#GDDHDD FGGGDDHH# Waktu eksekusi: 3 ms Banyak kasus yang ditinjau: 843 kasus
Hasil save format gambat (.png)	

3.2 Test Case 2

(Dari *test case* ini hingga selanjutnya tidak akan ditampilkan lagi untuk tampilan awal terminal dan GUI untuk menghindari data duplikat

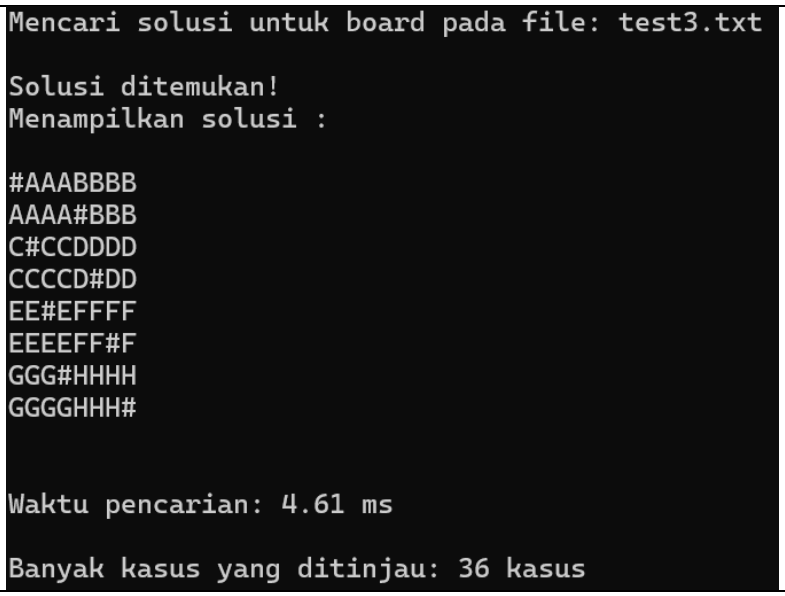
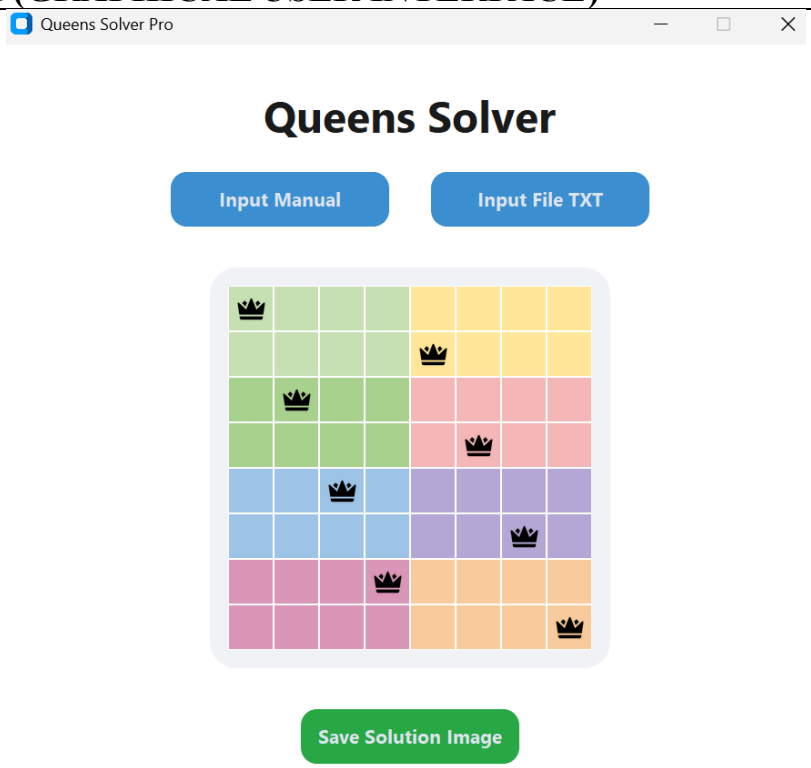
test2.txt Kasus normal dengan 8 *queen*	Tucil1_13524005 > input > test2 ABBCDDDE AABCCDDE FAAAADDE FFAAAADE GGAAAADA HGAAAAAA HHHAAAAA HHHHHAAA
TERMINAL	

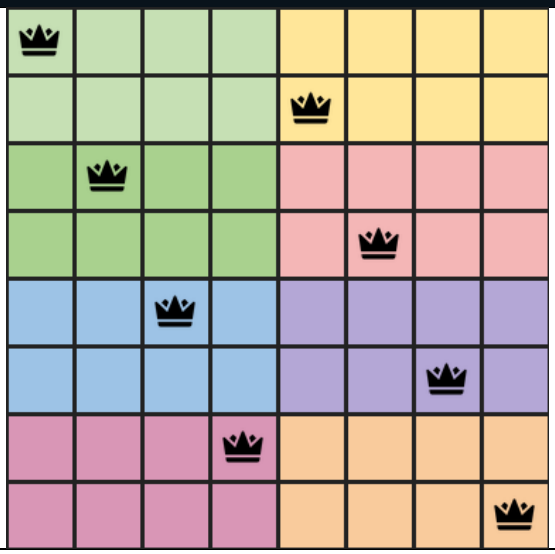
Pencarian solusi dan keluarannya serta opsi *save*		
GUI (GRAPHICAL USER INTERFACE)		
Pencarian solusi dan keluarannya serta opsi *save*		
Hasil Penyimpanan Solusi note : untuk format save dari terminal dan GUI sama		

Hasil save format teks (.txt)		Tucil1_13524005 > test > text > test2.txt AB#CDDDE AABC#DDE FAAAADD# #FAAAADE GGAAAA#A H#AAAAAA HHHAA#AA HHH#HAAA Waktu eksekusi: 0.5 ms Banyak kasus yang ditinjau: 105 kasus
Hasil save format gambat (.png)		

3.3 Test Case 3

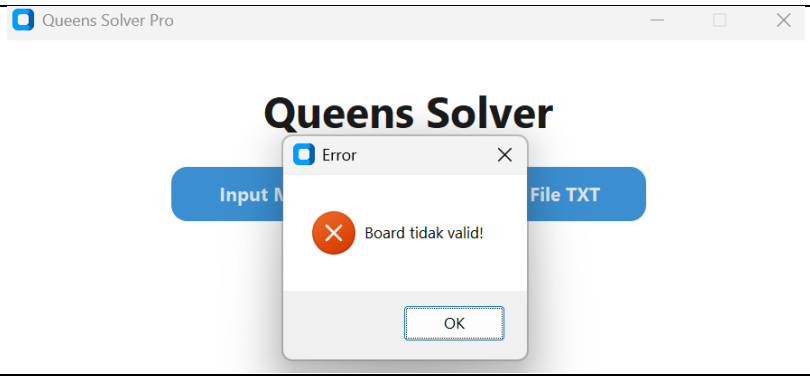
test3.txt Kasus normal dengan 8 *queen*		Tucil1_13524005 > input > test3.txt AAAABBBB AAAABBBB CCCCDDDD CCCCDDDD EEEEFFFF EEEEFFFF GGGGHHHH GGGGHHHH
TERMINAL		

Pencarian solusi dan keluarannya serta opsi *save*	
GUI (GRAPHICAL USER INTERFACE)	
Pencarian solusi dan keluarannya serta opsi *save*	
Hasil Penyimpanan Solusi note : untuk format save dari terminal dan GUI sama	

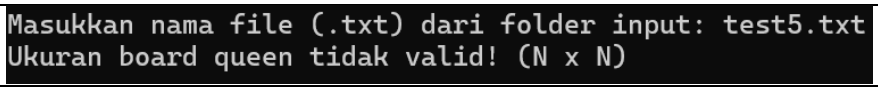
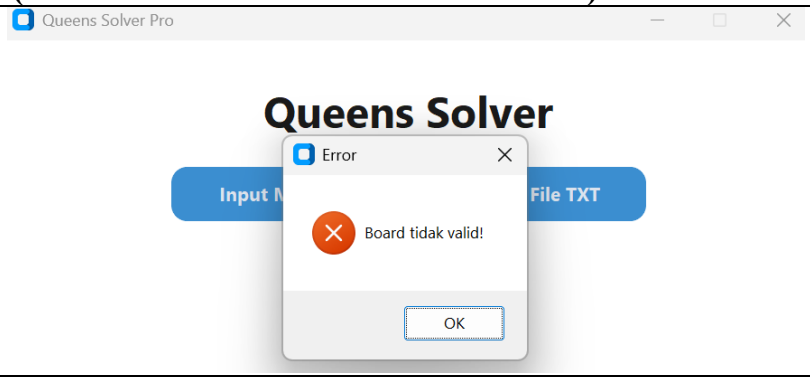
Hasil save format teks (.txt)	Tucil1_13524005 > test > text > test3.txt #AAABBBB AAAA#BBB C#CCDDDD CCCCD#DD EE#EFFFF EEEEFF#F GGG#HHHH GGGGHHH# Waktu eksekusi: 4.61 ms Banyak kasus yang ditinjau: 36 kasus
Hasil save format gambat (.png)	

3.4 Test Case 4

test4.txt Kasus dengan 3 *queen* tapi ukuran board tidak valid. Seharusnya berukuran N x N	AAABBCC AABBCCC ACCCCCC
TERMINAL	
Hasil saat mencoba memproses *board*	Masukkan nama file (.txt) dari folder input: test4.txt Ukuran board queen tidak valid! (N x N)
GUI (GRAPHICAL USER INTERFACE)	

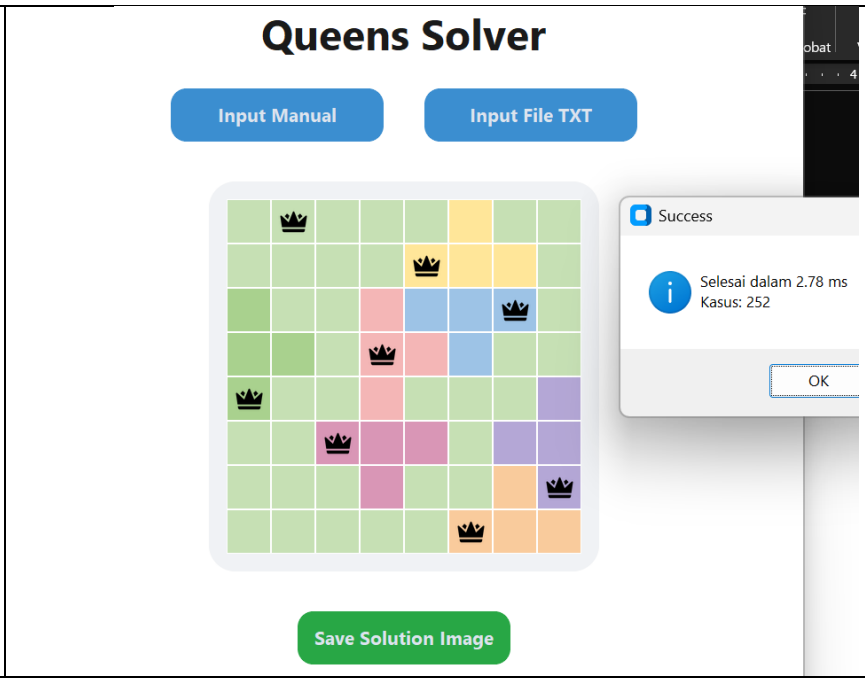
Pencarian solusi dan keluarannya saat *board* tidak valid	

3.5 Test Case 5

test5.txt Kasus dengan 3 *queen* tapi terdapat posisi-posisi yang hilang dalam board. Seharusnya berukuran N x N kosong	
TERMINAL	
Hasil saat mencoba memproses *board*	
GUI (GRAPHICAL USER INTERFACE)	
Pencarian solusi dan keluarannya saat *board* tidak valid	

3.6 Test Case 6

test6.txt Kasus normal dengan 8 *queen*		Tucil1_13524005 > input > test6.txt 1 AAAAABAA 2 AAAABBBA 3 CAADEEEA 4 CCADDEAA 5 CAADAAAF 6 AAGGGAFF 7 AAAGAAHF 8 AAAAAHHH
TERMINAL		
Pencarian solusi dan keluarannya serta opsi *save*		Solusi ditemukan! Menampilkan solusi : A#AAABAA AAAA#BBA CAADEE#A CCA#DEAA #AADAAAF AA#GGAFF AAAGAAH# AAAAA#HH Waktu pencarian: 8.68 ms Banyak kasus yang ditinjau: 252 kasus
GUI (GRAPHICAL USER INTERFACE)		

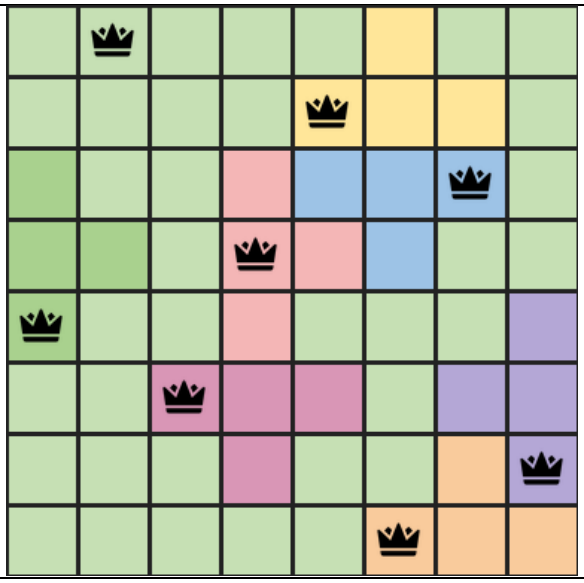
Pencarian solusi dan keluarannya serta opsi *save*	

Hasil Penyimpanan Solusi

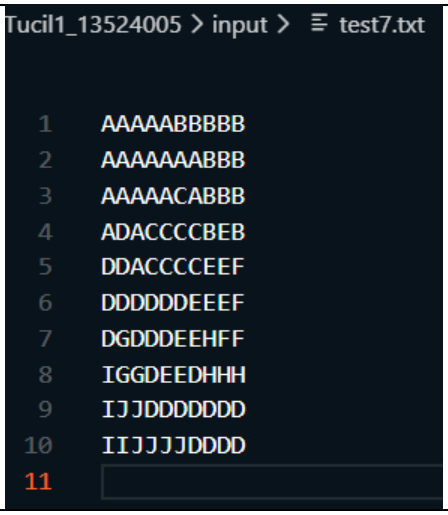
note : untuk format save dari terminal dan GUI sama

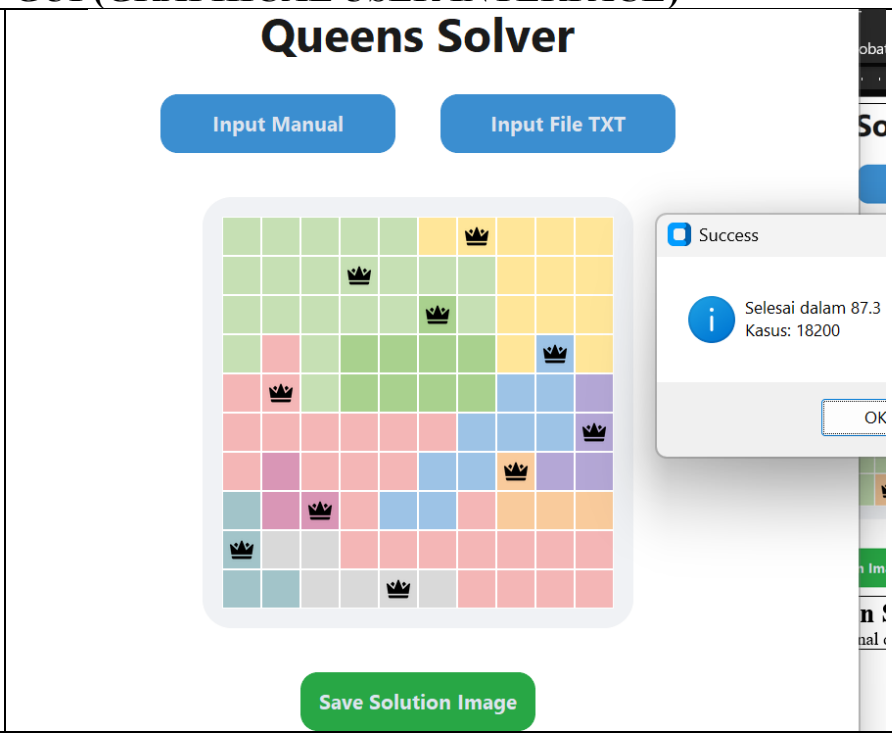
Hasil save format teks (.txt)

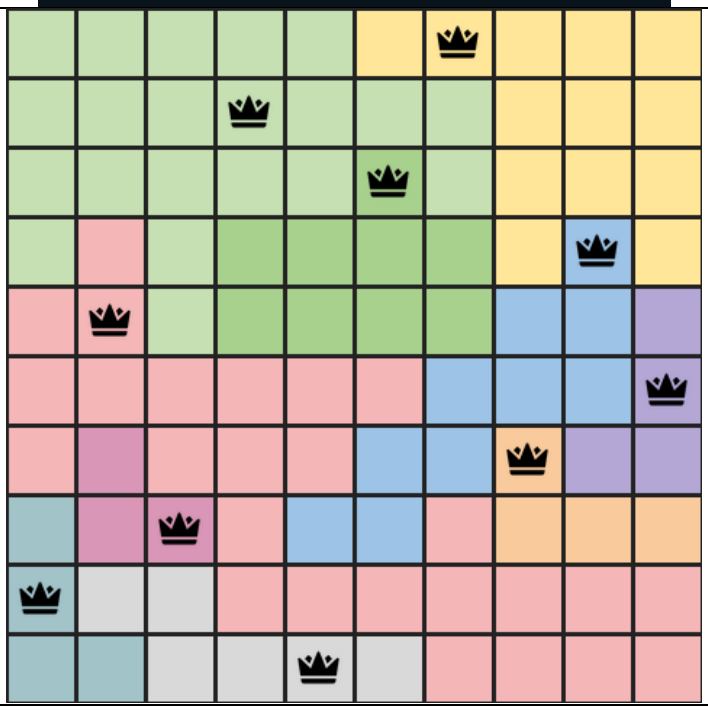
```
A#AAABAA
AAAA#BBA
CAADEE#A
CCA#DEAA
#AADAAAF
AA#GGAFF
AAAGAAH#
AAAAA#HH

Waktu eksekusi: 8.68 ms
Banyak kasus yang ditinjau: 252 kasus

```

Hasil save format gambat (.png)	

3.7 Test Case 7

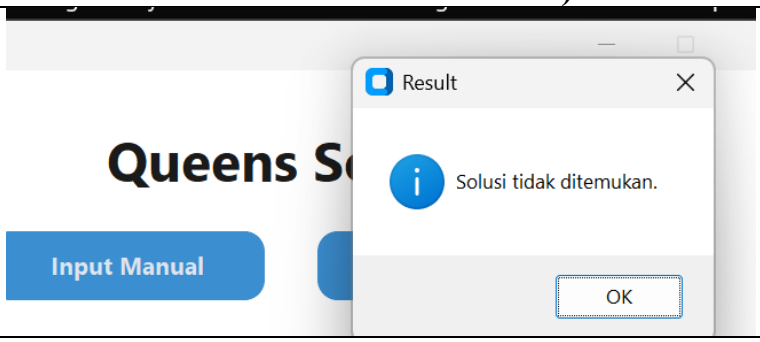
test7.txt Kasus normal dengan 10 *queen*	
TERMINAL	

Pencarian solusi dan keluarannya serta opsi *save*		Solusi ditemukan! Menampilkan solusi : AAAAAB#BBB AAA#AAABBB AAAAA#ABBB ADACCCCB#B D#ACCCCEEF DDDDDDEEE# DGDDDEE#FF IG#DEEDHHH #JJDDDDDDD IIJJ#JDDDD Waktu pencarian: 150.97 ms Banyak kasus yang ditinjau: 18200 kasus
GUI (GRAPHICAL USER INTERFACE)		
Pencarian solusi dan keluarannya serta opsi *save*		
Hasil Penyimpanan Solusi note : untuk format save dari terminal dan GUI sama		

Hasil save format teks (.txt)	Tucil1_13524005 > test > text > test7.txt 1 AAAAAB#BBB 2 AAA#AAABBB 3 AAAAA#ABBB 4 ADACCCCB#B 5 D#ACCCCEEF 6 DDDDDDEEE# 7 DGDDDEE#FF 8 IG#DEEDHHH 9 #JJDDDDDDD 10 IIJJ#JDDDD 11 12 Waktu eksekusi: 150.97 ms 13 Banyak kasus yang ditinjau: 18200 kasus 14
Hasil save format gambat (.png)	

3.8 Test Case 8

test8.txt Kasus dengan 4 *queen* tapi tidak terdapat solusi yang ditemukan.	Tucil1_13524005 > input > test8.txt 1 AABC 2 AABC 3 DDEE 4 DDEE
TERMINAL	

Hasil saat mencoba memproses *board*		Mencari solusi untuk board pada file: test8.txt Solusi tidak ditemukan! Waktu pencarian: 0.37 ms Banyak kasus yang ditinjau: 6 kasus
GUI (GRAPHICAL USER INTERFACE)		
Pencarian solusi dan keluarannya saat tidak ditemukan solusi		

BAB IV
LAMPIRAN

4.1 Pranala ke Repository Program

https://github.com/GeraldoA07/Tucil1_13524005

4.2 Tabel Spesifikasi Program

No	Poin	Ya	Tidak
1	Program berhasil di kompilasi tanpa kesalahan	✓	
2	Program berhasil di jalankan	✓	
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	✓	

4.3 Pernyataan Tidak Melakukan Kecurangan

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

Geraldo Artemius
(13524005)