

TUGAS KECIL 1 STRATEGI ALGORITMA

**Penyelesaian Permainan Queen's Linkedin
dengan Algoritma Brute Force**



Disusuh Oleh :
Valentino Daniel Kusumo - 13524104

INSTITUT TEKNOLOGI BANDUNG
2026

DAFTAR ISI

DAFTAR ISI.....	2
BAB 1.....	3
1. Permainan Queen's LinkedIn.....	3
2. Algoritma Brute Force.....	3
3. Algoritma Penyelesaian Permainan Queen's LinkedIn dengan Metode Brute Force.....	3
BAB 2.....	7
1. Link Repository Program.....	7
2. Source Code Program.....	7
BAB 3.....	26
1. Kasus Uji 1 - valid.....	26
2. Kasus Uji 2 - valid.....	28
3. Kasus Uji 3 - invalid (daerah terlalu banyak).....	30
4. Kasus Uji 4 - invalid (board tidak lengkap).....	30
5. Kasus Uji 5 - invalid (bukan alphabet).....	31
6. Kasus Uji 6 - valid.....	32
BAB IV.....	35

BAB 1

DESKRIPSI PERMAINAN DAN ALGORITMA

1. Permainan Queen's LinkedIn

Permainan Queen's LinkedIn merupakan permainan logika berbasis grid yang terinspirasi dari variasi *N-Queens*, tetapi memiliki aturan tambahan yang membuat permainan ini menjadi lebih kompleks. Pada permainan ini, pemain diminta untuk menempatkan sejumlah *queen* pada papan berukuran $n \times n$ dengan beberapa batasan tertentu.

Tujuan utama permainan ini adalah mengisi kotak - kotak yang terdapat dalam papan permainan sehingga terdapat 1 *queen* di setiap baris, kolom, dan wilayah berwarna tanpa ada *queen* yang saling bersentuhan, bahkan secara diagonal jarak dekat.

2. Algoritma Brute Force

Algoritma *brute force* adalah metode pemecahan suatu masalah yang menyelesaikan persoalan secara langsung (*straightforward*). Algoritma ini memecahkan persoalan secara sederhana, langsung, serta dengan cara yang jelas dan mudah dipahami. Tidak hanya itu, tetapi algoritma *brute force* menjamin akan menemukan solusi yang optimal jika solusi tersebut ada.

Meskipun begitu, algoritma *brute force* umumnya tidak “cerdas” dan tidak sangkil, karena ia membutuhkan *cost* komputasi yang besar dan waktu yang lama dalam penyelesaiannya. Oleh karenanya, algoritma *brute force* lebih cocok untuk persoalan yang ukuran masukannya kecil.

3. Algoritma Penyelesaian Permainan Queen's LinkedIn dengan Metode Brute Force

Algoritma *brute force* yang diimplementasikan pada permainan Queen's LinkedIn memiliki tahapan sebagai berikut :

1. Tentukan ukuran papan permainan (*dengan membaca jumlah baris input konfigurasi permainan*).
2. Representasikan posisi *queen* menggunakan array 1 dimensi, di mana indeks array menunjukkan baris dan nilai array menunjukkan kolom.
3. Cari semua kemungkinan kombinasi posisi *queen*.
4. Setiap kombinasi akan diperiksa apakah terdapat *queen* dalam baris atau kolom yang sama, apakah terdapat *queen* dalam wilayah daerah yang sama, dan apakah terdapat *queen* yang saling bersebelahan.
5. Jika semua batasan dipenuhi, maka solusi telah ditemukan dan proses pencarian berhenti.
6. Jika tidak dipenuhi, proses akan lanjut ke kombinasi berikutnya.
7. Jika semua kemungkinan telah dicoba dan tidak ada solusi, maka permainan dinyatakan tidak memiliki solusi.

Berikut adalah *pseudocode* untuk penyelesaian permainan Queen's LinkedIn menggunakan algoritma *brute force (murni)* :

```

function isValid(pos: list of integer, letters: matrix, n: integer) -> boolean
    i traversal [0..n]
        j traversal [0..n]
            if pos[i] = pos[j] then
                -> false
            if letters[i][pos[i]] =
                letters[j][pos[j]] then
                    -> false
            if abs(i - j) <= 1 and abs(pos[i] -
                pos[j]) <= 1 then
                    -> false
    -> true

** Board = Kelas buatan

function solve(board: Board) -> boolean, integer, integer
    n <- board.row
    letters <- board.board
    caseCount <- 0
    startTime <- time.time()

```

```

total <- n ^ n
num traversal [0..total]
    caseCount <- caseCount + 1
    i traversal [0..n]
        pos[i] <- 0
    x <- num
    i traversal [n-1..-1]
        pos[i] <- x % n
        x <- x // n

    if isValid(pos, letters, n) then
        i traversal [0..n]
            board.update(i, pos[i])

    totalTime = (time.time() - startTime)
                * 1000
    -> true, caseCount, totalTime

totalTime = (time.time() - startTime) * 1000
-> false, caseCount, totalTime

```

Adapun dalam praktiknya, algoritma *brute force* murni membutuhkan waktu yang sangat lama (*kompleksitas waktu $n!$*) sehingga tidak mungkin digunakan untuk menyelesaikan masalah dengan masukkan yang besar. Oleh karena itu, dibuatlah solusi yang lebih optimal menggunakan *backtracking*.

Backtracking merupakan pengembangan dari *brute force* yang tetap mencoba semua kemungkinan solusi, tetapi dengan cara yang lebih efisien. Pada permainan ini, *backtracking* bekerja dengan menempatkan queen secara bertahap pada setiap baris. Jika posisi tersebut melanggar aturan, maka proses akan kembali ke baris sebelumnya dan mencoba posisi lain.

Berikut *pseudocode* untuk algoritma *brute force* menggunakan *backtrack* :

```

function isValidBacktrack(pos: list of integer, letters:
matrix, row, col: integer) -> boolean
    i traversal [row]
        if pos[i] = pos[j] then
            -> false

```

```

if letters[i][pos[i]] = letters[j][pos[j]]
then
    -> false
if abs(i - j) <= 1 and abs(pos[i] - pos[j])
= 1 then
    -> false
-> true

function solve(board: Board) -> boolean, integer, integer
n <- board.row
letters <- board.board
caseCount <- 0
startTime <- time.time()

i traversal [0..n]
    pos[i] <- -1

function backtrack(row: integer) -> boolean
    if row = n then
        -> true
    col traversal [0..n]
        caseCount <- caseCount + 1
        if isValidBacktrack(pos,
            letters, row, col) then
                pos[row] <- col
                if backtrack(row+1)then
                    -> true
                pos[row] <- -1
    -> false

found <- backtrack(0)
if found then
    i traversal [0..n]
        board.update(i, pos[i])

totalTime = (time.time() - startTime) * 1000
-> true, caseCount, totalTime

totalTime = (time.time() - startTime) * 1000
-> false, caseCount, totalTime

```

BAB 2

SOURCE CODE PROGRAM

1. Link Repository Program

https://github.com/ValentinoDan/Tucil1_13524104.git

2. Source Code Program

Struktur program :

```
└── bin  
└── doc  
└── result  
└── src  
    ├── core  
    └── ui  
└── test
```

board.py

```
class Board:  
    def __init__(self, row, col):  
        self.row = row  
        self.col = col  
        self.board = [["" for _ in range(col)] for _ in  
                     range(row)]  
  
        # Update 1 posisi di board menjadi Queen  
    def update(self, row, col):  
        self.board[row][col] = "#"
```

helper.py

```
from .board import Board

# Kode warna
colors = {
    'A': 196, 'B': 202, 'C': 226, 'D': 46,
    'E': 51, 'F': 21, 'G': 93, 'H': 201,
    'I': 208, 'J': 118, 'K': 39, 'L': 129,
    'M': 220, 'N': 82, 'O': 27, 'P': 200,
    'Q': 214, 'R': 154, 'S': 75, 'T': 141,
    'U': 190, 'V': 49, 'W': 33, 'X': 165,
    'Y': 229, 'Z': 99
}

def validate(board):
    n = board.row

    # Jumlah daerah maksimal sama dengan sisi
    place = []
    for i in range(n):
        for j in range(n):
            if board.board[i][j].upper() not in place:
                place.append(board.board[i][j].upper())

    if len(place) > n:
        return False

    return True

def validateLine(line):
    if not line:
        return False, "Baris kosong tidak diperbolehkan"

    for char in line:
        if not char.isalpha():
            return False, f"Karakter wajib berupa alphabet"

    return True, ""

# Baca file dengan validasi
```

```
def readFile(file):
    try:
        with open(file, "r") as f:
            lines = [line.strip() for line in f if line.strip()]

        if not lines:
            return None, "File kosong!"

        n = len(lines)

        # Mengecek tiap baris
        for i, line in enumerate(lines):
            isValid, errorMsg = validateLine(line)
            if not isValid:
                return None, f"Baris {i+1}: {errorMsg}"

        # Mengecek board
        for line in lines:
            if len(line) != n:
                return None, f"Board harus berbentuk persegi."

        # Buat board
        board = Board(n, n)
        for i in range(n):
            for j in range(n):
                board.board[i][j] = lines[i][j]

        # Mengecek jumlah daerah
        if not validate(board):
            return None, f"Jumlah daerah tidak boleh lebih
dari {n}."

        return board, None

    except FileNotFoundError:
        return None, f"File {file} tidak ditemukan"
    except Exception as e:
        return None, f"Error membaca file: {str(e)}"
```

```

# Warna teks
def textColor(letter):
    number = colors.get(letter)
    return f"\033[38;5;{number}m{letter}\033[0m"

# Output solusi
def showResult(board):
    for i in range(board.row):
        for j in range(board.col):
            if board.board[i][j] == "#":
                print("#", end="")
            else:
                print(textColor(board.board[i][j]), end="")
    print()

```

solver.py

```

from .board import Board
import time

def isValid(pos, letters, n):
    for i in range(n):
        for j in range(i+1, n):
            if pos[i] == pos[j]: # 1 kolom
                return False
            if letters[i][pos[i]] == letters[j][pos[j]]: # tdk
                bisa di daerah sama
                return False
            if abs(i - j) <= 1 and abs(pos[i] - pos[j]) <= 1: # tdk boleh bersebelahan (diagonal dekat jd)
                return False

    return True

def isValidBacktrack(pos, letters, row, col):
    for i in range(row):
        prev = pos[i]

        if prev == col: # cek apakah sudah ada queen ditempat ini

```

```

        return False

    if letters[i][prev] == letters[row][col]: # cek daerah
        queen before
        return False

    if abs(i - row) <= 1 and abs(prev - col) <= 1: # cek
        apakah dekat
        return False

    return True

def solve(board, update=None, interval=None, backtracks=False):
    n = board.row
    letters = board.board # copy board
    caseCount = 0
    startTime = time.time()

    if backtracks:
        pos = [-1] * n

        def backtrack(row):
            nonlocal caseCount # caseCount global

            if row == n:
                return True

            for col in range(n):
                caseCount += 1

                if isValidBacktrack(pos, letters, row, col):
                    pos[row] = col

                    # Live update
                    if update and interval and caseCount %
                        interval == 0:
                        tempBoard = Board(n, n)
                        for i in range(n):
                            for j in range(n):

```

```

        tempBoard.board[i][j] =
letters[i][j]

        for i in range(n):
            if pos[i] != -1:
                tempBoard.update(i, pos[i])
        update(tempBoard)
        time.sleep(1e-6)

        if backtrack(row+1):
            return True

        pos[row] = -1 # gagal

    return False

found = backtrack(0)
if found:
    for i in range(n):
        board.update(i, pos[i])

    totalTime = (time.time() - startTime) * 1000
    return True, caseCount, totalTime

totalTime = (time.time() - startTime) * 1000
return False, caseCount, totalTime

else:
    total = n ** n

    for num in range(total):
        caseCount += 1

        pos = [0] * n # posisi queen
        x = num
        for i in range(n - 1, -1, -1):
            pos[i] = x % n
            x /= n

```

```

# Live update
if update and interval and caseCount % interval == 0:
    tempBoard = Board(n, n)
    for i in range(n):
        for j in range(n):
            tempBoard.board[i][j] = letters[i][j]

    for i in range(n):
        tempBoard.update(i, pos[i])
    update(tempBoard)

if isValid(pos, letters, n):
    for i in range(n):
        board.update(i, pos[i])

totalTime = (time.time() - startTime) * 1000
return True, caseCount, totalTime

totalTime = (time.time() - startTime) * 1000
return False, caseCount, totalTime

```

gui.py

```

import flet as ft
import os
from PIL import Image, ImageDraw, ImageFont
from core.board import Board
from core.solver import solve
from core.helper import readFile, validate, validateLine

def main(page: ft.Page):
    page.title = "Queen's Linkedin Game Solver"
    page.window.width = 1050
    page.window.height = 750
    page.padding = 30
    page.scroll = "auto"
    page.theme_mode = ft.ThemeMode.DARK

    # Text field untuk input board

```

```

boardInput = ft.TextField(
    cursor_color=ft.Colors.BLUE_200,
    multiline=True,
    max_lines=15,
    min_lines=9,
    width=350,
    label="Input Board",
    hint_text="Masukkan board atau load dari file",
    border_color=ft.Colors.BLUE_400,
    focused_border_color=ft.Colors.BLUE_700,
    border_radius=5
)

# Warna untuk tiap daerah
colors = {
    'A': '#EF4444', 'B': '#F97316', 'C': '#F59E0B', 'D': '#EAB308', 'E': '#84CC16', 'F': '#22C55E', 'G': '#10B981', 'H': '#14B8A6', 'I': '#06B6D4', 'J': '#0EA5E9', 'K': '#3B82F6', 'L': '#6366F1', 'M': '#8B5CF6', 'N': '#A855F7', 'O': '#D946EF', 'P': '#EC4899', 'Q': '#F43F5E', 'R': '#FB7185', 'S': '#FDA4AF', 'T': '#FCA5A5', 'U': '#FCD34D', 'V': '#BEF264', 'W': '#86EFAC', 'X': '#5EEAD4', 'Y': '#7DD3FC', 'Z': '#A5B4FC'
}

# Buat board grid
def createBoardGrid(board):
    n = board.row
    cellSize = min(40, 380 // n)

    grid = ft.Column(spacing=2,
                     horizontal_alignment=ft.CrossAxisAlignment.CENTER)

    for i in range(n):
        row = ft.Row(spacing=2,
                     alignment=ft.MainAxisAlignment.CENTER)
        for j in range(n):
            letter = board.board[i][j]
            isQueen = (letter == '#')

```

```
        if isQueen:
            bgcolor = "#FFD700"
            content = ft.Text("👑", size=cellSize*0.6,
                weight=ft.FontWeight.BOLD)
        else:
            bgcolor = colors.get(letter.upper(), "#94A3B8")
            content = ft.Text(size=cellSize*0.5,
                weight=ft.FontWeight.BOLD)

        cell = ft.Container(
            content=content,
            width=cellSize,
            height=cellSize,
            bgcolor=bgcolor,
            border_radius=3,
            alignment=ft.alignment.center,
        )
        row.controls.append(cell)
        grid.controls.append(row)

    return grid

# Grid Board
boardDisplay = ft.Column(spacing=10,
    horizontal_alignment=ft.CrossAxisAlignment.CENTER,
    scroll=ft.ScrollMode.AUTO)

# Text Result
result = ft.Text(size=12, selectable=True,
    weight=ft.FontWeight.W_500, text_align=ft.TextAlign.CENTER)

# Text untuk nama file
fileName = ft.Text("", size=12, color=ft.Colors.GREY_400)

# Result container
resultContainer = ft.Container(
    content=ft.Column([
```

```
        boardDisplay,
        ft.Divider(height=10, color=ft.Colors.TRANSPARENT),
        result
    ], spacing=5,
    horizontal_alignment=ft.CrossAxisAlignment.CENTER,
scroll=ft.ScrollMode.AUTO),
    padding=15,
    border_radius=8,
    bgcolor=ft.Colors.BLUE_100,
    border=ft.border.all(1, ft.Colors.GREY_300),
    width=480,
    height=550,
    animate=ft.Animation(300, "ease")
)

# checkbox backtrack
checkbox = ft.Checkbox(label = "Enable Backtracking (Faster &
Recommended)", value = False)

def validateBoard():
    if not boardInput.value or not boardInput.value.strip():
        return None, "Silakan masukkan konfigurasi board
terlebih dahulu!"

    lines = boardInput.value.strip().split("\n")
    n = len(lines)

    # Validasi tiap baris
    for i, line in enumerate(lines):
        isValid, errorMsg = validateLine(line)
        if not isValid:
            return None, f"Format board tidak valid!\nBaris
{i+1}: {errorMsg}"

    # Cek persegi
    for line in lines:
        if len(line) != n:
            return None, "Format board tidak valid!\nBoard
harus berbentuk persegi."
```

```
# Buat board
board = Board(n, n)
try:
    for i in range(n):
        for j in range(n):
            board.board[i][j] = lines[i][j]
except IndexError:
    return None, "Format board tidak valid!"

# Validasi jumlah daerah
if not validate(board):
    return None, "Format board tidak valid!"

return board, None

def exportTxt(e):
    board, error = validateBoard()
    boardDisplay.controls.clear()
    if error:
        result.value = error
        resultContainer.bgcolor = "#78350F"
        resultContainer.border = ft.border.all(1, "#F59E0B")
        result.color = "#FCD34D"
        page.update()
        return
    folder = "result" # folder save output
    if not os.path.exists(folder):
        os.makedirs(folder)

    files = os.listdir(folder)
    found, count, time = solve(board)
    filepath = os.path.join(folder, f"result{len(files)} + {1}.txt")

    with open(filepath, "w") as f:
        if found:
            for i in range(board.row):
                for j in range(board.col):
```

```
        f.write(board.board[i][j])
        f.write("\n")
        f.write("\n")
    else:
        f.write("Solusi tidak ditemukan\n")
    f.write(f"Waktu pencarian: {time:.2f} ms\n")
    f.write(f"Banyak kasus yang ditinjau: {count}
    kasus\n")

result.value = f"Berhasil diexport ke result{len(files)} +
1}.txt"
resultContainer.bgcolor = "#064E3B"
resultContainer.border = ft.border.all(1, "#10B981")
result.color = "#6EE7B7"
page.update()

def exportImage(e):
    board, error = validateBoard()
    boardDisplay.controls.clear()
    if error:
        result.value = error
        resultContainer.bgcolor = "#78350F"
        resultContainer.border = ft.border.all(1, "#F59E0B")
        result.color = "#FCD34D"
        page.update()
        return

    n = board.row

    found, count, time = solve(board)
    if not found:
        result.value = "Solusi tidak ditemukan, tidak bisa
        di-export"
        resultContainer.bgcolor = "#7F1D1D"
        resultContainer.border = ft.border.all(1, "#EF4444")
        result.color = "#FCA5A5"
        page.update()
        return
```

```

grid = [[0 for _ in range(n)] for _ in range(n)]
queens = []
for i in range(n):
    for j in range(n):
        if board.board[i][j] != "#":
            grid[i][j] =
colors.get(board.board[i][j].upper(), "#94A3B8")
else:
    grid[i][j] = "#FFD700"
queens.append((i, j))

cell = 80
font = ImageFont.truetype("seguiemj.ttf", cell // 2)
img = Image.new("RGB", (n * cell, n * cell))
draw = ImageDraw.Draw(img)

for i in range(n):
    for j in range(n):
        x1 = j * cell
        y1 = i * cell
        x2 = x1 + cell
        y2 = y1 + cell

        draw.rectangle([x1, y1, x2, y2], fill=grid[i][j],
outline="black")

        if (i, j) in queens:
            draw.text((x1 + cell // 6, y1 + cell // 4),
"👑", font=font)

folder = "result"
if not os.path.exists(folder):
    os.makedirs(folder)

img.save("result/game.png")

result.value = "Berhasil diexport ke game.png"
resultContainer.bgcolor = "#064E3B"
resultContainer.border = ft.border.all(1, "#10B981")

```

```
result.color = "#6EE7B7"
page.update()

# Load file
def loadFile(e: ft.FilePickerResultEvent):
    if e.files:
        filePath = e.files[0].path
        fileName.value = f"File: {e.files[0].name}"

        board, error = readFile(filePath)
        if board:
            lines = []
            for i in range(board.row):
                lines.append("".join(board.board[i]))
            boardInput.value = "\n".join(lines)
            boardDisplay.controls.clear()
            result.value = "File berhasil dimuat"
            resultContainer.bgcolor = "#064E3B"
            resultContainer.border = ft.border.all(1,
                                                "#10B981")
            result.color = "#6EE7B7"
        else:
            boardDisplay.controls.clear()
            result.value = f"Error: {error}"
            resultContainer.bgcolor = "#7F1D1D"
            resultContainer.border = ft.border.all(1,
                                                "#EF4444")
            result.color = "#FCA5A5"
    page.update()

filePicker = ft.FilePicker(on_result=loadFile)
page.overlay.append(filePicker)

def pickFile(e):
    filePicker.pick_files(
        allowed_extensions=["txt"],
        dialog_title="Pilih file konfigurasi"
    )
```

```
# Solve

def solves(e):
    board, error = validateBoard()
    if error:
        boardDisplay.controls.clear()
        result.value = error
        resultContainer.bgcolor = "#78350F"
        resultContainer.border = ft.border.all(1, "#F59E0B")
        result.color = "#FCD34D"
    page.update()
    return

backtracks = checkbox.value # ambil value dari checkbox

n = board.row
boardDisplay.controls.clear()
result.value = "Sedang mencari solusi..."
resultContainer.bgcolor = "#1E3A8A"
result.color = ft.Colors.WHITE
page.update()

# Live update
def updateProgress(tempBoard):
    boardDisplay.controls.clear()
    boardDisplay.controls.append(ft.Text("Mencari
solusi...", size=14, weight=ft.FontWeight.BOLD,
color="#60A5FA"))

    boardDisplay.controls.append(createBoardGrid(tempBoard))
    page.update()

found, caseCount, time = solve(board,
update=updateProgress, interval=1.5,
backtracks=backtracks)

if found:
    boardDisplay.controls.clear()
    boardDisplay.controls.append(
        ft.Text("Solusi Ditemukan!", size=16,
```

```
        weight=ft.FontWeight.BOLD, color="#10B981")
    )
boardDisplay.controls.append(createBoardGrid(board))

result.value = f"Waktu pencarian: {time:.2f} ms\nKasus
ditinjau: {caseCount} kasus"
resultContainer.bgcolor = "#064E3B"
resultContainer.border = ft.border.all(1, "#10B981")
result.color = "#6EE7B7"
else:
    boardDisplay.controls.clear()
    boardDisplay.controls.append(
        ft.Text("Solusi Tidak Ditemukan", size=16,
        weight=ft.FontWeight.BOLD, color="#EF4444")
    )
    result.value = f"Coba dengan konfigurasi board yang
berbeda.\nWaktu pencarian: {time:.2f} ms\nKasus
ditinjau: {caseCount} kasus"
    resultContainer.bgcolor = "#7F1D1D"
    resultContainer.border = ft.border.all(1, "#EF4444")
    result.color = "#FCA5A5"

page.update()

# UI
page.add(
    ft.Container(
        content=ft.Column([
            ft.Text(
                "Queen's Linkedin Game Solver",
                size=28,
                weight=ft.FontWeight.BOLD,
                color=ft.Colors.BLUE_700
            ),
            ft.Divider(height=20, color=ft.Colors.BLUE_200),

            ft.Row([
                ft.Container(
                    content=ft.Column([

```

```
        ft.Text("Input Board:", size=16,
            color="#E2E8F0",
            weight=ft.FontWeight.BOLD),
            boardInput,
            ft.Row([
                ft.ElevatedButton(
                    "Load dari File",
                    icon=ft(Icons.UPLOAD_FILE,
                    on_click=pickFile,
                    style=ft.ButtonStyle(
                        color=ft.Colors.WHITE,
                        bgcolor=ft.Colors.BLUE_600
                    )
                ),
                ft.ElevatedButton(
                    "Solve",
                    icon=ft(Icons.PLAY_ARROW,
                    on_click=solves,
                    style=ft.ButtonStyle(
                        color=ft.Colors.WHITE,
                        bgcolor=ft.Colors.GREEN_600
                    )
                ),
            ],
            spacing=10),
            ft.Row([
                ft.ElevatedButton(
                    "Export TXT",
                    icon=ft(Icons.SHARE,
                    on_click=exportTxt,
                    style=ft.ButtonStyle(
                        color=ft.Colors.WHITE,
                        bgcolor=ft.Colors.PURPLE_600
                    )
                ),
                ft.ElevatedButton(
                    "Export Image",
                    icon=ft(Icons.IMAGE,
```

```
        on_click=exportImage,
        style=ft.ButtonStyle(
            color=ft.Colors.WHITE,
            bgcolor=ft.Colors.PINK_600
        )
    ),
    ],
    spacing=10),
checkbox,
fileName,
],
spacing=15),
padding=20,
border_radius=10,
bgcolor="#1E293B"
),
ft.Container(
content=ft.Column([
ft.Text("Hasil:", size=16,
color=ft.Colors.WHITE54,
weight=ft.FontWeight.BOLD),
resultContainer,
],
spacing=10),
padding=20,
)
],
spacing=20, scroll=ft.ScrollMode.AUTO),
],
spacing=10,
horizontal_alignment=ft.CrossAxisAlignment.CENTER),
bgcolor=ft.Colors.BLACK38,
border_radius=15,
padding=20,
)
)
```

maincli.py

```
from core.board import Board
from core.solver import solve
from core.helper import readFile, showResult
import os
```

```
file = input("Silahkan masukkan nama file (tanpa .txt) : ")
path = os.path.join("test", file)
board, error = readFile(f"{path}.txt")

if board is None:
    print(f"Error: {error}")
    exit(1)

ans = input("Apakah ingin mengoptimalkan solusi (backtrack) Y/N: ")
if ans.upper() == "Y":
    found, caseCount, time = solve(board, None, None,
                                    backtracks=True)
else:
    found, caseCount, time = solve(board)

if found:
    showResult(board)
    print()
    print(f"Waktu pencarian: {time:.2f} ms")
    print(f"Banyak kasus yang ditinjau: {caseCount} kasus")
else:
    print("Solusi tidak ditemukan")
```

maingui.py

```
from ui.gui import main
import flet as ft

ft.app(target=main)
```

BAB 3

UJI PROGRAM

1. Kasus Uji 1 - *valid*

Input :

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

Output :

CLI

```
Silahkan masukkan nama file (tanpa .txt) : test1  
Apakah ingin mengoptimalkan solusi (backtrack) Y/N: Y  
AAABBCC#D  
ABBB#CECD  
ABBBDC#CD  
A#ABDCCCD  
BBBBD#DDD  
FGG#DDHDD  
#GIGDDHDD  
FG#GDDHDD  
FGGGDDHH#
```

```
Waktu pencarian: 8.67 ms  
Banyak kasus yang ditinjau: 7659 kasus
```

***Menggunakan backtrack untuk mempercepat*

GUI

Hasil:

Solusi Ditemukan!

Waktu pencarian: 6462.61 ms
Kasus ditinjau: 7659 kasus

Input Board:

```
Input Board:
AAABBCCCD
ABBBBCECD
ABBBDCECD
AAABDCCCD
BBBBDDDDD
FGGGDDHDD
FGIGDDHDD
FGGGDDHHH
```

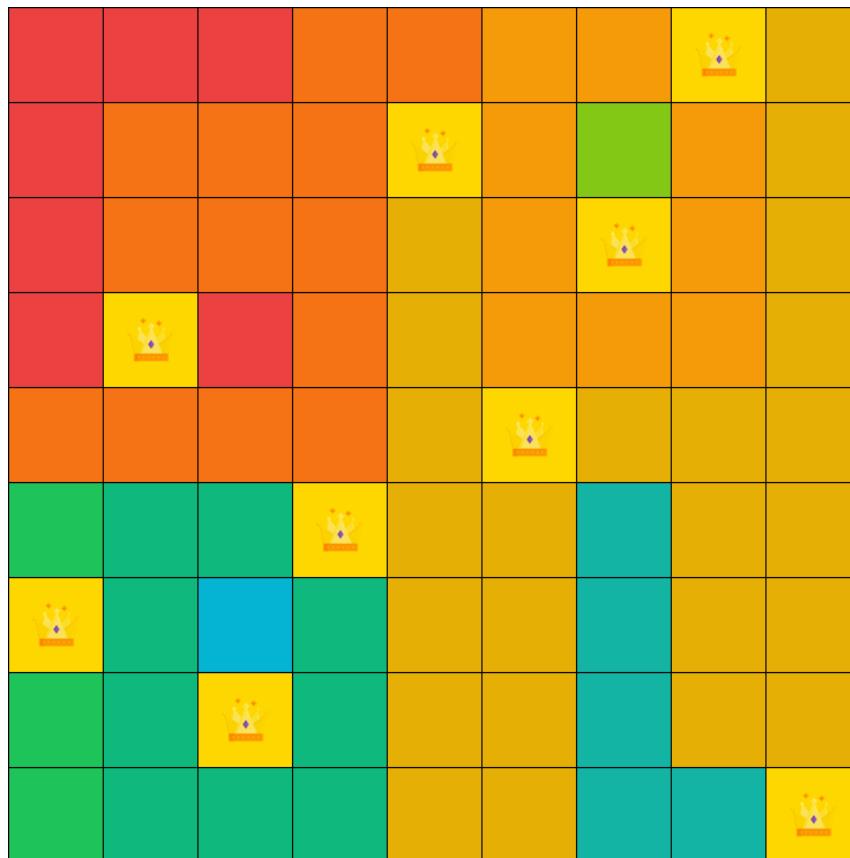
Load dari File Solve

Export TXT Export Image

Enable Backtracking (Faster & Recommended)

File: test1.txt

Export Image



2. Kasus Uji 2 - *valid*

Input :

ABBBB
CDCBC
CDCCC
CCCEE
CCCCC

Output :

CLI

```
Silahkan masukkan nama file (tanpa .txt) : test2
Apakah ingin mengoptimalkan solusi (backtrack) Y/N: N
#BBBB
CDC#C
C#CCC
CCCE#
CC#CC
```

```
Waktu pencarian: 1.64 ms
Banyak kasus yang ditinjau: 423 kasus
```

GUI

Hasil:

Solusi Ditemukan!

Waktu pencarian: 987.06 ms
Kasus ditinjau: 423 kasus

Input Board:

Input Board

```
A B B B B  
C D C B C  
C D C C C  
C C E E E  
C C C C C
```

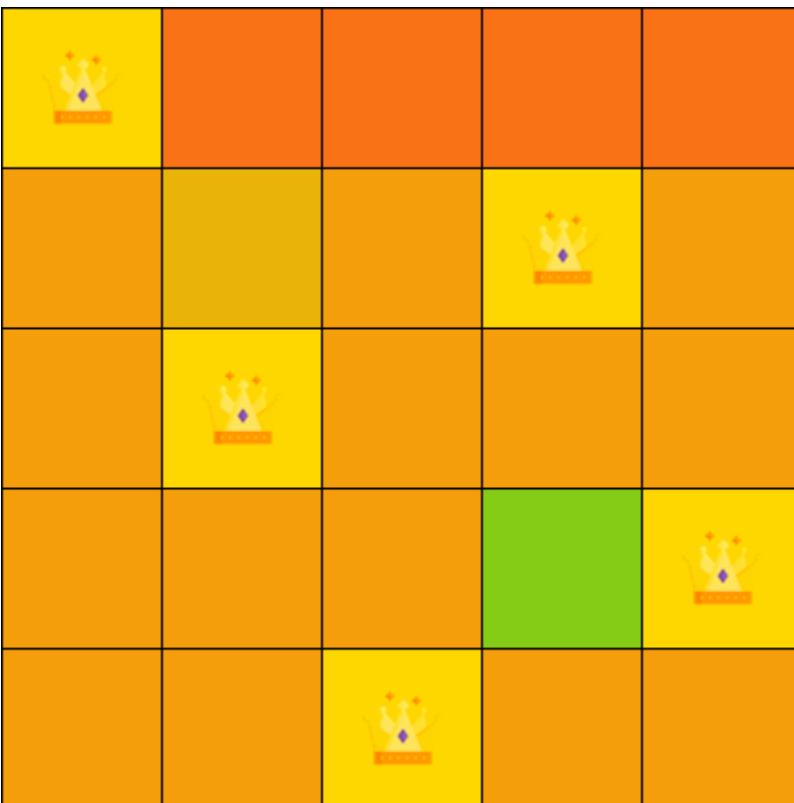
[Load dari File](#) [Solve](#)

[Export TXT](#) [Export Image](#)

Enable Backtracking (Faster & Recommended)

File: test2.txt

Export Image



3. Kasus Uji 3 - *invalid (daerah terlalu banyak)*

Input :

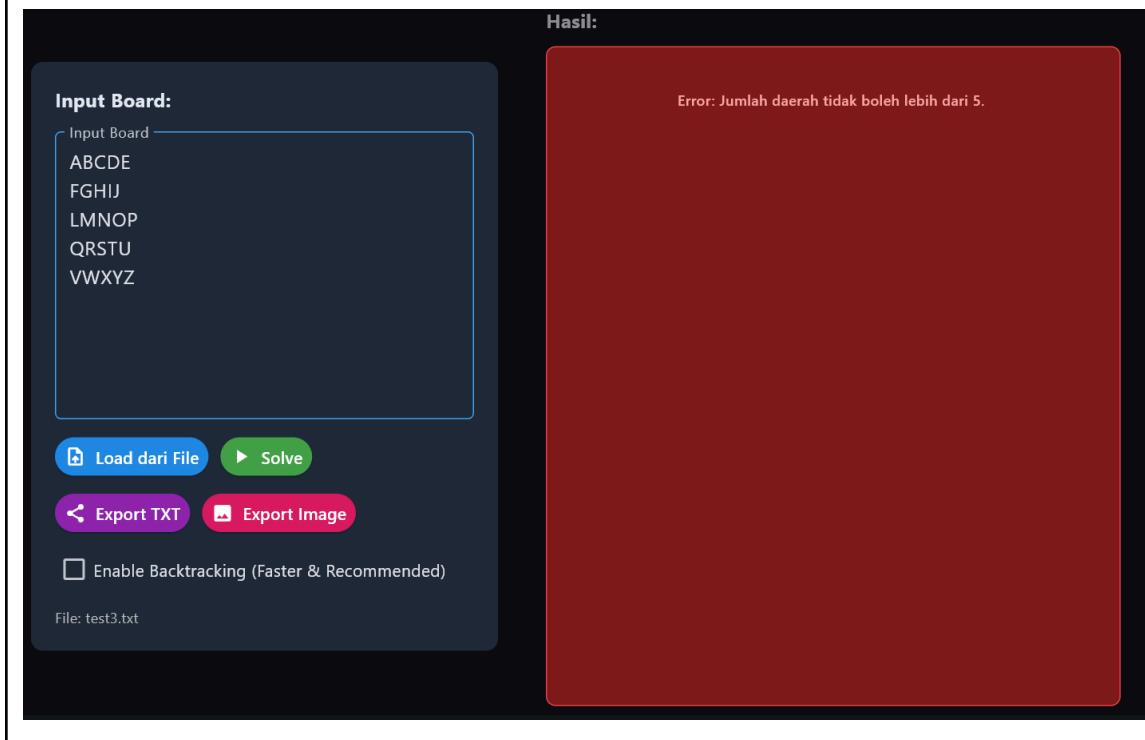
ABCDE
FGHIJ
LMNOP
QRSTU
VWXYZ

Output :

CLI

Silahkan masukkan nama file (tanpa .txt) : test3
Error: Jumlah daerah tidak boleh lebih dari 5.

GUI



4. Kasus Uji 4 - *invalid (board tidak lengkap)*

Input :

AAAD

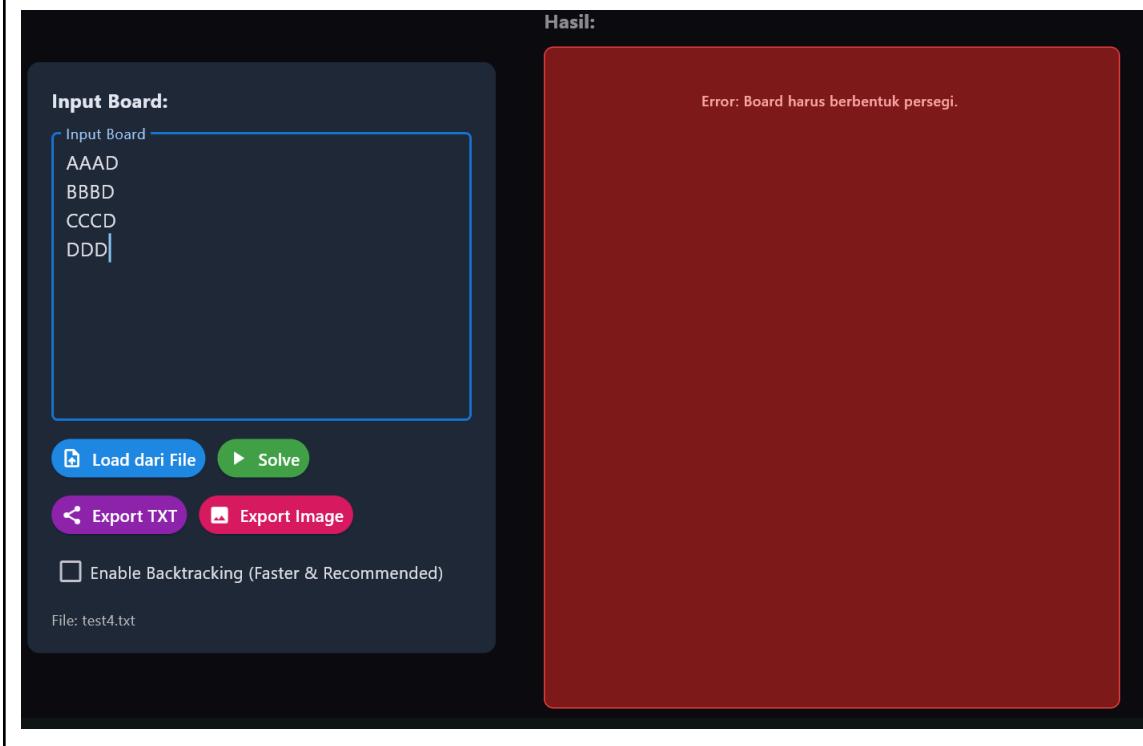
BBBD
CCCD
DDD

Output :

CLI

Silahkan masukkan nama file (tanpa .txt) : test4
Error: Board harus berbentuk persegi.

GUI



5. Kasus Uji 5 - *invalid (bukan alphabet)*

Input :

AAA

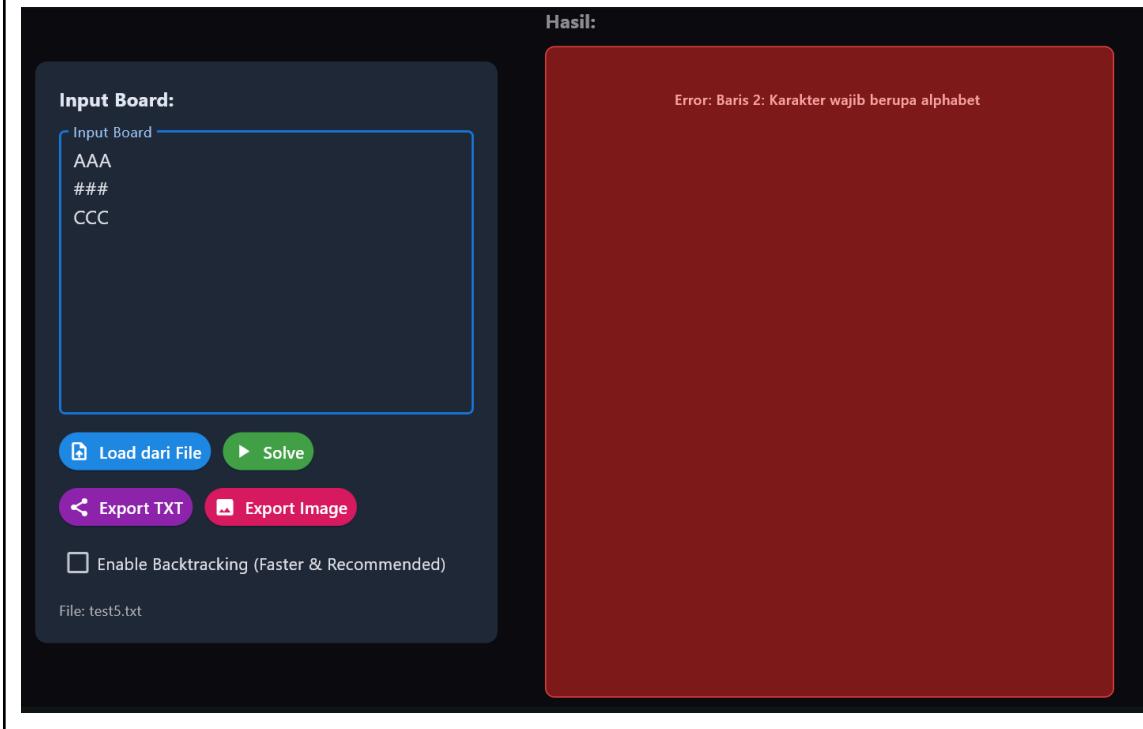
CCC

Output :

CLI

Silahkan masukkan nama file (tanpa .txt) : test5
Error: Baris 2: Karakter wajib berupa alphabet

GUI



6. Kasus Uji 6 - *valid*

Input :

A B C D E F G H I J K L M
B C D E F G H I J K L M A
C D E F G H I J K L M A B
D E F G H I J K L M A B C
E F G H I J K L M A B C D
F G H I J K L M A B C D E
G H I J K L M A B C D E F
H I J K L M A B C D E F G
I J K L M A B C D E F G H
J K L M A B C D E F G H I
K L M A B C D E F G H I J
L M A B C D E F G H I J K
M A B C D E F G H I J K L

Output :

CLI

```
Silahkan masukkan nama file (tanpa .txt) : test6
Apakah ingin mengoptimalkan solusi (backtrack) Y/N: Y
#BCDEFHIJKLM
BC#EFGHIJKLMNOP
CDEF#HIJKLMNOPAB
D#FGHIJKLMNOPABC
EFGHI#KLMABCD
FGHIJKLMNOP#ABCDE
GHIJKLMNOPAB#DEF
HIJKLMNOP#LMABCDE
IJ#LMABCDE
JKLMABCDEF#I
JKLMABCDE#GHIJ
LMABCDE#FGH#JK
MABCDEFHIJK#
```

Waktu pencarian: 2.73 ms

Banyak kasus yang ditinjau: 1651 kasus

GUI

Input Board:

```
Input Board
ABCDEFGHIJKLM
BCDEFGHIJKLMNOPA
CDEFGHIJKLMNOPB
DEFGHIJKLMNOPC
EFGHIJKLMNOPD
FGHIJKLMNOPE
GHIJKLMNOPF
HIJKLMNOPG
IJKLMNOPH
JKLMNOPGHI
JKLMNOPFGHIJ
JKLMNOPFGHIJK
MJKLMNOPFGHIKL
```

Enable Backtracking (Faster & Recommended)

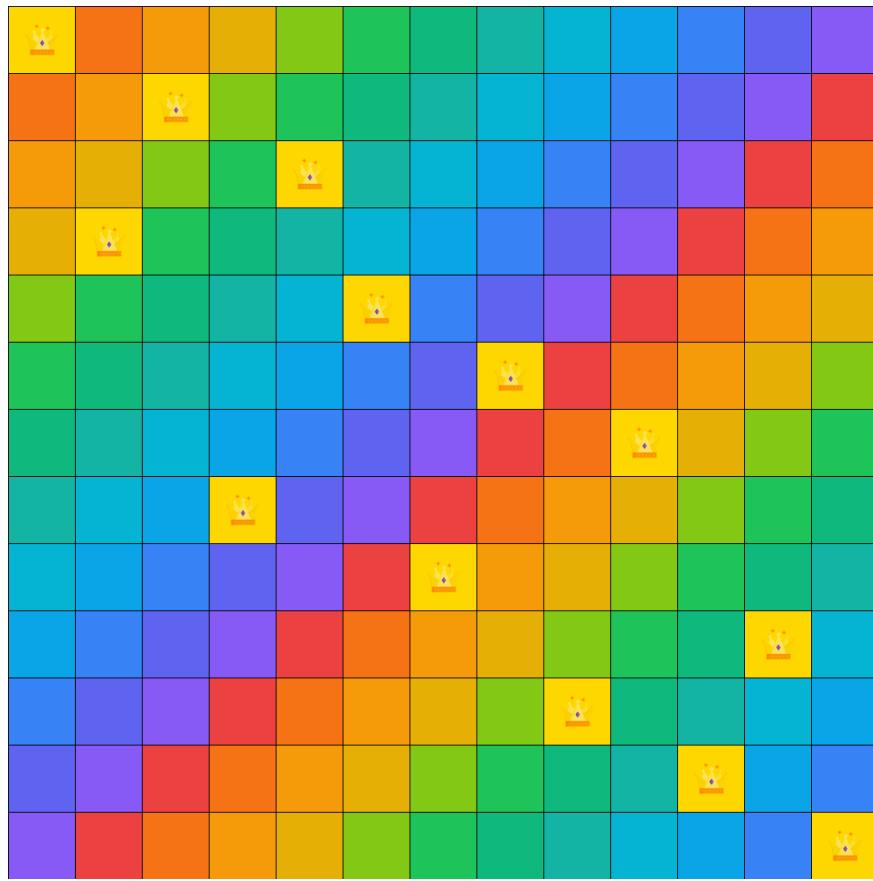
File: test6.txt

Hasil:

Solusi Ditemukan!

Waktu pencarian: 2010.30 ms
Kasus ditinjau: 1651 kasus

Export Image



BAB IV

LAMPIRAN

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	✓	

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



Valentino Daniel Kusumo