LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

PENYELESAIAN PERMAINAN QUEENS LINKEDIN

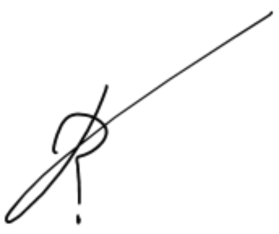
SEMESTER II TAHUN 2025/2026

Oleh:

Ramadhian Nabil Firdaus Gumay

13524126

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



Ramadhian Nabil Firdaus Gumay

1. ALGORITMA

1.1. Program merepresentasikan papan permainan sebagai array satu dimensi, di mana array mewakili baris dan nilai di dalamnya mewakili posisi kolom. Algoritma bekerja dengan menyusun semua kemungkinan permutasi angka dari 0 hingga N-1 untuk menempatkan Ratu di setiap baris.

1.2. Setiap kali satu susunan permutasi lengkap terbentuk, program menjalankan fungsi validasi untuk memeriksa dua aturan khusus. Pertama, program akan memeriksa kesamaan warna, yaitu memastikan tidak ada dua Ratu yang menempati kotak dengan kode warna yang sama. Kedua, program memeriksa diagonal, yaitu dengan menghitung selisih koordinat untuk memastikan tidak ada Ratu yang bersentuhan secara diagonal dalam jarak baris dan kolom bernilai satu dengan Ratu lainnya.

1.3. Pencarian dilakukan menggunakan fungsi rekursif yang menukar posisi elemen dalam array untuk mendapatkan kombinasi baru. Jika ditemukan susunan lolos dari validasi warna dan validasi diagonal, program akan menetapkannya sebagai solusi, menghentikan pencarian, dan menampilkan hasil akhirnya.

2. SOURCE PROGRAM

```
import time
import os
import sys

board =[]
q_position = []
count = 0

def baca_file(nama_input_file):
  board=[]
  try:
    with open(nama_input_file, 'r') as f:
      for line in f:
        row = list(line.strip().upper())
        if row:
          board.append(row)
    if not board:
      print(f"File '{nama_input_file}' kosong.")
      return None

    N = len(board)
    for r in range(N):
      if len(board[r]) != N:
        print(f"File '{nama_input_file}' tidak valid. Baris dan
kolom harus memiliki panjang yang sama.")
```

```
        return None

    return board

  except FileNotFoundError:
    print(f"File '{nama_input_file}' tidak ditemukan.")
    return None

def live_update(current_perm, N):
  os.system('cls' if os.name == 'nt' else 'clear')

  output_str = ""
  for r in range(N):
    line= ""
    for c in range(N):
      if current_perm[r] == c:
        line += "#"
      else:
        line += board[r][c] + ""
    output_str += line + "\n"
  print(output_str)
  time.sleep(0.1)

def validate(perm, N):
  used_colors = set()

  for r in range(N):
    c = perm[r]
    warna = board[r][c]
    if warna in used_colors:
      return False
    used_colors.add(warna)

  for r1 in range(N):
    for r2 in range(r1 + 1, N):
      c1 = perm[r1]
      c2 = perm[r2]
      if abs(r1 - r2) == abs(c1 - c2):
        if abs(r1 - r2) == 1:
          return False
  return True

def permute(arr, start_idx, end_idx):
  global count, q_position

  if start_idx == end_idx:
    count += 1
    current_comb = list(arr)

    q_position = current_comb

    if count % 100 == 0:
      live_update(current_comb, len(board))

    if validate(current_comb, len(board)):
      live_update(current_comb, len(board))
      return True
    return False
```

```
  for i in range (start_idx, end_idx + 1):
    arr[start_idx], arr[i] = arr[i], arr[start_idx]
    if permute(arr, start_idx + 1, end_idx):
      return True
    arr[start_idx], arr[i] = arr[i], arr[start_idx]
  return False

def save_file(nama_output_file, durasi, status, N):
  try:
    with open(nama_output_file, 'w') as f:
      if status:
        f.write("Solusi ditemukan:\n")
      else:
        f.write("Solusi tidak ditemukan:\n")
      for r in range(N):
        line = ""
        for c in range(N):
          if q_position[r] == c:
            line += "#"
          else:
            line += board[r][c] + ""
        f.write(line + "\n")
      f.write(f"\nWaktu pencarian: {durasi:.4f} ms\n")
      f.write(f"Banyak kasus yang ditinjau: {count} kasus\n")
    print(f"Solusi berhasil disimpan ke '{nama_output_file}'.")
  except Exception as e:
    print(f"Gagal menyimpan solusi ke '{nama_output_file}'.")

input_file = input("Masukkan nama file: ")
script_dir = os.path.dirname(os.path.abspath(__file__))
project_dir = os.path.dirname(script_dir)
test_folder = os.path.join(project_dir, 'test')
full_path = os.path.join(test_folder, input_file)
board_data = baca_file(full_path)

if board_data:
  board = board_data
  N = len(board)
  initial_arr = list(range(N))
  start_time = time.perf_counter()
  ketemu = permute(initial_arr, 0, N - 1)
  end_time = time.perf_counter()
  durasi = (end_time - start_time) * 1000

  os.system('cls' if os.name == 'nt' else 'clear')
  if ketemu:
    print("Solusi ditemukan:")
  else:
    print("Solusi tidak ditemukan:")
  for r in range(N):
    line = ""
    for c in range(N):
      if q_position[r] == c:
        line += "#"
      else:
        line += board[r][c] + ""
    print(line)
```

```
  print(f"\nWaktu pencarian: {durasi:.4f} ms")
  print(f"Banyak kasus yang ditinjau: {count} kasus")

  simpan = input("Apakah Anda ingin menyimpan solusi? (y/n):
").lower()
  if simpan == 'y':
    output_file = input("Masukkan nama file output: ")
    script_dir = os.path.dirname(os.path.abspath(__file__))
    project_dir = os.path.dirname(script_dir)
    test_folder = os.path.join(project_dir, 'test')
    if not os.path.exists(test_folder):
      os.makedirs(test_folder)
    full_path = os.path.join(test_folder, output_file)
    save_file(full_path, durasi, ketemu, N)
  else:
    print("Solusi tidak disimpan.")
```

3. PENGUJIAN

3.1. Uji 1

3.1.1. Data Uji 1

```
1  AAABBBCC
2  AAABBBCC
3  DDDEEEFF
4  DDDEEEFF
5  GGGHHHII
6  GGGHHHII
7  JJJKKKLL
8  JJJKKKLL
```

3.1.2. Solusi Uji 1

```
1  Solusi ditemukan:
2  #AABBBCC
3  AAA#BBCC
4  D#DEEEFF
5  DDDE#EFF
6  GGGHHH#I
7  GG#HHHII
8  JJJKK#LL
9  JJJKKKL#
10
11 Waktu pencarian: 1853.9171 ms
12 Banyak kasus yang ditinjau: 1599 kasus
13
```

3.2. Uji 2

3.2.1. Data Uji 2

```
AABBB
BAABB
BBBAA
BBBBA
ABBBB
```

3.2.2. Solusi Uji 2

```
Solusi tidak ditemukan:
AABB#
#AABB
B#BAA
BB#BA
ABB#B

Waktu pencarian: 128.7542 ms
Banyak kasus yang ditinjau: 120 kasus

```

3.3. Uji 3

3.3.1. Data Uji 3

```
AAAAAB
BBBBBC
CCCCCD
DDDDDE
EEEEEA
BBBBBC
```

3.3.2. Solusi Uji 3

```
Solusi tidak ditemukan:
AAAAA#
#BBBBC
C#CCCD
DD#DDE
EEE#EA
BBBB#C

Waktu pencarian: 850.3109 ms
Banyak kasus yang ditinjau: 720 kasus

```

3.4. Uji 4

3.4.1. Data Uji 4

```
ABC
DEF
GHI
```

3.4.2. Solusi Uji 4

```
Solusi tidak ditemukan:
AB#
#EF
G#I

Waktu pencarian: 0.0434 ms
Banyak kasus yang ditinjau: 6 kasus

```

3.5. Uji 5

3.5.1. Data Uji 5

```
AAAAABBB
CCCCCDDB
CEEEEEDB
CFFFFFDB
CGGGGGDB
CHHHHHDB
CHHHHHDB
CHHHHHDB
```

3.5.2. Solusi Uji 5

```
Solusi ditemukan:
#AAAABBB
CC#CCDDB
CEEE#EDB
C#FFFFDB
CGGGG#DB
CHHHHHD#
CHH#HHDB
CHHHHH#B

Waktu pencarian: 1177.5690 ms
Banyak kasus yang ditinjau: 996 kasus

```

4. LAMPIRAN

4.1. Pranala Repository

https://github.com/Rmadhian/Tucil1_13524126.git

4.2. Tabel Checklist

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		√