TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Penyelesaian Permainan Queens Linkedin dengan Algoritma Brute Force

Disusun oleh :

Kholida Rezki Khoiriah (13222071)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

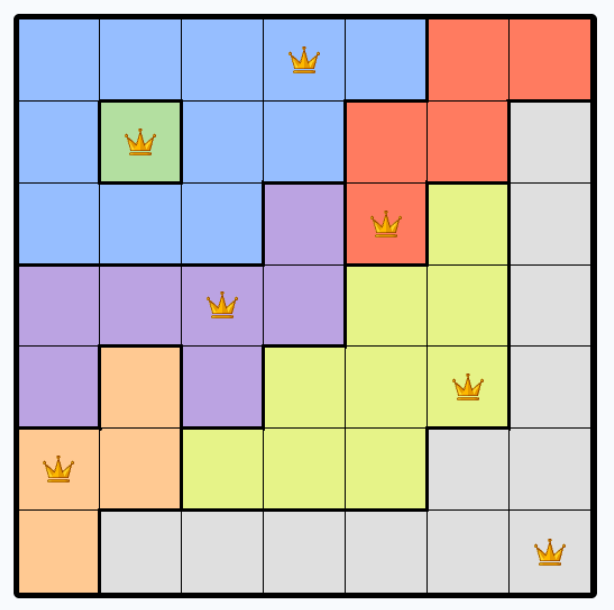
INSTITUT TEKNOLOGI BANDUNG

2023

Daftar Isi

BAB I
Algoritma Brute Force dalam Queens Linkedin

Queens adalah permainan logika yang tersedia pada situs LinkedIn. Permainan ini bertujuan menempatkan queen pada sebuah papan persegi berwarna sehingga hanya terdapat satu queen pada setiap baris, kolom, dan warna. Setiap queen tidak boleh bersebelahan secara horizontal vertikal, maupun diagonal. Program yang dibuat harus bisa menemukan satu solusi penempatan queen pada papan warna yang diberikan, atau menampilkan bahwa tidak ada solusi yang valid. Pencarian solusi dilakukan dengan memanfaatkan algoritma brute force. Algoritma ini memerika satu per satu dari seluruh kemungkinan yang ada untuk mencari solusi yang optimal.



Langkah-langkahnya sebagai berikut:

1. Input diterima dalam format .txt. Input dianggap selalu valid, yaitu memenuhi ketentuan berikut.
   - Ukuran input selalu n*n dengan n tidak lebih besar dari 26
   - Input selalu berupa karakter huruf kapital A-Z
   - Setiap karakter yang sama selalu bertetangga dan tidak ada yang terpisah
   - Variasi karakter sama dengan n
2. Input diubah menjadi matriks ukuran n*n, lalu dibuat matriks bayangan berukuran sama untuk melihat posisi queens
3. Masukkan kemungkinan pertama, yaitu queens terletak pada kolom berbeda untuk baris yang sama
4. Dilakukan pengecekan pelanggaran atau *violation*.
5. Jika terdapat violation, lakukan increment pada kemungkinan sebelumnya dengan memindahkan posisi queen ke sel selanjutnya.

6. Lakukan backtracking jika semua sel sudah dicek namun jumlah queen yang valid belum mencapai n
7. Menampilkan keluaran berupa posisi queen secara live update pada GUI serta posisi akhir pada GUI dan terminal
8. Algoritma berakhir saat sebuah solusi ditemukan (tidak ada pelanggaran dan semua queen sudah ditaruh pada board)
9. Menanmpilkan banyak kasus yang diuji dan waktu yang digunakan saat menjalankan algoritma brute force

BAB II
Source Program

Program ini ditulis dengan menggunakan bahasa pemrograman Python
Daftar Module/Library :

- Tkinter
- Numpy
- Time

Source code :

```
import numpy as np
import tkinter as tk
import time

def isSafe(queen_matrix, color_matrix, row, column):
    n = len(color_matrix)
    list_color, count_color = cek_warna(queen_matrix, color_matrix, n)

    # cek warna
    warna = color_matrix[row][column]
    if warna in list_color:
        indeks = list_color.index(warna)
        if count_color[indeks] == 1:
            return 1

    # cek apakah pada baris tersebut ada queen
    for i in range(n):
        if queen_matrix[row][i] == 1:
            return 0

    # cek apakah pada kolom tersebut ada queen
    for j in range(n):
        if queen_matrix[j][column] == 1:
            return 0

    # cek diagonal
        # kiri atas
        if row > 0 and column > 0:
            if queen_matrix[row-1][column-1] == 1:
                return 0
        # kanan atas
        if row > 0 and column < n-2:
            if queen_matrix[row-1][column+1] == 1:
                return 0
```

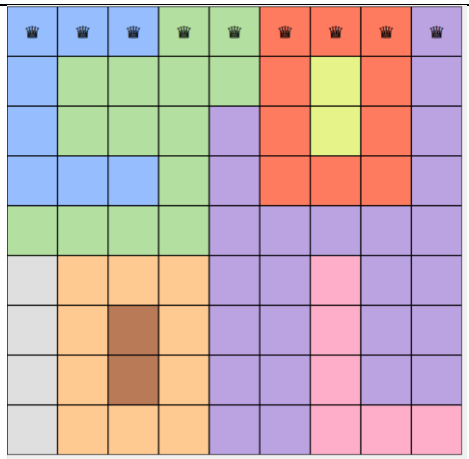
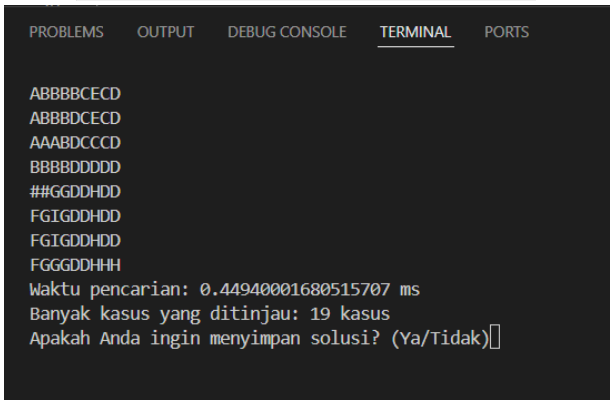
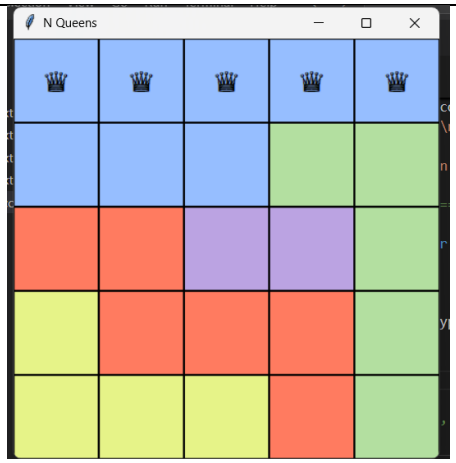
```
        # kiri bawah
        if row < n-2 and column > 0:
            if queen_matrix[row+1][column-1] == 1:
                return 0
        # kanan bawah
        if row < n-2 and column < n-2:
            if queen_matrix[row+1][column+1] == 1:
                return 0

    # lolos semua pengecekan
    return 1

def cek_warna(queen_matrix, color_matrix, n):
    list_color = []
    count_color = []

    for i in range(n):
        for j in range(n):
            if queen_matrix[i][j] == 1:
                warna = color_matrix[i][j]
                if warna not in list_color:
                    list_color.append(warna)
                    count_color.append(1)
                else:
                    indeks = list_color.index(warna)
                    count_color[indeks] += 1

    return list_color, count_color

def simpan_txt():
    with open("solusi.txt", "w") as f:
        for i in range(n):
            row_output = ""
            for j in range(n):
                if queen_matrix[i][j] == 1:
                    row_output += "#"
                else:
                    row_output += color_matrix[i][j]
            f.write(row_output + "\n")

    print("Solusi berhasil disimpan sebagai solusi.txt")


with open("input.txt", "r") as f:
    lines = [list(line.strip()) for line in f]
```

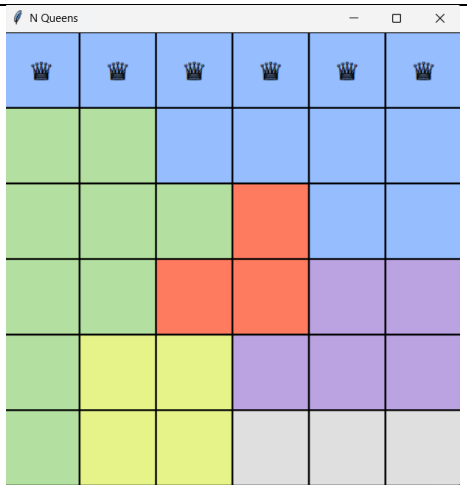
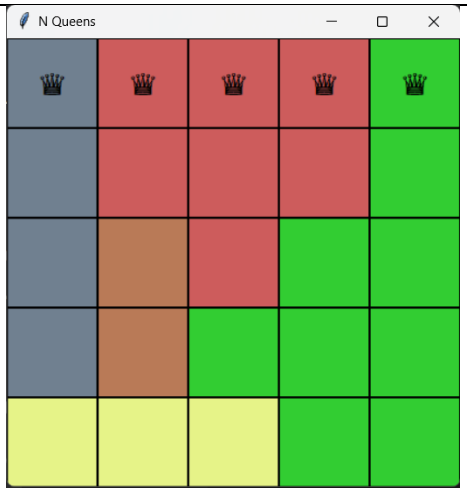
```
with open("input.txt", "r") as f:
    lines = [list(line.strip()) for line in f]

color_matrix = np.array(lines)
n = len(color_matrix)
queen_matrix = np.zeros((n, n), dtype=int)
row = 0
column = 0
queen_true = 0

# Initial case, pada baris pertama, ada semua queen
for i in range(n):
    queen_matrix[0][i] = 1

# GUI
# mapping warna
color_map = {
    "A": "#96BEFF",  # biru muda
    "B": "#B3DFA0",  # hijau muda
    "C": "#FF7B60",  # coral
    "D": "#BBA3E2",  # ungu muda
    "E": "#E6F388",  # lime
    "F": "#DFDFDF",  # abu terang
    "G": "#FFC992",  # peach
    "H": "#FFAEC9",  # pink
    "I": "#B97A57",  # coklat

    "J": "#00BFFF",  # deep sky blue
    "K": "#32CD32",  # lime green
    "L": "#FF1493",  # deep pink
    "M": "#FFD700",  # gold
    "N": "#8A2BE2",  # blue violet
    "O": "#FF4500",  # orange red
    "P": "#20B2AA",  # light sea green
    "Q": "#DC143C",  # crimson
    "R": "#708090",  # slate gray

    "S": "#00CED1",  # dark turquoise
    "T": "#ADFF2F",  # green yellow
    "U": "#FF69B4",  # hot pink
    "V": "#1E90FF",  # dodger blue
    "W": "#FF8C00",  # dark orange
    "X": "#9932CC",  # dark orchid
    "Y": "#3CB371",  # medium sea green
```

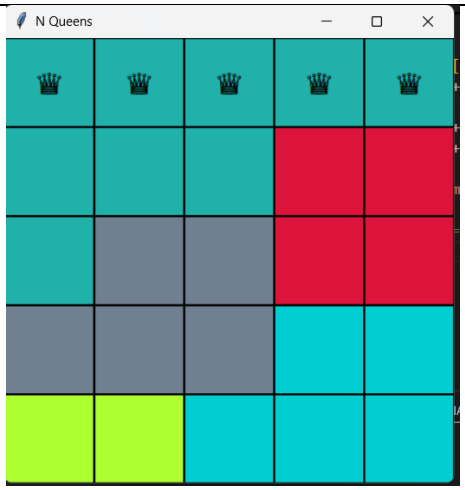
```
    "Z": "#CD5C5C"   # indian red
}

# buat window
root = tk.Tk()
root.title("N Queens")

# buat grid
for i in range(n):
    for j in range(n):
        letter = color_matrix[i][j]
        color = color_map.get(letter, "white")

        if queen_matrix[i][j] == 1:
            text_symbol = "♕"
        else:
            text_symbol = ""

        cell = tk.Label(
            root,
            text=text_symbol,
            fg="black",
            bg=color,
            width=5,
            height=2,
            font=("Segoe UI Symbol", 20),
            highlightthickness=1,
            highlightbackground="black"
        )
        cell.grid(row=i, column=j)
root.mainloop()

# Variabel yang dibutuhkan
waktu_pencarian = 0
banyak_kasus = 0

# Fungsi Utama
start = time.perf_counter()

# Cek
# cek violation
while queen_true < n:
    if not isSafe(queen_matrix, color_matrix, row, column):
        banyak_kasus += 1
        queen_matrix[row][column] = 0
```

```
        if row <= n-2:
            row += 1
        else:
            row = 0
            if column <= n-2:
                column += 1
            else:
                break
        queen_matrix[row][column] = 1
    else:
        queen_true += 1
        banyak_kasus += 1
        if row <= n-2:
            row += 1
        else:
            row = 0
            if column <= n-2:
                column += 1
            else:
                break

end = time.perf_counter()

waktu_pencarian = (end - start) * 1000  # jadi milisecond

# Output
for i in range(n):
    row_output = ""
    for j in range(n):
        if queen_matrix[i][j] == 0:
            row_output += color_matrix[i][j]
        else:
            row_output += "#"
    print(row_output)

print(f"Waktu pencarian: {waktu_pencarian} ms")
print(f"Banyak kasus yang ditinjau: {banyak_kasus} kasus")
simpan = input("Apakah Anda ingin menyimpan solusi? (Ya/Tidak)")

if simpan.lower() == "ya":
    simpan_txt()
```

BAB III: Pengujian dan Hasil Pengujian

<table>
<tr><td colspan="2">Input File</td></tr>
<tr><td>Input</td><td>Output</td></tr>
<tr><td>input.txt
AAABBCCCD
ABBBBCECD
ABBBDCECD
AAABDCCCD
BBBBDDDDD
FGGGDDHDD
FGIGDDHDD
FGIGDDHDD
FGGGDDHHH</td><td> 
  </td></tr>
<tr><td>input2.txt
AAAAA
AAABB
CCDDB
ECCCB
EEECB</td><td>  </td></tr>
</table>

	AAA## AAABB #C#DB ECCCB #EECB Waktu pencarian: 0.16140000661835074 ms Banyak kasus yang ditinjau: 15 kasus Apakah Anda ingin menyimpan solusi? (Ya/Tidak)
input3.txt AAAAAA BBAAAA BBBCAA BBCCDD BEEDDD BEEFFF	 AA#### #BAAAA BBBCAA BBCCDD B#EDDD BEEFFF Waktu pencarian: 0.14739998732693493 ms Banyak kasus yang ditinjau: 11 kasus Apakah Anda ingin menyimpan solusi? (Ya/Tidak)
input4.txt RZZZK RZZZK RIZKK RIKKK EEEKK	

	#Z### RZZZK R#ZKK RIKKK EEEKK Waktu pencarian: 0.21940001170150936 ms Banyak kasus yang ditinjau: 8 kasus Apakah Anda ingin menyimpan solusi? (Ya/Tidak)
input5.txt PPPPP PPPQQ PRRQQ RRRSS TTSSS	 PPPP# PPP#Q PRRQQ #RRSS T##SS Waktu pencarian: 0.18390000332146883 ms Banyak kasus yang ditinjau: 18 kasus Apakah Anda ingin menyimpan solusi? (Ya/Tidak)

Lampiran

1. *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 bentukfile gambar		✓

2. Pranala *repository*
   https://github.com/rezkikholida/Tucil1_13222071

3. Pernyataan tidak melakukan kecurangan yang di tandatangani

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

Kholida Rezki Khoiriah