Laporan Tugas Kecil 1

IF2211 Strategi Algoritma

Queens

Oleh:
Jonathan Alveraldo Bangun
13524120

1. Pendahuluan

1.1 Queens (LinkedIn)

Queens adalah sebuah gim teka-teki logika populer yang tersedia di platform LinkedIn. Permainan ini menantang pemain untuk mengatur posisi Ratu (Queen) pada sebuah papan persegi yang terbagi ke dalam berbagai wilayah berwarna.

Berdasarkan aturan resminya, solusi yang benar harus memenuhi kriteria berikut:

- Satu per Baris: Setiap baris horizontal hanya boleh memiliki tepat satu Ratu.
- Satu per Kolom: Setiap kolom vertikal hanya boleh memiliki tepat satu Ratu.
- Satu per Daerah Warna: Setiap kelompok sel dengan warna yang sama hanya boleh berisi satu Ratu.
- Aturan Tetangga (Langkah Raja): Ratu tidak boleh ditempatkan di petak yang bersentuhan satu sama lain, termasuk secara diagonal (atas, bawah, samping, maupun sudut).

1.2 Algoritma Brute Force

Brute Force adalah pendekatan yang paling mendasar untuk menyelesaikan sebuah masalah. Istilah ini secara harfiah berarti "kekuatan mentah". Sesuai namanya, algoritma ini tidak menggunakan strategi yang rumit atau logika yang cerdas untuk mempersempit pilihan. Sebaliknya, ia mengandalkan kekuatan pemrosesan komputer untuk mencoba setiap kemungkinan solusi yang ada satu per satu sampai solusi yang benar ditemukan.

Bayangkan kalau kita lupa kombinasi angka pada sebuah gembok koper yang terdiri dari 3 digit (000-999).

- Cara Cerdas: Mencoba mengingat tanggal lahir atau angka penting.
- Cara Brute Force: Mencoba mulai dari 000, 001, 002, 003, dan seterusnya sampai gembok terbuka. Anda pasti akan menemukan kodenya, tapi mungkin butuh waktu lama.

2. Algoritma Exhaustive Search

Def cari_exhausted

```
procedure cari_exhaustive(baris: integer, solusiSkrg: array of integer)
Algoritma
    if baris = N then
        totalIterasi <- totalIterasi + 1

        { Visualisasi setiap interval tertentu }
        if totalIterasi mod 2000 = 0 then
            Visualisasikan(solusiSkrg)

        { Validasi seluruh aturan sekaligus }
        if CekAman(solusiSkrg) then
            return True { Solusi ditemukan }
        else
            return False

    for col from 0 to N-1 do
        solusiSkrg[baris] <- col
        if CariExhaustive(baris + 1, solusiSkrg) then
            return True

    return False
```

Fungsi menerima parameter baris yang menentukan baris mana yang sedang diproses.
Untuk baris tersebut, program mencoba menempatkan Queen di setiap kolom dari 0 hingga (N-1).
Program terus memanggil dirinya sendiri secara rekursif (baris + 1) hingga mencapai dasar atau *base case*.
Ketika baris == self.n, sebuah konfigurasi papan yang lengkap telah terbentuk. Pada titik inilah program menambah hitungan total_iterasi dan memanggil fungsi cek_aman untuk memeriksa validitas konfigurasi tersebut secara menyeluruh.
Live Update: Untuk memenuhi spesifikasi tugas, setiap kelipatan 2000 iterasi, program mengirimkan data ke callback_visual agar proses pencarian dapat dipantau di layar secara *real-time*.

Def cek_aman

```
function cek_aman(solusi: array of integer) -> boolean

    kolomTerpakai <- array boolean [0..N-1] diisi False
    warnaTerpakai <- set kosong

    for r from 0 to N-1 do
        c <- solusi[r]

        { Aturan 1: Cek Kolom }
        if kolomTerpakai[c] then return False
        kolomTerpakai[c] <- True

        { Aturan 2: Cek Warna }
        warna <- papan[r][c]
        if warna dalam warnaTerpakai then return False
        Tambahkan warna ke warnaTerpakai

        { Aturan 3: Cek Tetangga (Diagonal & Bersebelahan) }
        for rLama from 0 to r-1 do
            cLama <- solusi[rLama]
            if abs(rLama - r) <= 1 AND abs(cLama - c) <= 1 then
                return False

    return True
```

Aturan 1: Menggunakan array boolean kolom_terpakai untuk mendeteksi jika ada dua Queen atau lebih yang berada di kolom yang sama.

Aturan 2: Mengakses self.papan[r][c] untuk mengambil identitas warna setiap kotak dan memastikan setiap warna hanya dihuni oleh tepat satu Queen menggunakan struktur data set.

Aturan 3: Melakukan iterasi mundur untuk membandingkan posisi Queen saat ini dengan Queen di baris-baris sebelumnya. Jika selisih posisi baris dan kolom antara dua Queen bernilai lebih dari 1, maka mereka dianggap bersentuhan (diagonal maupun tegak lurus), yang merupakan pelanggaran aturan.

3. Source Program

Bahasa pemrograman: python

algoritma.py

```python
class PapanQueens:

    def __init__(self, data_papan):
```

```
        self.papan = data_papan
        self.n = len(data_papan)
        self.total_iterasi = 0

    def cek_aman(self, baris_solusi):
        kolom_terpakai = [False] * self.n
        warna_terpakai = set()

        for r in range(self.n):
            c = baris_solusi[r]
            if kolom_terpakai[c]: return False
            kolom_terpakai[c] = True

            warna = self.papan[r][c]
            if warna in warna_terpakai: return False
            warna_terpakai.add(warna)

            for r_lama in range(r):
                c_lama = baris_solusi[r_lama]
                if abs(r_lama - r) <= 1 and abs(c_lama - c) <= 1:
                    return False
        return True
```

```
    def cari_exhaustive(self, baris, solusi_skrg, callback_visual):
        if baris == self.n:
            self.total_iterasi += 1
            if self.total_iterasi % 2000 == 0:
                callback_visual(solusi_skrg)
            return self.cek_aman(solusi_skrg)

        for col in range(self.n):
            solusi_skrg[baris] = col
            if self.cari_exhaustive(baris + 1, solusi_skrg,
callback_visual):
                return True
        return False
```

tampilan.py

```
import tkinter as tk
from tkinter import filedialog
import time
import threading
from algoritma import PapanQueens

class AplikasiQueens:
    def __init__(self, master):
        self.master = master
        self.master.title("Penyelesaian Permainan Queens Linkedin")
        self.master.geometry("700x900")

        self.data_papan = []
        self.n = 0
```

```
        self.solusi_akhir = []
        self.kotak_gui = []
        self.lagi_jalan = False
        self.waktu_mulai = 0
        self.warna_list = ["#FF595E", "#FFCA3A", "#8AC926", "#1982C4",
"#6A4C93",
                           "#FF924C", "#25A18E", "#FF6392", "#562C2C",
"#D8E2DC"]
        self.buat_ui()

    def buat_ui(self):
        tk.Label(self.master, text="QUEENS", font=("Helvetica", 22,
"bold"), pady=20).pack()

        frame_ctrl = tk.Frame(self.master)
        frame_ctrl.pack(pady=10)

        self.btn_pilih = tk.Button(frame_ctrl, text="LOAD FILE",
bg="#4CAF50", fg="white", font=("Arial", 10, "bold"), padx=15,
command=self.pilih_file)
        self.btn_pilih.pack(side=tk.LEFT, padx=10)

        self.btn_run = tk.Button(frame_ctrl, text="SOLVE", bg="#2196F3",
fg="white", font=("Arial", 10, "bold"), padx=15, state="disabled",
command=self.mulai_cari)
        self.btn_run.pack(side=tk.LEFT, padx=10)

        self.txt_status = tk.StringVar(value="Status: Siap")
        tk.Label(self.master, textvariable=self.txt_status,
font=("Consolas", 11)).pack()

        self.lbl_waktu = tk.Label(self.master, text="Waktu pencarian: 0
ms", font=("Consolas", 11))
        self.lbl_waktu.pack()
        self.lbl_iterasi = tk.Label(self.master, text="Banyak kasus yang
ditinjau: 0", font=("Consolas", 11))
        self.lbl_iterasi.pack()

        self.frame_simpan = tk.Frame(self.master)
```

```
        self.lbl_tanya = tk.Label(self.frame_simpan, text="Apakah Anda
ingin menyimpan solusi? (Ya/Tidak)",
                                  font=("Consolas", 11, "bold"), pady=5)
        self.lbl_tanya.pack()

        f_btn_simpan = tk.Frame(self.frame_simpan)
        f_btn_simpan.pack()
        tk.Button(f_btn_simpan, text="Ya", width=10,
command=self.aksi_simpan).pack(side=tk.LEFT, padx=5)
        tk.Button(f_btn_simpan, text="Tidak", width=10,
command=self.aksi_batal).pack(side=tk.LEFT, padx=5)

        self.area_papan = tk.Frame(self.master, borderwidth=1,
relief="solid")
        self.area_papan.pack(padx=20, pady=20)

    def pilih_file(self):
        path = filedialog.askopenfilename(filetypes=[("Text files",
"*.txt")])
        if path:
            self.frame_simpan.pack_forget()
            with open(path, 'r') as f:
                baris_teks = [l.strip() for l in f.readlines() if
l.strip()]

            self.data_papan = [list(l) for l in baris_teks]
            self.n = len(self.data_papan)
            self.buat_grid_papan()
            self.btn_run.config(state="normal")
            self.txt_status.set(f"Papan {self.n}x{self.n} dimuat.")

    def buat_grid_papan(self):
        for w in self.area_papan.winfo_children(): w.destroy()
        self.kotak_gui = []
        chars = sorted(list(set(c for r in self.data_papan for c in r)))
        cmap = {h: self.warna_list[i % len(self.warna_list)] for i, h in
enumerate(chars)}

        for r in range(self.n):
            baris_kotak = []
```

```
            for c in range(self.n):
                l = tk.Label(self.area_papan, text="",
bg=cmap[self.data_papan[r][c]],
                             width=4, height=2, relief="solid",
borderwidth=1, font=("Arial", 14, "bold"))
                l.grid(row=r, column=c)
                baris_kotak.append(l)
            self.kotak_gui.append(baris_kotak)

    def refresh_papan(self, solusi):
        for r in range(self.n):
            for c in range(self.n):
                self.kotak_gui[r][c].config(text="Q" if solusi[r] == c
else "")
        self.master.update_idletasks()

    def timer_loop(self):
        if self.lagi_jalan:
            skrg = (time.time() - self.waktu_mulai) * 1000
            self.lbl_waktu.config(text=f"Waktu pencarian: {skrg:.2f} ms")
            self.master.after(50, self.timer_loop)

    def mulai_cari(self):
        self.frame_simpan.pack_forget()
        self.btn_run.config(state="disabled")
        self.lagi_jalan = True
        self.waktu_mulai = time.time()
        self.timer_loop()
        threading.Thread(target=self.proses_hitung, daemon=True).start()

    def proses_hitung(self):
        solver = PapanQueens(self.data_papan)
        self.solusi_akhir = [0] * self.n
        ketemu = solver.cari_exhaustive(0, self.solusi_akhir,
self.refresh_papan)
        durasi = (time.time() - self.waktu_mulai) * 1000
        self.lagi_jalan = False

        self.master.after(0, lambda: self.lbl_waktu.config(text=f"Waktu
pencarian: {durasi:.2f} ms"))
```

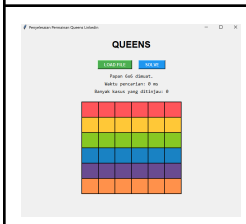
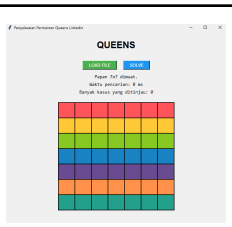
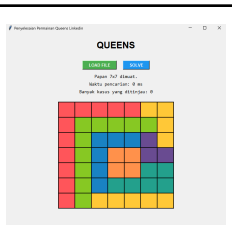
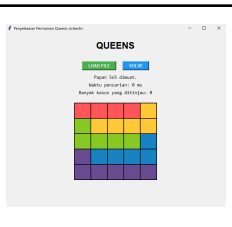
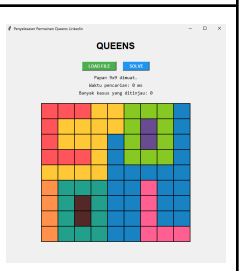
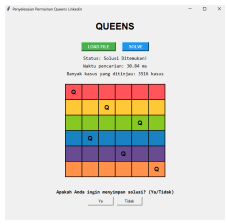
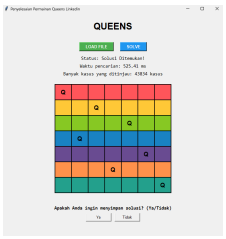
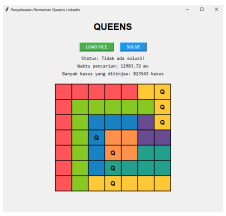
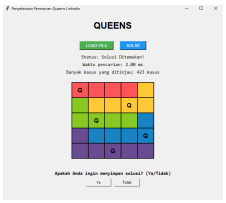
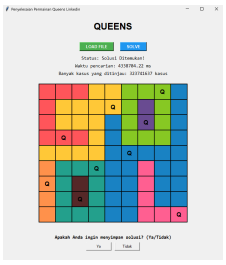
```
        self.master.after(0, lambda: self.lbl_iterasi.config(text=f"Banyak
kasus yang ditinjau: {solver.total_iterasi} kasus"))

        if ketemu:
            self.master.after(0, self.sukses_ketemu)
        else:
            self.master.after(0, lambda: self.txt_status.set("Status:
Tidak ada solusi!"))
        self.master.after(0, lambda: self.btn_run.config(state="normal"))

    def sukses_ketemu(self):
        self.refresh_papan(self.solusi_akhir)
        self.txt_status.set("Status: Solusi Ditemukan!")
        self.frame_simpan.pack(pady=10)

    def aksi_simpan(self):
        nama_file = "solusi.txt"
        with open(nama_file, 'w') as f:
            for r in range(self.n):
                row = "".join(["#" if self.solusi_akhir[r] == c else
self.data_papan[r][c] for c in range(self.n)])
                f.write(row + "\n")
        self.frame_simpan.pack_forget()
        self.txt_status.set(f"Status: Solusi berhasil disimpan di
{nama_file}!")

    def aksi_batal(self):
        self.frame_simpan.pack_forget()
        self.txt_status.set("Status: Penyimpanan dibatalkan.")
```

main.py

```
import tkinter as tk

from tampilan import AplikasiQueens


def main():
```

```
    root = tk.Tk()
    app = AplikasiQueens(root)
    root.mainloop()

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

4. Testing

test1.txt	test2.txt	test3.txt	test4.txt	test5.txt
AAAAAA BBBBBB CCCCCC DDDDDD EEEEEE FFFFFF	AAAAAAA BBBBBBB CCCCCCC DDDDDDD EEEEEEE FFFFFFF HHHHHHH	AAAAABB ACCCCCB ACDDDEB ACDFFEE ACDFFGG ACDGGGG ACBBBBB	AAAAB CBBBB CCCCD EDDDD EEEEE	AAABBCCCD ABBBBCECD ABBBDCECD AAABDCCCD BBBBDDDDD FGGGDDHDD FGIGDDHDD FGIGDDHDD FGGGDDHHH
				
				

5. Lampiran

Github: https://github.com/jonveral/Tucil1_13524120

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.

Jonathan Alveraldo Bangun