LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA
PENYELESAIAN PERMAINAN *QUEENS LINKEDIN* DENGAN ALGORITMA *BRUTE FORCE*

Oleh:
Fazri Arrashyi Putra
13524127
Rev.01

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2026

DAFTAR ISI

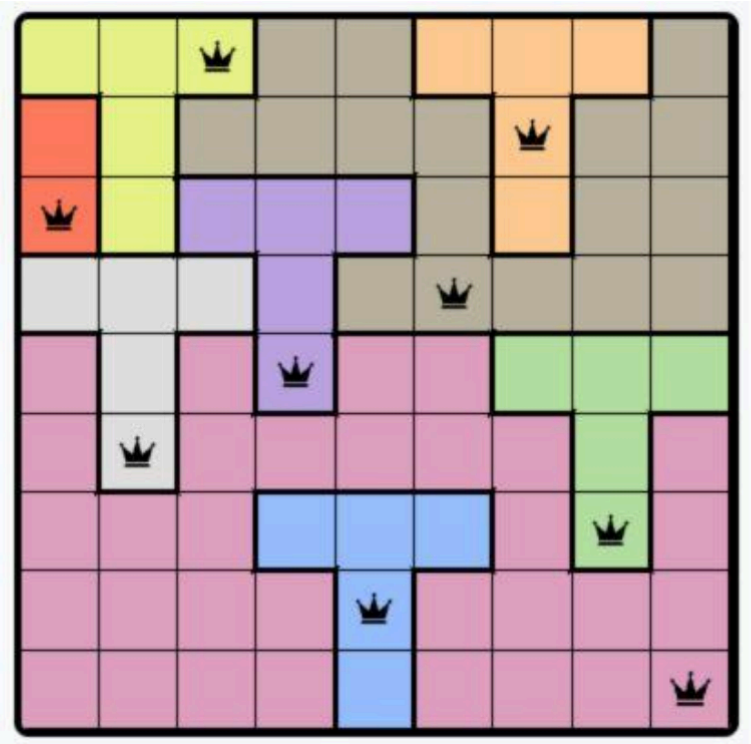
CATATAN PERUBAHAN

Rev.01	Memperbarui bagian kode sumber untuk menampilkan backtracking murni, alih-alih brute force dengan backtracking. Menambahkan pranala github

BAB I
DESKRIPSI PROGRAM

Permainan queens merupakan permainan yang terdapat pada platform profesional yaitu LinkedIn, permainan ini melatih logika dan cara berpikir bagaimana menempatkan sejumlah ratu (queens) pada papan catur berukuran N x N sehingga tidak ada dua ratu pun yang saling menyerang satu sama lain. Tujuan utama dari permainan ini adalah menemukan semua kemungkinan konfigurasi penempatan ratu yang valid pada papan.

Program ini bertujuan untuk menyelesaikan permainan Queens LinkedIn, yang pada dasarnya adalah masalah N-Queens, dengan menerapkan Algoritma Brute Force. Algoritma Brute Force akan mengecek secara sistematis setiap kemungkinan penempatan ratu pada papan catur untuk mencari solusi yang memenuhi syarat, yaitu tidak ada ratu yang berada pada baris, kolom, atau diagonal yang sama dengan ratu lainnya. Pendekatan ini dipilih untuk menganalisis dan membandingkan efektivitasnya dalam menyelesaikan masalah ini.



BAB II
ALGORITMA PEMROGRAMAN

Algoritma yang digunakan untuk menyelesaikan permainan *Queens LinkedIn* (masalah N-Queens) dalam tugas ini adalah **Algoritma Brute Force**.

A. Konsep Brute Force

Algoritma Brute Force adalah algoritma yang menyelesaikan persoalan secara lempang (straightforward), yang memecahkan persoalan secara sederhana, langsung, jelas caranya dan langkah-langkahnya mudah dipahami. Dalam konteks masalah N-Queens, ini berarti program akan:

1. Mencoba semua kemungkinan penempatan ratu yang mungkin pada papan sebanyak N x N.
2. Memverifikasi penempatan, memastikan hanya satu ratu yang ditempatkan pada baris, kolom dan warna daerahnya.
3. Jika konfigurasi valid dan semua N ratu tellah ditempatkan maka konfigurasi tersebut dicatat sebagai salah satu solusi.
4. Jika verifikasi penempatan tidak valid, maka akan mencoba kombinasi penempatan yang lainnya.

B. Implementasi Khusus untuk N-Queens

Untuk mengimplementasikan Brute Force pada masalah Queens Puzzle secara efisien, program menggunakan rekursi dan backtracking untuk membangun solusi secara berurutan, baris per baris.

1. **Representasi Papan**: Papan direpresentasikan sebagai array dua dimensi (list of lists), di mana setiap elemen menyimpan karakter yang mewakili daerah (region) pada posisi tersebut. Posisi ratu yang sudah ditempatkan disimpan dalam list **daftar_queen** yang berisi tuplle (baris, kolom).
2. **Fungsi Rekursif**: Fungsi **_brute_force_rekursif()** mencoba menempatkan ratu pada baris saat ini. Untuk setiap kolom yang mungkin di baris tersebut, fungsi akan:
    a. **Pengecekan Konflik**: Memeriksa apakah penempatan pada kolom tersebut valid dengan memanggil **_cek_penempatan_valid()**. Fungsi ini memeriksa tiga aturan:
        - Tidak ada ratu lain pada baris yang sama
        - Tidak ada ratu lain pada kolom yang sama
        - Tidak ada ratu lain pada daerah (region) yang sama
    b. **Maju**: Jika penempatan valid, ratu ditambahkan ke **daftar_queen** dengan **append**((baris, kolom)), dan fungsi rekursif dipanggil untuk mencoba menempatkan ratu pada baris berikutnya (baris + 1).
    c. **Mundur (Backtracking)**: Jika penempata pada baris berikutnya gagal menghasilkan solusi (fungsi rekursif mengembalikan False), ratu pada

baris saat ini dihapus dari **daftar_queen** menggunakan **pop()** untuk mencoba kolom lain.

Meskipun algoritma Brute Force menjamin penemuan solusi karena mengecek setiap kemungkinan, efisiensinya rendah (kompleksitas waktu eksponensial, O(N^N) untuk papan berukuran N×N), sehingga tidak praktis untuk nilai N yang besar.

BAB III
KODE SUMBER

A. Algoritma Brute Force yang digunakan

Berikut adalah Algoritma Brute Force yang digunakan, penjelasan ada dalam komentar kode berikut:

```python
def selesaikan(self, visualisasi=True, algoritma='backtracking'):
        # Inisialisasi: Reset counter dan daftar ratu
        self.kasus_ditinjau = 0

        if visualisasi:
            print(f"\nMemulai pencarian solusi...")
            time.sleep(1)

        # Pilih algoritma
        waktu_mulai = time.time()
        if algoritma == 'brute_force_murni':
            hasil = self._brute_force_murni(visualisasi)

        else:
            daftar_queen = []
            hasil = self._brute_force_rekursif(daftar_queen, 0, visualisasi)
        waktu_selesai = time.time()

        # Hitung waktu eksekusi (I/O diabaikan)
        self.waktu_pencarian_ms = int((waktu_selesai - waktu_mulai) * 1000)

        if visualisasi and hasil:
            self._tampilkan_papan(self.solusi, bersihkan_layar=True)

        return hasil
```

```python
def _brute_force_murni(self, visualisasi=True):
        # Generate semua permutasi kolom untuk n baris, setiap permutasi merepresentasikan penempatan ratu di setiap baris.
        semua_kolom = list(range(self.ukuran))
        semua_permutasi = self._generate_permutasi(semua_kolom)

        # Hitung total kemungkinan untuk progress
        total_kemungkinan = len(semua_permutasi)
        print(f"Total kemungkinan kombinasi: {total_kemungkinan}")

        # MEncoba setiap permutasi
        for idx, permutasi_kolom in enumerate(semua_permutasi):
            self.kasus_ditinjau += 1

            # Daftar queen dari permutasi
            daftar_queen = [(i, permutasi_kolom[i]) for i in range(self.ukuran)]
            # Visualisasi progress
            interval_update = max(1, total_kemungkinan // 100) if total_kemungkinan > 100 else 1
            if visualisasi and (self.kasus_ditinjau % interval_update == 0 or self.kasus_ditinjau <= 10):
                self._tampilkan_papan(daftar_queen)
                print(f"Progress: {self.kasus_ditinjau}/{total_kemungkinan} kombinasi")
                time.sleep(0.05)


            # Validasi solusi
            if self._validasi_solusi_lengkap(daftar_queen):
                self.solusi = daftar_queen[:]
                return True
        return False
```

```python
def _generate_permutasi(self, elemen):
        if len(elemen) <= 1:
            return [elemen]

        hasil = []
        for i in range(len(elemen)):
            # Ambil elemen ke-i sebagai elemen pertama
            elemen_pertama = elemen[i]
            elemen_sisa = elemen[:i] + elemen[i+1:]
            # Buat permutasi untuk elemen sisa.
            permutasi_sisa = self._generate_permutasi(elemen_sisa)

            # Gabungkan elemen pertama dengan setiap permutasi sisa
            for perm in permutasi_sisa:
                hasil.append([elemen_pertama] + perm)
        return hasil
```

```python
def _validasi_solusi_lengkap(self, daftar_queen):
        if len(daftar_queen) != self.ukuran:
            return False
        # Cek apakah semua baris unik.
        baris_set = set()
        for b, k in daftar_queen:
            if b in baris_set:
                return False
            baris_set.add(b)

        # Cek apakah semua kolom unik
        kolom_set = set()
        for b, k in daftar_queen:
            if k in kolom_set:
                return False

            kolom_set.add(k)

        # Cek apakah semua daerah unik,
        daerah_set = set()
        for b, k in daftar_queen:
            karakter_daerah = self.papan[b][k]
            if karakter_daerah in daerah_set:

                return False
            daerah_set.add(karakter_daerah)
        return True
```

B. Kode Sumber Lengkap

```python
import os
import time

class PuzzleQueens:
    def __init__(self, papan):
        self.papan = papan
        self.ukuran = len(papan)
        self.daerah = self._parse_daerah()
        self.solusi = None
        self.kasus_ditinjau = 0

    def _parse_daerah(self):
        daerah = {}
        for i in range(self.ukuran):
            for j in range(self.ukuran):
                karakter_daerah = self.papan[i][j]
                if karakter_daerah not in daerah:
                    daerah[karakter_daerah] = []
                daerah[karakter_daerah].append((i, j))
        return daerah

    def _cek_penempatan_valid(self, daftar_queen, baris, kolom):
        # Aturan 1: Cek apakah baris sudah ada ratu
        for b, k in daftar_queen:
            if b == baris:
                return False
        # Aturan 2: Cek apakah kolom sudah ada ratu
        for b, k in daftar_queen:
            if k == kolom:
                return False
        # Aturan 3: Cek apakah daerah (region) sudah ada ratu
        karakter_daerah = self.papan[baris][kolom]
        for b, k in daftar_queen:
            if self.papan[b][k] == karakter_daerah:
                return False

        return True
```

```python
def _validasi_solusi_lengkap(self, daftar_queen):
        if len(daftar_queen) != self.ukuran:
            return False
        # Cek apakah semua baris unik.
        baris_set = set()
        for b, k in daftar_queen:
            if b in baris_set:
                return False
            baris_set.add(b)

        # Cek apakah semua kolom unik
        kolom_set = set()
        for b, k in daftar_queen:
            if k in kolom_set:
                return False

            kolom_set.add(k)

        # Cek apakah semua daerah unik,
        daerah_set = set()
        for b, k in daftar_queen:
            karakter_daerah = self.papan[b][k]
            if karakter_daerah in daerah_set:

                return False
            daerah_set.add(karakter_daerah)
        return True

    def _tampilkan_papan(self, daftar_queen, bersihkan_layar=True):
        if bersihkan_layar:
            os.system('clear' if os.name != 'nt' else 'cls')

        print("\n" + "="*50)
        print("Pencarian Solusi (Live Update)")
        print("="*50 + "\n")
        papan_tampilan = [baris[:] for baris in self.papan]
        for b, k in daftar_queen:
            papan_tampilan[b][k] = '#'

        for baris in papan_tampilan:
            print(''.join(baris))

        print(f"\nRatu ditempatkan: {len(daftar_queen)}/{self.ukuran}")
        print(f"Kasus yang ditinjau: {self.kasus_ditinjau}")
        print("="*50 + "\n")
```

```python
def _brute_force_rekursif(self, daftar_queen, baris, visualisasi=True):
        # Base case: Jika semua baris sudah diproses
        if baris == self.ukuran:
            # Cek apakah semua ratu sudah ditempatkan
            if len(daftar_queen) == self.ukuran:
                self.solusi = daftar_queen[:]
                return True
            return False

        # Mencoba semua kolom untuk baris saat ini
        for kolom in range(self.ukuran):
            self.kasus_ditinjau += 1

            # Cek apakah penempatan ratu valid
            if self._cek_penempatan_valid(daftar_queen, baris, kolom):
                # MAJU: Tambahkan ratu ke daftar
                daftar_queen.append((baris, kolom))

                # Visualisasi progress
                if visualisasi and (self.kasus_ditinjau % 1000 == 0 or len(daftar_queen) <= 3):
                    self._tampilkan_papan(daftar_queen)
                    time.sleep(0.1)

                # Rekursi: Coba menempatkan ratu pada baris berikutnya
                if self._brute_force_rekursif(daftar_queen, baris + 1, visualisasi):
                    return True
                # BACKTRACK: untuk mencoba kolom lain
                daftar_queen.pop()
        # Jika semua kolom sudah dicoba dan tidak ada yang valid, return False
        return False

    def _generate_permutasi(self, elemen):
        if len(elemen) <= 1:
            return [elemen]

        hasil = []
        for i in range(len(elemen)):
            # Ambil elemen ke-i sebagai elemen pertama
            elemen_pertama = elemen[i]
            elemen_sisa = elemen[:i] + elemen[i+1:]
            # Buat permutasi untuk elemen sisa.
            permutasi_sisa = self._generate_permutasi(elemen_sisa)

            # Gabungkan elemen pertama dengan setiap permutasi sisa
            for perm in permutasi_sisa:
                hasil.append([elemen_pertama] + perm)
        return hasil

    def _brute_force_murn
```

```python
def _brute_force_murni(self, visualisasi=True):
        # Generate semua permutasi kolom untuk n baris, setiap permutasi merepresentasikan penempatan ratu di setiap baris.
        semua_kolom = list(range(self.ukuran))
        semua_permutasi = self._generate_permutasi(semua_kolom)

        # Hitung total kemungkinan untuk progress
        total_kemungkinan = len(semua_permutasi)
        print(f"Total kemungkinan kombinasi: {total_kemungkinan}")

        # MEncoba setiap permutasi
        for idx, permutasi_kolom in enumerate(semua_permutasi):
            self.kasus_ditinjau += 1

            # Daftar queen dari permutasi
            daftar_queen = [(i, permutasi_kolom[i]) for i in range(self.ukuran)]
            # Visualisasi progress
            interval_update = max(1, total_kemungkinan // 100) if total_kemungkinan > 100 else 1
            if visualisasi and (self.kasus_ditinjau % interval_update == 0 or self.kasus_ditinjau <= 10):
                self._tampilkan_papan(daftar_queen)
                print(f"Progress: {self.kasus_ditinjau}/{total_kemungkinan} kombinasi")
                time.sleep(0.05)


            # Validasi solusi
            if self._validasi_solusi_lengkap(daftar_queen):
                self.solusi = daftar_queen[:]
                return True
        return False

    def selesaikan(self, visualisasi=True, algoritma='backtracking'):
        # Inisialisasi: Reset counter dan daftar ratu
        self.kasus_ditinjau = 0

        if visualisasi:
            print(f"\nMemulai pencarian solusi...")
            time.sleep(1)

        # Pilih algoritma
        waktu_mulai = time.time()
        if algoritma == 'brute_force_murni':
            hasil = self._brute_force_murni(visualisasi)

        else:
            daftar_queen = []
            hasil = self._brute_force_rekursif(daftar_queen, 0, visualisasi)
        waktu_selesai = time.time()

        # Hitung waktu eksekusi (I/O diabaikan)
        self.waktu_pencarian_ms = int((waktu_selesai - waktu_mulai) * 1000)

        if visualisasi and hasil:
            self._tampilkan_papan(self.solusi, bersihkan_layar=True)

        return hasil
```

```python
def dapatkan_papan_solusi(self):
        if not self.solusi:
            return None

        papan_solusi = [baris[:] for baris in self.papan]
        for b, k in self.solusi:
            papan_solusi[b][k] = '#'

        return papan_solusi

    def cetak_solusi(self):
        if not self.solusi:
            print("Tidak ada solusi ditemukan!")
            return

        papan_solusi = self.dapatkan_papan_solusi()
        print("\n" + "="*50)
        print("SOLUSI FINAL")
        print("="*50 + "\n")

        for baris in papan_solusi:
            print(''.join(baris))

        print(f"\nWaktu pencarian: {self.waktu_pencarian_ms} ms")
        print(f"Banyak kasus yang ditinjau: {self.kasus_ditinjau} kasus")
        print("="*50 + "\n")


def baca_file_papan(nama_file):
    try:
        if not os.path.isabs(nama_file):
            base_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
            test_case_dir = os.path.join(base_dir, 'test', 'case')
            nama_file = os.path.join(test_case_dir, nama_file)

        with open(nama_file, 'r') as file:
            baris_baris = file.readlines()


        papan = []
        for baris in baris_baris:
            baris = baris.strip()
            if baris:
                papan.append(list(baris))

        return papan
    except FileNotFoundError:
        print(f"Error: File '{nama_file}' tidak ditemukan!")
        return None
    except Exception as e:
        print(f"Error membaca file: {e}")
        return None
```

```python
def validasi_papan(papan):
    if not papan:
        return False, "Papan kosong!"

    ukuran = len(papan)

    for i, baris in enumerate(papan):
        if len(baris) != ukuran:
            return False, f"Baris {i+1} memiliki panjang {len(baris)}, seharusnya {ukuran}"

    set_daerah = set()
    for baris in papan:
        for karakter in baris:
            set_daerah.add(karakter)

    if len(set_daerah) < ukuran:
        return False, f"Jumlah daerah ({len(set_daerah)}) kurang dari ukuran papan ({ukuran})"

    jumlah_daerah = {}
    for baris in papan:
        for karakter in baris:
            if karakter not in jumlah_daerah:
                jumlah_daerah[karakter] = 0
            jumlah_daerah[karakter] += 1

    for daerah, jumlah in jumlah_daerah.items():
        if jumlah == 0:
            return False, f"Daerah '{daerah}' tidak memiliki cell"

    return True, "Valid"


def simpan_solusi(nama_file, penyelesai):
    if not penyelesai.solusi:
        print("Tidak ada solusi untuk disimpan!")
        return

    papan_solusi = penyelesai.dapatkan_papan_solusi()

    if '.' in nama_file:
        nama_base = os.path.basename(nama_file).rsplit('.', 1)[0]

    else:
        nama_base = os.path.basename(nama_file)

    base_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    solutions_dir = os.path.join(base_dir, 'test', 'solutions')
    os.makedirs(solutions_dir, exist_ok=True)

    nama_file_output = os.path.join(solutions_dir, f"{nama_base}_solution.txt")

    try:
        with open(nama_file_output, 'w') as file:
            for baris in papan_solusi:
                file.write(''.join(baris) + '\n')
        print(f"Solusi berhasil disimpan ke '{nama_file_output}'")
    except Exception as e:
        print(f"Error menyimpan solusi: {e}")

```

```python
def main():
    print("="*60)
    print("QUEENS PUZZLE SOLVER - ALGORITMA BRUTE FORCE")
    print("Oleh: Fazri Arrashyi Putra - 13524127")
    print("="*60)
    print("Aturan: Setiap baris, kolom, dan daerah harus memiliki tepat satu ratu.\n")

    nama_file = input("Masukkan nama file test case (.txt): ").strip()

    if not nama_file.endswith('.txt'):
        nama_file += '.txt'
    papan = baca_file_papan(nama_file)
    if not papan:
        return

    valid, pesan = validasi_papan(papan)
    if not valid:
        print(f"Error validasi: {pesan}")
        return

    print(f"\nPapan berhasil dibaca! Ukuran: {len(papan)}x{len(papan)}")

    print("\nPilih algoritma:")
    print("1. Brute Force dengan Backtracking")
    print("2. Brute Force - Murni")
    input_algoritma = input("Masukkan pilihan (1/2): ").strip()

    if input_algoritma == '2':
        algoritma = 'brute_force_murni'
        print("\nPeringatan: Brute Force Murni akan mencoba SEMUA kombinasi!")
        print("   Untuk papan besar, ini bisa memakan waktu sangat lama.")
        konfirmasi = input("   Lanjutkan? (Ya/Tidak): ").strip().lower()
        if konfirmasi not in ['ya', 'y', 'yes']:
            print("Dibatalkan.")
            return
    else:
        algoritma = 'backtracking'

    input_visualisasi = input("\nTampilkan live update saat pencarian? (Ya/Tidak): ").strip().lower()
    visualisasi = input_visualisasi in ['ya', 'y', 'yes']

    penyelesai = PuzzleQueens(papan)

    if penyelesai.selesaikan(visualisasi=visualisasi, algoritma=algoritma):
        penyelesai.cetak_solusi()

        input_simpan = input("Apakah Anda ingin menyimpan solusi? (Ya/Tidak): ").strip().lower()
        if input_simpan in ['ya', 'y', 'yes']:
            simpan_solusi(nama_file, penyelesai)


    else:
        print("\n" + "="*50)
        print("TIDAK ADA SOLUSI DITEMUKAN")
        print("="*50)
        print(f"Waktu pencarian: {penyelesai.waktu_pencarian_ms} ms")
        print(f"Banyak kasus yang ditinjau: {penyelesai.kasus_ditinjau} kasus")
        print("="*50 + "\n")


if __name__ == "__main__":
    main()

```

Library yang digunakan pada program ini ada dua yakni OS untuk input/output file dan time untuk menghitung waktu eksekusi serta fungsi sleep.

BAB IV
MASUKKAN dan LUARAN PROGRAM

A. Test Case 1

Isi Test Case 1

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

Program dijalankan dan memilih test case 1

```
============================================================
Aturan: Setiap baris, kolom, dan daerah harus memiliki tepat satu ratu.

Masukkan nama file test case (.txt): test_case_1

Papan berhasil dibaca! Ukuran: 9x9

Tampilkan live update saat pencarian? (Ya/Tidak):
```

Output dan hasil .txt

```
==================================================
SOLUSI FINAL
==================================================

A#ABBCCCD
ABBB#CECD
ABBBDC#CD
AAABD#CCD
BBBBDDD#D
#GGGDDHDD
FG#GDDHDD
FGI#DDHDD
FGGGDDHH#

Waktu pencarian: 7249 ms
Banyak kasus yang ditinjau: 5652 kasus
==================================================

Apakah Anda ingin menyimpan solusi? (Ya/Tidak): Ya
Solusi berhasil disimpan ke '/home/fazridep/Documents/Kuliah/Strategi Algoritma/Tucil1/test/solutions/test_ca
_1_solution.txt'
~/Documents/Kuliah/Strategi Algoritma/Tucil1                    ✓ 53s Tucil1 Py 17:02:50
```

st > solutions > tes

```
1  A#ABBCCCD
2  ABBB#CECD
3  ABBBDC#CD
4  AAABD#CCD
5  BBBBDDD#D
6  #GGGDDHDD
7  FG#GDDHDD
8  FGI#DDHDD
9  FGGGDDHH#
```

B. Test Case 2

Isi Test Case 2

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

Program dijalankan dan memilih test case 2

```
============================================================
QUEENS PUZZLE SOLVER - ALGORITMA BRUTE FORCE
Oleh: Fazri Arrashyi Putra - 13524127
============================================================
Aturan: Setiap baris, kolom, dan daerah harus memiliki tepat satu ratu.

Masukkan nama file test case (.txt): test_case_2

Papan berhasil dibaca! Ukuran: 9x9

Tampilkan live update saat pencarian? (Ya/Tidak):
```

Output dan hasil .txt

```
============================================
SOLUSI FINAL
============================================

FGGGDDH#H
#GIGDDHDD
FG#GDDHDD
FGG#DDHDD
BBBBDDDD#
A#ABDCCCD
ABBBD#ECD
ABBBBC#CD
AAAB#CCCD

Waktu pencarian: 27435 ms
Banyak kasus yang ditinjau: 46242 kasus
============================================

Apakah Anda ingin menyimpan solusi? (Ya/Tidak):
```

solutions > tes

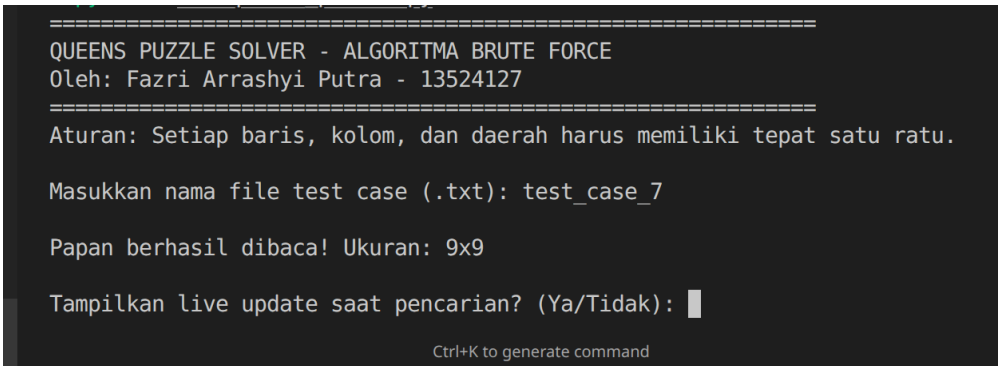
```
FGGGDDH#H
#GIGDDHDD
FG#GDDHDD
FGG#DDHDD
BBBBDDDD#
A#ABDCCCD
ABBBD#ECD
ABBBBC#CD
AAAB#CCCD
```

C. Test Case 3

Isi Test Case 3

```
AAABBBCCC
AAABBBCCC
DDDEEEFFF
DDDEEEFFF
GGGHHHIII
GGGHHHIII
DDDEEEFFF
GGGHHHIII
DDDEEEFFF
```

Program dijalankan dan memilih test case 7



Output dan hasil .txt

D. Test Case 4

Isi Test Case 4

```
DCCCBBAAA
DCECBBBBA
DCECDBBBA
DCCCDBAAA
DDDDDBBBB
DDHDDGGGF
DDHDDGIGF
DDHDDGIGF
HHHDDGGGF
```

Program dijalankan dan memilih test case 3

```
❯ python3 src/queens_puzzle.py
============================================================
QUEENS PUZZLE SOLVER - ALGORITMA BRUTE FORCE
Oleh: Fazri Arrashyi Putra - 13524127
============================================================
Aturan: Setiap baris, kolom, dan daerah harus memiliki tepat satu ratu

Masukkan nama file test case (.txt): test_case_3

Papan berhasil dibaca! Ukuran: 9x9

Tampilkan live update saat pencarian? (Ya/Tidak):
```

Output dan hasil .txt

```
==================================================
SOLUSI FINAL
==================================================

D#CCBBAAA
DCEC#BBBA
DC#CDBBBA
DCCCDBA#A
DDD#DBBBB
DDHDD#GGF
DDHDDG#GF
DDHDDGIG#
#HHDDGGGF

Waktu pencarian: 6683 ms
Banyak kasus yang ditinjau: 6165 kasus
==================================================

Apakah Anda ingin menyimpan solusi? (Ya/Tidak):
```

```
D#CCBBAAA
DCEC#BBBA
DC#CDBBBA
DCCCDBA#A
DDD#DBBBB
DDHDD#GGF
DDHDDG#GF
DDHDDGIG#
#HHDDGGGF
```

E. Test Case 5

Isi Test Case 5

```
AAABBBCCC
AAABBBCCC
DDDDAAABB
EEEFFFGGG
EEEFFFHHH
IIIEEEFFF
GGGHHHIII
GGGHHHIII
EEEFFFIII
```

Program dijalankan dan memilih test case 8

```
Oleh: Fazri Arrashyi Putra - 13524127
============================================================
Aturan: Setiap baris, kolom, dan daerah harus memiliki tepat satu ratu.

Masukkan nama file test case (.txt): test_case_8

Papan berhasil dibaca! Ukuran: 9x9

Tampilkan live update saat pencarian? (Ya/Tidak):
```

Ctrl+K to generate command

Output dan hasil .txt

```
AAABBBCC#
AAABB#CCC
DDDDAA#BB
EE#FFFGGG
EEEFFFH#H
#IIEEEFFF
G#GHHHIII
GGGHHHIII
EEEFFFIII

Ratu ditempatkan: 7/9
Kasus yang ditinjau: 369000
==================================================


==================================================
TIDAK ADA SOLUSI DITEMUKAN
==================================================
Waktu pencarian: 35362 ms
Banyak kasus yang ditinjau: 370260 kasus
==================================================

~/Doc/Ku/Strategi Algoritma/Tucil1          ✔ 41s Tucil1 Py
```

BAB V
LAMPIRAN

No	Poin	Ya	Tidak
1	Program berhasil di kompilasi tanpa kesalahan	V	
2	Program berhasil di jalankan	V	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	V	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	V	
5	Program memiliki Graphical User Interface (GUI)		V
6	Program dapat menyimpan solusi dalam bentuk file gambar		V

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

Fazri Arrashyi Putra

BAB VI
TAUTAN dan REFERENSI

A. **Pranala Github**
   Pranala Repository Github: https://github.com/fazridep/Tucil1_13524127
B. **Referensi:**
   - https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/02-Algoritma-Brute-Force-(2026)-Bag1.pdf
   - https://medium.com/@jeremy.ockenden/solving-linkedins-queens-puzzle-in-python-3f0d53925842