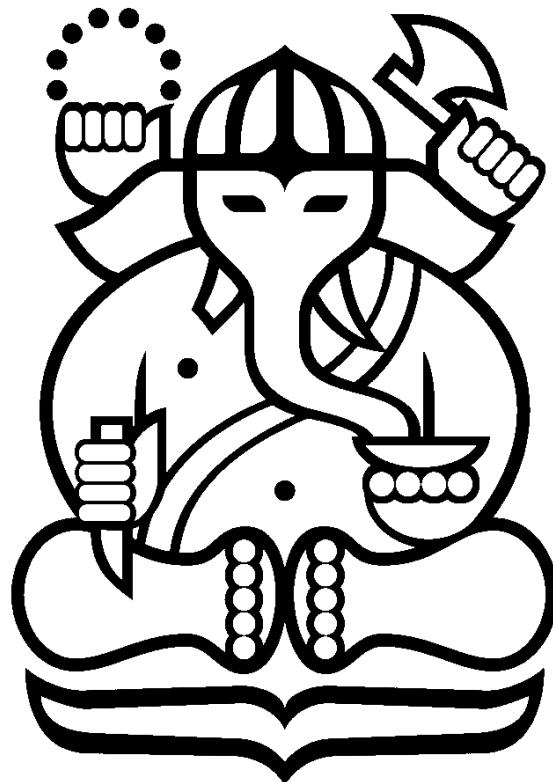


Tugas Kecil 2 IF2211 Strategi Algoritma

**Implementasi Algoritma Brute Force pada Permasalahan
Permainan Queens Linkedin**



Disusun oleh:

Dika Pramudya Nugraha (13524132)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2026

DAFTAR ISI

BAB I DESKRIPSI MASALAH.....	4
1.1. Algoritma <i>Brute Force</i>	4
1.2. Permainan Queens <i>LinkedIn</i>	5
1.3. Algoritma Penyelesaian Permainan <i>Queens LinkedIn</i> dengan Pendekatan Brute Force	5
BAB II IMPLEMENTASI ALGORITMA DALAM BAHASA PYTHON.....	7
2.1. File papan.py.....	7
2.2. File solverQueens.py	8
2.3. File imageGenerator.py.....	9
2.4. File imageParser.py.....	9
2.5. File main_gui.py	10
BAB III SOURCE CODE PROGRAM IMPLEMENTASI QUEENS LINKEDIN.....	13
3.1. Repository Program	13
3.2. Source Code.....	13
3.2.1. Papan.py.....	13
3.2.2. solverQueens.py.....	14
3.2.3. imageGenerator.py	19
3.2.4. imageParser.py	20
3.2.5. main_gui.py	25
BAB IV MASUKAN DAN LUARAN PROGRAM	43
4.1. Test Case Txt	43
4.1.1. Test Case 1 – 9x9	43
4.1.2. Test Case 2 – 4x4	43
4.1.3. Test Case 3 – 5x5 Unik	44
4.1.4. Test Case 4 – 4x4 Jumlah Wilayah Kurang Dari N	44
4.1.5. Test Case 5 – 3x2 Tidak Persegi	45
4.1.6. Test Case 6 – 4x4 Karakter Diluar Alfabet (A-Z)	45
4.1.7. Test Case 7 – N lebih dari 26	46
4.1.8. Test Case 8 – Warna Terpisah	47
4.1.9. Test Case 9 – Tidak Ada Solusi.....	47
4.2. Test Case Gambar	48
4.2.1. Test Case 1 – 9x9	48

4.2.2. Test Case 2 – 5x5	49
4.2.3. Test Case 3 – Jumlah Wilayah Tidak Sesuai Dengan N	49
4.2.4. Test Case 4 – 5x5 Solusi Tidak Ditemukan	50
4.2.5. Test Case 5 – 8x8	51
4.3. Download Output.....	51
BAB V LAMPIRAN.....	53
PERNYATAAN TIDAK MELAKUKAN KECURANGAN	54
DAFTAR PUSTAKA	55

BAB I

DESKRIPSI MASALAH

1.1. Algoritma *Brute Force*

Algoritma *brute force* adalah algoritma yang menyelesaikan persoalan secara sederhana, langsung, jelas caranya (*obvious way*), dan lempang (*straightforward*) yang dilakukan dengan cara mencoba seluruh kemungkinan solusi secara sistematis sampai ditemukan solusi yang memenuhi seluruh batasan yang diberikan[1]. Pendekatan ini juga sering disebut sebagai *exhaustive search* karena memeriksa seluruh ruang Solusi tanpa menggunakan heuristik atau strategi optimasi khusus. Dalam metode ini, setiap kemungkinan Solusi akan di periksa satu per satu berdasarkan batasan yang berlaku pada permasalahan yang sedang diselesaikan. Karena sifatnya yang menyeluruh, *brute force* menjamin bahwa apabila solusi memang ada di dalam ruang pencarian, maka solusi tersebut pasti dapat ditemukan.

Dalam proses eksekusinya, algoritma *brute force* tidak melakukan analisis mendalam terhadap struktur masalah untuk mempercepat pencarian, melainkan bergantung sepenuhnya pada kekuatan komputasi untuk memeriksa setiap opsi yang tersedia. Untuk setiap kemungkinan solusi, algoritma akan melakukan verifikasi apakah kemungkinan solusi tersebut memenuhi seluruh batasan yang telah ditetapkan. Jika sebuah kemungkinan solusi (kandidat) terbukti memenuhi semua syarat, maka ia akan ditetapkan sebagai solusi, sebaliknya jika kemungkinan solusi tersebut melanggar salah satu aturan, maka algoritma akan mengabaikannya dan segera beralih untuk memeriksa kemungkinan solusi yang potensial berikutnya.

Karakteristik paling menonjol dari algoritma ini adalah sifat kepastiannya. Namun, kelebihan ini diimbangi dengan kelemahan utama berupa efisiensi waktu yang rendah. Kompleksitas waktu algoritma *brute force* umumnya berbanding lurus dengan jumlah total kemungkinan solusi, yang pada kasus-kasus kombinatorial sering kali tumbuh seara eksponensial seiring bertambahnya ukuran masukan.

Secara spesifik dalam tugas ini, algoritma *brute force* yang diimplementasikan haruslah bersifat murni, yang berarti tidak diperbolehkan menggunakan bantuan *heuristik* apapun untuk memandu atau mempercepat pencarian solusi. Namun, saya membuatkan opsi optimasi dan backtracking jika ruangnya terlalu banyak dan itu akan sangat memakan waktu komputasi yang sangat lama, tetapi untuk solusi penyelesaian utamanya saya tetap menggunakan algoritma *brute force* murni. Selain itu, algoritma ini juga harus mampu mencatat metrik kinerja, seperti jumlah konfigurasi atau iterasi yang telah ditinjau serta waktu eksekusi yang dibutuhkan, untuk memberikan gambaran nyata mengenai beban komputasi yang diperlukan dalam menyelesaikan permainan *Queens LinkedIn*.

1.2. Permainan Queens LinkedIn

Permainan Queens LinkedIn adalah permainan logika yang tersedia pada situs jejaring professional LinkedIn. Tujuan dari permainan ini adalah untuk menempatkan ratu pada sebuah papan persegi berukuran $N \times N$ berwarna, sehingga terdapat hanya satu ratu pada tiap baris, kolom, dan daerah warna. Selain itu, satu ratu tidak dapat ditempatkan bersebelahan dengan ratu lainnya, termasuk secara diagonal[2]. Berbeda dengan permasalahan *N-Queens* klasik yang hanya menekankan larangan serangan antar ratu berdasarkan aturan pergerakan catur, permainan ini (*Queens LinkedIn*) menambahkan aturan tambahan berupa pembagian papan ke dalam beberapa *region* atau wilayah berwarna. Dengan demikian, permainan ini menggabungkan konsep penempatan ratu pada papan catur dengan pembatasan berbasis wilayah, sehingga menghasilkan kombinasi *constraint* atau batasan yang lebih kompleks dibandingkan versi klasiknya.

Dalam konteks tugas ini, papan permainan dapat diberikan dalam dua bentuk representasi, yaitu dalam bentuk teks (file .txt) maupun dalam bentuk gambar (.png atau .jpg). Pada representasi teks, setiap huruf merepresentasikan sebuah wilayah tertentu, sehingga huruf yang sama menunjukkan bahwa sel-sel tersebut berada dalam satu wilayah warna yang sama. Sementara itu, pada representasi gambar, wilayah ditentukan berdasarkan warna yang sama dan bersebelahan secara terhubung. Program yang dikembangkan harus mampu membaca kedua jenis masukan tersebut dan mengubahnya menjadi representasi matriks numerik yang dapat diproses oleh algoritma.

Permainan ini pada dasarnya merupakan permasalahan pencarian kombinatorial, karena solusi dicari dari seluruh kemungkinan konfigurasi penempatan ratu pada papan. Tantangan utama dalam permainan ini bukan hanya memastikan tidak adanya konflik antar ratu pada baris dan kolom, tetapi juga memastikan bahwa seluruh batasan atau *constraint* tambahan terpenuhi secara simultan. Oleh karena itu, permainan *Queens LinkedIn* menjadi studi kasus yang tepat untuk mengimplementasikan pendekatan *brute force*, karena solusi dicari melalui eksplorasi sistematis terhadap ruang kemungkinan yang ada.

1.3. Algoritma Penyelesaian Permainan *Queens LinkedIn* dengan Pendekatan Brute Force

Penyelesaian permainan *Queens LinkedIn* pada tugas ini dilakukan dengan menggunakan pendekatan *brute force* murni yang bersifat *exhaustive search*, yaitu dengan mencoba seluruh kemungkinan konfigurasi penempatan ratu pada papan berukuran $N \times N$ secara sistematis. Ide utama dari metode pendekatan *brute force* ini adalah menghasilkan seluruh kemungkinan terhadap seluruh aturan atau batasan yang berlaku.

Secara konseptual, ruang solusi dari permainan ini dapat direpresentasikan sebagai seluruh kemungkinan penempatan ratu pada setiap baris, kolom, dan daerah warna pada papan. Untuk papan berukuran $N \times N$, setiap baris memiliki N kemungkinan posisi kolom, sehingga secara teoritis terdapat N^N kombinasi konfigurasi yang mungkin.

Pada pendekatan *brute force* murni, setiap kombinasi tersebut dihasilkan satu per satu dan diuji apakah memenuhi seluruh batasan, yaitu tidak ada dua ratu pada kolom dan baris yang sama, tidak ada dua ratu dalam wilayah warna yang sama, serta tidak ada dua ratu yang saling berdekatan sesuai aturan permainan. Pemeriksaan dilakukan secara menyeluruh terhadap setiap konfigurasi tanpa menghentikan proses secara premature hanya karena Sebagian kondisi gagal, sehingga karakteristik *exhaustive search* tetap terjaga.

Langkah-langkah umum algoritma dapat dijelaskan sebagai berikut. Pertama, sistem menghasilkan sebuah konfigurasi kandidat yang merepresentasikan posisi ratu pada setiap baris. Konfigurasi ini dapat direpresentasikan dalam bentuk array satu dimensi, di mana indeks array menyatakan baris dan nilai pada indeks tersebut menyatakan kolom tempat ratu ditempatkan. Kedua, dilakukan pemeriksaan batasan kolom untuk memastikan tidak ada dua nilai kolom yang sama. Ketiga, dilakukan pemeriksaan batasan wilayah warna dengan mengecek bahwa setiap ratu berada pada wilayah warna yang berbeda. Keempat, dilakukan pemeriksaan aturan kedekatan untuk memastikan tidak ada dua ratu yang berada pada posisi bertetangga secara langsung. Apabila seluruh kondisi terpenuhi, maka konfigurasi tersebut dinyatakan sebagai solusi yang valid dan proses pencarian dapat dihentikan. Jika tidak, algoritma akan melanjutkan ke konfigurasi berikutnya hingga seluruh kemungkinan telah diperiksa atau solusi telah ditemukan.

Karena pendekatan ini mencoba seluruh kemungkinan konfigurasi, kompleksitas waktunya bersifat eksponensial terhadap ukuran papan. Untuk nilai N yang kecil, pendekatan ini masih dapat dijalankan dalam waktu yang wajar (cepat). Namun, ketika N bertambah besar, jumlah kombinasi meningkat sangat cepat sehingga waktu komputasi juga meningkat secara signifikan. Meskipun demikian, pendekatan *brute force* memberikan jaminan bahwa apabila solusi memang ada dalam ruang pencarian, maka solusi tersebut pasti akan ditemukan, karena tidak ada kemungkinan yang diabaikan selama proses pencarian berlangsung.

Dalam implementasi program tugas ini, pendekatan brute force dijadikan sebagai metode solusi utama penyelesaian sesuai dengan spesifikasi yang mewajibkan penggunaan pencarian menyeluruh. Fitur tambahan seperti optimasi atau teknik pemangkasan ruang solusi disediakan sebagai opsi terpisah, tetapi algoritma utama tetap mengikuti prinsip *brute force* murni. Dengan demikian, solusi yang diperoleh sepenuhnya berasal dari eksplorasi sistematis terhadap seluruh kemungkinan konfigurasi yang valid berdasarkan aturan permainan *Queens LinkedIn*.

BAB II

IMPLEMENTASI ALGORITMA DALAM BAHASA PYTHON

2.1. File papan.py

File ini berisi fungsi untuk membaca dan memproses input berbentuk file teks, membaca konfigurasi papan, melakukan validasi ukuran, serta mengonversi representasi huruf menjadi representasi numerik yang digunakan oleh algoritma pencarian solusi.

Nama Fungsi	Deskripsi
__init__(namaFile)	Konstruktor kelas Papan yang menerima nama file sebagai parameter. Konstruktor ini menginisialisasi atribut kelas seperti daftar baris asli, matriks warna, pemetaan warna, dan ukuran papan. Setelah itu, secara otomatis memanggil fungsi bacaFile(), validasiPapan(), konversiKeAngka(), validasiJumlahRegion(), dan validasiKonektivitasRegion() untuk memproses input.
bacaFile()	Membaca isi file input baris demi baris, menghapus spasi atau baris kosong, lalu menyimpannya ke dalam atribut daftarBarisAsli. Fungsi ini memastikan hanya baris yang valid yang diproses lebih lanjut.
validasiPapan()	Memastikan bahwa file input tidak kosong dan bahwa papan berbentuk persegi ($N \times N$). Fungsi ini juga membatasi ukuran maksimal papan hingga 26x26 sesuai jumlah huruf alfabet (A-Z). Selain itu, fungsi ini akan menghasilkan error jika jumlah karakter dalam setiap baris tidak sama dengan jumlah baris keseluruhan.
konversiKeAngka()	Mengubah representasi huruf pada papan menjadi angka. Setiap huruf unik akan diberikan ID numerik yang berbeda. Hasil konversi disimpan dalam atribut matriksWarna yang digunakan oleh algoritma solverQueens.
validasiJumlahRegion()	Memastikan bahwa jumlah region atau wilayah unik (jumlah huruf berbeda) sama dengan ukuran papan (N). Jika jumlah region tidak sama dengan N , maka input dianggap tidak valid dan akan menghasilkan error.
validasiKonektivitasRegion()	Memastikan bahwa setiap region atau wilayah membentuk satu kesatuan area yang terhubung. Proses ini dilakukan menggunakan pendekatan pencarian berbasis antrian. Jika ditemukan region atau wilayah yang terpecah menjadi beberapa bagian yang tidak terhubung, maka input dinyatakan tidak valid.

ambilMatriks()	Mengembalikan matriks numerik yang telah dikonversi dari huruf.
ambilUkuran()	Mengembalikan ukuran papan (N), yaitu jumlah baris atau kolom papan.
jumlahRegion()	Mengembalikan jumlah total wilayah unik yang terdapat pada papan berdasarkan pemetaan huruf ke angka.

2.2. File solverQueens.py

File ini berisi fungsi untuk pencarian solusi permainan *Queens LinkedIn*. Kelas ini mendukung dua pendekatan. Kelas ini mendukung dua pendekatan pencarian, yaitu *brute force (exhaustive search)* sebagai metode solusi utama, dan backtracking sebagai solusi tambahan, serta menyediakan opsi optimasi dan fitur live update untuk keperluan visualisasi.

Nama Fungsi	Deskripsi
<code>__init__(self, matriksWarna, hurufAsli, ukuran, live_k=0, aktifkanOptimasi=False, metodeSolusi="backtracking", callback=None)</code>	Konstruktor kelas SolverQueens yang berfungsi untuk menginisialisasi seluruh atribut seperti matriks warna (papanWarna), ukuran papan, posisi ratu, kolom dan wilayah yang sudah digunakan, jumlah iterasi, mode solusi, serta callback untuk live update GUI.
<code>jumlahRegion()</code>	Menghitung jumlah wilayah unik pada papan berdasarkan matriks warna. Digunakan untuk menginisialisasi array regionDipakai.
<code>mulai()</code>	Fungsi utama untuk menjalankan proses pencarian solusi. Fungsi ini mencatat waktu mulai dan waktu selesai, memilih metode pencarian sesuai parameter, mengembalikan hasil pencarian, jumlah iterasi, dan durasi eksekusi.
<code>membuatSemuaKombinasi()</code>	Generator yang menghasilkan seluruh kemungkinan kombinasi peletakan ratu sebanyak N^N secara manual tanpa menggunakan library eksternal. Setiap Kombinasi direpresentasikan sebagai tuple indeks kolom untuk setiap baris. Digunakan oleh metode <i>brute force</i> .
<code>bruteForce()</code>	Mengimplementasikan pencarian brute force dengan menghasilkan seluruh kemungkinan kombinasi peletakan ratu. Pada setiap kombinasi dilakukan pengecekan validitas sesuai constraint. Jika opsi optimasi aktif, maka menggunakan pengecekan dengan <i>early rejection</i> . Mendukung fitur <i>live update</i> dan <i>logging</i> iterasi melalui <i>callback</i> .
<code>cekTanpaOptimasi(kombinasi)</code>	Mengecek validasi kombinasi dengan memeriksa seluruh batasan tanpa <i>early return</i> .

	Semua aturan diperiksa sebelum hasil ditentukan.
Backtracking(baris)	Mengimplementasikan algoritma backtracking secara rekursif per baris. Setiap ratu ditempatkan satu per satu dengan pengecekan batasan.
cekPenempatanValid(barisTerakhir)	Memeriksa apakah konfigurasi sementara hingga baris tertentu masih valid. Digunakan pada backtracking tanpa optimasi.
AdaKonflikBertetangga(baris, kolom)	Mengecek apakah ratu yang ditempatkan langgar atau tidak kedekatan terhadap ratu lain.

2.3. File imageGenerator.py

File ini berisi fungsi yang digunakan untuk menghasilkan visualisasi solusi dalam bentuk gambar. Setelah algoritma menemukan posisi ratu pada papan, file ini bertugas untuk menambahkan ikon mahkota pada setiap sel yang berisi ratu, ekmudia menyimpan hasilnya sebagai file gambar baru. Namun, file ini awalnya digunakan pada versi CLI, tetapi setelah berpindah ke GUI, file ini tidak digunakan lagi, tetapi tetap saya cantumkan sebagai bukti pembuatan saja.

Nama Fungsi	Deskripsi
tambahkanMahkota(croppedImage, posisiQueen, N, pathOutput)	Menambahkan gambar mahkota pada setiap posisi ratu yang ditemukan dan menyimpan hasilnya sebagai file gambar baru.

2.4. File imageParser.py

File ini berisi fungsi dan kelas yang digunakan untuk memproses input berupa gambar papan permainan. Proses yang dilakukan meliputi konversi citra menjadi matriks wilayah, pengolahan warna, serta pengelolaan *cropping* dan konversi koordinat pada antarmuka GUI.

Nama Fungsi	Deskripsi
gambarKeMatriks(croppedImage, N)	Mengubah gambar hasil <i>crop</i> menjadi matriks wilayah berukuran N x N, membagi gambar menjadi sel-sel grid, menghitung representative setiap sel menggunakan median warna, serta mengelompokkan sel yang memiliki warna yang sama.
<code>__init__(self, lebarKanvas=450, tinggiKanvas=450)</code>	Konstruktor kelas ImageCropper yang menginisialisasi atribut seperti ukuran kanvas, referensi gambar asli dalam format OpenCV dan PIL, faktor skala tampilan, offset posisi gambar, serta dimensi baru gambar setelah diskalakan.

loadImage(image_path)	Memuat gambar dari file, mengonversi warna dari format BGR ke RGB, menghitung skala agar gambar sesuai dengan ukuran kanvas, serta menemukan posisi offset agar gambar tampil ditengah.
canvasToImageCoords(self, kanvas_x1, kanvas_y1, kanvas_x2, kanvas_y2)	Mengonversi koordinat dari sistem kavas GUI ke koordinat asli gambar, Proses ini memperhitungkan skala dan offset tampilan serta memastikan koordinat tetap berada dalam batas gambar.
cropImage(x1, y1, x2, y2)	Melakukan <i>cropping</i> pada gambar asli berdasarkan koordinat hasil konversi, serta memastikan area <i>crop</i> valid dan tidak kosong sebelum dikembalikan sebagai array baru.
processImage(cropped_image, N)	Memproses gambar hasil <i>crop</i> menjadi struktur data yang siap digunakan oleh algoritma pencarian solusi.
ekstrakWilayahWarna(self, croppedImage, N, matriks)	Menentukan warna asli di setiap wilayah berdasarkan median warna tiap sel. Warna kemudian dikonversi ke format HEX untuk keperluan visualisasi pada GUI.

2.5. File main_gui.py

File ini berisi fungsi dan kelas yang digunakan untuk memproses input berupa gambar papan permainan. Proses yang dilakukan meliputi konversi citra menjadi matriks wilayah, pengolahan warna, serta pengelolaan *cropping* dan konversi koordinat pada antarmuka GUI.

Nama Fungsi	Deskripsi
__init__(root)	Konstruktor utama GUI yang menginisialisasi seluruh atribut aplikasi, termasuk ukuran papan, objek solver, mode input (txt atau gambar), komponen visualisasi, cache gambar mahkota, serta membangun tampilan awal dengan memanggil createScrollableFrame() dan buildUI().
createScrollableFrame()	Membuat struktur utama GUI berbasis kanvas yang dapat di- <i>scroll</i> secara <i>vertical</i> . Seluruh komponen antarmuka ditempatkan dalam frame yang berada di dalam kanvas agar dapat menyesuaikan ukuran layar.
on_mousewheel(event)	Mengatur respons <i>scroll</i> menggunakan <i>mouse wheel</i> agar pengguna dapat meng gulir tampilan secara <i>vertical</i> .
buildUI()	Membangun seluruh komponen antarmuka, termasuk tombol load txt, load image, start, reset, pengaturan mode solusi, area input papan, area output solusi, log iterasi, serta

	<p>prompt penyimpanan solusi. Fungsi ini juga melakukan <i>binding event mouse</i> untuk fitur <i>crop</i> gambar.</p>
loadTxt()	Membaca file teks menggunakan kelas Papan, melakukan validasi otomatis, mengambil matriks numerik dan representasi huruf asli, lalu menampilkan papan input pada kanvas kiri. Mode input diatur menjadi txt.
loadImage()	Memuat file gambar menggunakan ImageCropper, menampilkan gambar pada canvas kiri, serta mengaktifkan mode crop agar pengguna dapat memilih area papan secara manual sebelum diproses menjadi matriks.
onCropStart(event)	Menyimpan titik awal saat pengguna mulai melakukan <i>drag</i> untuk menentukan area <i>crop</i> pada gambar.
onCropDrag(event)	Menampilkan persegi seleksi <i>crop</i> secara dinamis saat pengguna menggeser mouse.
onCropEnd(event)	Menandai akhir proses drag area <i>crop</i> .
confirmCrop()	Mengonversi area <i>crop</i> menjadi koordinat gambar asli, memproses gambar menjadi matriks wilayah menggunakan ImageCropper.processImage(), melakukan validasi jumlah region, lalu menampilkan papan hasil pada canvas kiri.
drawBoard(canvas, showQueens=False, queenPositions=None)	Menggambar papan permainan pada kanvas tertentu. Jika mode txt aktif, papan ditampilkan dengan huruf wilayah. Jika mode gambar aktif, warna wilayah ditampilkan berdasarkan hasil ekstraksi. Jika showQueens = True, posisi ratu divisualisasikan menggunakan ikon mahkota.
updateLiveVisual(positions, iterasi)	<i>Callback</i> yang dipanggil oleh <i>solver</i> untuk memperbarui tampilan secara <i>real-time</i> berdasarkan interval iterasi tertentu.
updateUI(positions, iterasi)	Memperbarui papan output serta label progres iterasi selama proses pencarian berlangsung.
tambahLogIterasi(positions, iterasi)	Menambahkan riwayat konfigurasi papan pada panel log setiap interval tertentu, termasuk pembatas antar iterasi.
bersihkanLog()	Menghapus seluruh isi log iterasi pada panel tampilan.
logIterasi(positions, iterasi)	Mengirim permintaan pembaruan log ke thread utama GUI menggunakan mekanisme <i>after()</i> agar aman terhadap <i>multithreading</i> .
logIterationUI(positions, iterasi)	Menjalankan proses pencatatan log iterasi pada thread GUI.
startSolver()	Memvalidasi bahwa papan telah dimuat, menginisialisasi objek <i>SolverQueens</i> dengan parameter yang sesuai (mode solusi, optimasi,

	live update), lalu menjalankan solver dalam thread terpisah agar GUI tidak freeze.
jalankanSOLver()	Menjalankan proses pencarian solusi, menghitung durasi eksekusi, memperbarui tampilan hasil, serta menampilkan kalimat penyimpanan jika solusi ditemukan.
tampilkanPromptSimpan(waktu_ms, iterasi)	Menampilkan panel konfirmasi penyimpanan solusi setelah solusi output ditemukan.
onSaveYes(waktu_ms, iterasi)	Menjalankan proses penyimpanan solusi ketika pengguna memilih untuk menyimpan.
onSaveNo()	Jika pengguna menolak untuk menyimpan output, maka menutup panel penyimpanan tanpa melakukan tindakan apa pun.
simpanSolusi(waktu_ms, iterasi)	Menentukan metode penyimpanan berdasarkan mode input (txt atau gambar).
simpanSolusiTxt(waktu_ms, iterasi)	Menyimpan solusi dalam format file teks yang berisi informasi ukuran papan, metode solusi, waktu eksekusi, jumlah iterasi, serta posisi queen pada setiap baris.
simpanSolusiGambar(waktu_ms, iterasi)	Menyimpan tampilan papan solusi dalam format gambar PNG dengan melakukan tangkapan layar area kanvas output.
resetBoard()	Menghapus papan input dan output, membersihkan log iterasi, serta mengembalikan status GUI ke kondisi awal tanpa menghapus konfigurasi program.

BAB III

SOURCE CODE PROGRAM IMPLEMENTASI QUEENS LINKEDIN

3.1. Repository Program

Berikut adalah pranala ke *repository* program:

https://github.com/Dikanugraha-hub/Tucil1_13524132

3.2. Source Code

3.2.1. Papan.py

```
class Papan:
    def __init__(self, namaFile):
        self.namaFile = namaFile
        self.daftarBarisAsli = []
        self.matriksWarna = []
        self.pemetaanWarna = {}
        self.ukuran = 0

        self.bacaFile()
        self.validasiPapan()
        self.konversiKeAngka()
        self.validasiJumlahRegion()
        self.validasiKonektivitasRegion()

    def bacaFile(self):
        with open(self.namaFile, 'r') as file:
            baris = [baris.strip() for baris in file if baris.strip()]
        self.daftarBarisAsli = baris

    def validasiPapan(self):
        if not self.daftarBarisAsli:
            raise ValueError("File kosong (Tidak Valid)")

        self.ukuran = len(self.daftarBarisAsli)

        if self.ukuran > 26:
            raise ValueError("Ukuran papan maksimal adalah 26x26 (sesuai dengan jumlah huruf A-Z)")

        for baris in self.daftarBarisAsli:
            if len(baris) != self.ukuran:
                raise ValueError("Papan harus berbentuk persegi (N x N)")

            for huruf in baris:
                if not huruf.isalpha():
                    raise ValueError("Wilayah harus alfabet (A-Z). Input tidak valid!")

    def konversiKeAngka(self):
        idWarna = 0
        self.matriksWarna = []

        for baris in self.daftarBarisAsli:
            barisBaru = []
            for huruf in baris:
                if huruf not in self.pemetaanWarna:
                    self.pemetaanWarna[huruf] = idWarna
                    idWarna += 1
```

```

        barisBaru.append(self.pemetaanWarna[huruf])
        self.matriksWarna.append(barisBaru)

    def validasiJumlahRegion(self):
        jumlah_region = len(self.pemetaanWarna)
        if jumlah_region != self.ukuran:
            raise ValueError(f"Jumlah region ({jumlah_region}) harus sama dengan ukuran papan ({self.ukuran}x{self.ukuran}). Input tidak valid!")

    def validasiKonektivitasRegion(self):
        region_cells = {}
        for i in range(self.ukuran):
            for j in range(self.ukuran):
                region_id = self.matriksWarna[i][j]
                if region_id not in region_cells:
                    region_cells[region_id] = []
                region_cells[region_id].append((i, j))

        for region_id, cells in region_cells.items():
            if len(cells) == 0:
                continue

            visited = set()
            queue = [cells[0]]
            visited.add(cells[0])

            while queue:
                row, col = queue.pop(0)
                for dr, dc in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
                    new_row, new_col = row + dr, col + dc
                    if 0 <= new_row < self.ukuran and 0 <= new_col < self.ukuran:
                        if self.matriksWarna[new_row][new_col] == region_id:
                            if (new_row, new_col) not in visited:
                                visited.add((new_row, new_col))
                                queue.append((new_row, new_col))
            if len(visited) != len(cells):
                raise ValueError(
                    "Terdapat region yang tidak terhubung (terpecah menjadi beberapa bagian). "
                    "Setiap region harus membentuk satu kesatuan yang terhubung. Input tidak valid!"
                )

    def ambilMatriks(self):
        return self.matriksWarna

    def ambilUkuran(self):
        return self.ukuran

    def jumlahRegion(self):
        return len(self.pemetaanWarna)

```

3.2.2. solverQueens.py

```

import time

class SolverQueens:

    def __init__(self, matriksWarna, hurufAsli, ukuran, live_k=0,
                 aktifkanOptimasi=False, metodeSolusi="backtracking", callback=None):
        self.papanWarna = matriksWarna

```

```

        self.hurufAsli = hurufAsli
        self.ukuran = ukuran
        self.posisiQueen = [-1] * ukuran
        self.kolomDipakai = [False] * ukuran
        jumlahRegion = self.jumlahRegion()
        self.regionDipakai = [False] * jumlahRegion
        self.jumlahIterasi = 0
        self.live_k = live_k
        self.aktifkanOptimasi = aktifkanOptimasi
        self.metodeSolusi = metodeSolusi.lower()
        self.solusiDitemukan = False
        self.callback = callback
        self.log_callback = None

    def jumlahRegion(self):
        semuaRegion = set()
        for baris in self.papanWarna:
            for nilai in baris:
                semuaRegion.add(nilai)
        return len(semuaRegion)

    def mulai(self):
        waktuMulai = time.perf_counter()

        if self.metodeSolusi == "brute_force":
            hasil = self.bruteForce()
        else:
            hasil = self.backtracking(0)

        waktuSelesai = time.perf_counter()
        durasi = (waktuSelesai - waktuMulai) * 1_000_000

        return hasil, self.jumlahIterasi, durasi

    def membuatSemuaKombinasi(self):
        kombinasi = [0] * self.ukuran

        while True:
            yield tuple(kombinasi)
            memuat = 1
            for i in range(self.ukuran - 1, -1, -1):
                if memuat:
                    kombinasi[i] += 1

```

```

        if kombinasi[i] >= self.ukuran:
            kombinasi[i] = 0
        else:
            memuat = 0
            break

        if memuat:
            break

    def bruteForce(self):
        for kombinasi in self.membuatSemuaKombinasi():
            self.jumlahIterasi += 1
            if self.log_callback and self.live_k > 0 and self.jumlahIterasi % self.live_k == 0:
                self.posisiQueen = list(kombinasi)
                self.log_callback(list(self.posisiQueen), self.jumlahIterasi)

            if self.live_k > 0 and self.jumlahIterasi % self.live_k == 0:
                self.posisiQueen = list(kombinasi)
                if self.callback:
                    self.callback(list(self.posisiQueen), self.jumlahIterasi)

                if self.aktifkanOptimasi:
                    if self.cekDenganOptimasi(kombinasi):
                        self.posisiQueen = list(kombinasi)
                        self.solusiDitemukan = True
                        if self.log_callback:
                            self.log_callback(list(self.posisiQueen), self.jumlahIterasi)
                    return True
                else:
                    if self.cekTanpaOptimasi(kombinasi):
                        self.posisiQueen = list(kombinasi)
                        self.solusiDitemukan = True
                        if self.log_callback:
                            self.log_callback(list(self.posisiQueen), self.jumlahIterasi)
                    return True

            return False

    def cekTanpaOptimasi(self, kombinasi):
        validKolom = True
        validRegion = True
        validTetangga = True

```

```

        if len(set(kombinasi)) != self.ukuran:
            validKolom = False

            regionDipakai = set()
            for baris, kolom in enumerate(kombinasi):
                region = self.papanWarna[baris][kolom]
                if region in regionDipakai:
                    validRegion = False
                regionDipakai.add(region)

            for baris in range(self.ukuran):
                kolom = kombinasi[baris]
                for barisLain in range(baris + 1, self.ukuran):
                    kolomLain = kombinasi[barisLain]
                    if abs(baris - barisLain) <= 1 and abs(kolom - kolomLain)
<= 1:
                    validTetangga = False

            return validKolom and validRegion and validTetangga

    def cekDenganOptimasi(self, kombinasi):
        if len(set(kombinasi)) != self.ukuran:
            return False

        regionDipakai = set()
        for baris, kolom in enumerate(kombinasi):
            region = self.papanWarna[baris][kolom]
            if region in regionDipakai:
                return False
            regionDipakai.add(region)

        for baris in range(self.ukuran):
            kolom = kombinasi[baris]
            for barisLain in range(baris + 1, self.ukuran):
                kolomLain = kombinasi[barisLain]
                if abs(baris - barisLain) <= 1 and abs(kolom - kolomLain)
<= 1:
                    return False

        return True

    def backtracking(self, baris):

```

```

        if baris == self.ukuran:
            self.solusiDitemukan = True
            if self.log_callback:
                self.log_callback(list(self.posisiQueen),
self.jumlahIterasi)
            return True

        for kolom in range(self.ukuran):
            self.jumlahIterasi += 1
            if self.log_callback and self.live_k > 0 and
self.jumlahIterasi % self.live_k == 0:
                self.log_callback(list(self.posisiQueen),
self.jumlahIterasi)
            if self.live_k > 0 and self.jumlahIterasi % self.live_k ==
0:
                if self.callback:
                    self.callback(list(self.posisiQueen),
self.jumlahIterasi)
            if self.aktifkanOptimasi:
                if self.kolomDipakai[kolom]:
                    continue
                regionSaatIni = self.papanWarna[baris][kolom]
                if self.regionDipakai[regionSaatIni]:
                    continue
                if self.adakanflikBertetangga(baris, kolom):
                    continue

                regionSaatIni = self.papanWarna[baris][kolom]
                self.posisiQueen[baris] = kolom
                self.kolomDipakai[kolom] = True
                self.regionDipakai[regionSaatIni] = True

            if not self.aktifkanOptimasi:
                if not self.cekPenempatanValid(baris):
                    self.posisiQueen[baris] = -1
                    self.kolomDipakai[kolom] = False
                    self.regionDipakai[regionSaatIni] = False
                    continue
                if self.backtracking(baris + 1):
                    return True

            self.posisiQueen[baris] = -1
            self.kolomDipakai[kolom] = False
            self.regionDipakai[regionSaatIni] = False

```

```

        return False

def cekPenempatanValid(self, barisTerakhir):
    kolomSet = set()
    regionSet = set()

    for baris in range(barisTerakhir + 1):
        kolom = self.posisiQueen[baris]
        if kolom in kolomSet:
            return False
        kolomSet.add(kolom)

        region = self.papanWarna[baris][kolom]
        if region in regionSet:
            return False
        regionSet.add(region)

        for barisLain in range(baris):
            kolomLain = self.posisiQueen[barisLain]
            if abs(baris - barisLain) <= 1 and abs(kolom - kolomLain)
<= 1:
                return False

    return True

def adaKonflikBertetangga(self, baris, kolom):
    for barisSebelum in range(baris):
        kolomSebelum = self.posisiQueen[barisSebelum]
        if kolomSebelum != -1:
            if abs(baris - barisSebelum) <= 1 and abs(kolom - kolomSebelum) <= 1:
                return True
    return False

```

3.2.3. imageGenerator.py

```

from PIL import Image

def tambahkanMahkota(croppedImage, posisiQueen, N, pathOutput):
    image = Image.fromarray(croppedImage[:, :, ::-1])

    lebar, tinggi = image.size

```

```

sel_w = lebar // N
sel_h = tinggi // N

mahkotaGambar = Image.open("assets/crown.png").convert("RGBA")

skala = 0.6
w_baru = int(sel_w * skala)
h_baru = int(sel_h * skala)

mahkota = mahkotaGambar.resize((w_baru, h_baru), Image.LANCZOS)

for baris in range(N):
    kolom = posisiQueen[baris]

    tengah_x = kolom * sel_w + sel_w // 2
    tengah_y = baris * sel_h + sel_h // 2

    x = tengah_x - w_baru // 2
    y = tengah_y - h_baru // 2

    image.paste(mahkota, (x, y), mahkota)

image.save(pathOutput)

```

3.2.4. imageParser.py

```

import cv2
import numpy as np
from PIL import Image

def gambarKeMatriks(croppedImage, N):
    tinggi, lebar, _ = croppedImage.shape
    sel_h = tinggi // N
    sel_w = lebar // N

    gambarFix = croppedImage // 32 * 32

    warnaSel = []
    for i in range(N):
        baris = []
        for j in range(N):
            y1 = i * sel_h
            y2 = y1 + sel_h

```

```

        x1 = j * sel_w
        x2 = x1 + sel_w

        area = gambarFix[y1:y2, x1:x2]

        warnaTengah = np.median(area.reshape(-1, 3), axis=0)
        baris.append(warnaTengah)
        warnaSel.append(baris)

    warnaSel = np.array(warnaSel)

    matriks = [[-1 for _ in range(N)] for _ in range(N)]
    idWilayah = 0
    threshold = 40

    def cariAreaSama(i, j, idWilayah, warnaAcuan):
        stack = [(i, j)]

        while stack:
            ci, cj = stack.pop()
            if ci < 0 or ci >= N or cj < 0 or cj >= N:
                continue
            if matriks[ci][cj] != -1:
                continue

            perbedaanWarna = np.linalg.norm(warnaSel[ci][cj] -
warnaAcuan)
            if perbedaanWarna > threshold:
                continue

            matriks[ci][cj] = idWilayah

            stack.append((ci-1, cj))
            stack.append((ci+1, cj))
            stack.append((ci, cj-1))
            stack.append((ci, cj+1))

        for i in range(N):
            for j in range(N):
                if matriks[i][j] == -1:
                    cariAreaSama(i, j, idWilayah, warnaSel[i][j])
                    idWilayah += 1

```

```

    return matriks

class ImageCropper:
    def __init__(self, lebarKanvas=450, tinggiKanvas=450):
        self.lebarKanvas = lebarKanvas
        self.tinggiKanvas = tinggiKanvas
        self.originalImageCv = None
        self.originalImagePil = None
        self.skala = 1.0
        self.offset_x = 0
        self.offset_y = 0
        self.lebarBaru = 0
        self.tinggiBaru = 0

    def loadImage(self, image_path):
        self.originalImageCv = cv2.imread(image_path)
        if self.originalImageCv is None:
            raise ValueError("Gambar tidak dapat dibaca")

        image_RGB      = cv2.cvtColor(self.originalImageCv,
cv2.COLOR_BGR2RGB)
        self.originalImagePil = Image.fromarray(image_RGB)

        lebarGambar, tinggiGambar = self.originalImagePil.size
        self.skala = min(self.lebarKanvas / lebarGambar,
self.tinggiKanvas / tinggiGambar)
        self.lebarBaru = int(lebarGambar * self.skala)
        self.tinggiBaru = int(tinggiGambar * self.skala)

        self.offset_x = (self.lebarKanvas - self.lebarBaru) // 2
        self.offset_y = (self.tinggiKanvas - self.tinggiBaru) // 2

        perubahanUkuran = self.originalImagePil.resize((self.lebarBaru,
self.tinggiBaru), Image.Resampling.LANCZOS)

        display_info = {
            'skala': self.skala,
            'offset_x': self.offset_x,
            'offset_y': self.offset_y,
            'display_width': self.lebarBaru,
            'display_height': self.tinggiBaru
        }

    return perubahanUkuran, self.originalImageCv, display_info

```

```

    def canvasToImageCoords(self, kanvas_x1, kanvas_y1, kanvas_x2,
    kanvas_y2):
        if self.originalImageCv is None:
            raise ValueError("Tidak ada gambar yang dimuat")
        if kanvas_x1 > kanvas_x2:
            kanvas_x1, kanvas_x2 = kanvas_x2, kanvas_x1
        if kanvas_y1 > kanvas_y2:
            kanvas_y1, kanvas_y2 = kanvas_y2, kanvas_y1

        original_x1 = int((kanvas_x1 - self.offset_x) / self.skala)
        original_y1 = int((kanvas_y1 - self.offset_y) / self.skala)
        original_x2 = int((kanvas_x2 - self.offset_x) / self.skala)
        original_y2 = int((kanvas_y2 - self.offset_y) / self.skala)

        tinggiGambar, lebarGambar = self.originalImageCv.shape[:2]
        original_x1 = max(0, min(original_x1, lebarGambar))
        original_y1 = max(0, min(original_y1, tinggiGambar))
        original_x2 = max(0, min(original_x2, lebarGambar))
        original_y2 = max(0, min(original_y2, tinggiGambar))

        return original_x1, original_y1, original_x2, original_y2

    def cropImage(self, x1, y1, x2, y2):
        if self.originalImageCv is None:
            raise ValueError("Tidak ada gambar yang dimuat")

        cropped = self.originalImageCv[y1:y2, x1:x2]

        if cropped.size == 0:
            raise ValueError("Area crop terlalu kecil")

        return cropped

    def processImage(self, cropped_image, N):
        if N > 26:
            raise ValueError("Ukuran N maksimal adalah 26 (sesuai dengan
jumlah huruf A-Z)")

        matriks = gambarKeMatriks(cropped_image, N)

        unique_regions = set()
        for row in matriks:

```

```

        for region_id in row:
            unique_regions.add(region_id)

    jumlah_region = len(unique_regions)
    if jumlah_region != N:
        raise ValueError(f"Jumlah region yang terdeteksi ({jumlah_region}) harus sama dengan N ({N}). Input tidak valid!")

    regionColors = self.ekstrakWilayahWarna(cropped_image, N,
matriks)
    huruf_awal = ord('A')
    hurufAsli = []

    for i in range(N):
        row = []
        for j in range(N):
            region_id = matriks[i][j]
            row.append(chr(huruf_awal + region_id))
        hurufAsli.append(row)

    return matriks, regionColors, hurufAsli

def ekstrakWilayahWarna(self, croppedImage, N, matriks):
    tinggi, lebar = croppedImage.shape[:2]
    sel_h = tinggi // N
    sel_w = lebar // N
    gambarFix = croppedImage // 32 * 32
    warnaSel = []

    for i in range(N):
        baris = []
        for j in range(N):
            y1 = i * sel_h
            y2 = y1 + sel_h
            x1 = j * sel_w
            x2 = x1 + sel_w
            area = gambarFix[y1:y2, x1:x2]
            warnaTengah = np.median(area.reshape(-1, 3), axis=0)
            baris.append(warnaTengah)
        warnaSel.append(baris)
    warnaSel = np.array(warnaSel)
    regionColors = {}

```

```

        for i in range(N):
            for j in range(N):
                region_id = matriks[i][j]
                if region_id not in regionColors:
                    regionColors[region_id] =
                    "#{:02x}{:02x}{:02x}".format(
                        int(warnaSel[i][j][2]),
                        int(warnaSel[i][j][1]),
                        int(warnaSel[i][j][0]))
                )
        return regionColors
    
```

3.2.5. main_gui.py

```

import tkinter as tk
from tkinter import filedialog, messagebox
import threading
import os
from datetime import datetime
from PIL import Image, ImageTk
from papan import Papan
from solverQueens import SolverQueens
from imageParser import ImageCropper

class GUIQueens:
    def __init__(self, root):
        self.root = root
        self.root.title("Visualisasi Penyelesaian N-Queens ---> Dika
Pramudya Nugraha (13524132)")
        self.root.geometry("1800x750")
        self.root.resizable(True, True)
        self.ukuran = 0
        self.papan = None
        self.solver = None
        self.matriks = None
        self.hurufAsli = None
        self.solving = False
        self.modeTxt = False
        self.imageCropper = ImageCropper(lebarKanvas=450,
tinggiKanvas=450)
        self.modeCrop = False
        self.cropStartX = None
        self.cropStartY = None
    
```

```

        self.cropRect = None
        self.gambarTampil = None
        self.regionColors = {}
        self.riwayatIterasi = []
        self.counterLog = 0
        self.crownImage = None
        self.fotoMahkotaInput = {}
        self.fotoMahkotaOutput = {}

    try:
        mahkota_path = os.path.join(os.path.dirname(__file__), "assets", "crown.png")
        self.crownImage = Image.open(mahkota_path)
    except Exception as e:
        pass

    self.colors = [
        "#FFB6C1", "#87CEEB", "#98FB98", "#FFD700", "#DDA0DD",
        "#F0E68C", "#FFE4B5", "#E0BBE4", "#FFDAB9", "#B0E0E6",
        "#FAFAD2", "#D8bfd8", "#F5DEB3", "#C1FFC1", "#FFE4E1"
    ]

    self.createScrollableFrame()
    self.buildUI()

def createScrollableFrame(self):
    self.mainCanvas = tk.Canvas(self.root, highlightthickness=0)
    scrollbar = tk.Scrollbar(self.root, orient="vertical",
                           command=self.mainCanvas.yview)
    self.scrollableFrame = tk.Frame(self.mainCanvas)

    self.scrollableFrame.bind(
        "<Configure>",
        lambda e:
self.mainCanvas.configure(scrollregion=self.mainCanvas.bbox("all"))
    )

    self.mainCanvas.create_window((0, 0),
window=self.scrollableFrame, anchor="nw")
    self.mainCanvas.configure(yscrollcommand=scrollbar.set)
    self.mainCanvas.pack(side="left", fill="both", expand=True)
    scrollbar.pack(side="right", fill="y")
    self.mainCanvas.bind_all("<MouseWheel>", self.on_mousewheel)

def on_mousewheel(self, event):

```

```

        self.mainCanvas.yview_scroll(int(-1*(event.delta/120)),
"units")

    def buildUI(self):
        frameAtas = tk.Frame(self.scrollableFrame)
        frameAtas.pack(pady=10)

        tk.Button(frameAtas, text="Load TXT", command=self.loadTxt,
width=12).pack(side=tk.LEFT, padx=5)
        tk.Button(frameAtas, text="Load Image", command=self.loadImage,
width=12).pack(side=tk.LEFT, padx=5)
        tk.Button(frameAtas, text="Start", command=self.startSolver,
width=12, bg="#90EE90").pack(side=tk.LEFT, padx=5)
        tk.Button(frameAtas, text="Reset", command=self.resetBoard,
width=12, bg="#FFB6C1").pack(side=tk.LEFT, padx=5)

        pengaturanFrame = tk.Frame(self.scrollableFrame)
        pengaturanFrame.pack(pady=5)

        tk.Label(pengaturanFrame,      text="Live      update
(k) :").pack(side=tk.LEFT)
        self.inputLive = tk.Entry(pengaturanFrame, width=8)
        self.inputLive.insert(0, "100")
        self.inputLive.pack(side=tk.LEFT, padx=5)
        self.opsiOptimasi = tk.BooleanVar(value=True)
        tk.Checkbutton(pengaturanFrame,      text="Optimasi",
variable=self.opsiOptimasi).pack(side=tk.LEFT, padx=10)
        tk.Label(pengaturanFrame,  text="Mode:").pack(side=tk.LEFT,
padx=5)
        self.pilihanMode = tk.StringVar(value="backtracking")
        tk.Radiobutton(pengaturanFrame,  text="Backtracking",
variable=self.pilihanMode, value="backtracking").pack(side=tk.LEFT)
        tk.Radiobutton(pengaturanFrame,  text="Brute Force",
variable=self.pilihanMode, value="brute_force").pack(side=tk.LEFT)

        frameUtama = tk.Frame(self.scrollableFrame)
        frameUtama.pack(pady=10)

        frameKiri = tk.Frame(frameUtama)
        frameKiri.pack(side=tk.LEFT, padx=10)

        tk.Label(frameKiri, text="INPUT BOARD", font=("Arial", 12,
"bold"), anchor="center").pack()
        self.canvasLeft = tk.Canvas(frameKiri, width=450, height=450,
bg="white",
highlightbackground="blue", highlightthickness=2)

```

```

        self.canvasLeft.pack(pady=5)
        self.frameCropControls = tk.Frame(frameKiri)
            tk.Label(self.frameCropControls,   text="N      (Grid
Size):").pack(side=tk.LEFT, padx=5)
        self.inputUkuran = tk.Entry(self.frameCropControls, width=8)
        self.inputUkuran.pack(side=tk.LEFT, padx=5)
        tk.Button(self.frameCropControls,   text="Confirm Crop",
command=self.confirmCrop,
            bg="#90EE90").pack(side=tk.LEFT, padx=5)

        frameKanan = tk.Frame(frameUtama)
        frameKanan.pack(side=tk.LEFT, padx=10)

        tk.Label(frameKanan,   text="OUTPUT BOARD (Live Update)",
font=("Arial", 12, "bold"), anchor="center").pack()
        self.canvasRight = tk.Canvas(frameKanan, width=450, height=450,
bg="#f0f0f0",
                           highlightthickness=2,
highlightbackground="green")
        self.canvasRight.pack(pady=5)

        frameLog = tk.Frame(frameUtama)
        frameLog.pack(side=tk.LEFT, padx=10)

        tk.Label(frameLog,   text="ITERATION LOG (History)",
font=("Arial", 12, "bold"), anchor="center").pack()
        logContainer = tk.Frame(frameLog,   width=450, height=450,
bg="white",
                           highlightthickness=2,
highlightbackground="orange")
        logContainer.pack(pady=5)
        logContainer.pack_propagate(False)

        self.logScrollbar = tk.Scrollbar(logContainer)
        self.logScrollbar.pack(side=tk.RIGHT, fill=tk.Y)

        self.textLog = tk.Text(logContainer, wrap=tk.WORD,
                           yscrollcommand=self.logScrollbar.set,
                           font=("Courier New", 9),
                           bg="#f9f9f9", fg="#333333",
                           state='disabled')
        self.textLog.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
        self.logScrollbar.config(command=self.textLog.yview)

        self.labelInfo = tk.Label(self.scrollableFrame, text="Belum ada
file dimuat", font=("Arial", 10), anchor="center")

```

```

        self.labelInfo.pack(pady=5)

        self.labelProgress = tk.Label(self.scrollableFrame, text="", font=("Arial", 9), fg="blue", anchor="center")
        self.labelProgress.pack()

        self.frameSimpan = tk.Frame(self.scrollableFrame, bg="#ffffcc", relief=tk.RAISED, borderwidth=2)

        self.labelSaveQuestion = tk.Label(self.frameSimpan, text=" Solusi ditemukan! Apakah Anda ingin menyimpan solusi?", font=("Arial", 10, "bold"), bg="#ffffcc", fg="#006400", anchor="center")
        self.labelSaveQuestion.pack(pady=8)

        frameTombolSimpan = tk.Frame(self.frameSimpan, bg="#ffffcc")
        frameTombolSimpan.pack(pady=5)

        self.btnSaveYa = tk.Button(frameTombolSimpan, text="Ya, Simpan", width=15, bg="#90EE90", font=("Arial", 10, "bold"), cursor="hand2")
        self.btnSaveYa.pack(side=tk.LEFT, padx=10)

        self.btnSaveTidak = tk.Button(frameTombolSimpan, text="Tidak", width=15, bg="#FFB6C1", font=("Arial", 10, "bold"), cursor="hand2")
        self.btnSaveTidak.pack(side=tk.LEFT, padx=10)

        tk.Label(self.scrollableFrame, text="", height=1).pack()

        self.canvasLeft.bind("<ButtonPress-1>", self.onCropStart)
        self.canvasLeft.bind("<B1-Motion>", self.onCropDrag)
        self.canvasLeft.bind("<ButtonRelease-1>", self.onCropEnd)

    def loadTxt(self):
        path = filedialog.askopenfilename(filetypes=[("Text Files", "*.txt")])
        if not path:
            return

    try:
        self.modeCrop = False
        self.frameCropControls.pack_forget()

        self.papan = Papan(path)

```

```

        self.ukuran = self.papan.ambilUkuran()
        self.matriks = self.papan.ambilMatriks()
        self.hurufAsli = [[char for char in row] for row in
self.papan.daftarBarisAsli]
        self.modeTxt = True

        self.drawBoard(self.canvasLeft, showQueens=False)
        self.canvasRight.delete("all")
        self.labelInfo.config(text=f"✓  TXT  Loaded | Ukuran:
{self.ukuran}x{self.ukuran}")
        self.labelProgress.config(text="")
    except Exception as e:
        messagebox.showerror("Error", str(e))

    def loadImage(self):
        path = filedialog.askopenfilename(filetypes=[("Image Files",
".png *.jpg *.jpeg")])
        if not path:
            return
        try:
            resized, cv_image, display_info =
self.imageCropper.loadImage(path)
            self.gambarTampil = ImageTk.PhotoImage(resized)
            self.canvasLeft.delete("all")
            self.canvasLeft.create_image(225, 225,
image=self.gambarTampil)
            self.modeCrop = True
            self.frameCropControls.pack(pady=5)
            self.labelInfo.config(text="Gambar telah dimuat. Seret
(drag) untuk memilih area pemotongan gambar, masukkan N, lalu klik
Confirm Crop.")
            self.labelProgress.config(text="")
        except Exception as e:
            messagebox.showerror("Error", str(e))

    def onCropStart(self, event):
        if not self.modeCrop:
            return
        self.cropStartX = event.x
        self.cropStartY = event.y
        if self.cropRect:
            self.canvasLeft.delete(self.cropRect)

    def onCropDrag(self, event):
        if not self.modeCrop or self.cropStartX is None:

```

```

        return

    if self.cropRect:
        self.canvasLeft.delete(self.cropRect)
    self.cropRect = self.canvasLeft.create_rectangle(
        self.cropStartX, self.cropStartY, event.x, event.y,
        outline="red", width=3
    )

def onCropEnd(self, event):
    if not self.modeCrop:
        return

def confirmCrop(self):
    if not self.modeCrop or self.imageCropper.originalImageCv is None:
        messagebox.showwarning("Warning", "Load image terlebih dahulu!")
        return

    if self.cropRect is None:
        messagebox.showwarning("Warning", "Pilih area crop terlebih dahulu!")
        return

    try:
        N = int(self.inputUkuran.get())
        if N <= 0:
            raise ValueError("N harus positif")
        if N > 26:
            raise ValueError("N maksimal adalah 26 (sesuai dengan jumlah huruf A-Z)")
    except ValueError as e:
        messagebox.showerror("Error", str(e))
        return

    try:
        coords = self.canvasLeft.coords(self.cropRect)
        canvas_x1, canvas_y1, canvas_x2, canvas_y2 = coords

        orig_x1, orig_y1, orig_x2, orig_y2 = self.imageCropper.canvasToImageCoords(
            canvas_x1, canvas_y1, canvas_x2, canvas_y2
        )

        cropped = self.imageCropper.cropImage(orig_x1, orig_y1,
orig_x2, orig_y2)
    
```

```

        self.matriks, self.regionColors, self.hurufAsli =
self.imageCropper.processImage(cropped, N)

        self.ukuran = N
        self.modeTxt = False
        self.drawBoard(self.canvasLeft, showQueens=False)
        self.canvasRight.delete("all")
        self.modeCrop = False
        self.frameCropControls.pack_forget()

        self.labelInfo.config(text=f"✓ Image Cropped & Processed |"
Ukuran: {N}x{N}")
        self.labelProgress.config(text="")

    except Exception as e:
        messagebox.showerror("Error", f"Gagal memproses crop: {str(e)}")

def drawBoard(self, canvas, showQueens=False, queenPositions=None):
    canvas.delete("all")
    if self.ukuran == 0:
        return

    cell = 450 // self.ukuran

    if canvas == self.canvasLeft:
        crownCache = self.fotoMahkotaInput
    else:
        crownCache = self.fotoMahkotaOutput

    if self.modeTxt:
        for i in range(self.ukuran):
            for j in range(self.ukuran):
                x1 = j * cell
                y1 = i * cell
                x2 = x1 + cell
                y2 = y1 + cell

                canvas.create_rectangle(x1, y1, x2, y2,
outline="black", fill="white", width=1)

                if showQueens and queenPositions and i <
len(queenPositions) and queenPositions[i] == j:
                    display_char = "#"
                    color = "red"
                    font_size = max(14, cell//2)

```

```

        else:
            display_char = self.hurufAsli[i][j] if
self.hurufAsli else ""
            color = "black"
            font_size = max(10, cell//3)

            canvas.create_text(x1 + cell//2, y1 + cell//2,
                               text=display_char,
                               font=("Courier New", font_size,
"bold")),
                               fill=color)

        else:
            for i in range(self.ukuran):
                for j in range(self.ukuran):
                    x1 = j * cell
                    y1 = i * cell
                    x2 = x1 + cell
                    y2 = y1 + cell

                    region_id = self.matriks[i][j]
                    if self.regionColors and region_id in
self.regionColors:
                        color = self.regionColors[region_id]
                    else:
                        color = self.colors[region_id % len(self.colors)]

                    canvas.create_rectangle(x1, y1, x2, y2,
outline="black", fill=color, width=1)

                    if self.hurufAsli:
                        label = self.hurufAsli[i][j]
                        canvas.create_text(x1 + cell//2, y1 + cell//2,
                                           text=label,
                                           font=("Arial", max(8,
cell//4)),
                                           fill="gray")

            if showQueens and queenPositions:
                for i in range(len(queenPositions)):
                    kol = queenPositions[i]
                    if kol != -1:
                        x = kol * cell + cell // 2
                        y = i * cell + cell // 2

```

```

        if self.crownImage:
            crown_size = int(cell * 0.8)

            if crown_size not in crownCache:
                crown_rgba = self.crownImage.convert('RGBA')
                resized = crown_rgba.resize((crown_size,
                                              crown_size), Image.Resampling.LANCZOS)
                crownCache[crown_size] = ImageTk.PhotoImage(resized)

            canvas.create_image(x, y,
                                image=crownCache[crown_size],
                                tags="queen")

        else:
            canvas.create_text(x, y, text="■",
                               font=("Arial", max(12,
                                                 cell//2)),
                               fill="darkred",
                               tags="queen")

    def updateLiveVisual(self, positions, iterasi):
        self.root.after(0, lambda: self.updateUI(positions, iterasi))

    def updateUI(self, positions, iterasi):
        self.drawBoard(self.canvasRight, showQueens=True,
                      queenPositions=positions)
        self.labelProgress.config(text=f"Solving... Iterasi: {iterasi:,}")

        self.tambahLogIterasi(positions, iterasi)

    def tambahLogIterasi(self, positions, iterasi):
        log_entry = f"Iterasi ke-{iterasi:,}\n"

        if self.modeTxt:
            for i in range(self.ukuran):
                row = ""
                for j in range(self.ukuran):
                    if positions[i] == j:
                        row += "# "
                    else:
                        row += self.hurufAsli[i][j] + " "
                log_entry += row.strip() + "\n"
        else:

```

```

        for i in range(self.ukuran):
            row = ""
            for j in range(self.ukuran):
                if positions[i] == j:
                    row += "# "
                else:
                    row += self.hurufAsli[i][j] + " "
            log_entry += row.strip() + "\n"

        log_entry += "-" * 30 + "\n"

        self.textLog.config(state='normal')
        self.textLog.insert(tk.END, log_entry)

        lines = int(self.textLog.index('end-1c').split('.')[0])
        max_lines = 500 * (self.ukuran + 2)
        if lines > max_lines:
            self.textLog.delete('1.0', f'{lines - max_lines}.0')

        self.textLog.config(state='disabled')
        self.textLog.see(tk.END)

    def bersihkanLog(self):
        self.textLog.config(state='normal')
        self.textLog.delete('1.0', tk.END)
        self.textLog.config(state='disabled')
        self.riwayatIterasi = []

    def logIterasi(self, positions, iterasi):
        self.root.after(0, lambda: self.logIterationUI(positions, iterasi))

    def logIterationUI(self, positions, iterasi):
        self.tambahLogIterasi(positions, iterasi)

    def startSolver(self):
        if self.matriks is None:
            messagebox.showwarning("Warning", "Load file terlebih dahulu!")
        return

        if self.solving:
            messagebox.showwarning("Warning", "Solver sedang berjalan!")

```

```

        return

    try:
        live_k = int(self.inputLive.get())
    except:
        live_k = 0

    self.solving = True
    self.labelInfo.config(text="Memulai solver...")
    self.labelProgress.config(text="")
    self.frameSimpan.pack_forget()
    self.bersihkanLog()

    self.solver = SolverQueens(
        matriksWarna=self.matriks,
        hurufAsli=self.hurufAsli,
        ukuran=self.ukuran,
        live_k=live_k,
        aktifkanOptimasi=self.opsiOptimasi.get(),
        metodeSolusi=self.pilihanMode.get(),
        callback=self.updateLiveVisual if live_k > 0 else None
    )

    self.solver.log_callback = self.logIterasi

    thread = threading.Thread(target=self.jalankanSolver,
daemon=True)
    thread.start()

def jalankanSolver(self):
    try:
        hasil, total, durasi = self.solver.mulai()
        waktu_ms = durasi / 1000

        if hasil:
            self.root.after(0, lambda:
self.drawBoard(self.canvasRight, showQueens=True,
queenPositio
ns=self.solver.posisiQueen))

            self.root.after(0, lambda: self.labelInfo.config(
                text=f"✅ Solusi ditemukan! | Waktu Pencarian:
{waktu_ms:.3f} ms | Iterasi: {total:,}"
            ))
    
```

```

                self.root.after(0, lambda:
self.labelProgress.config(text=""))

                self.root.after(100, lambda:
self.tampilkanPromptSimpan(waktu_ms, total))

            else:

                self.root.after(0, lambda:
self.canvasRight.delete("all"))

                self.root.after(0, lambda: self.labelInfo.config(
text=f"🔴 Tidak ada solusi | Waktu Pencarian:
{waktu_ms:.3f} ms | Iterasi: {total:,}")

            )

                self.root.after(0, lambda:
self.labelProgress.config(text=""))

            except Exception as e:

                self.root.after(0, lambda: messagebox.showerror("Error",
f"Solver error: {str(e)}"))

            finally:

                self.solving = False


    def tampilkanPromptSimpan(self, waktu_ms, iterasi):

        self.btnExitYa.config(command=lambda: self.onSaveYes(waktu_ms,
iterasi))

        self.btnExitTidak.config(command=self.onSaveNo)

        self.frameSimpan.pack(pady=10, padx=20, fill=tk.X)

        self.root.after(200, lambda: self.mainCanvas.yview_moveto(1.0))



    def onSaveYes(self, waktu_ms, iterasi):
        self.frameSimpan.pack_forget()
        self.simpanSolusi(waktu_ms, iterasi)


    def onSaveNo(self):
        self.frameSimpan.pack_forget()


    def simpanSolusi(self, waktu_ms, iterasi):
        if self.modeTxt:
            self.simpanSolusiTxt(waktu_ms, iterasi)
        else:
            self.simpanSolusiGambar(waktu_ms, iterasi)


    def simpanSolusiTxt(self, waktu_ms, iterasi):
        default_name =
f"solution_{datetime.now().strftime('%Y%m%d_%H%M%S')}.txt"
        filepath = filedialog.asksaveasfilename(
            defaultextension=".txt",
            filetypes=[("Text Files", "*.txt"), ("All Files", "*.*")],

```

```

        initialfile=default_name
    )

    if not filepath:
        return

    try:
        with open(filepath, 'w', encoding='utf-8') as f:
            f.write("=" * 50 + "\n")
            f.write("N-QUEENS SOLVER - SOLUTION\n")
            f.write("=" * 50 + "\n\n")
            f.write(f"Ukuran Grid: {self.ukuran}x{self.ukuran}\n")
            f.write(f"Mode: {'Backtracking' if self.solver.metodeSolusi == 'backtracking' else 'Brute Force'}\n")
            f.write(f"Optimasi: {'Ya' if self.solver.aktifkanOptimasi else 'Tidak'}\n")
            f.write(f"Waktu Eksekusi: {waktu_ms:.3f} ms\n")
            f.write(f"Jumlah Iterasi: {iterasi:,}\n")
            f.write(f"Tanggal: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n\n")
            f.write("SOLUSI:\n")
            f.write("-" * 50 + "\n\n")

            for i in range(self.ukuran):
                row = ""
                for j in range(self.ukuran):
                    if self.solver.posisiQueen[i] == j:
                        row += "# "
                    else:
                        row += self.hurufAsli[i][j] + " "
                f.write(row.strip() + "\n")

            f.write("\n" + "-" * 50 + "\n")

            f.write("\nPOSISI QUEEN (Baris, Kolom):\n")
            for i in range(self.ukuran):
                f.write(f"Baris {i+1}: Kolom {self.solver.posisiQueen[i]+1}\n")

            messagebox.showinfo("Sukses", f"Solusi berhasil disimpan ke:{filepath}")

    except Exception as e:
        messagebox.showerror("Error", f"Gagal menyimpan file:\n{str(e)}")

```

```

def simpanSolusiGambar(self, waktu_ms, iterasi):
    default_name = f"solution_{datetime.now().strftime('%Y%m%d_%H%M%S')}.png"
    filepath = filedialog.asksaveasfilename(
        defaultextension=".png",
        filetypes=[("PNG Files", "*.png"), ("All Files", "*.*")],
        initialfile=default_name
    )

    if not filepath:
        return

    try:
        from PIL import Image as PILImage, ImageDraw, ImageFont
        img_size = 450
        img = PILImage.new('RGB', (img_size, img_size), 'white')
        draw = ImageDraw.Draw(img)

        cell = img_size // self.ukuran

        if self.modeTxt:
            for i in range(self.ukuran):
                for j in range(self.ukuran):
                    x1 = j * cell
                    y1 = i * cell
                    x2 = x1 + cell
                    y2 = y1 + cell

                    draw.rectangle([x1, y1, x2, y2], outline='black',
fill='white', width=1)

                    if self.solver.posisiQueen[i] == j:
                        display_char = "#"
                        color = "red"
                        font_size = max(14, cell//2)
                    else:
                        display_char = self.hurufAsli[i][j] if
self.hurufAsli else ""
                        color = "black"
                        font_size = max(10, cell//3)

        try:

```

```

        font = ImageFont.truetype("cour.ttf",
font_size)

    except:
        font = ImageFont.load_default()

        bbox = draw.textbbox((0, 0), display_char,
font=font)

        text_width = bbox[2] - bbox[0]
        text_height = bbox[3] - bbox[1]
        text_x = x1 + (cell - text_width) // 2
        text_y = y1 + (cell - text_height) // 2

        draw.text((text_x, text_y), display_char,
fill=color, font=font)
    else:
        for i in range(self.ukuran):
            for j in range(self.ukuran):
                x1 = j * cell
                y1 = i * cell
                x2 = x1 + cell
                y2 = y1 + cell

                region_id = self.matriks[i][j]
                if self.regionColors and region_id in
self.regionColors:
                    color = self.regionColors[region_id]
                else:
                    color = self.colors[region_id %
len(self.colors)]

                draw.rectangle([x1, y1, x2, y2], outline='black',
fill=color, width=1)

                if self.hurufAsli:
                    label = self.hurufAsli[i][j]
                    font_size = max(8, cell//4)
                    try:
                        font = ImageFont.truetype("arial.ttf",
font_size)
                    except:
                        font = ImageFont.load_default()

                    bbox = draw.textbbox((0, 0), label,
font=font)

                    text_width = bbox[2] - bbox[0]

```

```

        text_height = bbox[3] - bbox[1]
        text_x = x1 + (cell - text_width) // 2
        text_y = y1 + (cell - text_height) // 2

                draw.text((text_x, text_y), label,
fill='gray', font=font)

        for i in range(len(self.solver.posisiQueen)):
            kol = self.solver.posisiQueen[i]
            if kol != -1:
                x = kol * cell + cell // 2
                y = i * cell + cell // 2

                if self.crownImage:
                    crown_size = int(cell * 0.8)
                    crown_rgba = self.crownImage.convert('RGBA')
                    crown_resized =
crown_rgba.resize((crown_size, crown_size), Image.Resampling.LANCZOS)

                    crown_x = x - crown_size // 2
                    crown_y = y - crown_size // 2

                    img.paste(crown_resized, (crown_x, crown_y),
crown_resized)
                else:
                    queen_char = "♛"
                    font_size = max(12, cell//2)
                    try:
                        font = ImageFont.truetype("arial.ttf",
font_size)
                    except:
                        font = ImageFont.load_default()

                    bbox = draw.textbbox((0, 0), queen_char,
font=font)
                    text_width = bbox[2] - bbox[0]
                    text_height = bbox[3] - bbox[1]
                    text_x = x - text_width // 2
                    text_y = y - text_height // 2

                    draw.text((text_x, text_y), queen_char,
fill='darkred', font=font)

            img.save(filepath, 'PNG')

```

```
        messagebox.showinfo("Sukses", f"Solusi berhasil disimpan ke:\n{filepath}")

    except Exception as e:
        messagebox.showerror("Error", f"Gagal menyimpan gambar:\n{str(e)}")

    def resetBoard(self):
        if self.matriks is not None:
            self.canvasLeft.delete("all")
            self.canvasRight.delete("all")
            self.frameSimpan.pack_forget()
            self.bersihkanLog()
            self.labelInfo.config(text=f"Board direset | Ukuran: {self.ukuran}x{self.ukuran}")
            self.labelProgress.config(text="")
        else:
            self.canvasLeft.delete("all")
            self.canvasRight.delete("all")
            self.frameSimpan.pack_forget()
            self.bersihkanLog()
            self.labelInfo.config(text="Belum ada file dimuat")
            self.labelProgress.config(text="")

if __name__ == "__main__":
    root = tk.Tk()
    app = GUIQueens(root)
    root.mainloop()
```

BAB IV

MASUKAN DAN LUARAN PROGRAM

4.1. Test Case Txt

4.1.1. Test Case 1 – 9x9

Data Input
AAABBCCCD ABBBBCECD ABBDCECD AAABDCCCD BBBBDDDDD FGGGDDHDD FGIGDDHDD FGIGDDHDD FGGGDDHHH

Output

The screenshot shows a software interface for solving the N-Queens problem. At the top, there is a header with the title "Visualisasi Penyelesaian N-Queens" and the author "Dika Pramudya Nugraha (13524132)". Below the header are buttons for "Load TXT", "Load Image", "Start", and "Reset". There is also a "Live update (k): 10000" slider and mode selection buttons for "Optimasi", "Backtracking", and "Brute Force".

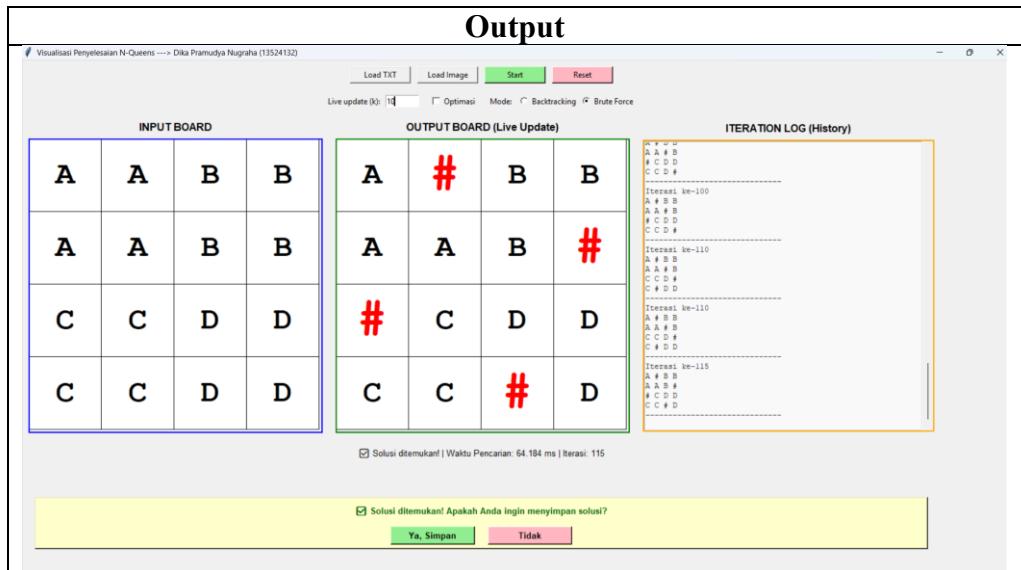
The main area is divided into three sections:

- INPUT BOARD:** An 8x8 grid where each cell contains a letter from A to H. The letters are arranged in a specific pattern: A1=A, A2=B, A3=B, A4=B, A5=C, A6=C, A7=C, A8=D; B1=B, B2=B, B3=B, B4=B, B5=C, B6=C, B7=C, B8=D; C1=B, C2=B, C3=D, C4=C, C5=E, C6=E, C7=C, C8=D; D1=C, D2=C, D3=C, D4=C, D5=D, D6=D, D7=D, D8=D; E1=C, E2=D, E3=D, E4=D, E5=D, E6=D, E7=D, E8=D; F1=D, F2=D, F3=D, F4=D, F5=D, F6=D, F7=D, F8=D; G1=D, G2=D, G3=D, G4=D, G5=D, G6=D, G7=D, G8=D; H1=D, H2=D, H3=D, H4=D, H5=D, H6=D, H7=D, H8=D.
- OUTPUT BOARD (Live Update):** An 8x8 grid showing a solution. The letters are: A1=A, A2=B, A3=B, A4=B, A5=C, A6=C, A7=C, A8=D; B1=B, B2=B, B3=B, B4=B, B5=C, B6=C, B7=C, B8=D; C1=B, C2=B, C3=D, C4=C, C5=E, C6=E, C7=C, C8=D; D1=C, D2=C, D3=C, D4=C, D5=D, D6=D, D7=D, D8=D; E1=C, E2=D, E3=D, E4=D, E5=D, E6=D, E7=D, E8=D; F1=D, F2=D, F3=D, F4=D, F5=D, F6=D, F7=D, F8=D; G1=D, G2=D, G3=D, G4=D, G5=D, G6=D, G7=D, G8=D; H1=D, H2=D, H3=D, H4=D, H5=D, H6=D, H7=D, H8=D.
- ITERATION LOG (History):** A text area showing the history of iterations. It includes two sections: Iterasi ke-323,740,000 and Iterasi ke-323,740,437. Each section lists a series of 8x8 boards, each with a different arrangement of letters (A-H) in each row and column, representing a step in the search space.

At the bottom, there is a message box asking if the user wants to save the found solution, with "Ya, Simpan" and "Tidak" buttons.

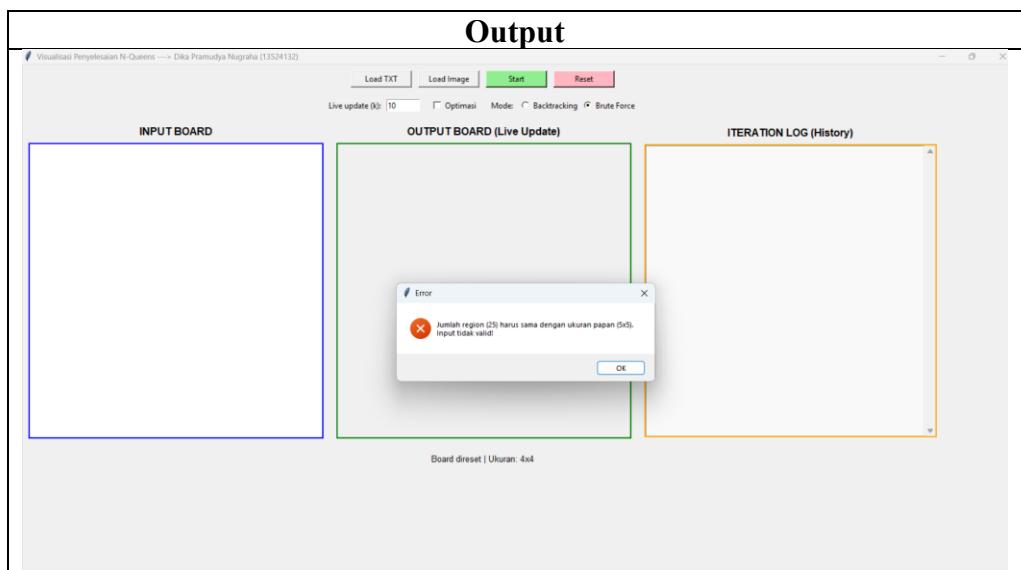
4.1.2. Test Case 2 – 4x4

Data Input
AABB
AABB
CCDD
CCDD



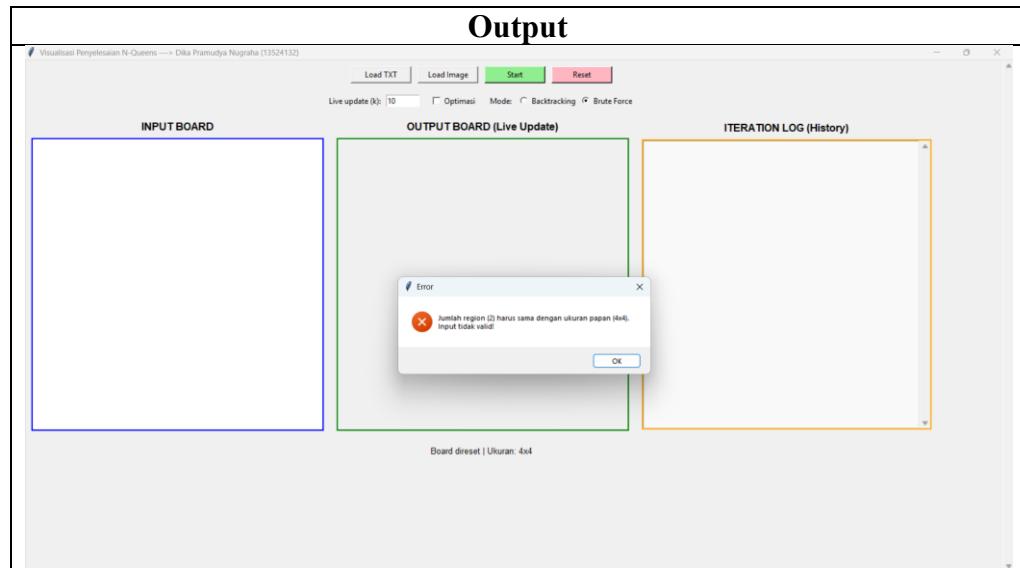
4.1.3. Test Case 3 – 5x5 Unik

Data Input
ABCDE FGHIJ KLMNO PQRST UVWXY



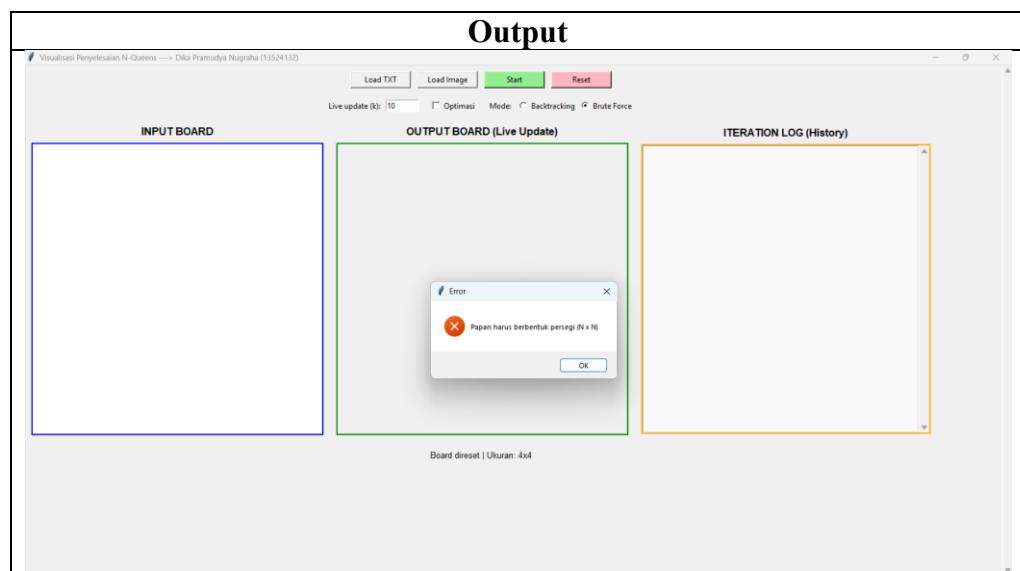
4.1.4. Test Case 4 – 4x4 Jumlah Wilayah Kurang Dari N

Data Input
AAAA AAAA BBBB BBBB



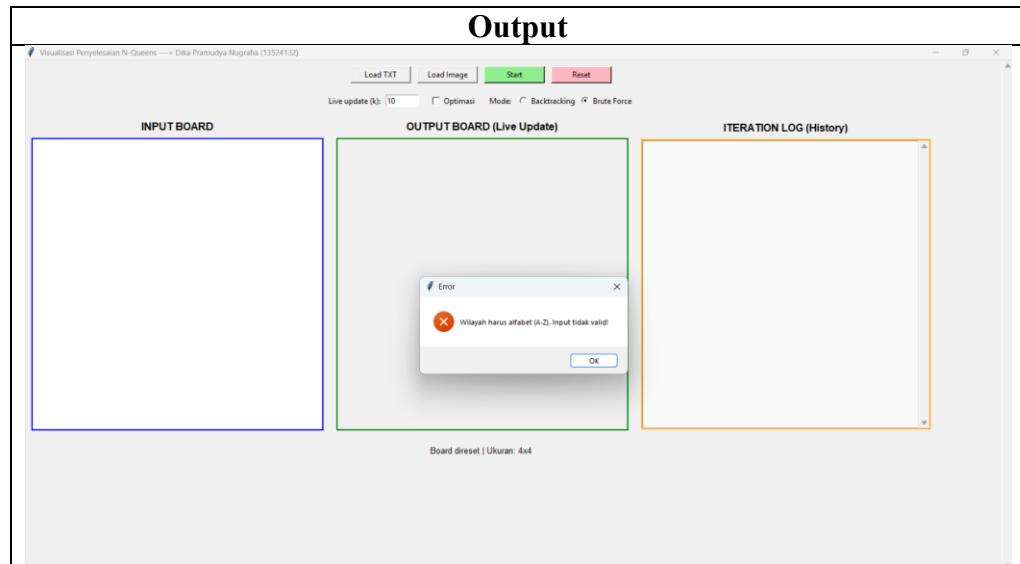
4.1.5. Test Case 5 – 3x2 Tidak Persegi

Data Input
AABB AABB CCDD



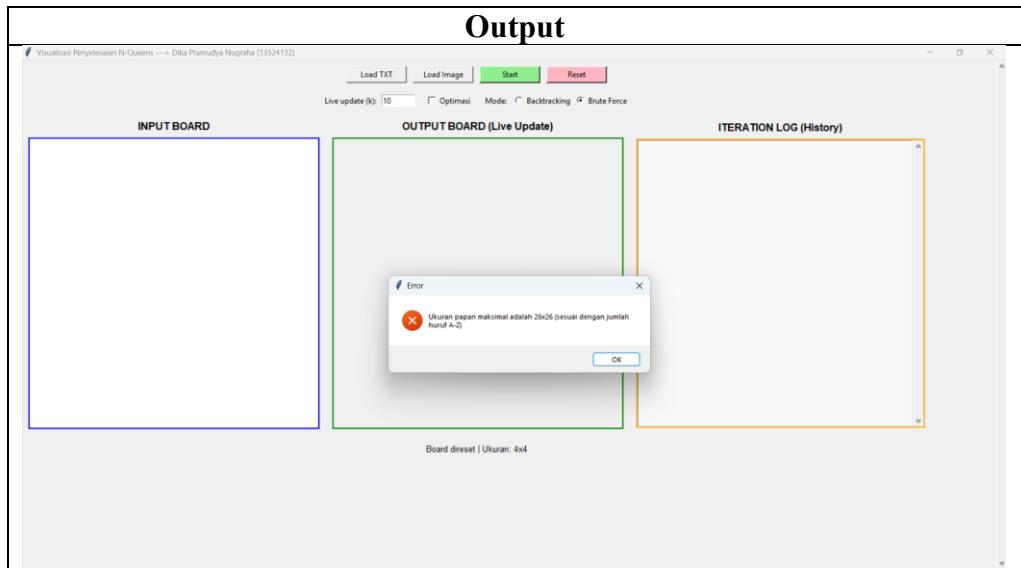
4.1.6. Test Case 6 – 4x4 Karakter Diluar Alfabet (A-Z)

Data Input
AAB1 AABB CCDD CCDD



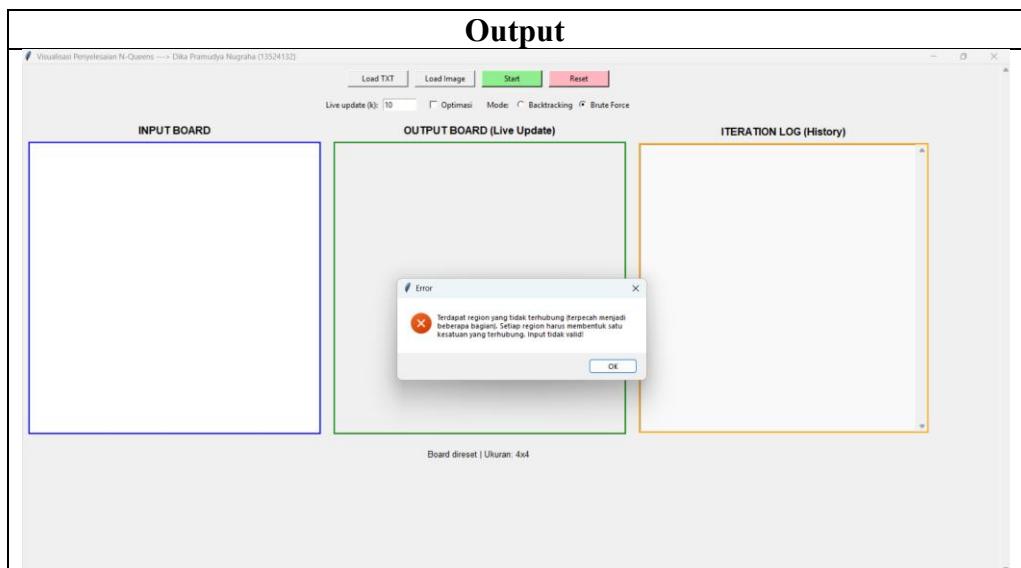
4.1.7. Test Case 7 – N lebih dari 26

Data Input
AAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAA BBBBBBBBBBBBBBBBBBBBBBBBBBB CCCCCCCCCCCCCCCCCCCCCCCC DDDDDDDDDDDDDDDDDDDDDDDDDDDD EEEEEEEEEEEEEEEEEEEEEEE FFFFFFF FFFFFFF FFFFFFF GGGGGGGGGGGGGGGGGGGGGGGGGGGG HHHHHHHHHHHHHHHHHHHHHHHHHHHHH IIIIIIIIIIIIIIIIIIII JJJJJJJJJJJJJJJJJJJJJJ KKKKKKKKKKKKKKKKKKKKKKKKKKK LLLLLLLLLLLLLLLLLLLLLLLL MMMMMMMMMMMMMMMMMMMMMMMM NNNNNNNNNNNNNNNNNNNNNNNNNN OOOOOOOOOOOOOOOOOOOOOOOO PPPPPPPPPPPPPPPPPPPPPPPPPP QQQQQQQQQQQQQQQQQQQQQQQQ RRRRRRRRRRRRRRRRRRRRRRRRR SSSSSSSSSSSSSSSSSSSSSSSS TTTTTTTTTTTTTTTTTTTT UUUUUUUUUUUUUUUUUUUU VVVVVVVVVVVVVVVVVVVV WWWWWWWWWWWWWWWWWWWWWWWWWW XXXXXXXXXXXXXXXXXXXX YYYYYYYYYYYYYYYYYYYYYYYY ZZZZZZZZZZZZZZZZZZZZZZZZ



4.1.8. Test Case 8 – Warna Terpisah

Data Input
AACC DBDC DBBB AAAA



4.1.9. Test Case 9 – Tidak Ada Solusi

Data Input
AAACC BBCCC DDDDE DDEEE EEEEEE

Output

Visualisasi Penyelesaian N-Queens ---> Dika Pramudya Nugraha (13524132)

Live update (k): 14 Optimasi Mode: Backtracking Brute Force

INPUT BOARD				
A	A	A	C	C
B	B	C	C	C
D	D	D	D	E
D	D	E	E	E
E	E	E	E	E

OUTPUT BOARD (Live Update)				

ITERATION LOG (History)				
Iterasi ke-3,110				
A A A C #				
B B C #				
D D D D #				
D E E E #				
E E E E #				

Iterasi ke-3,110				
A A A C #				
B B C C #				
D D D D #				
D E E E #				
E E E E #				

Iterasi ke-3,120				
A A A C #				
B B C C #				
D D D D #				
D E E E #				
E E E E #				

Iterasi ke-3,120				
A A A C #				
B B C C #				
D D D D #				
D E E E #				
E E E E #				

✖ Tidak ada solusi | Waktu Pencarian: 1668.595 ms | Iterasi: 3,125

4.2. Test Case Gambar

4.2.1. Test Case 1 – 9x9

Data Input

A 10x10 grid representing a sparse binary matrix. The grid contains the following colored cells:

- Red: (1, 2)
- Yellow: (1, 1), (2, 1), (3, 1)
- Purple: (2, 2), (3, 2), (4, 2)
- Blue: (5, 2), (5, 3), (5, 4), (6, 3), (6, 4)
- Green: (4, 3), (4, 4), (5, 5), (6, 5), (6, 6)
- Grey: (3, 1) through (3, 5), (4, 1) through (4, 5), (5, 1) through (5, 5), (6, 1) through (6, 5)
- Pink: (1, 1) through (1, 5), (2, 1) through (2, 5), (3, 1) through (3, 5), (4, 1) through (4, 5), (5, 1) through (5, 5), (6, 1) through (6, 5), (7, 1) through (7, 5), (8, 1) through (8, 5), (9, 1) through (9, 5), (10, 1) through (10, 5)

Output

Visualisasi Penyelesaian N-Queens ---> Dika Pramudya Nugraha (13524132)

Load TXT
Load Image
Start
Reset

Live update (k): 10000 Optimasi Mode: Backtracking Brute Force

INPUT BOARD							
A	A	A	B	B	C	C	C
D	A	B	B	B	C	B	B
D	A	E	E	E	B	C	B
F	F	F	E	B	B	B	B
G	F	G	E	G	G	H	H
G	F	G	G	G	G	H	G
G	G	G	I	I	I	G	H
G	G	G	G	I	G	G	G
G	G	G	G	I	G	G	G

OUTPUT BOARD (Live Update)							
A	A	A	Q	B	B	C	C
D	A	B	B	B	B	Q	B
Q	A	E	E	E	B	C	B
F	F	F	E	B	B	Q	B
G	F	G	E	G	G	H	H
G	F	G	G	G	G	H	H
G	G	G	I	I	I	Q	G
G	G	G	G	Q	G	G	G
G	G	G	G	I	G	G	Q

ITERATION LOG (History)							
Iterasi ke-115,100,000							
#	T	U	V	W	X	Y	Z
#	F	G	G	G	H	G	H
#	G	G	#	I	G	G	G
#	G	G	I	G	G	G	G
#	O	G	I	G	G	G	G

Iterasi ke-115,100,000							
#	A	B	C	C	B	B	B
#	A	B	B	B	A	B	B
#	F	F	E	B	B	B	B
#	F	F	E	B	B	B	B
#	G	F	G	H	H	H	H
#	F	G	G	G	H	H	H
#	G	G	G	G	G	H	H
#	G	G	G	G	G	H	H

Iterasi ke-115,107,525							
#	A	A	B	C	C	B	B
#	A	A	B	C	C	B	B
#	A	E	E	C	B	B	B
#	F	F	E	B	B	B	B
#	F	F	E	B	B	B	B
#	G	G	G	G	H	H	H
#	G	G	G	G	H	H	H
#	G	G	G	G	Q	G	G

Solusi ditemukan! | Waktu Pencarian: 566983.774 ms | Iterasi: 115,107,525

Solusi ditemukan! Apakah Anda ingin menyimpan solusi?
Ya, Simpan
Tidak

4.2.2. Test Case 2 – 5x5

Data Input									

Output

Visualisasi Penyelesaian N-Queens —> Dika Pramudya Nugraha (13524132)

INPUT BOARD					OUTPUT BOARD (Live Update)					ITERATION LOG (History)	
A	B	B	B	B	A	B	👑	B	B	Iterations ke-1,200	
A	A	B	C	C	👑	A	B	C	C	A # B B B A # A C D A # B C D A # E D D # A # E D D #	
A	B	B	C	D	A	B	B	👑	C	Iterations ke-1,300	
A	E	B	D	D	A	👑	B	D	D	A # B B B A # B C C A # B C D A # E D D # A # E D D #	
A	E	D	D	D	A	E	D	D	👑	Iterations ke-1,300	

Live update (k): 100 Optimasi Mode: Backtracking Brute Force

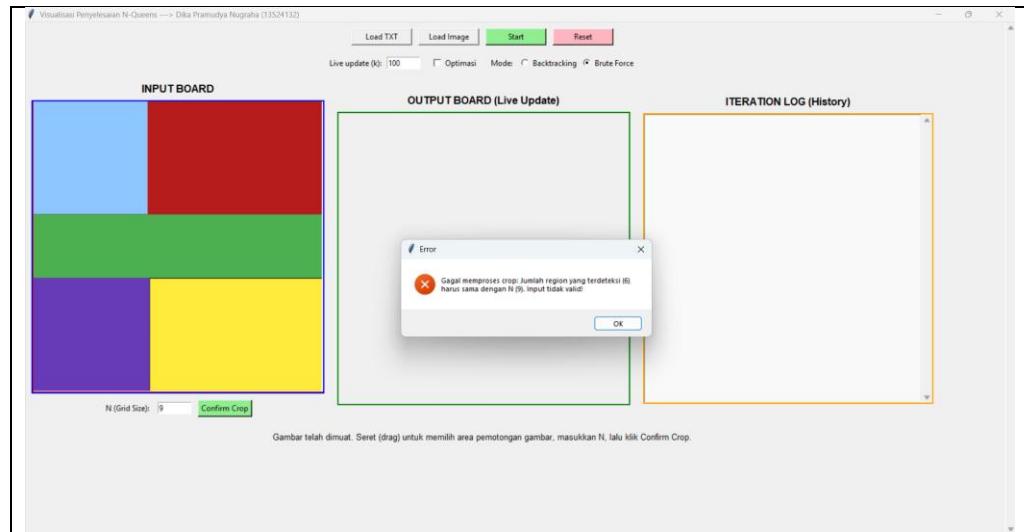
Solusi ditemukan! | Waktu Pencarian: 80.503 ms | Iterasi: 1.335

Solusi ditemukan! Apakah Anda ingin menyimpan solusi?

4.2.3. Test Case 3 – Jumlah Wilayah Tidak Sesuai Dengan N

Data Input									

Output



4.2.4. Test Case 4 – 5x5 Solusi Tidak Ditemukan

Data Input				

Output				
<p>The 'INPUT BOARD' section shows a 5x5 grid labeled A through E. The 'OUTPUT BOARD (Live Update)' and 'ITERATION LOG (History)' sections are visible.</p> <p>ITERATION LOG (History):</p> <pre> Iterasi ke-3,000 A A A B # C C B # B D D D D # E E E E # E E E E # Iterasi ke-3,000 A A A B # C C B # B D D D D # E E E E # E E E E # Iterasi ke-3,100 A A A B # C C B # B D D D D # E E E E # E E E E # Iterasi ke-3,100 A A A B # C C B # B D D D D # E E E E # E E E E # </pre> <p>Message: ✘ Tidak ada solusi Waktu Pencarian: 198.610 ms Iterasi: 3.125</p>				

4.2.5. Test Case 5 – 8x8

Data Input

Output

Visualisasi Penyelesaian N-Queens ---> Dika Pramudya Nugraha (13524132)

Load TXT | Load Image | Start | Reset | Live update (k): 100000 | Optimasi Mode: Backtracking | Brute Force

INPUT BOARD

A	A	A	A	A	A	A	A
A	B	B	B	C	C	C	A
A	B	D	D	D	D	C	A
A	B	D	E	E	D	C	F
A	B	D	E	E	G	C	F
F	B	D	D	G	G	H	F
F	B	B	B	B	H	H	F
F	F	F	F	F	F	F	F

OUTPUT BOARD (Live Update)

A	A	A	A	A	A	A	#
A	B	B	B	#	C	C	A
A	#	D	D	D	D	C	A
A	B	D	#	E	D	C	F
A	B	D	E	E	#	O	F
F	B	#	D	G	G	H	F
F	B	B	B	B	H	#	F
F	#	F	F	F	F	F	F

ITERATION LOG (History)

```

A B S A C C C A
A # D E D C F
# B D E G C F
# C D E G F
# D E G F
# E G F
# F G F
# G H F
# H F F F F F
-----  

Iterasi ke-15,700,000
A A A A A A A A
A B D D D C A
A # D E E C C F
# C D E G F
# D D G G F
# B D D G G F
# F B B B H F
# F F F F F F F
-----  

Iterasi ke-15,776,433
A A A A A A A A
A B D D D C A
A # D D D C A
A B D # E D C F
# C D E G F
# D D G G F
# B D D G G F
# F B B B H F
# F F F F F F F
-----  


```

Solusi ditemukan! Waktu Pencarian: 57183.136 ms | Iterasi: 15,776

Solusi ditemukan! Apakah Anda ingin menyimpan solusi?

Ya, Simpan | **Tidak**

4.3. Download Output

Memilih Folder Mana yang Menjadi Tempat Penyimpanan

Visualisasi Penyelesaian N-Queens ---> Dika Pramudya Nugraha (13524132)

Save As | Start | Reset | Backtracking | Brute Force | Live update (k): 100000 | Optimasi Mode: Backtracking | Brute Force

Organize | New folder | output

File name: solution_20260217_212710.txt

Save as type: Text Files (*.txt)

ITERATION LOG (History)

```

A T U
# A B
# C D
C C D #
-----  

Iterasi ke-100
A # B B
A A B
# C D
C C D #
-----  

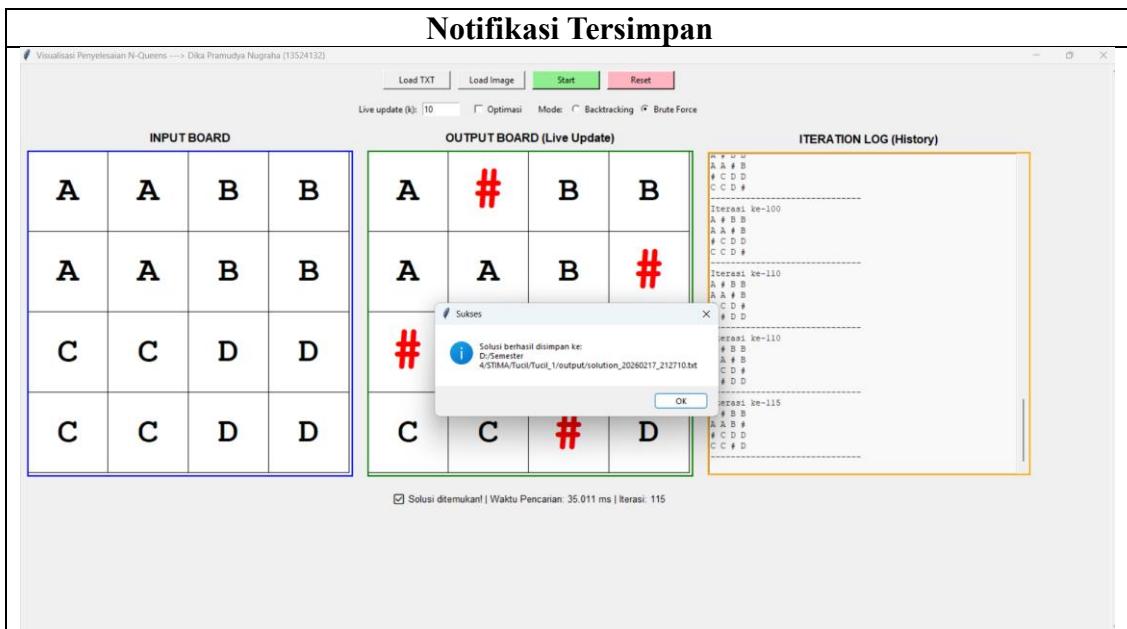
Iterasi ke-110
A # B
A A B
C C A
C # D D
-----  

Iterasi ke-110
A # B
A A B
C C A
C # D D
-----  

Iterasi ke-115
A # B B
A A B
# C D
C C D
-----  


```

Solusi ditemukan! Waktu Pencarian: 35.011 ms | Iterasi: 115



BAB V

LAMPIRAN

No	Poin	Ya	Tidak
1	Program berhasil di kompilasi 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	✓	

PERNYATAAN TIDAK MELAKUKAN KECURANGAN

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

A handwritten signature in black ink, appearing to read "Dika Pramudya Nugraha".

Dika Pramudya Nugraha - 13524132

DAFTAR PUSTAKA

- [1] [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/02-Algoritma-Brute-Force-\(2026\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/02-Algoritma-Brute-Force-(2026)-Bag1.pdf)
- [2] https://edunexcontentprodhot.blob.core.windows.net/edunex/2026/89717-Algorithm-Strategy/413619-Mgg-01-Pengantar-Strategi-Algoritma-Algoritma-Brute-Force/file/1770804587124_Tucil1-IF2211-2026?sv=2024-11-04&spr=https&st=2026-02-11T10%3A09%3A47Z&se=2028-02-11T10%3A09%3A47Z&sr=b&sp=r&sig=fLBK0u5mk%2F3dLRIBLgXWePv1mTE0XNyktJF4Yq0ucT0%3D&rsct=application%2Fpdf
- [3] <https://sheeptester.github.io/words-go-here/misc/queens.html>
- [4] freeCodeCamp.org. (2022, June 20). *Brute force algorithms explained*.
<https://www.freecodecamp.org/news/brute-force-algorithms-explained/>