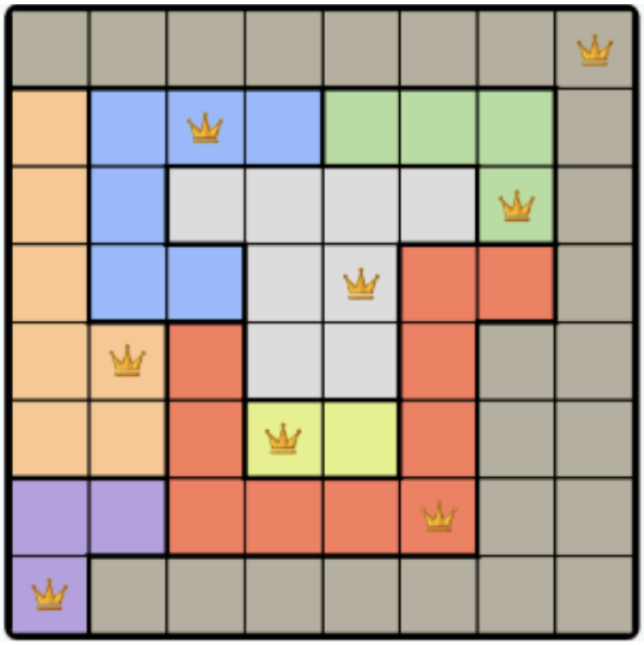
Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II tahun 2025/2026

Penyelesaian Permainan ***Queens*** Linkedin



Nama : Dzaki Ahmad Al Hussainy

NIM : 13524084

Program Studi Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung 2025/2026

Deskripsi Permasalahan

Queens adalah gim logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari gim ini adalah menempatkan *queen* pada sebuah papan persegi berwarna sehingga terdapat hanya satu *queen* pada tiap baris, kolom, dan daerah warna. Selain itu, satu *queen* tidak dapat ditempatkan bersebelahan dengan *queen* lainnya, termasuk secara diagonal.

Analisis Permasalahan

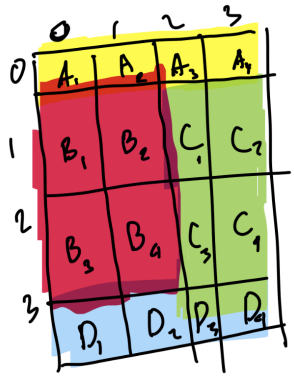
Dekomposisi

Pada setelah mencoba permainan ini kita dapat menentukan ada beberapa aturan yang harus diikuti

1. Tidak boleh ada *Queens* yang ditaruh pada baris dan kolom yang sama
2. Tidak boleh ada *Queens* yang ditaruh pada bersebelahan dengan queen lainnya
3. Tidak boleh ada *Queens* yang ditaruh pada warna yang sama

Bagaimana cara untuk merepresentasikan papan ini dalam komputer?
Papan ini akan menyimpan beberapa informasi yaitu posisi yang terdiri dari koordinat x dan y kemudian warna dari kotak itu sendiri.

Mari kita analisis lebih lanjut bagaimana cara kita menyelesaikan peraturan yang harus diikuti dalam penyelesaian algoritma ini misalnya kita akan coba dengan menyelesaikan untuk papan kecil berukuran 4x4



Perhatikan gambar di kiri, gambar tersebut bisa kita dituliskan menjadi pasangan warna kemudian posisi masing-masing koordinat yang memiliki warna yang sama.

A : {A1{0,0}, A2 {0,1}, A3 {0,2}, A4 {0,3}}
B : {B1{1,0}, B2{1,1}, B3 {2,0}, B4 {2,1}}
C : {C1{1,2}, C2 {1,3}, C3 {2,2}, C4 {2,3}}
D : {D1{3,0}, D2 {3,1}, D3 {3,2}, D4 {3,3}}

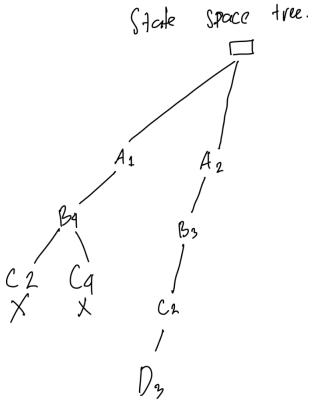
Pengenalan Pola

Pada misalnya kita memilih A1 sebagai pilihan awal kita maka kita bisa melihat nilai yang memiliki baris 0 dan juga kolom 0 tidak bisa dipilih lagi, sehingga pilihan B1, B3, D1 tidak bisa dipilih karena tidak sesuai dengan aturan 1, dan tidak bisa memilih A2,A3,A4 karena tidak sesuai dengan aturan 3.

Artinya kita bisa rumuskan

1. Jika ingin menaruh sebuah queen pada posisi sebelumnya kita perlu cek apakah posisi itu bisa ditempatkan jika pos.row = queen1.row atau pos.col = queen1.col artinya tidak dapat ditempatkan di posisi tersebut.
2. Kemudian untuk melakukan cek aturan 2, pada aturan ini harus melakukan cek bersebelahan dengan membuat sebuah grid sebesar papan kemudian melakukan iterasi dari baris + 1 hingga baris - 1 yang di dalamnya melakukan iterasi kolom + 1 hingga kolom - 1.
3. Masing-masing warna akan memiliki pasangan senarai koordinat dari kotak yang memiliki warna yang sama.

Dalam pembuatan penyelesaian pencarian ini misalnya secara pencarian manusia akan melakukan mencari warna dengan kotak terkecil terlebih dahulu kemudian melakukan iterasi per warna mana saja yang harus ditaruh *queen* kemudian mengeceknya apakah sesuai dengan aturan atau tidak.



Di samping kiri adalah contoh misalnya memilih A1 terlebih dahulu, kemudian kita akan melakukan cek warna apa saja yang bisa ditaruh berikutnya, kebetulan disini adalah warna B, pada warna B posisi mana saja yang bisa ditaruh *queen* jika sudah menaruh di A1 yaitu hanya di B4. kemudian kita melakukan cek lagi warna berikutnya untuk warna C warna yang bisa ditaruh berikutnya adalah di posisi C2 atau C4. Kemudian kita pilih misalnya C2, apakah ada tempat yang bisa ditaruh *queen* di D? D3 tidak bisa karena melanggar aturan 2 dan D4 tidak bisa juga karena sudah sebaris dengan C2. Karena C2 tidak ada pilihan yang valid untuk warna D dilakukan kembali ke pilihan C4 jika dipilih maka D3 dan D4 tidak valid karena melanggar aturan ke-2. Sehingga melakukan putar balik ke pilihan warna B yang tidak ada pilihan lain kemudian kembali ke pilihan A dan memilih A2. Setelah itu akan mencoba memilih warna B yang tersedia dan valid yaitu B3. Setelah memilih B3 lanjut ke warna C pada warna cek akan melakukan cek apakah tempat yang tersedia hanya ada C2 yang mengikuti ketiga aturan itu. Kemudian hanya tersisa D3 untuk warna terakhir sehingga mendapatkan urutan penyelesaian A2, B3, C2 dan D3 yaitu menempatkan *queen* di {0,1},{2,0}, {1,3} dan {3,2}.

Dalam penyelesaian masalah ini saya akan menggunakan bahasa pemrograman java karena memiliki library yang lengkap dan pembuatan GUI yang sudah familiar.

Abstraksi

Mari kita abstraksi penyelesaian dari penjelasan pengenalan pola sebelumnya. Mari kita mulai membahas agar dapat membuat algoritma secara efektif.

Abstraksi Penyelesaian Permasalahan

1. Cara Menggambarkan Posisi
   Setelah melakukan dekomposisi kita telah mengetahui setiap kotak memiliki atribut ada koordinat x, y dan juga warna yang digambarkan dengan sebuah char.
2. Cara Menggambarkan Papan
   Papan akan memiliki atribut berisi semuaPosisi yang berisi posisi masing-masing setiap kota. semuaPosisi ini akan dipakai untuk menampilkan papan dan keadaan papan. Karena masing-masing papan akan berisi posisi yang memiliki warna unik diperlukan juga atribut untuk memberikan masing-masing warna unik pada papan tersebut. Atribut warna ini akan dituliskan dengan warnaUnik dan digunakan untuk melakukan iterasi masing-masing warna. Diperlukan juga ukuran papan jadi dituliskan juga ukuran baris dan kolom agar mempermudah mengakses ukuran papan.
3. Cara Menyelesaikan Aturan 1
   Saat melakukan cek posisi misalnya ada posisi p. Perlu dilakukan cek apakah sudah ada kolom atau baris yang sudah dipakai. Sehingga lebih mudah untuk melakukan akses kita buat sebuah array boolean yang digunakan untuk melakukan cek. Kita buat array boolean pada Class solver

colsTerpilih dan rowsTerpilih kemudian kita inisalisasi dengan false. Saat melakukan cek apakah queen bisa ditaruh di titik p valid atau tidak, kita bisa melakukan akses dengan colsTerpilih[p.col] atau rowTerpilih[p.row] jika sudah ditempati maka akan menghasilkan false untuk menandakan tempatnya tidak valid.

4. Cara Menyelesaikan Aturan 2
   Kemudian untuk melakukan cek ketetanggaan dari masing-masing *queen*, kita buat juga matriks boolean yang berisi lokasi di mana *queen* dapat ditaruh sehingga bisa dilakukan cek dengan cepat. Sehingga dilakukan iterasi misalnya dari posisi p, dilakukan iterasi i dari p.row + 1 hingga p.row -1 kemudian iterasi j dari p.col + 1 hingga p.col - 1, kemudian jika grid[i][j] = true maka tidak valid karena berada di tetangga suatu *queen.*
5. Cara Menyelesaikan Aturan 3
   Aturan ketiga menyatakan *queen* harus ditaruh di warna yang unik, hal ini dapat diatasi dengan membuat sebuah map yang memiliki kunci warna dan value berisi list yang berisi posisi dengan warna yang sama, hal ini mempermudah dalam melakukan iterasi mengutamakan warna untuk melakukan menaruhkan *queen.*

Penyusunan Algoritma

Dalam menyusun permasalahan ini akan melakukan iterasi *brute force* yang mengutamakan warna terlebih dahulu, alasannya dari penyelesaian manusia biasanya akan menyelesaikan dari warna yang memiliki warna dengan jumlah posisi paling sedikit terlebih dahulu kemudian melakukan penyelesaian ke warna yang terkecil berikutnya hingga selesai mencoba semua kombinasi sehingga mendapatkan solusi yang valid. Algoritma ini akan melakukan pengecekan penaruhan *queen* pada board jika valid akan melanjutkan ke warna berikutnya jika tidak akan lanjut ke warna yang sama pada posisi yang berbeda.

Heuristik Algoritma

Penyelesaian algoritma ini dapat lebih cepat jika melakukan pengurutan posisiPerWarna dengan posisi pada warna tertentu yang paling sedikit terlebih dahulu. Saya membuat *toggle button* untuk dapat menjalankan heuristik ini.

Penyelesaian Algoritma

Fungsi atau Prosedur Pembantu

1. Fungsi isSafe(Position p)
   Digunakan untuk melakukan cek apakah posisi tersebut dapat ditempatkan queen atau tidak

```
private boolean[] rowsTerpilih;
private boolean[] colsTerpilih;
private boolean[][] grid;

public boolean isSafe(Position p) {
    int r = p.row;
    int c = p.col;

    if (rowsTerpilih[r] || colsTerpilih[c]) {
        return false;
    }

    // Melakukan cek tetangga
    for (int i = r - 1; i <= r + 1; i++) {
        for (int j = c - 1 ; j <= c + 1; j++) {
            if (i >= 0 && i < boardSizeRow && j>= 0 && j <
                boardSizeCol) {
                if (grid[i][j]) {
                    return false;
                }
            }
        }
    }
    return true;
}
```

2. Prosedur placeQueen(Position p)
   Digunakan untuk mempermudah dalam menaruh sebuah queen dan memperbaharui keadaan papan sekarang

```
public void placeQueen(Position p) {
        int r = p.row;
        int c = p.col;

        rowsTerpilih[r] = true;
        colsTerpilih[c] = true;
        grid[p.row][p.col] = true;

        solusi.add(p);
    }
```

3. Prosedur removeQueen(Position p)
   Digunakan untuk mempermudah jika queen tidak bisa ditaruh maka queen akan diambil kembali dan mengulang iterasi berikutnya.

```java
public void removeQueen(Position p) {
        int r = p.row;
        int c = p.col;

        rowsTerpilih[r] = false;
        colsTerpilih[c] = false;
        grid[p.row][p.col] = false;

        // Delete elemen terakhir
        solusi.remove(solusi.size() - 1);
    }
```

4. Fungsi solveRecursive(int idxWarna, List<Character> warna)
   Pada fungsi ini akan meminta idxWarna untuk mendapatkan indeks dari list karakter warna kemudian secara rekursif akan menambahkan idxWarna dan melakukan iterasi pada list warna lagi. Iterasi pada warna ini akan dipakai untuk menjadi kunci dalam mengakses posisi pada warna tersebut dan melakukan pencarian dari semua kombinasi secara looping rekursif.

```java
private boolean solveRecursive(int idxWarna, List<Character> warna){
        this.iteration++;
        // Basis ketika index dari warna = ukuran warna
        if (idxWarna == warna.size()){
            return true;
        }

        char currWarna = warna.get(idxWarna);

        List<Position> posisiAvailableWarna = board.posisiPerWarna.get
            (currWarna);
        for (Position p : posisiAvailableWarna) {
            if (isSafe(p)) {
                placeQueen(p);
                updateVisual();
                if (solveRecursive(idxWarna + 1, warna)) {
                    return true;
                }
                removeQueen(p);
                updateVisual();
            }
        }
        return false;
    }
```

   Prosedur updateVisual() di sini hanya digunakan untuk menampilkan *live update* visualisasi pada GUI

Algoritma Utama

Pada algoritma ini memanggil fungsi solveRekursif dengan input 0 untuk index warna awal kemudian mengambil warna untuk terurut atau tidak terurut berdasarkan menggunakan heuristik atau tidak

```java
public boolean solve(boolean heuristic){
        this.iteration = 0;
        if (heuristic) {
            return solveRecursive(0, board.getWarnaTerutut());
        }
        else{
            return solveRecursive(0, board.getWarna());
        }

    }
```

Analisis Algoritma

Untuk melakukan analisis kompleksitas algoritma *worst case* misalnya ukuran papan akan berbentuk persegi dan terdapat n jumlah warna sehingga dapat diformulasikan

n1 + n2 .. + n_i = n

dan kita kombinasinya ditulis dengan

n1 * n2 * … * n_i

Maka kita ingin memaksimalkan hasil dari kedua permasalahan ini sehingga n yang dipakai itu harus sama, maka kita dapatkan O(n^n).

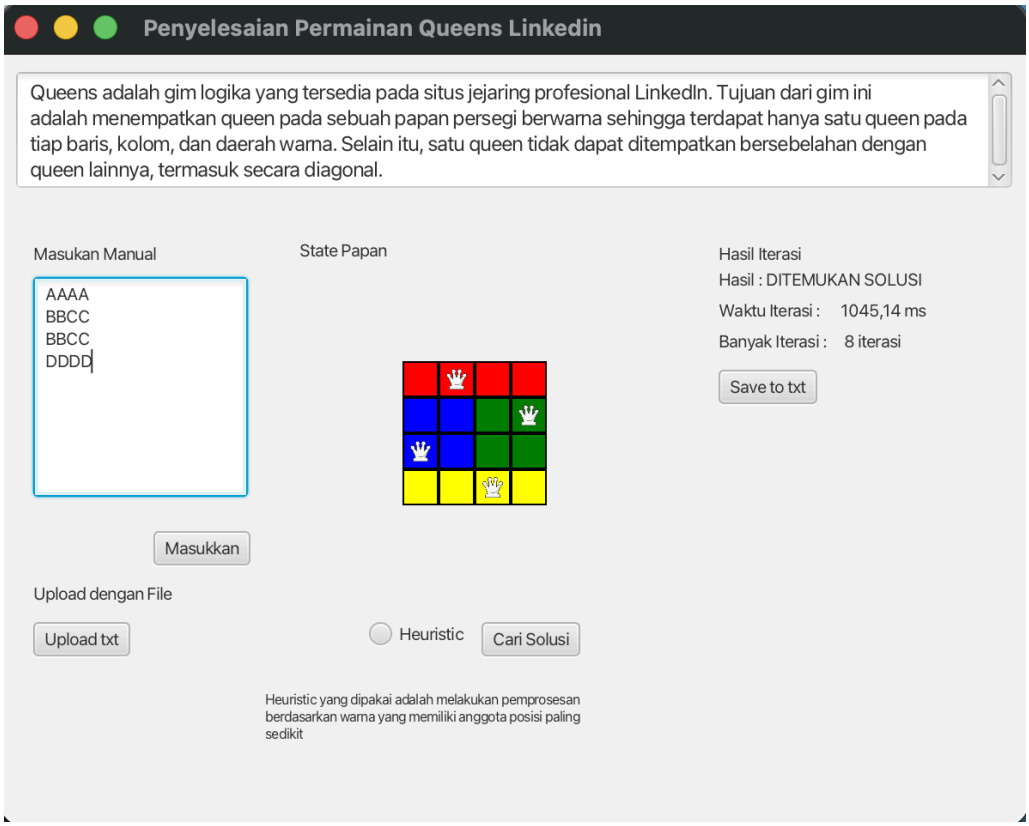
Graphical User Interface GUI

GUI yang dibuat disini menggunakan *SceneBuilder* untuk membuat layout dan mendesain fungsi tambahan untuk melakukan tampilan papan. Digunakan juga *interface* dan *threading* pada java untuk menampilkan live update dalam menaruh queen. Jujur ini merupakan pertama kali saya melakukan *Threding* pada java dan sangat menarik bagaimana GUI dapat memanggil fungsi dengan menggunakan *thread* yang berbeda.

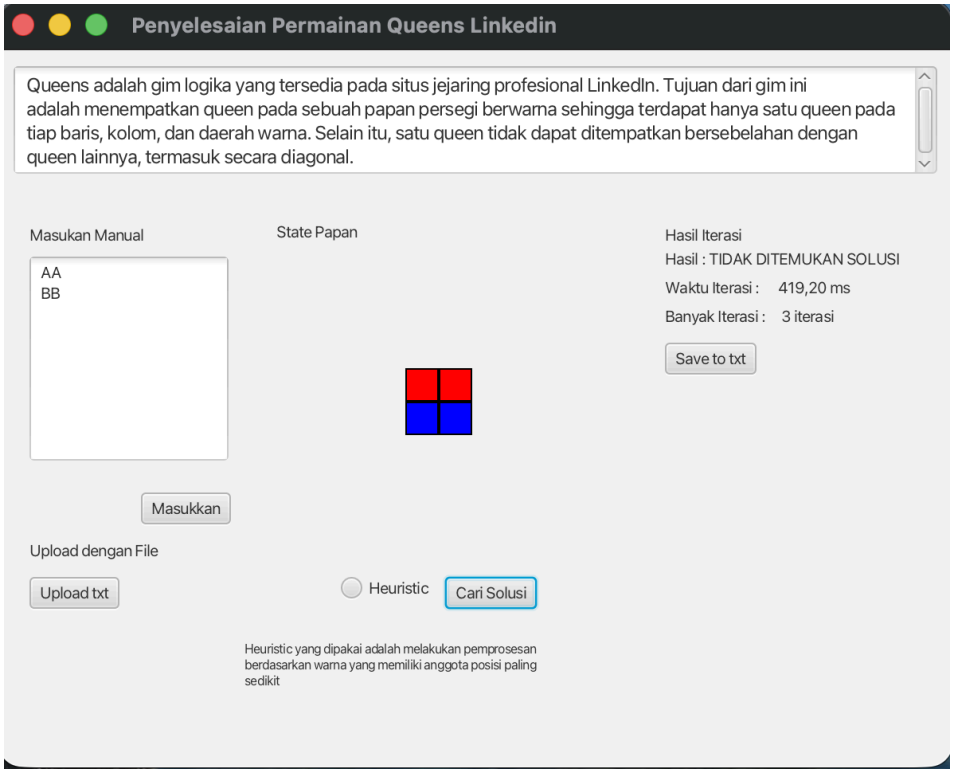
Hasil Pengujian

Pengujian dari program ini ada beberapa yang dibuat oleh saya dan ada juga yang diambil dari website https://queensgame.vercel.app/ untuk pengujian dengan papan yang besar.

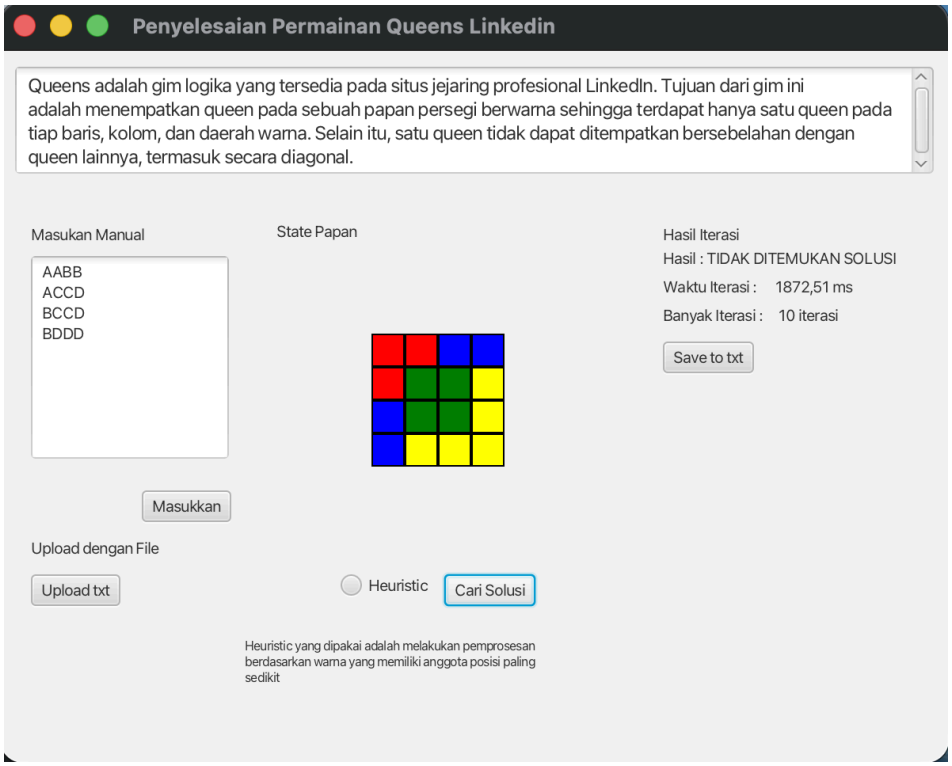
Kasus Uji 1



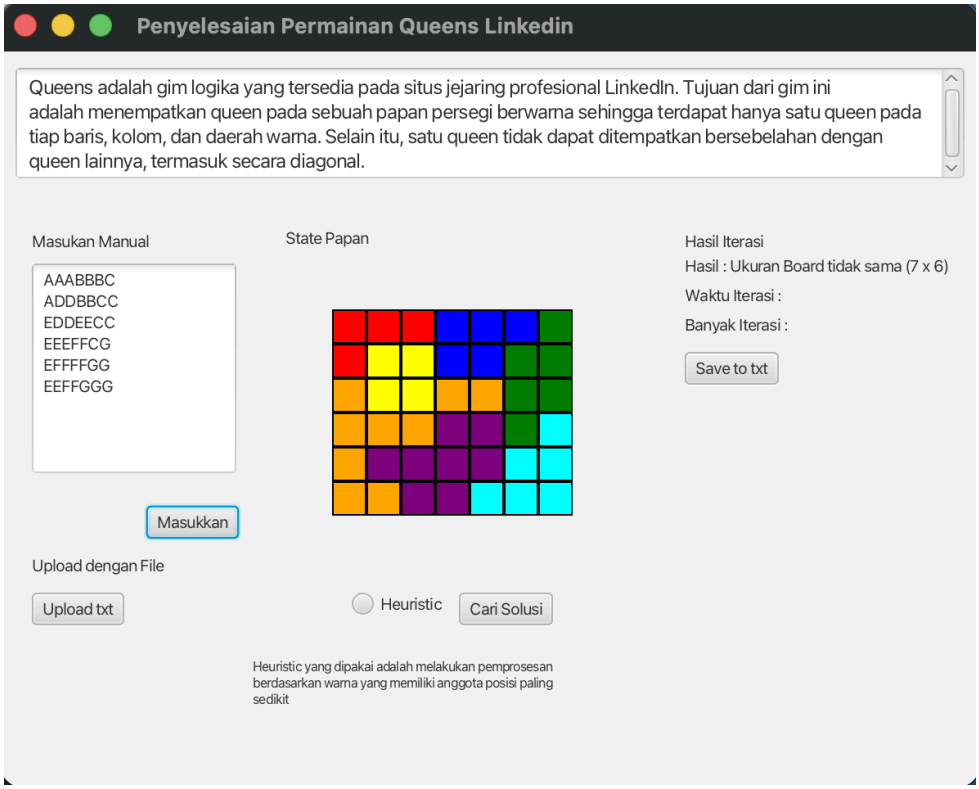
Kasus Uji 2



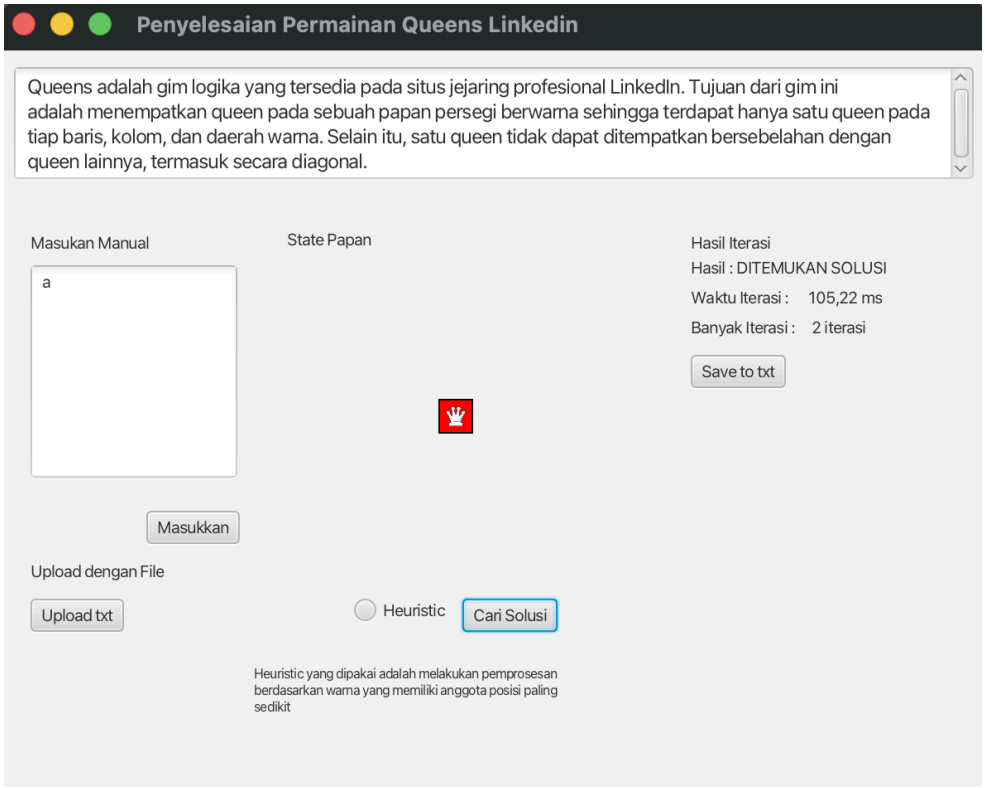
Kasus Uji 3



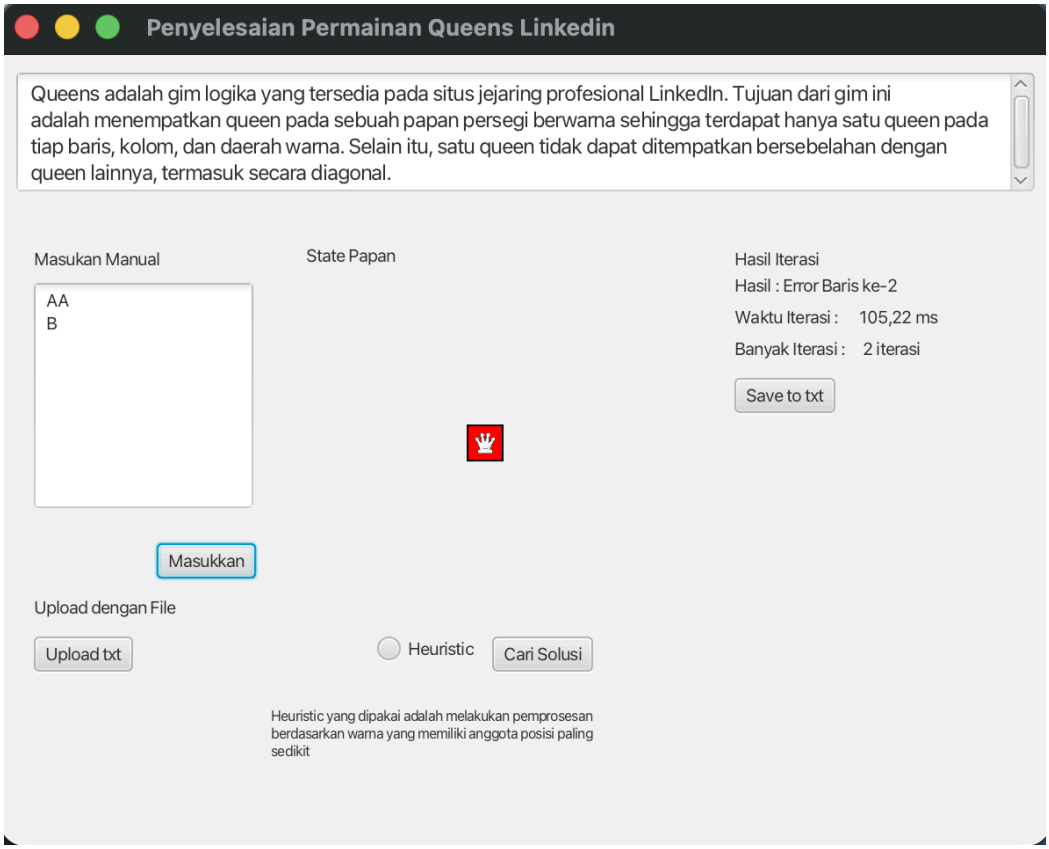
Kasus Uji 4



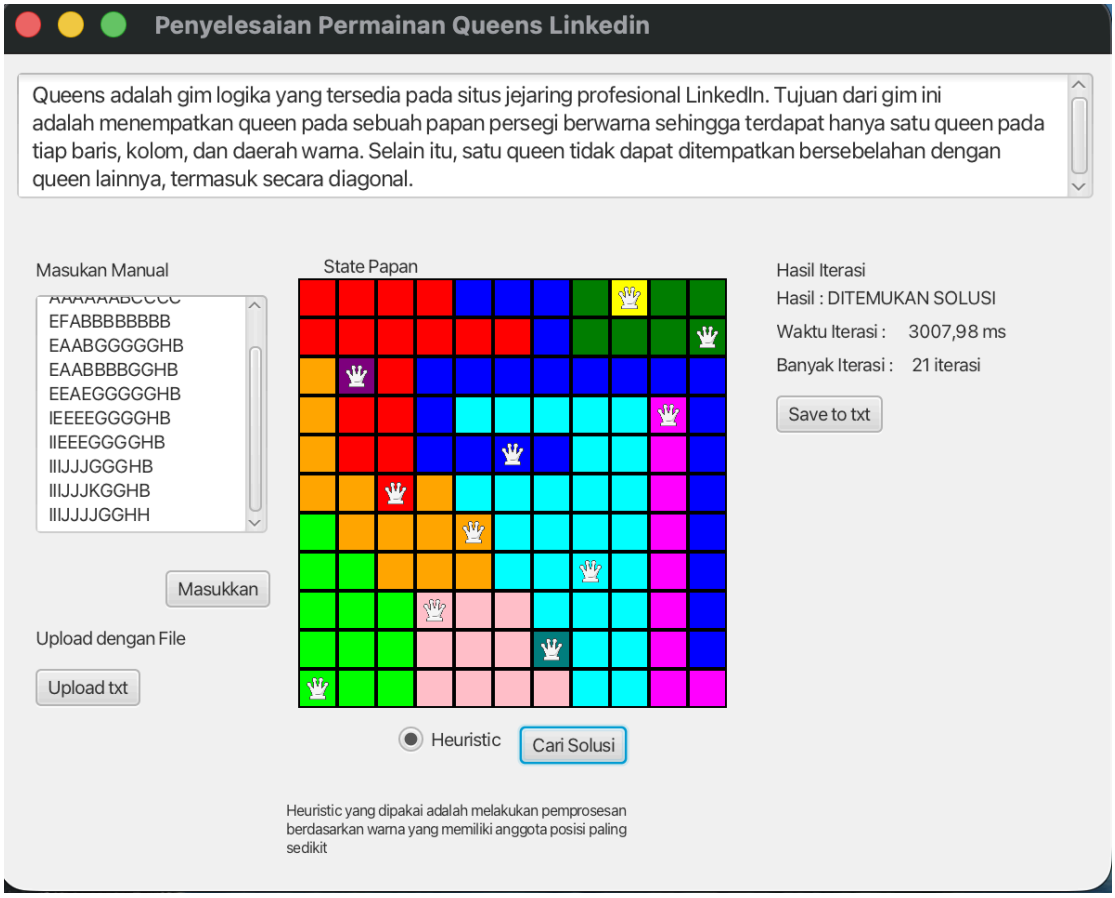
Kasus Uji 5



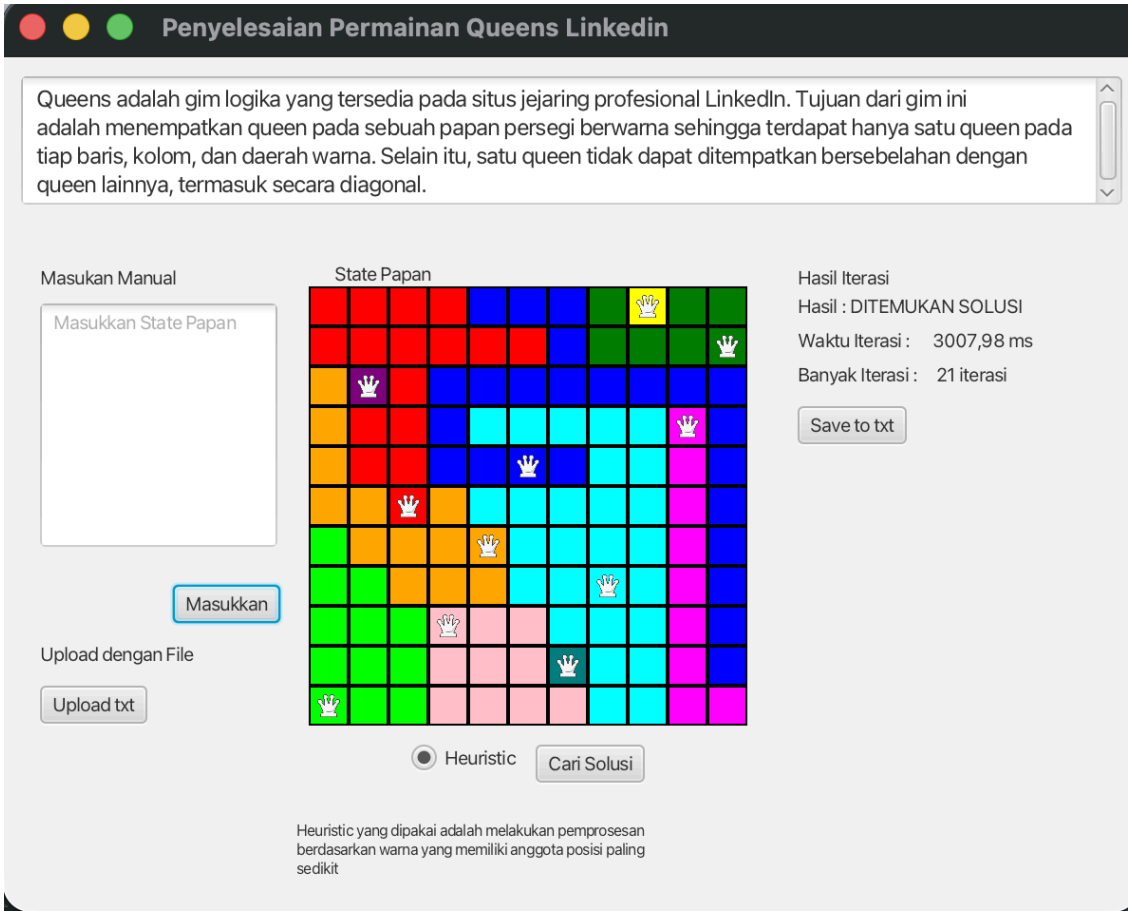
Kasus Uji 6


Penyelesaian Permainan Queens Linkedin
Queens adalah gim logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari gim ini adalah menempatkan queen pada sebuah papan persegi berwarna sehingga terdapat hanya satu queen pada tiap baris, kolom, dan daerah warna. Selain itu, satu queen tidak dapat ditempatkan bersebelahan dengan queen lainnya, termasuk secara diagonal.
Masukan Manual
AA
B
Masukkan
Upload dengan File
Upload txt
State Papan
Heuristic
Cari Solusi
Heuristic yang dipakai adalah melakukan pemprosesan berdasarkan warna yang memiliki anggota posisi paling sedikit
Hasil Iterasi
Hasil : Error Baris ke-2
Waktu Iterasi : 105,22 ms
Banyak Iterasi : 2 iterasi
Save to txt

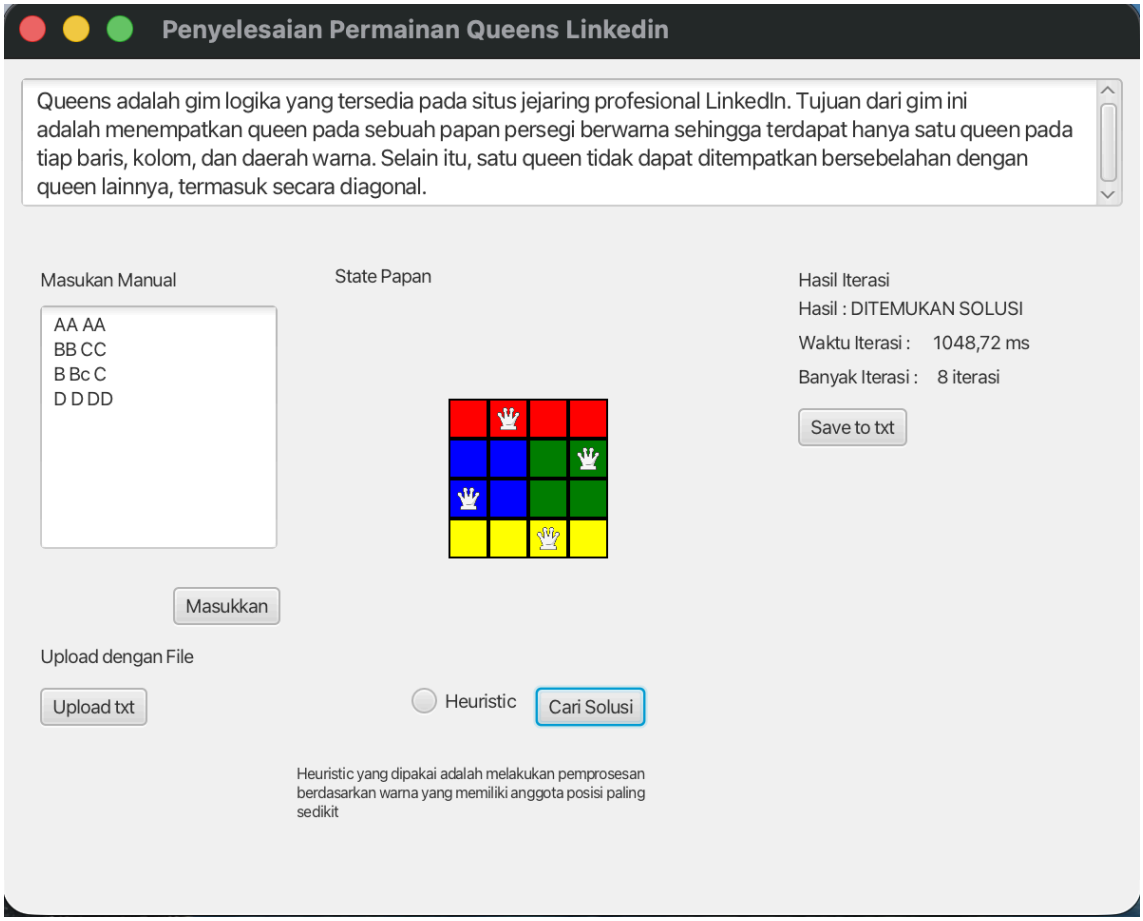
Kasus Uji 7


Penyelesaian Permainan Queens Linkedin
Queens adalah gim logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari gim ini adalah menempatkan queen pada sebuah papan persegi berwarna sehingga terdapat hanya satu queen pada tiap baris, kolom, dan daerah warna. Selain itu, satu queen tidak dapat ditempatkan bersebelahan dengan queen lainnya, termasuk secara diagonal.
Masukan Manual
EFABBBBBBBB
EAABGGGGGHB
EAABBBBGGHB
EEAEGGGGGHB
IEEEEGGGGHB
IIEEEGGGGHB
IIIJJJGGGHB
IIIJJJKGGHB
IIIJJJJGGHH
Masukkan
Upload dengan File
Upload txt
State Papan
Heuristic
Cari Solusi
Heuristic yang dipakai adalah melakukan pemprosesan berdasarkan warna yang memiliki anggota posisi paling sedikit
Hasil Iterasi
Hasil : DITEMUKAN SOLUSI
Waktu Iterasi : 3007,98 ms
Banyak Iterasi : 21 iterasi
Save to txt

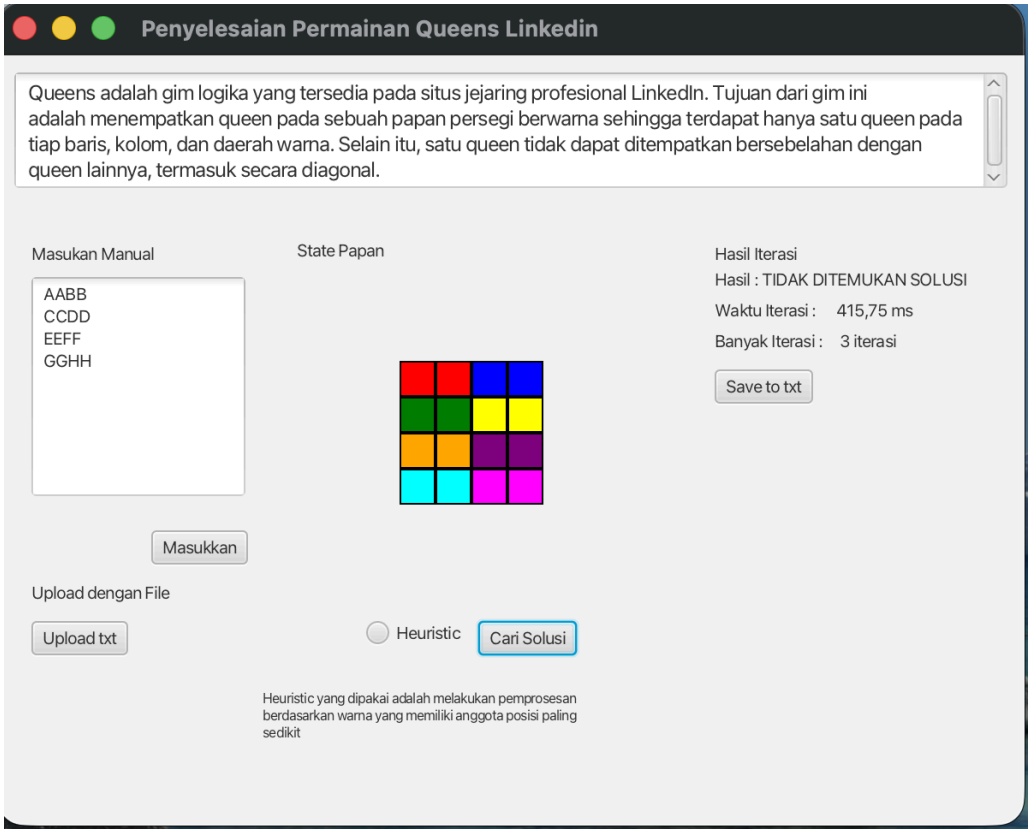
Kasus Uji 8


Penyelesaian Permainan Queens Linkedin
Queens adalah gim logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari gim ini adalah menempatkan queen pada sebuah papan persegi berwarna sehingga terdapat hanya satu queen pada tiap baris, kolom, dan daerah warna. Selain itu, satu queen tidak dapat ditempatkan bersebelahan dengan queen lainnya, termasuk secara diagonal.
Masukan Manual
Masukkan State Papan
Masukkan
Upload dengan File
Upload txt
State Papan
Heuristic
Cari Solusi
Heuristic yang dipakai adalah melakukan pemprosesan berdasarkan warna yang memiliki anggota posisi paling sedikit
Hasil Iterasi
Hasil : DITEMUKAN SOLUSI
Waktu Iterasi : 3007,98 ms
Banyak Iterasi : 21 iterasi
Save to txt

Kasus Uji 9


Penyelesaian Permainan Queens Linkedin
Queens adalah gim logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari gim ini adalah menempatkan queen pada sebuah papan persegi berwarna sehingga terdapat hanya satu queen pada tiap baris, kolom, dan daerah warna. Selain itu, satu queen tidak dapat ditempatkan bersebelahan dengan queen lainnya, termasuk secara diagonal.
Masukan Manual
AA AA
BB CC
B Bc C
D D DD
Masukkan
Upload dengan File
Upload txt
State Papan
Heuristic
Cari Solusi
Heuristic yang dipakai adalah melakukan pemprosesan berdasarkan warna yang memiliki anggota posisi paling sedikit
Hasil Iterasi
Hasil : DITEMUKAN SOLUSI
Waktu Iterasi : 1048,72 ms
Banyak Iterasi : 8 iterasi
Save to txt

Kasus Uji 10



Kesimpulan

Algoritma *brute force* algoritma yang tidak begitu cerdas dan langsung menyelesaikan masalah berdasarkan alur pemikiran programmer berdasarkan alur berpikir komputasional dan pendekatan dengan algoritma dan data struktur. Pendekatan ini secara mudah dapat dijalankan

Lampiran

Repositori : https://github.com/HussainDzaki/tucil1_13524084

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

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

Dzaki Ahmad Al Hussainy 13524084

Daftar Pusaka

Penjelasan algoritma Brute force : Algoritma Brute Force dan Algoritma Brute Force 2
Abdul Bari 6.1 N Queens Problem using Backtracking
Dokumentasi java : Overview (JavaFX 8)