N-Queen Linkedin Game Solver
Tugas Kecil 1

Penulis:
Syaqina Octavia Rizha (13524088)

MATA KULIAH IF2211 STRATEGI AlGORITMA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2025

BAB I: PENDAHULUAN

1.1 Deskripsi Persoalan

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. Program yang dibuat ini dapat menemukan satu solusi penempatan queen pada suatu papan berwarna yang diberikan, atau menampilkan bahwa tidak ada solusi yang valid. Program melakukan pencarian solusi menggunakan algoritma brute force.

BAB II: ISI

2.1 Penjelasan Algoritma

A. Brute Force Murni

Algoritma ini mencoba semua kemungkinan dengan melakukan peletakan N-Queen pada papan N*N tanpa memperhatikan aturan peletakannya terlebih dahulu. Sehingga kompleksitas dari algoritma ini sama dengan kombinasi $C(N^2, N)$.

Alur:

1. Papan diasumsikan sebagai 1 array dengan indeks [0..N*N-1]
2. N-Queen diletakkan di baris pertama papan
3. Validasi N-Queen
4. Jika belum, geser Queen[N], ke indeks selanjutnya. Validasi
5. Ulangi hingga Queen[N] mencapai indeks teakhir (N*N-1)
6. Jika sudah di indeks terujung, majukan Queen[N-1], ke indeks selanjutnya, dan Queen[N], diletakkan disebelahnya.
7. Ulangi proses

B. Brute Force dengan Optimasi

Algoritma ini menerapkan 1 aturan sejak awal, yaitu “Hanya ada 1 Queen pada 1 baris”, sehingga dilakukan peletakan 1 Queen pada setiap baris, lalu iterasi dari baris ke-N. Kompleksitas dari algoritma ini sama dengan kombinasi O(N^N).

Alur:

1. Iterasi dilakukan dengan Array 1D, indeks sebagai row, dan hanay column yang berubah-ubah.
2. 1-Queen diletakkan di setiap baris pada kolom pertama papan
3. Validasi N-Queen
4. Jika belum, geser Queen[N] (columm++). Validasi
5. Ulangi hingga Queen[N] mencapai kolom teakhir (N-1)
6. Jika sudah di kolom terujung, majukan Queen[N-1], ke indeks selanjutnya, dan Queen[N], diletakkan dari kolom 0 lagi.
7. Ulangi proses

2.2 Kode Penyelesaian

A. Brute Force Murni

```
void algoritmaBF(Point* coordinate, char* color, int n, char** map, bool* valid) {
    // Pure brute force, kompleksitas O(C(n*n, n))
    int d1[n];
    for(int i = 0; i < n; i++){
        coordinate[i] = {0, i};
        color[i] = map[0][i];
        d1[i] = i;
    }

    int shifted = n-1;

    if(isValid(coordinate, color, n)){
        *valid = true;
        steps++;
        return;
    }else{
        bool found = false;

        while(!found){
            shifted = n - 1;
            while(shifted >= 0 && d1[shifted] >= (n*n - n + shifted)) shifted--;
            //kalau indeks udh mentok, mundurin
            if(shifted < 0) found = true; // sudah mentok, tidak ada hasil
            else{
                d1[shifted]++;
                coordinate[shifted] = {d1[shifted] / n, d1[shifted] % n};
                color[shifted] = map[coordinate[shifted].row][coordinate[shifted].col];

                if(isValid(coordinate, color, n)){
                    *valid = true;
                    found = true;
                }

                int temp = shifted + 1;
                while(temp < n){
                    d1[temp] = d1[temp-1] + 1; // mulai dari queen [shifted-1], buat sebelahan semua sampai queen[n]
                    coordinate[temp] = {d1[temp]/n, d1[temp]%n};
                    color[temp] = map[coordinate[temp].row][coordinate[temp].col];
                    temp++;
                }
                steps++;
            }
            // ==== LIVE UPDATE ====
            if (steps % (2*n*n*n) == 0) printBoard(coordinate, n, n);
        }
    }
}
```

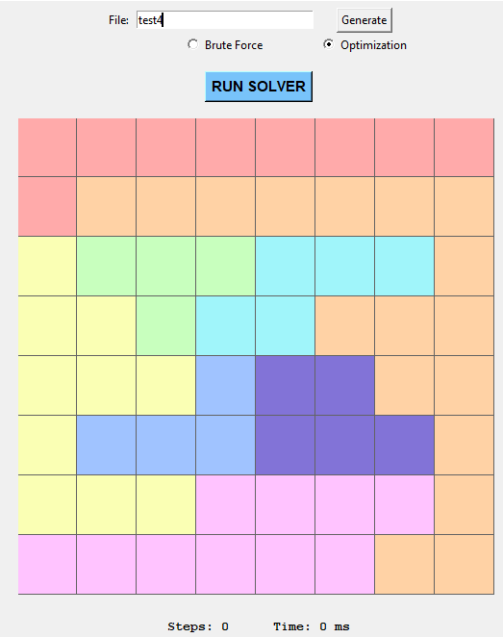
B. Brute Force dengan Optimasi

```
void optimized(Point* coordinate, char* color, int n, char** map, bool* valid){
    // Optimized: search by row O(n^n)
    int rowNow = 0, colNow = 0;

    for(int i = 0; i < n; i++){
        coordinate[i] = {i, 0};
        color[i] = map[i][0];
    }
    int shifted = n-1;

    if(isValid(coordinate, color, n)){
        *valid = true;
        steps++;
        return;
    }else{
        bool found = false;

        while(!found){
            shifted = n - 1;
            while(shifted >= 0 && coordinate[shifted].col == n-1) shifted--;
            //kalau indeks udh mentok, mundurin row yg digeser
            if(shifted < 0) found = true; // sudah mentok, tidak ada hasil
            else{
                coordinate[shifted].col++;
                color[shifted] = map[shifted][coordinate[shifted].col];

                if(isValid(coordinate, color, n)){
                    *valid = true;
                    found = true;
                }

                int temp = shifted + 1;
                while(temp < n){
                    coordinate[temp].col = 0; // mulai dari queen [shifted-1], mulai dari col = 0 semua.
                    color[temp] = map[temp][coordinate[temp].col];
                    temp++;
                }
                steps++;
            }
            // ==== LIVE UPDATE ====
            if (steps % (5*n*n+1) == 0) printBoard(coordinate, n, n);
        }
    }
}
```

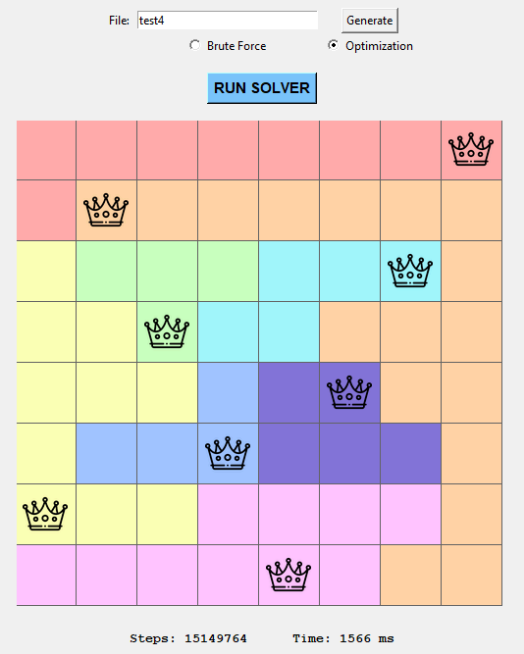
BAB III: HASIL

3.1 Test 1

1. Input

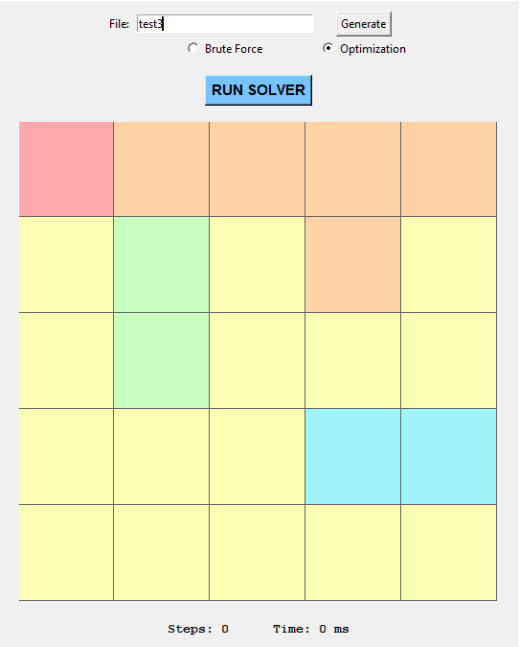


2. Output

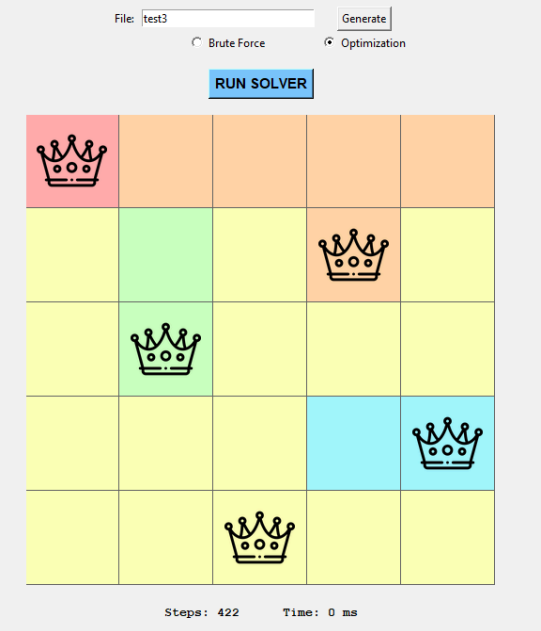


3.2 Test 2

1. Input

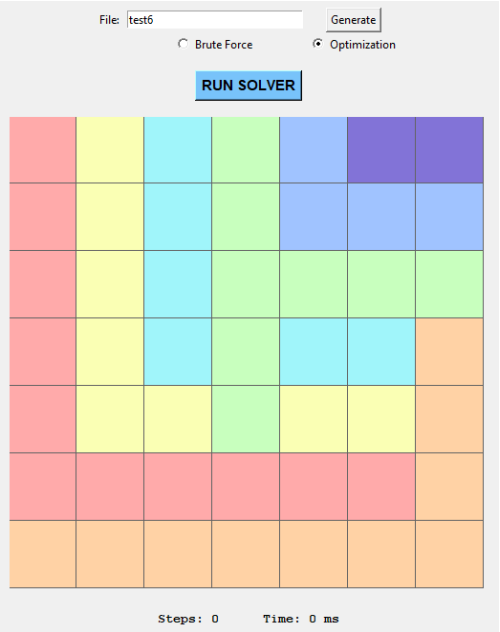


2. Output

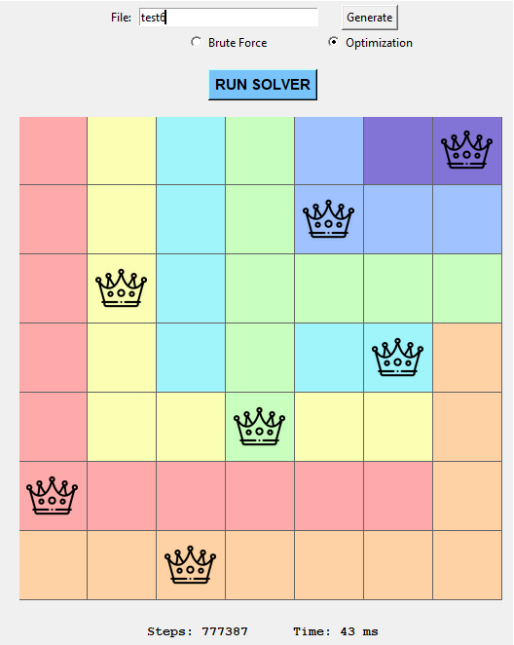


3.3 Test 3

1. Input

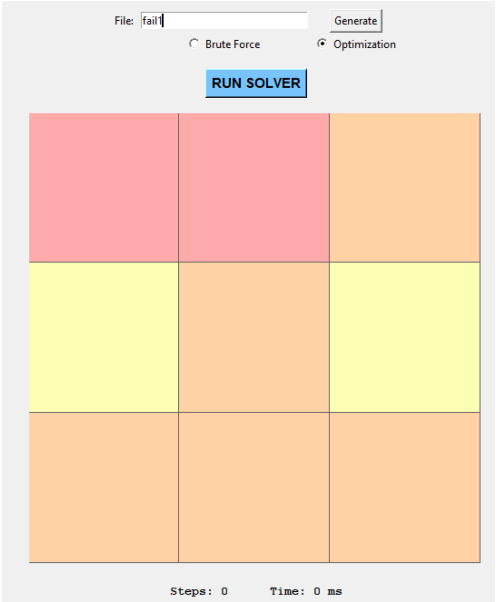


2. Output

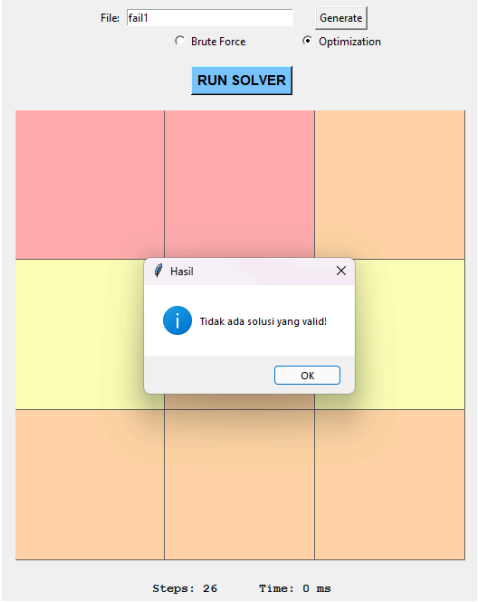


3.4 Test 4

1. Input

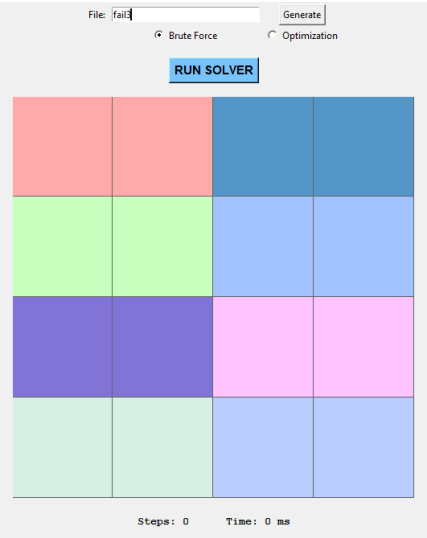

File: fail1
Generate
Brute Force
Optimization
RUN SOLVER
Steps: 0
Time: 0 ms

2. Output

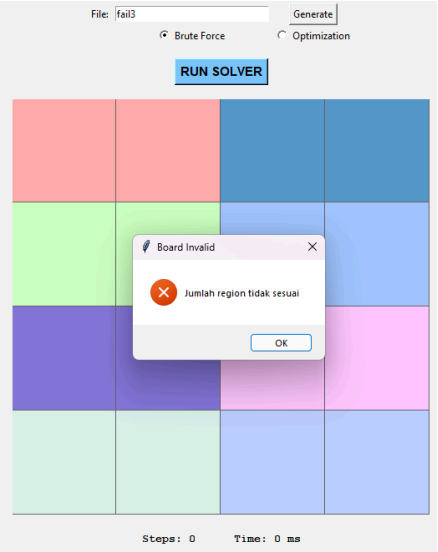

File: fail1
Generate
Brute Force
Optimization
RUN SOLVER
Hasil
Tidak ada solusi yang valid!
OK
Steps: 26
Time: 0 ms

3.5 Test 5

1. Input



2. Output



BAB IV:LAMPIRAN

Link Repo:

https://github.com/SyaqinaOctavia/Tucil1_13524088

Pernyataan:

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

Syaqina Octavia Rizha

13524088