Laporan Tugas Kecil 1 — N-Queens dengan Pewarnaan Menggunakan Algoritma Brute Force

Nashiruddin Akram (13524090)

17 Februari 2026

1 Penjelasan Algoritma

1.1 Deskripsi Masalah

Diberikan grid $N \times N$ dimana setiap sel memiliki warna (karakter A—Z). Tugas adalah menempatkan N queens dengan aturan:

- Satu queen per baris
- Satu queen per kolom
- Tidak ada dua queens dengan warna sama
- Tidak ada dua queens yang bersebelahan (horizontal/vertikal/diagonal)

1.2 Pemilihan Basis Brute Force

Saya melihat bahwa dalam kasus ini terdapat tiga basis yang dapat digunakan untuk melakukan brute force, yaitu berdasarkan baris, kolom, atau grid warna. Pada akhirnya saya memilih menggunakan baris sebagai acuan karena lebih terbiasa dengan pendekatan tersebut dan kompleksitas waktunya lebih mudah dianalisis.

Memang, menggunakan warna sebagai basis berpotensi lebih cepat pada kasus tertentu, namun pendekatan berbasis baris lebih stabil karena patokannya langsung bergantung pada nilai N (ukuran papan).

Jika menggunakan baris sebagai acuan dan menerapkan brute force murni tanpa optimasi, maka kompleksitas maksimalnya adalah $O(N^2)$ – Sebelum Pengecekan. Oleh karena itu, saya menambahkan sedikit optimasi dengan menggunakan array boolean `col[]` untuk melacak kolom mana saja yang sudah digunakan. Dengan cara ini, tidak perlu mencoba kolom yang sudah terpakai dan kompleksitasnya tetap $O(N!)$.

1.3 Penjelasan Langkah-Langkah Algoritma

Langkah 1: Validasi Awal

- Baca input grid $N \times N$
- Hitung jumlah warna unik di grid
- Jika jumlah warna $\neq N$, maka tidak ada solusi (return)

Langkah 2: Rekursif — `fungsi(x)`
Gunakan rekursi untuk mengisi baris x satu per satu. Untuk setiap baris:

- **Basis**: Jika $x = N$ (semua baris terisi), validasi dengan `cek()`
- **Rekursi**: Untuk setiap kolom i, jika kolom belum terpakai:
  - Tempatkan queen di (x, i)
  - Rekursi ke baris berikutnya `fungsi(x+1)`
  - Jika solusi ditemukan, hentikan proses
  - Jika tidak ditemukan, hapus queen dari (x, i) dan lanjutkan ke kolom berikutnya

Langkah 3: Validasi Constraint — `cek()`

- Untuk setiap queen di baris x dan kolom y, periksa semua queen di baris sebelumnya
- Jika ada yang memiliki warna sama atau bersebelahan $|x - i| \leq 1 \wedge |y - j| \leq 1$: return false
- Jika semua lolos: Maka itu solusinya, kemudian return true dan program pun selesai

Untuk selengkapnya, lihat kode program di bagian berikutnya.

2 Source Program

Listing 1: main.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
#include <chrono>

char grid[100][100];
long long n, iterasi, waktu;
bool ada[100][100], solusi, frek[26], col[100];
auto lastUpdate = chrono::steady_clock::now();
bool cek();

void printgrid(){
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            if (ada[i][j]) cout << "#"; else cout << grid[i][j];
        }
        cout << endl;
    }
}

void fungsi(int x){
    if (x == n){
        iterasi++;
        solusi = cek();
        if (solusi == true) { printgrid(); return; }
        auto now = chrono::steady_clock::now();
        if (chrono::duration_cast<chrono::milliseconds>
            (now - lastUpdate).count() >= 50) {
            lastUpdate = now;
            string line = "ITER:" + to_string(iterasi) + ":";
            for (int i = 0; i < n; i++){
                if (i > 0) line += "|";
                for (int j = 0; j < n; j++){
                    if (ada[i][j]) line += "#";
                    else line += grid[i][j];
                }
            }
            line += "\n";
            fwrite(line.c_str(), 1, line.size(), stderr);
            fflush(stderr);
        }
        return;
    }

    for (int i = 0; i < n; i++){
        if (solusi) return;
        if (col[i]) continue;

        ada[x][i] = true;
        col[i] = true;
        fungsi(x + 1);
        if (solusi) return;
```

```
            ada[x][i] = false;
            col[i] = false;
        }
    }

bool cek(){
    for (int x = 0; x < n; x++){
        int y = -1;
        for (int j = 0; j < n; j++){
            if (ada[x][j]){ y = j; break; }
        }
        char warna = grid[x][y];
        for (int i = 0; i < x; i++){
            for (int j = 0; j < n; j++){
                if (ada[i][j] &&
                    (warna == grid[i][j] ||
                     (abs(x-i) <= 1 && abs(y-j) <= 1))){
                    return false;
                }
            }
        }
    }
    return true;
}

int main(){
    string st;
    cin >> st;
    n = st.length();
    for (int i = 0; i < n; i++){
        grid[0][i] = st[i];
        ada[0][i] = false;
        col[i] = false;
    }
    for (int i = 1; i < n; i++){
        string s;
        cin >> s;
        for (int j = 0; j < n; j++){
            grid[i][j] = s[j];
            ada[i][j] = false;
        }
    }

    auto mulai = chrono::high_resolution_clock::now();

    memset(frek, 0, sizeof(frek));
    int cnt = 0;
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            if (frek[grid[i][j] - 'A'] == 0) {
                frek[grid[i][j] - 'A'] = 1;
                cnt++;
            }
        }
    }
    cout << endl;
```

```
    if (cnt != n){
        cout << "Tidak␣Ada␣Solusi␣Yang␣Memenuhi" << endl;
        return 0;
    }

    iterasi = 0;
    solusi = false;
    fungsi(0);

    if (!solusi){
        cout << "Tidak␣Ada␣Solusi␣Yang␣Memenuhi" << endl;
    }

    auto end = chrono::high_resolution_clock::now();
    waktu = chrono::duration_cast<chrono::milliseconds>
            (end - mulai).count();

    cout << "Waktu␣pencarian:␣" << waktu << "␣ms" << endl;
    cout << "Banyak␣kasus␣yang␣ditinjau:␣" << iterasi << "␣kasus" << endl;

    return 0;
}
```

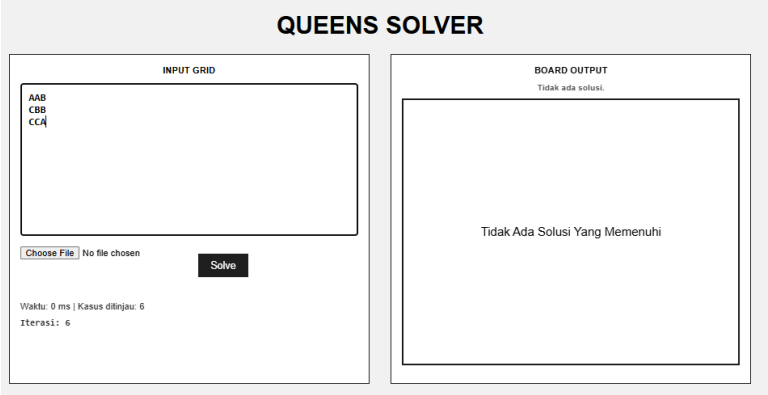
3 Test Case

3.1 Test Case 1 (3×3 Grid)

Input:

```
AAB
CBB
CCA
```

Output:

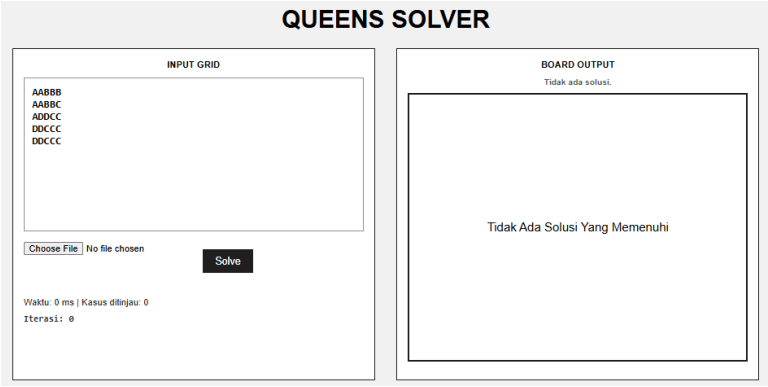


3.2 Test Case 2 (5×5 Grid)

Input:

```
AAABB
AAABB
CCCDD
CCCDD
CCCDD
```

Output:

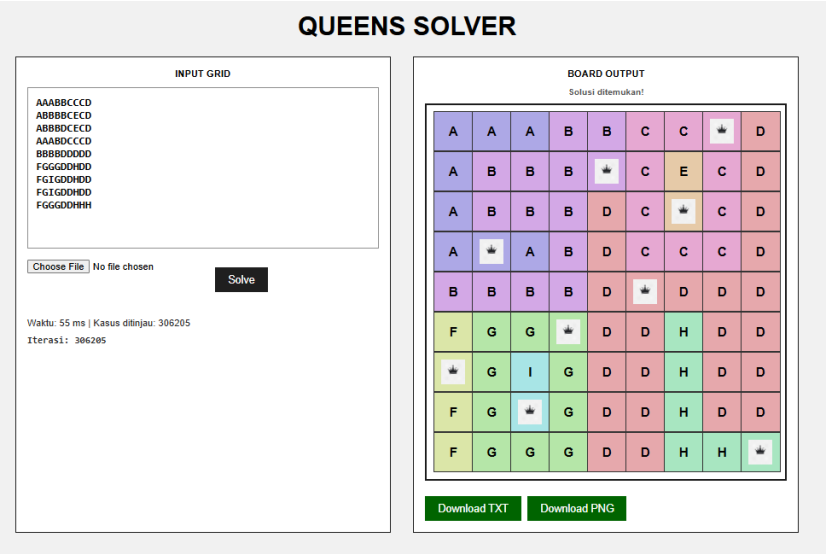


3.3 Test Case 3 (9×9 Grid)

Input:

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

Output:

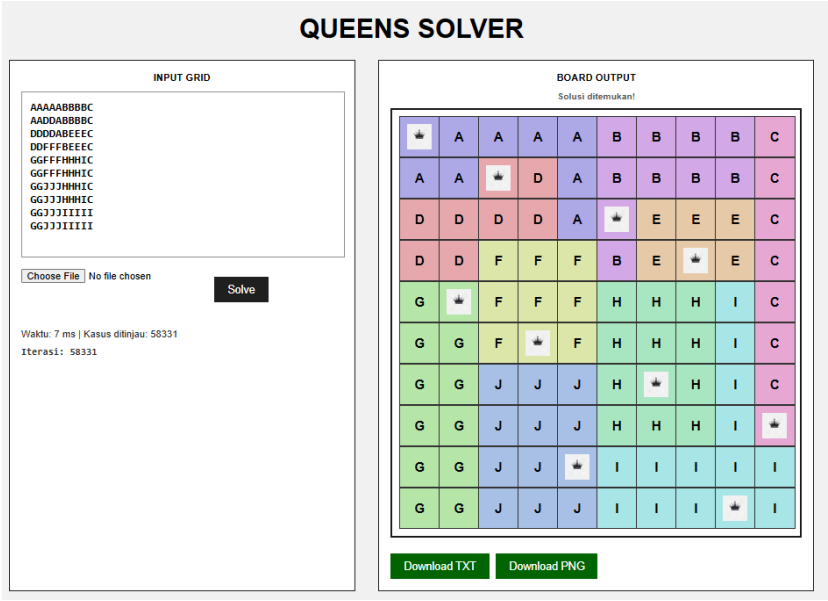


3.4 Test Case 4 (10×10 Grid)

Input:

```
AAAAAbBBBC
AADDAbBBBC
DDDDAbEEEC
DDFFFbEEEC
GGFFFhHHIC
GGFFFhHHIC
GGJJJhHHIC
GGJJJhHHIC
GGJJJIIIII
GGJJJIIIII
```

Output:

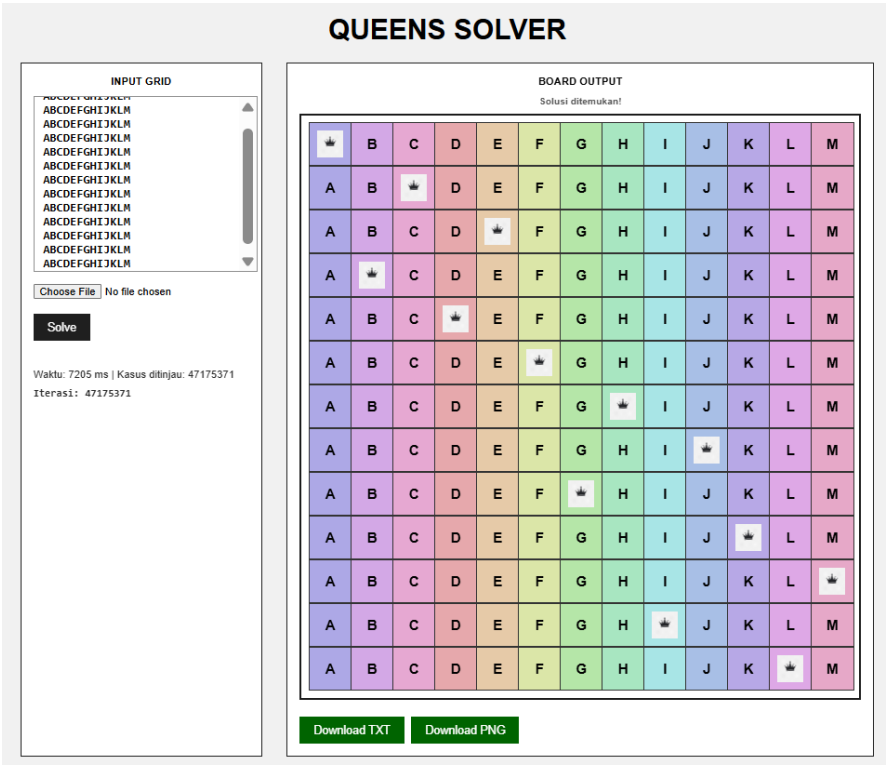


3.5 Test Case 5 (13×13 Grid)

Input:

```
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
```

Output:


QUEENS SOLVER
INPUT GRID
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
ABCDEFGHIJKLM
Choose File No file chosen
Solve
Waktu: 7205 ms | Kasus ditinjau: 47175371
Iterasi: 47175371
BOARD OUTPUT
Solusi ditemukan!
Download TXT
Download PNG

4 Repository GitHub

Kode program lengkap tersedia di:

Link: `https://github.com/Akram17t/Tucil1_13524090`

Repository berisi file-file berikut:

- `main.cpp` — Solver dengan algoritma Brute Force
- `server.js` — Backend Node.js (opsional)
- `index.html` — Frontend web (opsional)
- `README.md` — Dokumentasi lengkap

5 Pernyataan Tidak Melakukan Kecurangan

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

Nashiruddin Akram (13524090)

6 Lampiran

6.1 Checklist Penilaian

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	✓	
2	Program berhasil dijalankan	✓	
3	Solusi memenuhi aturan permainan N-Queens	✓	
4	Input/output berkas `.txt`	✓	
5	Graphical User Interface (GUI)	✓	
6	Dapat menyimpan visualisasi dalam format gambar	✓	

Tabel 1: Checklist Penilaian Tugas Kecil 1