Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II Tahun 2025/2026
Penyelesaian N- Queens Linkedin

Daniel Anindito Nugroho
13524002
K02

I. Pendahuluan

A. Latar Belakang

Permainan Queens pada platform LinkedIn merupakan pengembangan dari masalah klasik N-Queens yang sering dipelajari dalam materi strategi algoritma. Pada permainan ini, pemain harus menempatkan Ratu (Queen) pada papan berukuran $N \times N$ dengan beberapa aturan:

1. Setiap baris hanya boleh berisi satu Queen
2. Setiap kolom hanya boleh berisi satu Queen
3. Setiap daerah warna (karakter/alphabet) hanya boleh berisi satu Queen
4. Queen tidak boleh saling bersebelahan, termasuk secara diagonal (tidak boleh berjarak horizontal/vertikal ≤ 1 dalam 8 arah terdekat)

Berbeda dengan N-Queens klasik, variasi LinkedIn ini menambahkan dimensi baru: constraint warna. Setiap sel pada papan memiliki warna unik yang ditandai dengan karakter alphabet (A-Z), dan setiap warna hanya boleh ditempati oleh satu Queen. Hal ini membuat problem menjadi lebih kompleks dan berkurang jumlah solusi validnya.

Pada tugas ini digunakan algoritma Brute Force Iteratif untuk mencari solusi. Pendekatan ini dilakukan dengan mencoba semua kemungkinan konfigurasi secara sistematis dengan memanfaatkan teknik increment dan decrement untuk menghasilkan variasi posisi Queen yang berbeda-beda. Meskipun tidak seefisien backtracking dengan pruning, algoritma ini pasti akan menemukan satu solusi jika ada.

B. Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam tugas ini adalah:

1. Bagaimana merancang algoritma brute force iteratif untuk menemukan solusi permainan Queens yang memenuhi keseluruhan constraint sekaligus?
2. Bagaimana memastikan bahwa konfigurasi yang dihasilkan valid terhadap seluruh aturan constraint yang kompleks?
3. Bagaimana menangani error input dengan validasi yang komprehensif mencakup format, ukuran, karakter, dan keunikan alfabet/warna?
4. Bagaimana menghitung jumlah konfigurasi yang ditinjau selama proses pencarian?
5. Bagaimana mengukur dan menampilkan waktu eksekusi algoritma untuk berbagai ukuran papan?

C. Tujuan

Tujuan dari pelaksanaan tugas ini adalah :

1. Mengimplementasikan algoritma brute force untuk menyelesaikan permainan Queens sesuai dengan spesifikasi yang diberikan.
2. Menghasilkan solusi yang valid atau menyatakan bahwa tidak terdapat solusi.
3. Menghitung dan menampilkan jumlah iterasi yang diperlukan untuk mencari suatu solusi.
4. Mengukur waktu eksekusi algoritma brute force pencarian solusi.
5. Menyimpan solusi hasil algoritma brute force ke file output.

II. Algoritma Brute Force

A. Ide Dasar Algoritma

Algoritma brute force iteratif untuk menyelesaikan N-Queens Problem dengan constraint warna bekerja dengan asumsi fundamental bahwa setiap baris harus berisi tepat satu Queen. Oleh karena itu, struktur data yang digunakan adalah array satu dimensi queens[N], di mana queens[i] menyimpan nomor kolom tempat Queen di baris i berada. Dengan asumsi ini, constraint pertama (setiap baris tepat satu Queen) sudah terpenuhi secara otomatis oleh struktur data, sehingga program hanya perlu memvalidasi tiga constraint lainnya: kolom, diagonal/adjacency, dan warna.

B. Pseudocode

```
ALGORITMA Brute Force
        loadBoard()
        N = ukuran papan
        board[][] = isi papan dari file

        for i = 0 to N-1
           queens[i] = 0
        endfor

        iterations = 0
        found = false
        start_time = current_time()

        while true
           iterations++

           if iterations mod 1000000 = 0 then
              clearScreen()
              display("Iterasi ke-" + iterations)
              printBoard()
           endif

           valid = true
           for row = 0 to N-1
              if NOT isValid(row, queens[row]) then
                 valid = false
                 break
              endif
           endfor

           if valid then
              found = true
              break
           endif

           row = N-1
           while row >= 0
              queens[row]++
```

```
            if queens[row] < N then
                break
            else
                queens[row] = 0
                row--
            endif
        endwhile

        if row < 0 then
            break
        endif
    endwhile

    end_time = current_time()
    duration = end_time - start_time

    display("Iterasi: " + iterations)
    display("Waktu: " + duration + " ms")

    if found then
        printBoard()
        display("Solusi ditemukan!")
        saveSolution()
    else
        display("Tidak ada solusi")
    endif
END


FUNGSI isValid(row, col) -> boolean
Input: row (baris), col (kolom)
Output: true jika posisi aman, false jika tidak

    for otherRow = 0 to N-1
        if otherRow != row then
            otherCol = queens[otherRow]

            if otherCol = col then
                return false
            endif

            rowDiff = |row - otherRow|
            colDiff = |col - otherCol|

            if rowDiff <= 1 AND colDiff <= 1 then
                return false
            endif
```

```
            if board[row][col] = board[otherRow][otherCol] then
               return false
            endif
         endif
      endfor

      return true
   END
```

C. Langkah-langkah Algoritma

Proses pencarian solusi dimulai dengan tahap inisialisasi di mana array queens[N] dimana n adalah ukuran panjang atau lebar suatu board yang kemudian diatur ke nilai nol untuk semua elemen guna menciptakan konfigurasi awal yang akan divalidasi. Algoritma kemudian melakukan iterasi sistematis untuk memeriksa setiap konfigurasi yang ada. Jika sebuah konfigurasi memenuhi semua aturan maka solusi dinyatakan ditemukan, namun jika tidak valid maka program akan beralih ke konfigurasi berikutnya. Mekanisme transisi konfigurasi ini bekerja menyerupai sistem bilangan basis N yang dimulai dengan menaikkan posisi kolom queen pada baris terakhir. Apabila posisi tersebut melebihi batas maka nilai akan dikembalikan ke nol dan kenaikan posisi akan berpindah ke baris di atasnya secara berulang. Dalam setiap tahap, konfigurasi divalidasi berdasarkan tiga aturan utama yaitu tidak boleh ada dua queen di kolom yang sama, tidak boleh ada queen yang berdekatan baik secara diagonal maupun horizontal dan vertikal, serta tidak boleh ada dua queen yang menempati petak dengan warna atau karakter yang sama.

D. Kompleksitas Waktu

Dalam algoritma ini, setiap Queen di baris sendiri (asumsi one queen per row). Setiap Queen dapat berada di N kolom berbeda. Oleh karena itu, total possible configuration N^N, dan melalui validasi per konfigurasi melalui loop N baris dan masing-masing baris membandingkan dengan (N-1) baris lainnya sehingga total time complexity = O(N^N * N^2).

III. Implementasi Program

A. Struktur Folder Repository

```
├── docs
│   └── LaporanStima_Tucil1.pdf
├── test
│   ├── tc1.txt
│   ├── tc2.txt
│   ├── tc3.txt
│   ├── tc4.txt
│   ├── tc5.txt
│   ├── solusitc1.txt
│   ├── solusitc2.txt
│   ├── solusitc3.txt
```

```
│   ├── solusitc4.txt
│   └── solusitc5.txt
├── src
│   ├── queens.cpp
├── bin
│   └── main.exe
└── README.md
```

Struktur direktori ini menunjukkan susunan file untuk sebuah proyek pemrograman yang terbagi menjadi beberapa folder utama. Folder docs berisi file dokumentasi laporan tugas kecil bernama LaporanStima_Tucil1.pdf. Folder test menyimpan sepuluh file yang terdiri dari lima file kasus uji bernama tc1.txt hingga tc5.txt dan lima file solusi yang sesuai bernama solusitc1.txt hingga solusitc5.txt. Folder src memuat kode sumber program dalam file queens.cpp, sedangkan folder bin berisi file eksekusi program bernama main.exe. Terakhir, terdapat file README.md yang terletak langsung di direktori utama proyek tersebut.

B. Source Code

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <chrono>
#include <cstdlib>
#include <limits>
#include <set>

using namespace std;

int N = 0;
const int maxN = 26;
char board[maxN][maxN];
int queens[maxN];
bool solution = false;

bool sizeValidation(){
    set<char> uniqueChars;
    for (int i = 0; i < N; i++){
        for (int j = 0; j < N; j++){
            uniqueChars.insert(board[i][j]);
        }
    }

    if ((int)uniqueChars.size() != N){
```

```
        cout << "Jumlah karakter/warna pada papan tidak sesuai dengan
ukuran papan." << endl;
        return false;
    }
    return true;
}

void printBoard(ostream& out){
    for (int i = 0; i <= N-1; i++){
        for (int j = 0; j <= N-1; j++){
            if (queens[i] == j){
                out << "# ";
            } else {
                out << board[i][j] << " ";
            }
        }
        out << endl;
    }
}

void loadBoard(){
    N = 0;
    ifstream file;
    string filename;
    bool validFilename = false;

    while (!validFilename){
        cout << "Masukkan nama file input atau path (misal case.txt
atau testcase/tc.txt): ";
        getline(cin, filename);

        if (!filename.empty() && filename.front() == '"')
filename.erase(0, 1);
        if (!filename.empty() && filename.back() == '"')
filename.pop_back();

        if (filename.empty()){
            cout << "Nama file tidak boleh kosong." << endl;
            continue;
        }

        string txtExt = ".txt";
        if (filename.size() < txtExt.size() ||
```

```
            filename.substr(filename.size() - txtExt.size()) !=
txtExt){
            cout << "Format file tidak valid. Hanya file .txt yang
diterima." << endl;
            continue;
        }

        validFilename = true;
    }

    file.open(filename);
    while (!file.is_open()){
        cout << "File tidak ditemukan. Masukkan nama file yang benar:
";
        getline(cin, filename);
        if (!filename.empty() && filename.front() == '"')
filename.erase(0, 1);
        if (!filename.empty() && filename.back() == '"')
filename.pop_back();
        file.clear();
        file.open(filename);
    }
    cout << "File " << filename << " berhasil dibuka." << endl;

    string line;
    int row = 0;

    while (getline(file,line)){
        while (!line.empty() && isspace((unsigned char)line.back())){
            line.pop_back();
        }
        if (!line.empty() && line[line.length()-1] == '\r'){
            line.erase(line.length()-1);
        }
        if (line.empty()){
            continue;
        }
        if (N == 0){
            N = line.length();
            if (N > maxN){
                cout << "Ukuran papan melebihi batas maksimal." <<
endl;
                exit(1);
```

```
            }
        }
        if ((int)line.length()!=N){
            cout << "Format input tidak valid." << endl;
            exit(1);
        }
        for (int col = 0; col < N; col++){
            char currentChar = line[col];

            if (currentChar == ' ' || currentChar == '\t'){
                cout << "Error: Format papan cacat!" << endl;
                exit(1);
            }


            if (currentChar < 'A' || currentChar > 'Z'){
                cout << "Error: Karakter tidak valid ditemukan!" <<
endl;
                exit(1);
            }

            board[row][col] = currentChar;
        }
        row++;
    }
    file.close();

    if (row == 0){
        cout << "File kosong!" << endl;
        exit(1);
    }
    if (row != N){
        cout << "Board tidak berbentuk persegi (n x n)!" << endl;
        exit(1);
    }
    if (!sizeValidation()){
        exit(1);
    }
    cout << "Board " << N << "x" << N << " berhasil dimuat." << endl;
}

void clearScreen() {
    #ifdef _WIN32
```

```
        system("cls");
    #else
        system("clear");
    #endif
}

bool isValid(int row, int col){
    for (int otherRow = 0; otherRow < N; otherRow++){
        if (otherRow == row){
            continue;
        }
        int otherCol = queens[otherRow];
        if (otherCol == col){
            return false;
        }
        int rowDiff = row - otherRow;
        int colDiff = col - otherCol;

        if (rowDiff < 0){
            rowDiff = -1 * rowDiff;
        }
        if (colDiff < 0){
            colDiff = -1 *colDiff;
        }
        if (rowDiff<=1 && colDiff <=1){
            return false;
        }
        if (board[row][col] == board[otherRow][otherCol]){
            return false;
        }
    }
    return true;
}

void saveSolution(long long iteration,double duration){
    string response;
    cout << "Apakah Anda ingin menyimpan solusi? (y/n): ";
    cin >> response;
    if (response == "y" || response == "Y"){
        string filename;
        bool validFilename = false;

        while (!validFilename){
```

```
            cout << "Masukkan nama file/path untuk menyimpan file
output (misal hasil.txt): ";
            cin >> filename;

            if (filename.empty()){
                cout << "Nama file tidak boleh kosong." << endl;
                continue;
            }

            string txtExt = ".txt";
            if (filename.size() < txtExt.size() ||
                filename.substr(filename.size() - txtExt.size()) !=
txtExt){
                cout << "Format file tidak valid. Hanya file .txt yang
diterima." << endl;
                continue;
            }

            validFilename = true;
        }

        ofstream file(filename);

        if (!file.is_open()){
            cout << "Gagal membuka file untuk menyimpan solusi." <<
endl;
            return;
        }else{
            printBoard(file);
            file << "\nIterasi: " << iteration << endl;
            file << "Waktu eksekusi: " << duration << " ms" << endl;
            file.close();
            cout << "Solusi berhasil disimpan ke " << filename << endl;
        }
    }
}

int main() {
    loadBoard();
    for (int i = 0; i < N; i++){
        queens[i] = 0;
    }
    long long iterations = 0;
```

```
    bool found = false;

    cout << "Mencari solusi..." << endl;
    auto start = chrono::high_resolution_clock::now();

    while (true){
        iterations++;

        if (iterations % 1000000==0){
            clearScreen();
            cout << "Iterasi ke-" << iterations << endl;
            printBoard(cout);
        }
        bool valid = true;
        for (int i=0; i < N; i++){
            if(isValid(i, queens[i]) == false){
                valid = false;
                break;
            }
        }

        if (valid){
            found = true;
            break;
        }

        int row = N-1;
        while (row>=0){
            queens[row]++;
            if (queens[row] < N){
                break;
            }else{
                queens[row] = 0;
                  row--;
            }
        }
        if (row < 0){
            break;
        }
    }

    auto end = chrono::high_resolution_clock::now();
    chrono::duration<double, milli> duration = end - start;
```

```
    cout << "\nIterasi: " << iterations << endl;
    cout << "Waktu eksekusi: " << duration.count() << " ms" << endl;

    cout << "HASIL AKHIR:" << endl;
    if (found){
        printBoard(cout);
        cout << "\nSolusi ditemukan!" << endl;
        saveSolution(iterations, duration.count());
    } else {
        cout << "\nTidak ada solusi yang ditemukan." << endl;
    }
}
```

Fungsi sizeValidation bertugas untuk memvalidasi apakah jumlah karakter unik atau warna yang ada pada papan permainan sesuai dengan ukuran papan N. Fungsi ini menggunakan struktur data set untuk menyimpan setiap karakter unik yang ditemukan saat menelusuri seluruh papan. Setelah itu, fungsi membandingkan ukuran set tersebut dengan nilai N. Jika jumlahnya tidak sama, fungsi akan mencetak pesan kesalahan dan mengembalikan nilai false, yang menandakan bahwa papan tidak valid untuk diproses lebih lanjut.

Fungsi printBoard bertanggung jawab untuk menampilkan status papan permainan ke output stream yang diberikan, baik itu layar konsol maupun file teks. Fungsi ini melakukan iterasi baris demi baris dan kolom demi kolom. Jika posisi kolom pada baris tersebut sesuai dengan posisi ratu yang tersimpan dalam array queens, maka program akan mencetak simbol pagar. Namun jika tidak ada ratu di posisi tersebut, program akan mencetak karakter asli dari papan yang merepresentasikan warna atau wilayah petak tersebut.

Fungsi loadBoard menangani proses pembacaan data awal dari file eksternal. Fungsi ini meminta pengguna memasukkan nama file dan melakukan validasi untuk memastikan nama file tidak kosong dan berakhiran .txt. Setelah file berhasil dibuka, fungsi membaca setiap baris untuk menentukan ukuran papan N dan mengisi array board dengan karakter yang ada. Fungsi ini juga melakukan serangkaian pemeriksaan keamanan, seperti memastikan file tidak kosong, papan berbentuk persegi sempurna, karakter yang valid hanya huruf kapital, serta memanggil fungsi sizeValidation untuk verifikasi akhir.

Fungsi clearScreen adalah fungsi utilitas sederhana yang digunakan untuk membersihkan tampilan layar konsol. Fungsi ini mendeteksi sistem operasi yang digunakan saat kompilasi. Jika program dijalankan di Windows, ia akan memanggil perintah sistem cls, sedangkan jika dijalankan di sistem operasi lain seperti Linux atau macOS, ia akan memanggil perintah clear. Hal ini berguna untuk memberikan efek visual yang lebih rapi saat program menampilkan progres iterasi.

Fungsi isValid berfungsi untuk memeriksa apakah posisi ratu pada baris dan kolom tertentu melanggar aturan permainan atau tidak. Fungsi ini membandingkan posisi ratu saat ini dengan posisi ratu di baris-baris lainnya. Terdapat tiga kondisi pelarangan yang diperiksa, yaitu ratu tidak boleh berada di kolom yang sama dengan ratu lain, ratu tidak boleh berada di petak yang berdekatan atau bersentuhan langsung dengan ratu lain baik secara vertikal, horizontal maupun diagonal, dan ratu tidak boleh menempati petak dengan karakter atau warna yang sama dengan petak ratu lainnya.

Fungsi saveSolution memberikan opsi kepada pengguna untuk menyimpan hasil solusi ke dalam file teks. Fungsi ini menerima parameter jumlah iterasi dan durasi waktu eksekusi. Jika pengguna memilih untuk menyimpan, fungsi akan meminta nama file output, memvalidasinya, dan kemudian menuliskan konfigurasi papan terakhir beserta statistik kinerja program ke dalam file tersebut menggunakan format yang sama dengan fungsi printBoard.

Fungsi main merupakan program utama yang mengatur alur eksekusi keseluruhan. Fungsi ini memulai dengan memuat papan, menginisialisasi posisi ratu, dan mencatat waktu awal eksekusi. Algoritma utamanya menggunakan pendekatan brute force iteratif di dalam loop while, di mana posisi ratu digeser terus-menerus layaknya sistem bilangan untuk mencoba setiap kemungkinan kombinasi. Program akan menampilkan progres setiap satu juta iterasi. Jika konfigurasi yang valid ditemukan, loop akan berhenti, menampilkan hasil akhir dan waktu tempuh, serta menawarkan penyimpanan solusi. Jika semua kemungkinan habis dicoba tanpa hasil, program akan memberitahu bahwa tidak ada solusi.

IV. Hasil Pengujian

A. Test Case 1

Test case merupakan test case valid 9x9

Soal:

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

Solusi:

```
A A A B B C C # D
A B B B # C E C D
A B B B D C # C D
A # A B D C C C D
B B B B D # D D D
F G G # D D H D D
# G I G D D H D D
F G # G D D H D D
F G G G D D H H #

Iterasi: 323741637
Waktu eksekusi: 48810.7 ms
```

B. Test Case 2

Test case 9x9 error dengan satu baris memiliki lebih dari batas maksimal karakter (10).

Soal:

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

Solusi:

```
Masukkan nama file input (misal case.txt): ../testcase/tc2.txt
File ../testcase/tc2.txt berhasil dibuka.
Format input tidak valid.
```

C. Test Case 3

Test case 9x9 error dengan warna/alfabet melebihi batas maksimal board.

Soal:

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

Solusi:

```
Masukkan nama file input atau path (misal case.txt atau testcase/tc.txt): ../testcase/tc3.txt
File ../testcase/tc3.txt berhasil dibuka.
Jumlah karakter/warna pada papan tidak sesuai dengan ukuran papan.
```

D. Test Case 4

Test Case 10x9 error karena tidak NxN / berbentuk persegi.

Soal:

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

Solusi :

```
Masukkan nama file input atau path (misal case.txt atau testcase/tc.txt): ../testcase/tc4.txt
File ../testcase/tc4.txt berhasil dibuka.
Board tidak berbentuk persegi (n x n)!
```

E. Test Case 5

Test case 9x9 error berisi white space di tengah-tengah.

Soal:

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

Solusi:

```
Masukkan nama file input atau path (misal case.txt atau testcase/tc.txt): ../testcase/tc5.txt
File ../testcase/tc5.txt berhasil dibuka.
Error: Format papan cacat!
```

V. Analisis

Program ini menggunakan metode brute force yang mencoba setiap kemungkinan posisi ratu satu per satu layaknya sistem penghitungan angka. Algoritma ini baru memeriksa aturan permainan seperti larangan saling bersentuhan atau kesamaan wilayah warna setelah semua ratu ditempatkan di papan. Pendekatan ini menjadikan logika program mudah dipahami namun mengakibatkan kinerja yang lambat karena komputer harus memproses jutaan kombinasi yang sebenarnya sudah salah sejak awal. Meskipun demikian struktur penulisan kode sudah sangat rapi karena memisahkan proses input validasi dan logika inti ke dalam fungsi yang berbeda.

VI. Kesimpulan

Program ini telah berhasil menerapkan aturan permainan N-Queens variasi warna dan ketetanggaan dengan benar serta memiliki fitur pendukung yang baik seperti pembacaan file dan pencatatan waktu. Kode ini sangat cocok untuk pembelajaran logika dasar namun kurang efektif untuk menyelesaikan papan berukuran besar karena metode pencariannya yang memeriksa semua kemungkinan tanpa terkecuali. Agar program dapat berjalan lebih cepat dan efisien disarankan untuk mengubah algoritma menjadi metode backtracking yang dapat memangkas langkah yang salah lebih dini.

VII. Pranala Repository

https://github.com/le1n-gif/Tucil1_13524002

VIII. Pernyataan

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

Daniel Anindito Nugroho

IX. 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		✓