LAPORAN TUGAS KECIL 1

IF 2211 Strategi Algoritma

Penyelesaian Permainan Queens Linkedin

18 Februari 2026

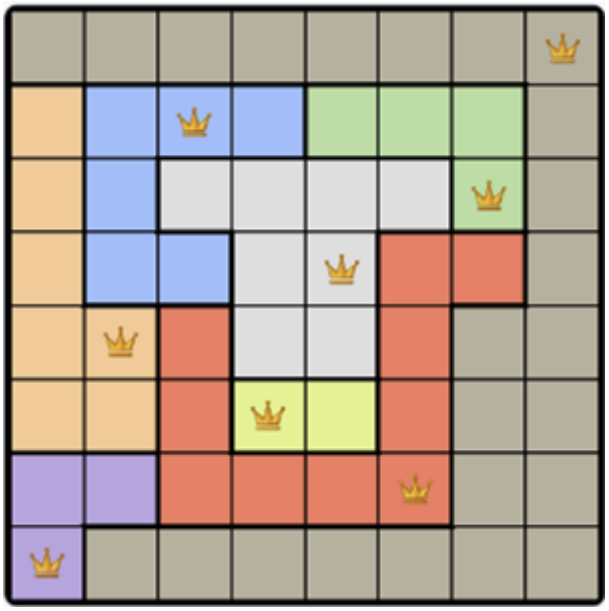
D
I
S
U
S
U
N

O
L
E
H

Mahatma Brahmana / 13524015

Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung

Deskripsi Tugas



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.

Tugas kami adalah membuat program yang 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.

Berikut adalah contoh file .txt yang akan dijadikan sebagai input.

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

Tiap karakter akan merepresentasikan warna yang berbeda-beda.

Penjelasan Kode

Pengerjaan tugas ini murni menggunakan algoritma brute force dan menggunakan bahasa pemogramanm C++.

Mula-mula program akan membaca array berukuran n x n dari sebuah file yang berisi kumpulan karakter.Tujuan program adalah menempatkan tepat n queen sehingga:

1. Tepat 1 queen di setiap baris dan kolom
2. Tepat 1 queen di setiap warna
3. Queen tidak boleh bersebelahan, termasuk bersebelahan secara diagonal

Program yang saya kerjakan menyediakan 2 metode:

1. Brute force murni
2. Optimisasi (Backtracking)

Program dimulai dengan :

Pembacaan file input

Fungsi :

```
bool salin(const string& filename, vector<vector<char>>& array)
```

Fungsi ini akan membaca isi dari file yang dituju, dan akan disimpan ke array. Di fungsi ini, akan dilakukan validasi sebagai berikut :

1. File tidak kosong
2. Berbentuk persegi (jumlah kolom harus sama dengan jumlah baris)
3. Jumlah warna sama dengan n
4. Hanya berisi alphabet, spasi juga tidak diperbolehkan.

Setelah valid, isi array akan dicetak ke layar

Penyelesaian Brute Force

Untuk penyelesaian Brute Force, solusinya (queen) direpresentasikan sebagai:

```
vector<int> col(n, 0);
```

Artinya, `col[i] = j,` dimana pada baris ke-i, queen ditempatkan di kolom-j. Karena tiap baris hanya punya tepat 1 nilai kolom (queen), maka aturan 1 queen di setiap baris dan kolom terpenuhi.

Selanjutnya ada fungsi:

```
bool nextCombination(vector<int>& col, int n)
```

Fungsi ini akan menghasilkan semua kemungkinan untuk posisi queen. Cara kerjanya adalah:

1. `col[0]` adalah basis awal lalu dilanjutkan dengan `col[1]` dan seterusnya
2. Saat fungsi dipanggil, tambah `col[0]`. Jika belum mencapai n-1, tambah dan fungsi selesai.

3. Kalau saat `col[0]` sudah mencapai n-1, reset nilai `col[0]` ,menjadi 0 dan tambah 1 nilai dari `col[0+1]`.
4. Jika semua nilai sudah menjadi 0 kembali, dan nilai dari `col[n+1]` sudah diluar array, maka semua kombinasi sudah dicoba, dan `return false`

Dengan menggunakan fungsi ini, total konfigurasi yang diuji adalah n^n.

Selanjutnya ada fungsi:

```
bool isValid(const vector<int>& col, const
vector<vector<char>>& array)
```

Fungsi ini akan memeriksa apakah posisi semua queen memenuhi seluruh aturan. Aturan yang dicek:

1. Kolom tidak boleh sama
   Dipakai `vector<bool> queenCol(n,false)` sebagai penanda kolom terisi.
   Dilakukan iterasi i dari 0 sampai n, lalu cek `queenCol[col[i]]`.Jika ada queenCol yang `true`, maka `return false`.
2. Warna tidak boleh sama
   Dipakai `unordered_set<char>` colours untuk menyimpan warna yang sudah dipakai.Jika warna sudah ada di set maka `return false`.
3. Tidak boleh bersebelahan secara diagonal
   Cek baris tetangga menggunakan `if (i > 0 && abs(col[i] - col[i - 1]) == 1) return false;`

Selanjutnya ada fungsi:

```
void solutionBruteForce(vector<vector<char>> array)
```

Fungsi ini adalah fungsi utama Brute Force. Alur kerjanya adalah:

1. Inisialisasi col = [0, 0, 0, …] dan count = 0 (untuk menghitung total konfigurasi yang diuji)
2. Dilakukan loop, dan disetiap iterasi:
   - `++count`
   - cek `isValid(col,array)`
   - jika valid maka cetak hasil dan selesai (`return`)
   - jika belum valid, lanjutkan ke iterasi berikutnya dengan `nextCombination(col,n)`.

   Di setiap iterasi kelipatan 100, akan ditampilkan hasil progress ke CLI.
3. Jika semua iterasi habis, atau saat `nextCombination(col,n)` menghasilkan `false`, tampilkan “Tidak ada solusi.
4. Menampilkan jumlah iterasi yang diuji dan waktu eksekusi program

Jika ditemukan solusi, `printResult` akan mencetak hasilnya, dengan cara posisi queen akan dicetak sebagai X. Setelah itu, pengguna diberikan pilihan untuk menyimpan hasil ke folder result atau tidak.

Penyelesaian Optimisasi (Backtracking)

Program ini awalnya saya kerjakan mengikuti cara pengerjaan ‘sudoku’ yang ada di PPT. Namun, karena ini adalah tambahan, maka saya tidak akan menjelaskannya lebih dalam. Ide dari program ini adalah, kita akan mulai dengan menempatkan queen pertama di baris pertama dan kolom pertama.

Kita cek, apakah posisi queen tersebut valid atau tidak. Disini akan otomatis menjadi valid karena hanya ada 1 queen, maka kita akan lanjut ke baris selanjutnya (kolomnya dimulai dari pertama).

Disini kita cek ulang, apakah posisi queen tersebut valid atau tidak, jika iya, maka kita lanjut ke baris selanjutnya (kolomnya dimulai dari pertama). Jika tidak valid, maka kita akan memajukan kolomnya sampai dia valid. Jika kolomnya sudah habis, maka kembali ke baris sebelumnya (backtracking) dan memajukan kolom posisi queen, dan begitu seterusnya.

Solusi dianggap tidak ada jika posisi kolom di baris pertama sudah melewati batas. Karena jika kolom di baris selain baris pertama melewati batas, maka dia akan melakukan backtracking ke baris selanjutnya, tetapi untuk kasus baris pertama, tidak ada baris selanjutnya, maka solusinya tidak ada.

SOURCE CODE

main

```
#include <fstream>
#include <iostream>
#include <filesystem>
#include <vector>
#include <algorithm>
#include <unordered_set>
#include <cctype>
#include <chrono>
using namespace std;

bool salin(const string& filename, vector<vector<char>>& array){
    filesystem::path p = filesystem::current_path() / "test" /
(filename + ".txt");

    ifstream in(p);
    if (!in) {
        cout << "Tidak ada file bernama " << p << endl;
        return false;
    }
```

```
    string line;
    unordered_set<char> colours;
    while (getline(in, line)) {
        if (line.empty()) {
            continue;
        }
        array.emplace_back(line.begin(), line.end());
        for (int i = 0; i < line.size(); ++i) {
            if (!isalpha(static_cast<unsigned char>(line[i]))) {
                cout << "Karakter tidak valid (harus alfabet): " <<
line[i] << endl;
                return false;
            }
            colours.insert(line[i]);
        }
    }

    if (array.empty()) {
        cout << "File kosong." << endl;
        return false;
    }

    if (array[1].size() != array.size()) {
        cout << "Isi file tidak berbentuk persegi.\n";
        return false;
    }

    if (colours.size() != array.size()) {
        cout << "Jumlah warna/alphabet harus tepat," << endl << "Besar
array :  " << array.size() << endl << "Jumlah warna/alphabet :  " <<
colours.size() << "." << endl;
        return false;
    }

    for (const auto& row : array) {
        for (char ch : row){
            cout << ch;
        }
        cout << endl;
    }
    cout << endl;
    return true;
```

```
}

void solutionOptimization(vector<vector<char>> array){
    int n = array.size();
    int i = 0 , j = 0;

    vector<char> colour;
    vector<vector<bool>> queen(n, vector<bool>(n, false));
    int countqueen = 0;

    vector<int> rowposition(n, 0);

    int count = 0;
    auto start = chrono::high_resolution_clock::now();

    while ((countqueen < n) && (i < n) ){
        if (rowposition[0] >= n){
            cout<<"Tidak ada solusi\n";
            return;
        }else if (rowposition[i] >= n){
            rowposition[i] = 0;
            i -= 1;
            queen[i][rowposition[i]] = false;

            colour.pop_back();
            rowposition[i] += 1;
            countqueen -= 1;

            cout << endl << "Kembali ke baris " << i << ", coba posisi
berikutnya: " << rowposition[i] << endl;
            count++;
            continue;
        }
        while (rowposition[i] < n) {
            bool conflict = false;

            for (int r = 0; r < i; ++r) {
                if (rowposition[r] == rowposition[i]) {
                    conflict = true;
                    break;
                }
            }
            if (conflict){
```

```
                rowposition[i]++;
                count++;
                continue;
            }

            for (char used : colour) {
                if (used == array[i][rowposition[i]]) {
                    conflict = true;
                    break;
                }
            }
            if (conflict){
                rowposition[i]++;
                count++;
                continue;
            }

            if (i != 0){
                if (abs(rowposition[i] - rowposition[i-1]) == 1){
                    conflict = true;
                }
                // if (rowposition[i] == 0){
                //     if (queen[i-1][rowposition[i] + 1] == true){
                //         conflict = true;
                //     }
                // }
                // else if (rowposition[i] == n-1){
                //     if (queen[i-1][rowposition[i] - 1] == true){
                //         conflict = true;
                //     }
                // }
                // else{
                //     if (queen[i-1][rowposition[i] - 1] == true ||
queen[i-1][rowposition[i] + 1] == true){
                //         conflict = true;
                //     }
                // }
            }

            if (conflict){
                rowposition[i]++;
                count++;
                continue;
```

```
            }

            break;
        }
        if (rowposition[i] >= n){
            continue;
        }

        queen[i][rowposition[i]] = true;
        countqueen += 1;
        colour.push_back(array[i][rowposition[i]]);
        i += 1;
        count++;


        for (int k = 0 ; k < n ; k++){
            for (int l = 0 ; l < n ; l++){
                if (queen[k][l] == true){
                    cout << "X";
                }else{
                    cout << array[k][l];
                }
            }
            cout << endl;
        }
        cout << endl;
    }

    cout << "Hasil Akhir : \n";

    for (int k = 0 ; k < n ; k++){
        for (int l = 0 ; l < n ; l++){
            if (queen[k][l] == true){
                cout << "X";
            }else{
                cout << array[k][l];
            }
        }
        cout << endl;
    }
    cout << endl << "Banyak kasus yang ditinjau: " << count << endl;

    auto finish = chrono::high_resolution_clock::now();
```

```
    auto duration = chrono::duration_cast<chrono::milliseconds>(finish
- start).count();
    cout << "Waktu eksekusi backtracking: " << duration << " ms" <<
endl;

    char choice;
    cout << "Apakah ingin menyimpan hasil ke folder result? (y/n): ";
    cin >> choice;

    if (choice == 'y' || choice == 'Y'){
        string resultName;
        cout << "Masukkan nama file (tanpa .txt): ";
        cin >> resultName;

        filesystem::path resultLoc = filesystem::current_path() /
"result";
        filesystem::create_directories(resultLoc);
        filesystem::path outPath = resultLoc / (resultName + ".txt");
        ofstream out(outPath);
        for (int k = 0 ; k < n ; k++){
            for (int l = 0 ; l < n ; l++){
                if (queen[k][l] == true){
                    out << "X";
                }else{
                    out << array[k][l];
                }
            }
            out << endl;
        }
        out << endl;
        cout << "Hasil disimpan di " << outPath << endl;
    }
    return;
}

bool nextCombination(vector<int>& col, int n) {
    int i = 0;
    while (i < col.size()) {
        if (col[i] + 1 < n) {
            ++col[i];
            return true;
        }
        col[i] = 0;
```

```
        ++i;
    }
    return false;
}


bool isValid(const vector<int>& col, const vector<vector<char>>&
array){
    int n = array.size();
    vector<bool> queenCol(n, false);
    unordered_set<char> colours;

    for (int i = 0; i < n; ++i){

        if (col[i] < 0 || col[i] >= n){
            return false;
        }

        if (queenCol[col[i]]){
            return false;
        }
        queenCol[col[i]] = true;

        char colour = array[i][col[i]];
        if (colours.count(colour)){
            return false;
        }
        colours.insert(colour);

        if (i > 0 && abs(col[i] - col[i - 1]) == 1){
            return false;
        }
    }

    return true;
}


void printResult(const vector<int>& col, const vector<vector<char>>&
array){
    int n = array.size();
    for (int i = 0; i < n; ++i){
        for (int j = 0; j < n; ++j){
```

```
            if (col[i] == j){
                cout << 'X';
            }else{
                cout << array[i][j];
            }
        }
        cout << endl;
    }
    cout << endl;
}


void solutionBruteForce(vector<vector<char>> array){
    int n = array.size();
    vector<int> col(n, 0);
    long long count = 0;
    auto start = chrono::high_resolution_clock::now();

    do {
        ++count;
        if (isValid(col, array)){
            cout << "Hasil Akhir (brute force):" << endl;
            printResult(col, array);
            cout << "Banyak kasus yang ditinjau : " << count << endl;
            auto finish = chrono::high_resolution_clock::now();
            auto duration =
chrono::duration_cast<chrono::milliseconds>(finish - start).count();
            cout << "Waktu eksekusi brute force: " << duration << " ms"
<< endl;
            char choice;
            cout << "Apakah ingin menyimpan hasil ke folder result?
(y/n): ";
            cin >> choice;

            if (choice == 'y' || choice == 'Y'){
                string resultName;
                cout << "Masukkan nama file (tanpa .txt): ";
                cin >> resultName;

                filesystem::path resultLoc = filesystem::current_path()
/ "result";
                filesystem::create_directories(resultLoc);
```

```
                filesystem::path outPath = resultLoc / (resultName +
".txt");
                ofstream out(outPath);
                for (int i = 0; i < n; ++i){
                    for (int j = 0; j < n; ++j){
                        if (col[i] == j){
                            out << 'X';
                        }else{
                            out << array[i][j];
                        }
                    }
                    out << endl;
                }
                out << endl;
                cout << "Hasil disimpan di " << outPath << endl;
            }
            return;
        }
        if (count % 1000 == 0){
            cout << "Sudah meninjau " << count << " kasus." << endl;
            printResult(col, array);
        }
    } while (nextCombination(col, n));

    cout << "Tidak ada solusi dengan brute force." << endl;
    cout << "Banyak kasus yang ditinjau : " << count << endl;
    auto finish = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::milliseconds>(finish
- start).count();
    cout << "Waktu eksekusi brute force: " << duration << " ms" <<
endl;

}


int main() {
    string filename;
    cout<<"Masukkan nama file yang ingin dibuka: ";
    cin>>filename;

    vector<vector<char>> array;
    if (!salin(filename, array)) {
        return 1;
```

```
    }

    int pilihan;
    cout << "Pilih metode penyelesaian:\n";
    cout << "1. Optimisasi (backtracking)\n";
    cout << "2. Brute force murni\n";
    cout << "Masukkan pilihan (1/2): ";
    cin >> pilihan;

    cout << endl;

    if (pilihan == 2){
        solutionBruteForce(array);
    }else if (pilihan == 1){
        solutionOptimization(array);
    }else{
        cout << "Pilihan tidak valid." << endl;
        return 1;
    }
    return 0;
}
```

TEST CASE

1. Input

```
Masukkan nama file yang ingin dibuka: input
AAABBCCCD
ABBBBCECD
ABBBDCECD
AAABDCCCD
BBBBDDDDD
FGGGDDHDD
FGIGDDHDD
FGIGDDHDD
FGGGDDHHH

Pilih metode penyelesaian:
1. Optimisasi (backtracking)
2. Brute force murni
Masukkan pilihan (1/2):
```

2. Progres

```
FGGXDDHDD
XGIGDDHDD
FGXGDDHDD
FGGGDDHHX

Sudah meninjau 354148000 kasus.
AAABBCXCD
ABBBXCECD
ABBBDCXCD
AAABDCXCD
BBBBXDDDD
FGGXDDHDD
XGIGDDHDD
FGXGDDHDD
FGGGDDHHX
```

3. Output

```
Banyak kasus yang ditinjau: 8505
Waktu eksekusi brute force: 2186 ms
Apakah ingin menyimpan hasil ke folder result? (y/n):
```

4. Save Output

```
Apakah ingin menyimpan hasil ke folder result? (y/n): y
Masukkan nama file (tanpa .txt): result
Hasil disimpan di "D:\\Kuliah\\sem4\\BASDAT\\result\\result.txt"
```

result > result.txt

```
1   AAABBCCXD
2   ABBBXCECD
3   ABBBDCXCD
4   AXABDCCCD
5   BBBBDXDDD
6   FGGXDDHDD
7   XGIGDDHDD
8   FGXGDDHDD
9   FGGGDDHHX
10
11
```

5. Backtracking

```
Hasil Akhir :
AAABBCCXD
ABBBXCECD
ABBBDCXCD
AXABDCCCD
BBBBDXDDD
FGGXDDHDD
XGIGDDHDD
FGXGDDHDD
FGGGDDHHX

Banyak kasus yang ditinjau: 8505
Waktu eksekusi brute force: 2042 ms
Apakah ingin menyimpan hasil ke folder result? (y/n):
PS D:\Kuliah\sem4\BASDAT>
```

6. File Kosong

```
PS D:\Kuliah\sem4\BASDAT> ./bin/main
Masukkan nama file yang ingin dibuka: a
File kosong.
PS D:\Kuliah\sem4\BASDAT>
```

7. Isi file diluar alphabet

```
PS D:\Kuliah\sem4\BASDAT> ./bin/main
Masukkan nama file yang ingin dibuka: input
Karakter tidak valid (harus alfabet): 6
PS D:\Kuliah\sem4\BASDAT>
```

8. Contoh soal lain(1)

```
Hasil Akhir (brute force):
AAXB
XABB
CCDX
CXDD

Banyak kasus yang ditinjau : 115
Waktu eksekusi brute force: 0 ms
Apakah ingin menyimpan hasil ke folder result? (y/n):
```

9. Contoh soal lain(2)

```
Hasil Akhir (brute force):
AAABBX
AAXBBB
CCCDXD
CXCDDD
EEEXFF
XEEFFF

Banyak kasus yang ditinjau : 4266
Waktu eksekusi brute force: 11 ms
Apakah ingin menyimpan hasil ke folder result? (y/n):
```

10. Contoh soal lain (3)

```
Sudah meninjau 823000 kasus.
AAXBBBB
AAABBBX
AAXBBBB
CCCDDXD
CCCDDDX
EEEFFFX
EEEFFGX

Tidak ada solusi dengan brute force.
Banyak kasus yang ditinjau : 823543
Waktu eksekusi brute force: 1711 ms
PS D:\Kuliah\sem4\BASDAT>
```

11. Contoh soal lain (4)

```
Hasil Akhir (brute force):
AAAABBBX
AAAXBBBB
CCCCDDXD
CCXCDDDD
EEEEFXFF
EXEEFFFF
GGGGXHHH
XGGGHHHH

Banyak kasus yang ditinjau : 1103264
Waktu eksekusi brute force: 3175 ms
Apakah ingin menyimpan hasil ke folder result? (y/n):
1 ⚠ 0
```

Pernyataan tidak melakukan kecurangan

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

Mahatma Brahmana
13524015

Tabel Checklist

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		✓