LAPORAN TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA
SEMESTER II Tahun 2025/2026
Penyelesaian Permainan Queens LinkedIn

Leonardus Brain Fatolosja
13524146

Institut Teknologi Bandung
2026

Nama **:** Leonardus Brain Fatolosja
NIM **:** 13524146

I. Penyelesaian Menggunakan Algoritma Brute Force

Pada tugas kecil ini, saya menggunakan algoritma brute force murni dengan backtracking. Langkah-langkah dari algoritma brute force ini adalah sebagai berikut.

a. Membaca file .txt yang berisi papan awal untuk ditempati ratu nantinya. Akan ada satu fungsi yang menangani hal ini dan mendeteksi apabila ada kondisi yang menyebabkan input menjadi tidak *valid*. Beberapa kondisi ketika input menjadi tidak *valid* adalah sebagai berikut.
   - Jumlah baris dan kolom tidak sama.
   - Jumlah baris dan daerah tidak sama.

b. Inisiasi dari program ini adalah pembentukan matriks dan array yang berisi calon *queen* yang akan dicek dan di*update* terus menerus. Selain itu, ada variabel untuk mengatur penampilan kondisi papan secara *live*. Terakhir, ada variabel untuk menghitung waktu eksekusi dari algoritma brute forcenya.

c. Program akan memanggil fungsi rekursif untuk mencoba semua kemungkinan posisi *queen* secara murni dalam artian tidak adanya heuristik sama sekali. Efek samping dari menggunakan algoritma brute force murni adalah kompleksitas waktu yang sangat tinggi sehingga membutuhkan waktu yang sangat lama seiring bertambahnya jumlah variabel (jumlah *queen*).

d. Fungsi rekursif akan mengecek posisi queen satu per satu dari n x n posisi di papan. Tentunya sebelum ditaruh, posisi tersebut akan dicek untuk mencari tahu apakah posisi tersebut merupakan posisi yang valid atau tidak. Artinya adalah posisi tersebut tidak mengandung warna daerah yang sudah ditaruh *queen*, dan tidak berada di kolom/baris yang sama dengan queen lain.

e. Jika fungsi sudah menemui satu posisi yang *valid*, maka rekursif akan dipanggil kembali dan mengulangi hal yang sama sampai array yang berisi calon *queen* penuh atau tidak ada posisi yang *valid*. Jika tidak ada posisi yang *valid,* maka fungsi akan melakukan backtracking dan menghapus posisi queen yang sudah dimasukkan sebelumnya ke array

yang berisi posisi calon *queen*. Tujuannya adalah untuk mencari posisi baru demi kombinasi posisi queen yang baru juga.

f. Fungsi rekursif akan berhenti ketika array yang berisi calon queen sudah terisi penuh. Tanda fungsi akan berhenti dengan mengembalikan nilai true. Pada main, jika fungsi rekursif mengembalikan true, maka hasil akan ditampilkan. Sedangkan jika fungsi mengembalikan false, maka akan ada pesan yang bertuliskan bahwa tidak ada solusi yang tepat. Selain itu, ada perhitungan waktu eksekusi dan jumlah percobaan yang gagal.

Algoritma ini memang sangat tidak efisien secara waktu dan memori. Namun, algoritma ini membuktikan bahwa brute force bisa menyelesaikan hampir semua persoalan walau *naif*. Kompleksitas waktu dari algoritma brute force yang direalisasikan adalah O(n. n . n!), yang terdiri dari rekursif sebanyak n! dan iterasi di dalam rekursif sebanyak n x n. Ini menjadi salah satu alasan yang membuat algoritma ini sangat lambat dalam menyelesaikan persoalnnya.

II. *Source Code* Program dalam C

```
typedef struct {
    int x;
    int y;
} Point;

typedef struct {
    char area;
    Point point;
} Result;
```

Kode di atas merupakan tipe data untuk menyimpan posisi calon queen. Nantinya array of Result akan selalu dicek dan di*update*.

```
char** bacaFile (int *rows, int *cols, bool *valid, int *area){
    FILE *file;

    char areas[MAX];
    *area = 0;
    int buffer;
    *rows=0,*cols=0;

    char nama[144];
    printf("Masukkan nama file (di folder test): ");
    scanf("%s", nama);

    char path[200];
    sprintf(path, "../test/%s", nama);
    file = fopen(path, "r");
    if (file == NULL){
        printf("Error boys!");
        exit(1);
    }
```

```
// Dimensi File
buffer = fgetc(file);
while (buffer!=EOF){
    bool ada = false;
    if (buffer=='\n'){
        *rows+=1;
    }
    if (*rows==0 && buffer != '\n' && buffer !=' '){
        *cols+=1;
    }
    for (int i=0;i<*area;i++){
        if (areas[i]==(char)buffer){
            ada = true;
            break;
        }else if (buffer=='\n' || buffer==' '){
            ada = true;
            break;
        }
    }
    if (!ada) {
        areas[*area]=(char)buffer;
        *area+=1;
    }
    buffer = fgetc(file);
}
*rows +=1;
if (*rows!=*cols || *rows!=*area ){
    *valid = false;
    return 0;
}
rewind(file);
```

```
// Assign ke Matrix
char **A =  malloc(*rows * sizeof(char *));
for (int i=0;i< *rows;i++){
    A[i] = malloc((*cols) * sizeof(char));
}
int i=0,j=0;
buffer = fgetc(file);
while (buffer!=EOF){
    if (buffer == '\n'){
        i+=1;
        j=0;
    }else if (buffer == ' '){
    }else{
        A[i][j] = buffer;
        j+=1;
    }
    buffer = fgetc(file);
}
fclose(file);

*valid = true;
return A;
```

Fungsi bacaFile akan mengubah variabel jumlah kolom (cols), jumlah baris (rows), jumlah area warna (area), dan kondisi ke-valid-an input (valid). Di dalamnya, fungsi ini akan membaca file yang namanya ditulis oleh pengguna. Selanjutnya, isi karakter yang melambangkan area akan disimpan di dalam matriks.

```
bool accept(Result *RL, char area, int row, int col, int n){
    for (int i=0;i<n;i++){
        if (RL[i].area==area || RL[i].point.x==row || RL[i].point.y==col){
            return false;
        }
    }
    return true;
}

void putQueen(Result *RL, char area, int row, int col, int n){
    for (int i=0;i<n;i++){
        if (RL[i].area=='0'){
            RL[i].area=area;
            RL[i].point.x=row;
            RL[i].point.y=col;
            return;
        }
    }
}

void removeQueen(Result *RL, char area, int n){
    for (int i=0;i<n;i++){
        if (RL[i].area==area){
            RL[i].area='0';
            RL[i].point.x=-1;
            RL[i].point.y=-1;
            return;
        }
    }
}
```

Kode di atas berisi 3 fungsi, yaitu fungsi accept untuk mengecek apakah calon posisi merupakan posisi yang valid atau tidak, fungsi putQueen untuk menaruh queen ke dalam array yang berisi calon queen, dan fungsi removeQueen untuk menghapus posisi queen dari array yang berisi calon queen. Ketiganya akan mengakses array of Result yang berisi calon queen.

```
bool cekResult(Result *RL, int n){
    for (int i=0;i<n;i++){
        if (RL[i].area=='0'){
            return false;
        }
    }
    return true;
}

void printHasil(char** matrix, Result *RL, int rows, int area){
    for (int i=0;i<rows;i++){
        for (int j=0;j<rows;j++){
            bool found = false;
            for (int k=0;k<area;k++){
                if (RL[k].point.x==i && RL[k].point.y==j){
                    found = true;
                    break;
                }
            }
            if (found) printf("#");
            else printf("%c",matrix[i][j]);
        }
        printf("\n");
    }
    return;
}
```

Kode di atas berisi 2 fungsi, yaitu fungsi cekResult untuk mengecek apakah array yang berisi calon queen sudah terisi penuh atau belum dengan tujuan untuk mengakhiri algoritma brute force ini, dan fungsi printHasil untuk mencetak isi matrix dengan posisi calon queen pada kondisi saat itu berdasarkan isi dari array of Result.

```
void tulisFile(char **matrix, Result *RL, int n, char *nama, int try, int time){
    FILE *file = fopen(nama, "a");

    fprintf(file, "Hasil untuk %d x %d\n", n, n);
    for (int i=0;i<n;i++){
        for (int j=0;j<n;j++){
            bool found = false;
            for (int k=0;k<n;k++){
                if (RL[k].point.x==i && RL[k].point.y==j){
                    found = true;
                    break;
                }
            }
            if (found) fprintf(file, "#");
            else fprintf(file, "%c",matrix[i][j]);
        }
        fprintf(file, "\n");
    }
    fprintf(file, "Jumlah percobaan: %d\n", try);
    fprintf(file, "Waktu total: %d ms\n", time);
    fclose(file);
}
```

Kode di atas merupakan fungsi untuk menyimpan hasil akhir ke dalam file. Jika nama file sudah ada, maka hanya akan menambahkan isi. Sedangkan jika tidak ada file, maka akan membuat file yang baru dan mengisinya.

```
bool rekursif(char** matrix, Result *RL, int rows, int cols, int *try, int freq){
    if (cekResult(RL,rows)) {
        return true;
    }

    for (int i=0;i<rows;i++){
        for (int j=0;j<cols;j++){
            if (accept(RL,matrix[i][j],i,j,rows)){
                putQueen(RL,matrix[i][j],i,j,cols);

                if (rekursif(matrix,RL,rows,cols,try,freq)){
                    return true;
                }
                *try += 1;

                if (freq>0 && *try % freq == 0){
                    system("cls");
                    printHasil(matrix,RL,rows,cols);
                    printf("\n");
                }

                removeQueen(RL,matrix[i][j],rows);
            }
        }
    }
    return false;
}
```

Fungsi terakhir adalah fungsi rekursif yang merupakan algoritma utama dari program ini. Fungsi ini mempunyai pengecekan keberakhiran rekursif sebagai *line code* pertama yang dijalankan. Jika belum, fungsi akan mencoba semua kemungkinan posisi queen dengan menggunakan fungsi-fungsi sebelumnya untuk mengecek kevalidan posisi, menaruh queen ke dalam calon, menghapus queen dari calon, dan mencetak matriks sementara dari queen.

III. Input dan Output

Semua input dan output dapat dilihat di dalam repository github, terkhususnya di dalam folder test.

A. Testcase1.txt

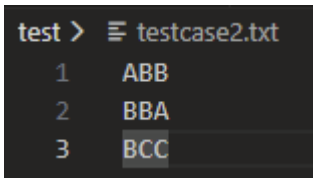
test > testcase1.txt

```
1  AAABBCCCD
2  ABBBBCECD
3  ABBBDCECD
4  AAABDCCCD
5  BBBBDDDDD
6  FGGGDDHDD
7  FGIGDDHDD
8  FGIGDDHDD
9  FGGGDDHHH
```

```
Ada solusi dengan 15040324 percobaan
A#ABBCCCD
ABBB#CECD
ABBBDC#CD
AAABD#CCD
BBBBDDD#D
#GGGDDHDD
FG#GDDHDD
FGI#DDHDD
FGGGDDHH#

Waktu total: 20055 ms
Apakah ingin menyimpan hasil? (y/n) :y
Masukkan nama file untuk menyimpan hasil (di folder test): result1
```

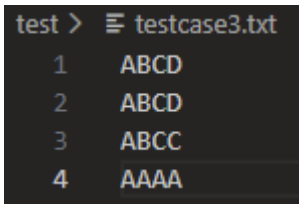
B. Testcase2.txt




```
[PERINGATAN] Live Update akan memperlambat algoritma, masukkan 0 untuk menonaktifkan Live Update
[SARAN] Gunakan frekuensi 100.000 untuk n=8 supaya tidak terlalu lambat
Masukkan frekuensi iterasi untuk Live Update : 10
Masukkan nama file (di folder test): testcase2
Ada solusi dengan 0 percobaan
#BB
B#A
BC#

Waktu total: 1 ms
Apakah ingin menyimpan hasil? (y/n) :y
Masukkan nama file untuk menyimpan hasil (di folder test): result2
```

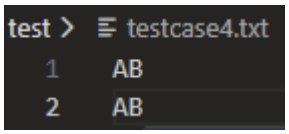
C. Testcase3.txt




```
[PERINGATAN] Live Update akan memperlambat algoritma, masukkan 0 untuk menonaktifkan Live Update
[SARAN] Gunakan frekuensi 100.000 untuk n=8 supaya tidak terlalu lambat
Masukkan frekuensi iterasi untuk Live Update : 100
Masukkan nama file (di folder test): testcase3
Ada solusi dengan 25 percobaan
A#CD
ABC#
AB#C
#AAA

Waktu total: 1 ms
Apakah ingin menyimpan hasil? (y/n) :y
Masukkan nama file untuk menyimpan hasil (di folder test): result3
```

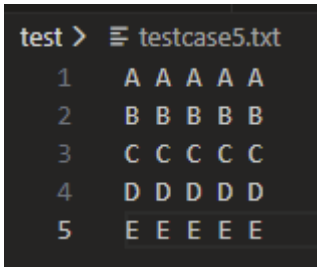
D. Testcase4.txt




```
[PERINGATAN] Live Update akan memperlambat algoritma, masukkan 0 untuk menonaktifkan Live Update
[SARAN] Gunakan frekuensi 100.000 untuk n=8 supaya tidak terlalu lambat
Masukkan frekuensi iterasi untuk Live Update : 1
Masukkan nama file (di folder test): testcase4
Ada solusi dengan 0 percobaan
#B
A#

Waktu total: 1 ms
Apakah ingin menyimpan hasil? (y/n) :y
Masukkan nama file untuk menyimpan hasil (di folder test): result4
```

E. Testcase5.txt




```
[PERINGATAN] Live Update akan memperlambat algoritma, masukkan 0 untuk menonaktifkan Live Update
[SARAN] Gunakan frekuensi 100.000 untuk n=8 supaya tidak terlalu lambat
Masukkan frekuensi iterasi untuk Live Update : 100
Masukkan nama file (di folder test): testcase5
Ada solusi dengan 0 percobaan
#AAAA
B#BBB
CC#CC
DDD#D
EEEE#

Waktu total: 1 ms
Apakah ingin menyimpan hasil? (y/n) :y
Masukkan nama file untuk menyimpan hasil (di folder test): result5
```

IV. Pranala Repository

https://github.com/leains/tucil1_13524146

V. Pernyataan

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

Leonardus Brain Fatolosja
13524146