Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II tahun 2025/2026

Penyelesaian Permainan Queens Linkedin dengan Algoritma Brute Force

Ariel Cornelius Sitorus 13524085

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2026

DAFTAR ISI

Contents

BAB II 4
ALGORITMA BRUTE FORCE 4
BAB III SOURCE CODE 5
REFERENSI 17
LAMPIRAN 18
Tabel Hasil 18
Pranala Github 18
Pernyataan 18

BAB I
DESKRIPSI PERMASALAHAN

Queens adalah permainan logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari permainan ini adalah menempatkan *Queen* pada sebuah papan persegi berwarna (*grid*). Komponen pada permainan antara lain :

1. Papan, yaitu matriks berukurang persegi yang teridiri atas sel-sel kosong
2. Wilayah (*Regions*), yaitu area berwarna yang membagi papan menjadi bagian tersendiri
3. *Queens,* bidak yang ditempatkan pada posisi yang valid papan
4. Timer, yaitu untuk menghitung waktu

Aturan permainan *minigame* ini sebagai berikut:

1. Hanya terdapat satu *Queen* pada setiap baris,
2. Hanya terdapat satu *Queen* pada setiap kolom,
3. Hanya terdapat satu *Queen* pada setiap daerah warna, dan
4. Satu *Queen* tidak dapat ditempatkan bersebelahan dengan *Queen* lainnya, termasuk secara diagonal.

Tugas kecil ini bertujuan untuk merancang algoritma *brute force* untuk menemukan solusi dengan menempatkan queen pada posisi yang tepat. Pada Tugas Kecil 1 ini, akan digunakan bahasa pemrograman C++.

BAB II

ALGORITMA BRUTE FORCE

Algoritma brute force adalah pendekatan yang naif untuk memecahkan suatu persoalan. Algoritma ini didasarkan pada problem statement dan definisi atau konsep yang dilibatkan. Algoritma ini akan memecahkan persoalan dengan langsung, jelas, dan sangat sederhana. Pada permainan *queen* ini, algoritma brute force yang digunakan pada program ini bekerja dengan mencoba semua kemungkinan susunan penempatan queen di papan, kemudian memeriksa apakah susunan tersebut memenuhi semua aturan permainan.

Algoritma *brute force* yang dibuat untuk menyelesaikan *queen* tersebut memiliki langkah-langkah sebagai berikut:

1. Inisialisasi array perm = [0, 1, 2, ..., N-1] sebagai permutasi pertama yang merepresentasikan posisi awal queen di setiap baris. Hitung total permutasi yang akan dicoba sebesar N!.

2. Untuk setiap permutasi, lakukan pengecekan validitas susunan queen. Pengecekan dilakukan dalam dua tahap:

   a) Cek tidak bersebelahan: untuk setiap pasang queen di barisberurutan i dan i+1, hitung selisih kolomnya. Jika selisih bernilai kurang dari atau sama dengan 1, berarti kedua queen bersebelahan atau berdiagonal langsung sehingga permutasi dinyatakan tidak valid.

   b) Cek region berbeda: ambil karakter warna dari grid[i][perm[i]] untuk setiap queen. Jika ditemukan dua queen dengan karakter warna yang sama, permutasi dinyatakan tidak valid. Jika kedua pengecekan lolos, solusi ditemukan.

3. Jika permutasi saat ini tidak valid, generate permutasi berikutnya menggunakan algoritma nextPerm: cari posisi i paling kanan dimana perm[i] < perm[i+1], cari posisi j paling kanan dimana perm[j] > perm[i], tukar perm[i] dan perm[j], lalu balik semua elemen dari posisi i+1 sampai akhir array.

4. Ulangi langkah 2 dan 3 sampai solusi ditemukan atau seluruh N! permutasi telah habis dicoba. Jika semua permutasi sudah dicoba dan tidak ada yang valid, papan dinyatakan tidak memiliki solusi.

Berdasarkan bahan ajar, program ini tidak menggunakan Teknik heuristic karena:

Slide	Program

heuristik mengeliminasi kemungkinan solusi tanpa mengeksplorasi seluruhnya secara penuh.	semua N! permutasi dieksplorasi satu per satu secara penuh tanpa ada yang dilewati atau diprioritaskan.
heuristik berbasis terkaan, intuisi, atau common sense yang tidak dapat dibuktikan secara matematis.	satu-satunya alasan sebuah permutasi dinyatakan tidak valid adalah karena gagal pengecekan matematis , yaitu selisih kolom <= 1 (bersebelahan) atau dua berada di warna yang sama.
heuristik tidak menjamin selalu dapat memecahkan persoalan.	dijamin selalu menemukan solusi jika Solusi memang ada, dan dijamin menyatakan tidak ada solusi jika memang tidak ada, karena seluruh ruang pencarian diperiksa.

BAB III
SOURCE CODE

Alur Program Secara Keseluruhan

1. main.cpp: program mulai, minta path file input dari user
2. input.cpp: baca file, parsing papan, validasi input
3. main.cpp: tampilkan papan yang dibaca
4. solutions.cpp: jalankan brute force (inisialisasi perm)
5. solutions.cpp: loop coba semua permutasi
   a. tiap 20 percobaan: tampilkan progress di layar
   b. cek valid tiap permutasi
   c. kalau valid: catat solusi
6. solutions.cpp: tampilkan hasil akhir + statistik waktu
7. solutions.cpp: tanya user mau simpan ke file atau tidak
   a. kalau ya: minta nama file, simpan ke test/output/
8. program selesai

Input.h

```
#ifndef INPUT_H
#define INPUT_H

#include <iostream>
#include <fstream>
#include <cstring>
#include <cstdlib>
#include <ctime>
using namespace std;

#define MAX_N 26

// struct buat nyimpen data papan
struct Board{
    int N;
    char grid[MAX_N][MAX_N];
    char regions[MAX_N];       // list region unik
    int numRegions;
    int regionCounts[MAX_N];               // jumlah kotak per region
    int regionCoords[MAX_N][MAX_N][2];     // koordinat tiap region
[regionIdx][kotakKe][0=baris/1=kolom]
};

Board readInput(char filename[]);
void fillRegionCoords(Board &board);
void printBoard(Board board);
void clearScreen();

#endif
```

input.cpp

```
#include "input.h"

Board readInput(char fname[]){
    Board b;
    b.N=0; b.numRegions=0;

    ifstream fin(fname);
    if(!fin.is_open()){ cout<<"gabisa buka file\n"; exit(1); }

    char lines[MAX_N][MAX_N+2];
    int nl=0;
    char buf[256];
    while(fin.getline(buf,256)){
        int len=strlen(buf);
        while(len>0&&(buf[len-1]=='\r'||buf[len-1]=='\n'||buf[len-1]==' '))
len--;
        buf[len]='\0';
        if(len>0){
            for(int i=0;i<len;i++) lines[nl][i]=buf[i];
            lines[nl][len]='\0';
            nl++;
        }
    }
    fin.close();
```

```
    if(nl==0){ cout<<"file kosong\n"; exit(1); }
    b.N=nl;

    for(int i=0;i<b.N;i++){
        int len=strlen(lines[i]);
        if(len!=b.N){ cout<<"ukuran baris salah\n"; exit(1); }
        for(int j=0;j<b.N;j++){
            b.grid[i][j]=lines[i][j];
            int ada=0;
            for(int k=0;k<b.numRegions;k++) if(b.regions[k]==lines[i][j]){
ada=1; break; }
            if(!ada) b.regions[b.numRegions++]=lines[i][j];
        }
    }

    if(b.numRegions!=b.N){ cout<<"jumlah tidak cocok dengan ukuran papan\n";
exit(1); }
    return b;
}

void printBoard(Board b){
    for(int i=0;i<b.N;i++){
        for(int j=0;j<b.N;j++) cout<<b.grid[i][j];
        cout<<'\n';
    }
}

void clearScreen(){ system("cls"); }

void fillRegionCoords(Board &b){
    for(int i=0;i<b.numRegions;i++) b.regionCounts[i]=0;
    for(int i=0;i<b.N;i++){
        for(int j=0;j<b.N;j++){
            char c=b.grid[i][j];
            int idx=-1;
            for(int k=0;k<b.numRegions;k++) if(b.regions[k]==c){ idx=k;
break; }
            if(idx!=-1){
                int cnt=b.regionCounts[idx];
                b.regionCoords[idx][cnt][0]=i;
                b.regionCoords[idx][cnt][1]=j;
                b.regionCounts[idx]++;
            }
        }
    }
}
```

Solutions.h

```
#ifndef SOLUTIONS_H
#define SOLUTIONS_H

#include "input.h"

void tampilHasil(Board board, int perm[]);
void simpanSolusi(Board board, int perm[], double waktu, long long kasus);
bool nextPerm(int arr[], int n);
int cekValid(Board &board, int perm[]);
void solve(Board board);

#endif
```

Solutions.cpp

```
#include "solutions.h"

void tampilHasil(Board board, int perm[]){
    for(int i=0;i<board.N;i++){
        for(int j=0;j<board.N;j++){
            if(j==perm[i]) cout<<'#';
            else cout<<board.grid[i][j];
        }
        cout<<'\n';
    }
}

void simpanSolusi(Board board, int perm[], double waktu, long long iter){
    char fname[64];
    char fullpath[256];
    cout<<"nama file output (akan disimpan di test/output/, tanpa .txt): ";
    cin.getline(fname,64);
    // gabungin path otomatis ke test/output/ dan tambahin .txt
    strcpy(fullpath,"test/output/");
    strcat(fullpath,fname);
    strcat(fullpath,".txt");
    ofstream fout(fullpath);
    if(!fout.is_open()){ cout<<"gagal buka file\n"; return; }
    for(int i=0;i<board.N;i++){
        for(int j=0;j<board.N;j++){
            if(j==perm[i]) fout<<'#';
            else fout<<board.grid[i][j];
        }
        fout<<'\n';
    }
    fout<<'\n';
```

```
    fout<<"Waktu pencarian: "<<waktu<<" ms\n";
    fout<<"Banyak kasus yang ditinjau: "<<iter<<" kasus\n";
    fout.close();
    cout<<"tersimpan di "<<fname<<'\n';
}

bool nextPerm(int a[], int n){
    int i=n-2;
    while(i>=0 && a[i]>=a[i+1]) i--;
    if(i<0) return false;
    int j=n-1;
    while(a[j]<=a[i]) j--;
    int t=a[i]; a[i]=a[j]; a[j]=t;
    int l=i+1,r=n-1;
    while(l<r){ t=a[l];a[l]=a[r];a[r]=t; l++;r--; }
    return true;
}

int cekValid(Board &board, int perm[]){
    int n=board.N;
    // cek baris berurutan ga boleh bersebelahan/diagonal langsung
    for(int i=0;i<n-1;i++){
        int d=perm[i]-perm[i+1];
        if(d<0) d=-d;
        if(d<=1) return 0;
    }
    // cek tiap queen beda region
    char used[MAX_N]; int cnt=0;
    for(int i=0;i<n;i++){
        char c=board.grid[i][perm[i]];
        for(int k=0;k<cnt;k++) if(used[k]==c) return 0;
        used[cnt++]=c;
    }
    return 1;
}

void solve(Board board){
    int n=board.N;
    int perm[MAX_N];
    for(int i=0;i<n;i++) perm[i]=i;

    long long total=1;
    for(int i=1;i<=n;i++) total*=i;

    cout<<"nyari solusi...\n";
```

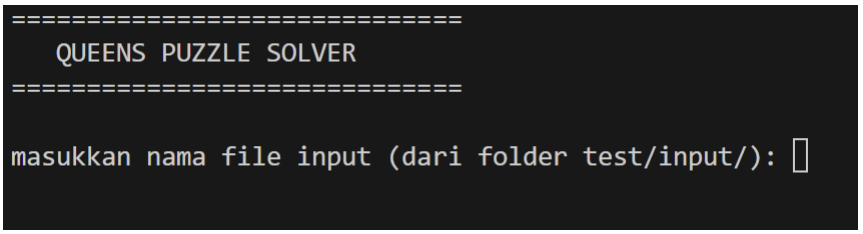
```
    cout<<"total permutasi: "<<total<<"\n\n";

    long long iter=0;
    int sol[MAX_N], found=0;
    clock_t t0=clock();

    do{
        iter++;
        if(iter%20==0){
            clearScreen();
            printHeader();
            cout<<"percobaan ke "<<iter<<" dari "<<total<<"\n\n";
            tampilHasil(board,perm);
            cout<<'\n';
        }
        if(cekValid(board,perm)){
            found=1;
            for(int i=0;i<n;i++) sol[i]=perm[i];
            break;
        }
    }while(nextPerm(perm,n));

    clock_t t1=clock();
    double waktu=(double)(t1-t0)/CLOCKS_PER_SEC*1000;

    clearScreen();
    printHeader();

    if(found){
        cout<<"solusi ketemu!\n\n";
        tampilHasil(board,sol);
    } else {
        cout<<"ga ada solusi\n";
    }

    cout<<"\nWaktu pencarian: "<<waktu<<" ms\n";
    cout<<"Banyak kasus yang ditinjau: "<<iter<<" kasus\n";

    if(found){
        cout<<"\nmau simpan? (y/n): ";
        char jwb[4]; cin.getline(jwb,4);
        if(jwb[0]=='y'||jwb[0]=='Y') simpanSolusi(board,sol,waktu,iter);
    }
}
```

main.cpp

```
#include "input.h"
#include "solutions.h"

int main(){
    char fname[64];
    char fullpath[256];

    printHeader();
    cout<<"masukkan nama file input (dari folder test/input/): ";
    cin.getline(fname,64);

    // gabungin path otomatis ke test/input/
    strcpy(fullpath,"test/input/");
    strcat(fullpath,fname);

    Board papan = readInput(fullpath);
    cout<<"\npapan yang dibaca:\n";
    printBoard(papan);
    cout<<'\n';

    solve(papan);

    return 0;
}
```

BAB IV
UJI COBA PROGRAM

Program untuk menyelesaikan permainan *Queen* dikompilasi dengan menggunakan makefile. make untuk kompilasi, make run untuk jalankan program dan make clean untuk hapus hasil kompilasi.



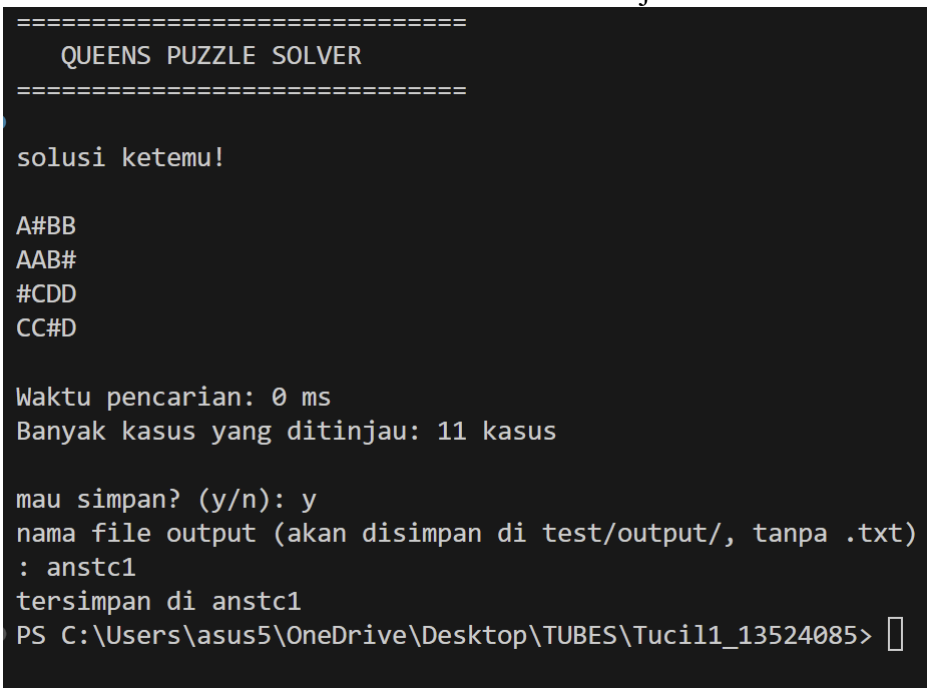
Gambar 5.1 Tampilan Awal Program

Berikutnya pengguna dapat memilih sesuai pilihan yang tersedia pada menu. Berikut beberapa data uji sebagai contoh input output program.

Data Uji 1

```
AABB
AABB
CCDD
CCDD
```

Gambar 5.2 Masukkan Data Uji 1



Gambar 5.3 Keluaran Data Uji 1

1. Data Uji 2

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

Gambar 5.4 Data Uji 2

```
   QUEENS PUZZLE SOLVER
==============================

solusi ketemu!

AAABBCC#D
ABBB#CECD
ABBBDC#CD
A#ABDCCCD
BBBBD#DDD
FGG#DDHDD
#GIGDDHDD
FG#GDDHDD
FGGGDDHH#

Waktu pencarian: 170396 ms
Banyak kasus yang ditinjau: 306205 kasus

mau simpan? (y/n): y
nama file output (akan disimpan di test/output/, tanpa .txt)
: anstc2
tersimpan di anstc2
PS C:\Users\asus5\OneDrive\Desktop\TUBES\Tucil1_13524085>
```

Gambar 5.5 Keluaran Data Uji 2

2. Data Uji 3

```
AABB
CCDD
AABB
CCDD
```

Gambar 5.6 Data Uji 3

```
==============================
   QUEENS PUZZLE SOLVER
==============================

ga ada solusi

Waktu pencarian: 15 ms
Banyak kasus yang ditinjau: 24 kasus
```

Gambar 5.7 Keluaran Data Uji 3

3. Data Uji 4

```
AABBC
ADBBC
DDDEC
DDEEC
DDECC
```

Gambar 5.8 Data Uji 4

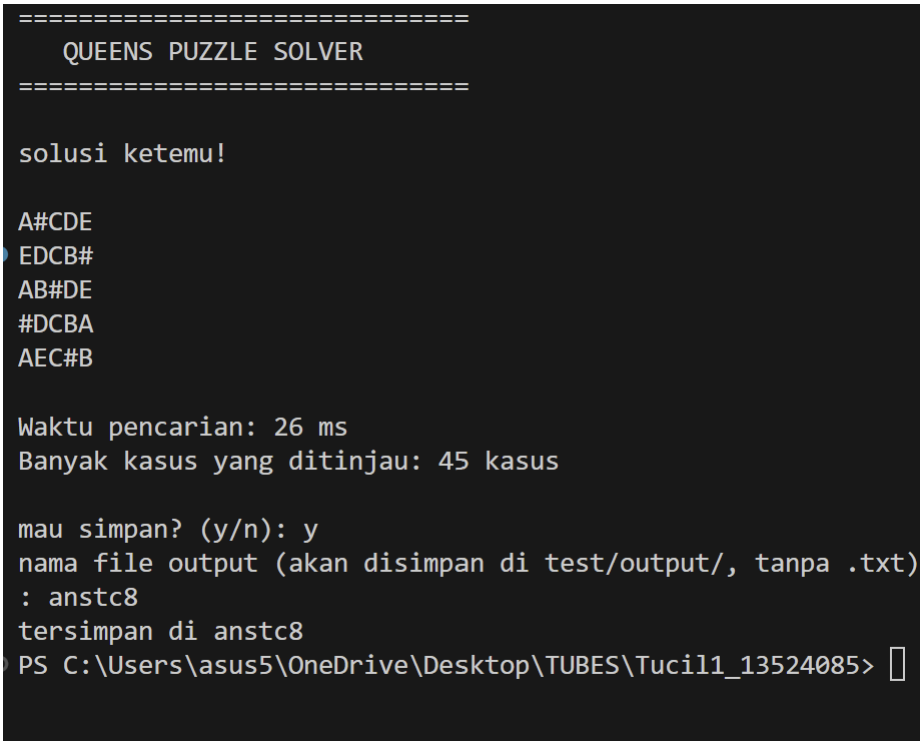
```
==============================
   QUEENS PUZZLE SOLVER
==============================

solusi ketemu!

#ABBC
ADB#C
D#DEC
DDEE#
DD#CC

Waktu pencarian: 0 ms
Banyak kasus yang ditinjau: 14 kasus

mau simpan? (y/n): y
nama file output (akan disimpan di test/output/, tanpa .txt)
: anstc7
tersimpan di anstc7
PS C:\Users\asus5\OneDrive\Desktop\TUBES\Tucil1_13524085>
```

Gambar 5.9 Keluaran Data Uji 4

4. Data Uji 5

```
ABCDE
EDCBA
ABCDE
EDCBA
AECDB
```

Gambar 5.10 Data Uji 5



Gambar 5.11 Keluaran Data Uji 5

REFERENSI

R. Munir, “Algoritma Brute Force (Bagian I),” *IF2211 Strategi Algoritma*, 2026. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/02-Algoritma-Brute-Force-(2026)-Bag1.pdf

R. Munir, “Algoritma Brute Force (Bagian 2),” *IF2211 Strategi Algoritma*, 2026. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/03-Algoritma-Brute-Force-(2026)-Bag2.pdf

LAMPIRAN

Tabel Hasil

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**		√

Pranala Github

Repository Github dapat diakses melalui pranala berikut:

https://github.com/Sakazu01/Tucil1_13524085

Pernyataan

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

Ariel Cornelius Sitorus