LAPORAN TUGAS KECIL

IF2211 Strategi Algoritma

Queens Board Solver dengan Algoritma Brute Force

Dipersiapkan oleh:
13524134 - Salman Faiz Assidqi

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganesha 10, Bandung 40132

1 Implementasi Algoritma Brute Force

Algoritma Brute Force digunakan dalam program melalui iterasi dan *backtracking*. Program akan mencari posisi ratu yang valid melalui iterasi pada setiap warna. Jika pencarian posisi ratu pada suatu warna gagal, program akan mencari posisi ratu kembali pada warna sebelumnya untuk mencari kemungkinan lain (karena gagal). Agar dapat lebih jelas, berikut adalah langkah-langkah algoritma Brute Force yang digunakan program.

1. Pilih warna pertama, yaitu yang berada di paling kiri atas matriks papan.
2. Pilih posisi ratu di petak pertama warna, yaitu yang berada di paling kiri atas warna.
3. Jika posisi ratu valid (tidak ada ratu di baris dan kolom yang sama serta di dekatnya), letakkan ratu di petak tersebut.
4. Jika langkah 2 tidak valid, pilih posisi ratu di petak kedua warna dan seterusnya sampai sampai valid.
5. Jika posisi ratu yang valid tidak ditemukan sampai akhir pencarian pada langkah 4, warna sebelumnya akan dipilih untuk dicari kembali posisi ratunya (Posisi yang dicari melanjutkan dari posisi sebelumnya).
6. Jika dipilih warna pertama (pada awal atau hasil *backtracking*) dan posisi ratu yang valid tidak ditemukan, papan tidak memiliki solusi.
7. Melanjutkan langkah 3, warna berikutnya akan dipilih untuk dicari posisi ratunya. Jika posisi ratu yang valid berhasil ditemukan sampai warna terakhir, papan akan memiliki solusi.

2 Kode Sumber Program

Berikut adalah kode sumber Program Queens Board Solver.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <time.h>

typedef struct {
    int baris;
    int kolom;
} Posisi;

Posisi buatPosisi(int baris, int kolom) {
    Posisi posisi;
    posisi.baris = baris;
    posisi.kolom = kolom;
    return posisi;
}

typedef struct {
    Posisi posisiSekarang;
```

```
    int nextAlphabet;
} Alphabet;

char matriks[600][600];
Alphabet alphaList[26];
int currentAlphabetIndex = -1;
int firstAlphabetIndex = -1;

void buatAlphaList(Alphabet *alphabet) {
    for (int i = 0; i < 26; i++) {
        alphabet[i].posisiSekarang.baris = -1;
        alphabet[i].posisiSekarang.kolom = -1;
        alphabet[i].nextAlphabet = -1;
    }
}

int dapatBanyakAlphabet() {
    int banyakAlphabet = 0;
    for (int i = 0; i < 26; i++) {
        if (alphaList[i].nextAlphabet != -1) {
            banyakAlphabet++;
        }
    }
    return banyakAlphabet;
}

void displayMatriks(int jumlahBaris, int jumlahKolom) {
    for (int i = 0; i < jumlahBaris; i++) {
        for (int j = 0; j < jumlahKolom; j++) {
            printf("%c ", matriks[i][j]);
        }
        printf("\n");
    }
}

int panjangBarisValiddanSimpan(char *baris, int panjangBaris, int
indeksBaris) {
    int panjangBarisTemp = 0;
    int iAlpha = 0;
    for (int i = 0; i < panjangBaris; i++) {
```

```
        char karakter = baris[i];
        int karakterInt = (int) baris[i];
        if (karakterInt >= 65 && karakterInt <= 90 /* huruf besar */) {
            panjangBarisTemp++;
        } else if (karakterInt == 32 /* spasi */ || karakterInt == 13 /*
enter */) {
            continue;
        } else {
            panjangBarisTemp = -1;
            break;
        }
        matriks[indeksBaris][iAlpha] = karakter;
        iAlpha++;
        int indeksAlphabet = karakterInt - 65;
        if (alphaList[indeksAlphabet].nextAlphabet == -1) {
            if (firstAlphabetIndex == -1) {
                firstAlphabetIndex = indeksAlphabet;
                currentAlphabetIndex = firstAlphabetIndex;
                alphaList[indeksAlphabet].nextAlphabet =
firstAlphabetIndex;
            } else {
                alphaList[currentAlphabetIndex].nextAlphabet =
indeksAlphabet;
                currentAlphabetIndex = indeksAlphabet;
                alphaList[indeksAlphabet].nextAlphabet =
firstAlphabetIndex;
            }
        }
    }
    return panjangBarisTemp;
}

bool apkhPosisiDekat(Posisi posisi1, Posisi posisi2) {
    if (abs(posisi1.baris - posisi2.baris) <= 1 && abs(posisi1.kolom -
posisi2.kolom) <= 1) {
        return true;
    }
    return false;
}
```

```
bool apkhPosisiValid(Posisi posisi) {
    int i = firstAlphabetIndex;
    bool selesai = false;

    while(!selesai) {
        if (alphaList[i].posisiSekarang.baris != -1 &&
alphaList[i].posisiSekarang.kolom != -1) {
            if (alphaList[i].posisiSekarang.baris == posisi.baris ||
                alphaList[i].posisiSekarang.kolom == posisi.kolom ||
                apkhPosisiDekat(alphaList[i].posisiSekarang, posisi)) {
                    return false;
            }
        }
        i = alphaList[i].nextAlphabet;
        if (i == firstAlphabetIndex) {
            selesai = true;
        }
    }
    return true;
}

bool cariPosisiSelanjutnya(int dimensi) {
    int startBaris, startKolom;
    int currIdx = currentAlphabetIndex;
    char charTujuan = (char) (currIdx + 65);

    if (alphaList[currIdx].posisiSekarang.baris == -1) {
        startBaris = 0;
        startKolom = 0;
    } else {
        int barisSekarangTemp = alphaList[currIdx].posisiSekarang.baris;
        int kolomSekarangTemp = alphaList[currIdx].posisiSekarang.kolom;
        matriks[barisSekarangTemp][kolomSekarangTemp] = charTujuan;
        startBaris = barisSekarangTemp;
        startKolom = kolomSekarangTemp + 1;
        alphaList[currIdx].posisiSekarang = buatPosisi(-1, -1);
    }

    for (int i = startBaris; i < dimensi; i++) {
        int starBaris2 = (i == startBaris) ? startKolom : 0;
```

```
        for (int j = starBaris2; j < dimensi; j++) {
            if (matriks[i][j] == charTujuan) {
                Posisi cekPosisi = buatPosisi(i, j);
                if (apkhPosisiValid(cekPosisi)) {
                    alphaList[currIdx].posisiSekarang = cekPosisi;
                    matriks[i][j] = charTujuan + 32;
                    return true;
                }
            }
        }
    }
    return false;
}

int findAlphabetBefore(int indeksAlphabet) {
    int i = firstAlphabetIndex;
    bool selesai = false;
    while (!selesai) {
        if (alphaList[i].nextAlphabet == indeksAlphabet) {
            return i;
        }
        i = alphaList[i].nextAlphabet;
        if (i == firstAlphabetIndex) {
            selesai = true;
        }
    }
    return -1;
}

int main() {
    FILE *fptr;
    char namaFile[1000];
    printf("%s", "Selamat datang di program Solusi Papan Queens!\n");
    printf("%s", "Masukkan nama file: ");
    fgets(namaFile, sizeof(namaFile), stdin);
    size_t len = strlen(namaFile);
    if (len > 0 && namaFile[len-1] == '\n') {
        namaFile[len-1] = '\0';
    }
```

```
    printf("\nMembuka file...\n");
    fptr = fopen(namaFile, "r");
    if (fptr == NULL) {
        char namaFileTxt[1005];
        strcpy(namaFileTxt, namaFile);
        strcat(namaFileTxt, ".txt");
        fptr = fopen(namaFileTxt, "r");
        if (fptr == NULL) {
            printf("File tidak ditemukan!\n");
            return 1;
        }
    }
    printf("File berhasil dibuka\n");
    printf("\nMembaca file...\n");
    int jumlahKolom = 0;
    int jumlahBaris = 0;
    char barisKarakter[600];
    buatAlphaList(alphaList);
    firstAlphabetIndex = -1;
    currentAlphabetIndex = -1;

    while (fgets(barisKarakter, sizeof(barisKarakter), fptr) != NULL) {
        int panjangBarisAsliTemp = strlen(barisKarakter);
        if (panjangBarisAsliTemp > 0 &&
barisKarakter[panjangBarisAsliTemp-1] == '\n') {
            barisKarakter[panjangBarisAsliTemp-1] = '\0';
        }
        int panjangBarisValidTemp =
panjangBarisValiddanSimpan(barisKarakter, strlen(barisKarakter),
jumlahBaris);
        if (panjangBarisValidTemp == 0) {
            continue;
        }
        jumlahBaris++;
        if (panjangBarisValidTemp != -1 && jumlahBaris == 1) {
            jumlahKolom = panjangBarisValidTemp;
        }
        if (jumlahBaris > jumlahKolom) {
            printf("Kesalahan: dimensi papan tidak persegi\n");
            fclose(fptr); return 1;
```

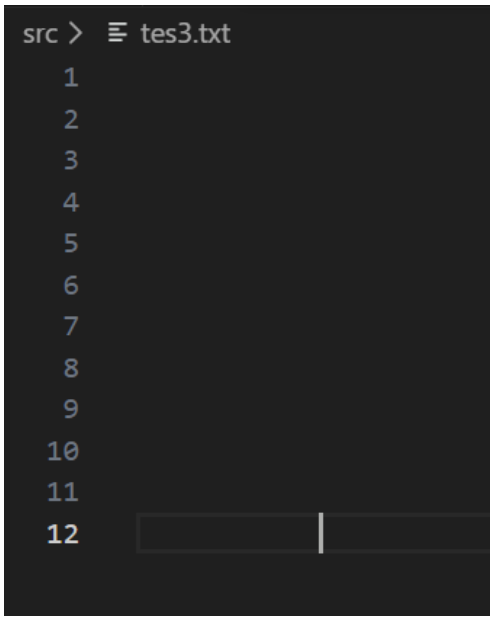
```
        }
        if (panjangBarisValidTemp == -1) {
            printf("Kesalahan: ditemukan karakter tidak valid pada baris
%d papan\n", jumlahBaris);
            fclose(fptr); return 1;
        }
        if (panjangBarisValidTemp != jumlahKolom) {
            printf("Kesalahan: baris %d papan tidak konsisten\n",
jumlahBaris);
            fclose(fptr); return 1;
        }
    }

    int totalAlfabet = dapatBanyakAlphabet();
    printf("Dimensi: %d x %d\nJumlah warna: %d\n", jumlahBaris,
jumlahKolom, totalAlfabet);
    if (jumlahBaris != jumlahKolom) {
        printf("Kesalahan: dimensi papan tidak persegi\n");
        fclose(fptr); return 1;
    }
    if (totalAlfabet != jumlahBaris) {
        printf("Kesalahan: jumlah warna (huruf) tidak sama dengan dimensi
papan\n");
        fclose(fptr); return 1;
    }
    if (jumlahBaris == 0) {
        printf("File kosong\n");
        fclose(fptr); return 1;
    }

    printf("\nPapan awal:\n");
    displayMatriks(jumlahBaris, jumlahKolom);

    printf("\nMencari solusi...\n");
    currentAlphabetIndex = firstAlphabetIndex;
    bool selesai = false;
    bool isSolved = false;
    int i = 1;
    int iterasi = 0;
    clock_t waktuMulai = clock();
```

```
    while (!selesai) {
        iterasi++;
        if (currentAlphabetIndex == firstAlphabetIndex) {
            printf("Pencarian ke-%d (dari warna awal)\n", i);
            i++;
        }
        bool isFound = cariPosisiSelanjutnya(jumlahBaris);

        if (isFound) {
            if (alphaList[currentAlphabetIndex].nextAlphabet ==
firstAlphabetIndex) {
                isSolved = true;
                selesai = true;
            } else {
                currentAlphabetIndex =
alphaList[currentAlphabetIndex].nextAlphabet;
            }
        } else {
            alphaList[currentAlphabetIndex].posisiSekarang =
buatPosisi(-1, -1);

            if (currentAlphabetIndex == firstAlphabetIndex) {
                selesai = true;
            } else {
                currentAlphabetIndex =
findAlphabetBefore(currentAlphabetIndex);
            }
        }
    }
    clock_t waktuSelesai = clock();
    double waktuPencarian = ((double)(waktuSelesai - waktuMulai) /
CLOCKS_PER_SEC) * 1000.0;

    printf("Pencarian solusi selesai\n");
    printf("Waktu pencarian: %.2f ms\n", waktuPencarian);
    printf("Jumlah iterasi: %d\n", iterasi);
    if (isSolved) {
        printf("\nSolusi ditemukan (Huruf kecil adalah Ratu):\n");
        displayMatriks(jumlahBaris, jumlahKolom);
    } else {
```

```
        printf("Tidak ditemukan solusi\n");
    }

    fclose(fptr);
    return 0;
}
```

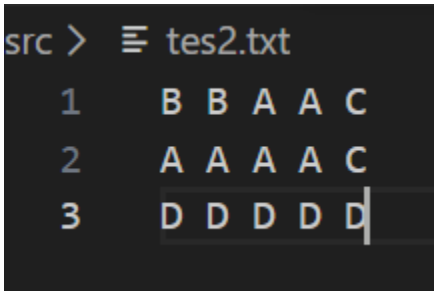
3 Tangkapan Layar Pengujian

3.1 Tangkapan Layar Pengujian 1



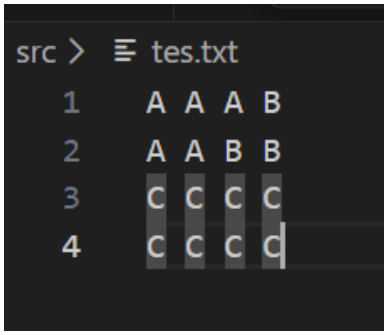

```
Selamat datang di program Solusi Papan Queens!
Masukkan nama file: tes3

Membuka file...
File berhasil dibuka

Membaca file...
Dimensi: 0 x 0
Jumlah warna: 0
File kosong
```

3.2 Tangkapan Layar Pengujian 2



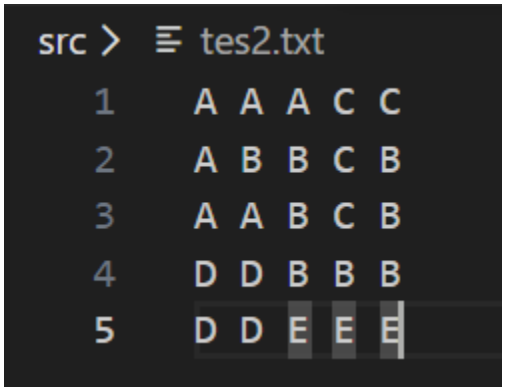

```
Selamat datang di program Solusi Papan Queens!
Masukkan nama file: tes2.txt

Membuka file...
File berhasil dibuka

Membaca file...
Dimensi: 3 x 5
Jumlah warna: 4
Kesalahan: dimensi papan tidak persegi
```

3.3 Tangkapan Layar Pengujian 3




```
Selamat datang di program Solusi Papan Queens!
Masukkan nama file: tes

Membuka file...
File berhasil dibuka

Membaca file...
Dimensi: 4 x 4
Jumlah warna: 3
Kesalahan: jumlah warna (huruf) tidak sama dengan dimensi papan
```

3.4 Tangkapan Layar Pengujian 4




```
Selamat datang di program Solusi Papan Queens!
Masukkan nama file: tes2

Membuka file...
File berhasil dibuka

Membaca file...
Dimensi: 5 x 5
Jumlah warna: 5

Papan awal:
A A A C C
A B B C B
A A B C B
D D B B B
D D E E E

Mencari solusi...
Pencarian ke-1 (dari warna awal)
Pencarian ke-2 (dari warna awal)
Pencarian ke-3 (dari warna awal)
Pencarian ke-4 (dari warna awal)
Pencarian ke-5 (dari warna awal)
Pencarian ke-6 (dari warna awal)
Pencarian ke-7 (dari warna awal)
Pencarian solusi selesai
Waktu pencarian: 0.00 ms
Jumlah iterasi: 115
Tidak ditemukan solusi
```

3.5 Tangkapan Layar Pengujian 5

src > tes2.txt

```
1   A A A F F G G G F
2   B A F F F F G F F
3   B A E E E F G F F
4   C C C E F F F F F
5   D C D E H H I I I
6   D C D D D D D I D
7   D D D H H H D I D
8   D D D D H D D D D
9   D D D D H D D D D
```

```
Papan awal:
A A A F F G G G F
B A F F F F G F F
B A E E E F G F F
C C C E F F F F F
D C D E H H I I I
D C D D D D D I D
D D D H H H D I D
D D D D H D D D D
D D D D H D D D D

Mencari solusi...
Pencarian ke-1 (dari warna awal)
Pencarian ke-2 (dari warna awal)
Pencarian ke-3 (dari warna awal)
Pencarian solusi selesai
Waktu pencarian: 0.00 ms
Jumlah iterasi: 179

Solusi ditemukan (Huruf kecil adalah Ratu):
A A a F F G G G F
B A F F F F g F F
b A E E E F G F F
C C C E F f F F F
D C D e H H I I I
D c D D D D D I D
D D D H H H D i D
D D D D h D D D D
D D D D H D D D d
```

3.6 Tangkapan Layar Pengujian 6

src > tes3.txt

```
1   A     B B B B
2   C C C C C
3
4   C C C C C
5   D D E E E
6
7
8
9   D D D E  E
```

```
Selamat datang di program Solusi Papan Queens!
Masukkan nama file: tes3

Membuka file...
File berhasil dibuka

Membaca file...
Dimensi: 5 x 5
Jumlah warna: 5

Papan awal:
A B B B B
C C C C C
C C C C C
D D E E E
D D D E E

Mencari solusi...
Pencarian ke-1 (dari warna awal)
Pencarian ke-2 (dari warna awal)
Pencarian solusi selesai
Waktu pencarian: 0.00 ms
Jumlah iterasi: 3
Tidak ditemukan solusi
```

4 Link Repositori Github

https://github.com/faiz410/QueensBoardSolver.git

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

Salman Faiz Assidqi