Laporan Tugas Kecil 1 :
Penyelesaian Permainan Queens Linkedin dengan Algoritma Brute Force

1. Algoritma Brute Force

   Algoritma yang digunakan pada tugas ini adalah algoritma brute force dengan basis permutasi. Di mana langkah langkah logikanya adalah sebagai berikut :

   - Pembentukkan representasi, di mana posisi Queen di simpan dalam array q dengan indeks dalam array mewakili baris, dan nilai dalam array mewakili kolom. Contoh q[0] = 1, berarti Queen berada di baris ke-0 dan kolom ke-1.
   - Dengan menggunakan permutasi dari angka 0 sampai N-1, program secara otomatis menjamin bahwa tidak akan ada Queen yang berada di satu baris atau kolom yang sama dengan Queen lainnya.
   - Program membuat susunan posisi Queen yang baru, di mana setiap susunan akan dianggap sebagai salah satu kandidat solusi
   - Setiap kandidat yang muncul, dilakukan pengecekan terhadap 2 aturan, yaitu aturan diagonal yang mengecek apakah Queen menempel secara diagonal. Lalu aturan warna yang mengecek apakah huruf (A-Z) pada posisi Queen tersebut sudah pernah digunakan oleh Queen lain.
   - Program akan berhenti ketika menemukan satu susunan yang memenuhi semua syarat atau semua kemungkinan susunan telah diperiksa.

2. Validasi Input

   Sebelum algoritma dijalankan, program melakukan validasi untuk memastikan kebenaran data input, yaitu mengecek apakah baris = kolom (persegi) atau tidak. Jika bukan persegi, maka program akan memberikan pesan error dan berhenti.

3. Source Program (menggunakan bahasa C)

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

bool isValid(int q[], int n, char **grid){
    bool color_used[26] = {false};
    for (int r = 0; r < n; r++){
        int c = q[r];
         int color_idx = grid[r][c] - 'A';//buat A jadi 0, B jadi
1, dan seterusnya

        //validasi agar hanya A-Z yang dibaca
        if (color_idx < 0 || color_idx > 26){
            continue;
        }
        //jika warna sudah dipakai maka return false
        if (color_used[color_idx]){
```

```
            return false;
        }
         color_used[color_idx] = true;//jika warna belum dipakai,
maka buat menjadi true

        if (r > 0){
            if (abs(q[r] - q[r - 1]) <= 1){
                return false;
            }

        }

    }
    return true;
}


// fungsi untuk mencari semua kemungkinan posisi queen dengan
baris dan kolom sudah pasti berbeda
bool nextPerm(int q[], int n){
    int i = n -2;
    while (i >= 0 && q[i] >= q[i + 1]){
        i--;
    }
     if (i < 0){ //jika i < 0, maka semua variasi posisi queen
sudah dicoba
        return false;
    }
    int j = n -1;
    while (q[j] <= q[i]){
        j--;
    }
    int temp = q[i];
    q[i] = q[j];
    q[j] = temp;

    for (int k = i +1,l = n -1; k < l; k++, l--){
        temp = q[k];
        q[k] = q[l];
        q[l] = temp;
    }
    return true;
}
```

```c
int main(){
    char nama_file[100], character;
    int n; //n sebagai ukuran grid (n x n)
    printf("masukkan nama file yang ingin dibuka : ");
    scanf("%s", nama_file);

    FILE *file = fopen(nama_file, "r");
    if (file == NULL)
    {
          printf("Gagal membuka file %s. Pastikan nama file sudah sesuai\n", nama_file);
        return 1;
    }
    printf("File %s berhasil dibuka\n", nama_file);

    //Validasi baris dan kolom
    int count = 0; int rows = 0;
    char line[1024];
    while (fgets((line), sizeof(line), file))
    {
        int character_in_line = 0;
        for (int i = 0; i < line[i] != '\0'; i++)
        {
            if (line[i] >= 'A' && line[i] <= 'Z')
            {
                character_in_line++;;
            }

        }
        if (character_in_line > 0)
        {
            rows++;
            count += character_in_line;
        }

    }
    if (rows * rows != count){
          printf("Error, Matriks harus berbentuk persegi (baris = kolom)\n");
        fclose(file);
        return 1;
    }
```

```
    n = rows;

    rewind(file);//muat file dari baris awal
    char **grid = (char **)malloc(n * sizeof(char *));
    for (int i = 0; i < n; i++)
    {
        grid[i] = (char *)malloc(n * sizeof(char));
    }

    int *q = (int *)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
                fscanf(file, " %c", &grid[i][j]);//memasukkan isi
dari file ke grid
        }
          q[i] = i; //inisialisasi queen awal dengan indeks array
sebagai baris, dan nilai array sebagai kolom
    }
    fclose(file);

      printf("Mencari solusi untuk persoalan Queens secara Brute
Force...\n\n");
    clock_t start_time = clock();
    unsigned long long attempts = 0;
    bool found = false;

    do
    {
        attempts++;
            if (attempts % 1000 == 0) //live update setiap 1000
percobaan
        {
            printf("percobaan ke %llu : \n", attempts);
            for (int i = 0; i < n; i++){
                for (int j = 0; j < n; j++){
                    if (q[i] == j){
                        printf("#");
                    }else{
                        printf("%c", grid[i][j]);
                    }
```

```
                }
                printf("\n");
            }
            printf("\n");
        }

        if (isValid(q, n, grid))
        {
            found = true;
            break;
        }

    } while (nextPerm(q, n)); //mencoba percobaan berikutnya jika masih valid

    clock_t end_time = clock();
          double  time  =  ((double)(end_time  -  start_time)*1000)/CLOCKS_PER_SEC;
    if (found)
    {
        printf("Solusi ditemukan : \n");
        for (int i = 0; i < n; i++){
            for (int j = 0; j < n; j++){
                if (q[i] == j){
                    printf("#");
                }else{
                    printf("%c", grid[i][j]);
                }

            }
            printf("\n");
        }
        char pilihan[10];
        printf("Waktu pencarian : %.2f ms\n", time);
            printf("Banyak kasus yang ditinjau : %llu kasus\n", attempts);
          printf("Apakah Anda ingin menyimpan solusi? (Ya/Tidak)");
        scanf("%s", pilihan);
        if (strcmp(pilihan, "Ya") == 0)
        {
            char nama_file_solusi[100];
            char path[150];
```

```
            printf("Masukkan nama file solusi : ");
            scanf("%s", nama_file_solusi);

            sprintf(path, "test/%s", nama_file_solusi);

            FILE *solusi = fopen(path, "w");
            if (solusi == NULL)
            {
                    printf("Gagal menyimpan, pastikan folder sudah
tersedia\n");
            }else{
                for (int i = 0; i < n; i++){
                    for (int j = 0; j < n; j++){
                        if (q[i] == j){
                            fprintf(solusi,"#");
                        }else{
                            fprintf(solusi, "%c", grid[i][j]);
                        }
                    }
                    fprintf(solusi, "\n");
                }
                fclose(solusi);
                printf("Solusi berhasil disimpan di %s\n", path);
            }

        }else if (strcmp(pilihan, "Tidak") == 0){
            printf("Solusi tidak disimpan\n");
        }else{
            printf("input tidak valid\n");
        }
    }else{
        printf("Tidak ada solusi\n");
    }

    for (int i = 0; i < n; i++)
    {
        free(grid[i]);
    }
    free(grid);
    free(q);
    return 0;
}
```

4. Contoh Input dan Output

Test case 1 :

Input :

```
A A A B B
C A B B B
C C E E B
C D D E E
D D D D E
```

Output :

```
Solusi ditemukan :
#AABB
CAB#B
C#EEB
CDDE#
DD#DE
Waktu pencarian : 0.00 ms
Banyak kasus yang ditinjau : 14 kasus
Apakah Anda ingin menyimpan solusi? (Ya/Tidak) Ya
Masukkan nama file solusi : solusi1.txt
Solusi berhasil disimpan di test/solusi1.txt
```

Test Case 2 :

Input :

```
A A A A A A B B B B B B
A A A A A A B B B B B B
C C C C C C D D D D D D
C C C C C C D D D D D D
E E E E E E F F F F F F
E E E E E E F F F F F F
G G G G G G H H H H H H
G G G G G G H H H H H H
I I I I I I J J J J J J
I I I I I I J J J J J J
K K K K K K L L L L L L
K K K K K K L L L L L L
```

Output :

```
Solusi ditemukan :
#AAAAABBBBBB
AAAAAA#BBBBB
C#CCCCDDDDDD
CCCCCCD#DDDD
EE#EEEFFFFFF
EEEEEEFF#FFF
GGG#GGHHHHHH
GGGGGGHHH#HH
IIII#IJJJJJJ
IIIIIIJJJJ#J
KKKKK#LLLLLL
KKKKKKLLLLL#
Waktu pencarian : 45225.00 ms
Banyak kasus yang ditinjau : 18307491 kali
Apakah Anda ingin menyimpan solusi? (Ya/Tidak)
```

Test Case 3 :

Input :

```
A A A A A A A
B B B B B B B
C C C C C C C
D D D D D D D
E E E E E E E
F F F F F F F
G G G G G G G
```

Output :

```
Solusi ditemukan :
#AAAAAA
BB#BBBB
CCCC#CC
D#DDDDD
EEEEE#E
FFF#FFF
GGGGGG#
Waktu pencarian : 0.00 ms
Banyak kasus yang ditinjau : 171 kasus
Apakah Anda ingin menyimpan solusi? (Ya/Tidak) Ya
Masukkan nama file solusi : solusi3.txt
Solusi berhasil disimpan di test/solusi3.txt
```

Test Case 4 :

Input :

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

Output :

```
Solusi ditemukan :
AAABBCC#D
ABBB#CECD
ABBBDC#CD
A#ABDCCCD
BBBBD#DDD
FGG#DDHDD
#GIGDDHDD
FG#GDDHDD
FGGGDDHH#
Waktu pencarian : 657.00 ms
Banyak kasus yang ditinjau : 306205 kasus
Apakah Anda ingin menyimpan solusi? (Ya/Tidak) Ya
Masukkan nama file solusi : solusi4.txt
Solusi berhasil disimpan di test/solusi4.txt
```

Test Cse 5 :

Input :

```
A B C D E
B A C D E
C C C D E
D D D D E
E E E E E
```

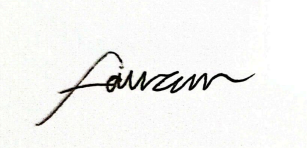
Output :

```
Mencari solusi untuk persoalan Queens secara Brute Force...

Tidak ada solusi
```

5. Pranala ke Repository :
   https://github.com/Zannn210/Tucil1_13524113

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



Fauzan Mohamad Abdul Ghani

Lampiran

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi 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		✓