TUGAS KECIL 1 IF2211

MATA KULIAH STRATEGI ALGORITMA

Laporan Penyelesaian Permainan Queens Linkedin

Disusun Oleh

Moreno Syawali Ganda Sugita

NIM 13524096

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA - KOMPUTASI

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2026

Daftar Isi

BAB I

Deskripsi Algoritma

Algoritma Brute Force memiliki keunggulan dalam menemukan solusi jika di persoalan tersebut memang ada. Walaupun begitu, Brute Force memiliki kekurangan dalam beban komputasi dan waktu pencarian. Sehingga penulisan program akan terlihat sederhana karena setiap persoalan diselesaikan secara *straightforward*. Dalam persoalan *Queens* tersebut, diperlukan metode *Generate and Test* untuk mencoba seluruh kombinasi bidak *Queen* sampai menemukan sebuah kombinasi yang merupakan solusi dari sebuah papan soal tersebut. Berikut merupakan langkah-langkah program penyelesaian permainan *Queens* Linkedin.

1. Inisialisasi

```
def ReadFiles(name):
    if os.path.exists(os.path.join("test", name)):
        path = os.path.join("test", name)
    else:
        return None, 0, f"File {name} tidak ditemukan di folder test/."

    try:
        with open(path, 'r') as file:
            lines = [line.strip() for line in file if line.strip()]

            if not lines:
                return None, 0, "File kosong"

            board = [list(line) for line in lines]
            m = len(board)
            n = len(board[0])

            # Ngecek papannya valid atau tidak
            for i, row in enumerate(board):
                if len(row) != n:
                    return None, 0, f"Baris {i+1} memiliki panjang
berbeda"
            if m != n:
                return None, 0, "Papan berbentuk persegi panjang. Tidak
Valid"

            for r in range(n):
                for c in range(n):
                    if board[r][c] == '#':
                        return None, 0, "File mengandung karakter '#'"

            unik = set()
            for r in range(n):
                for c in range(n):
                    unik.add(board[r][c])

            colorCnt = len(unik)
            if colorCnt != n:
```

```
                return None, 0, f"Terdapat {colorCnt} warna, seharusnya
{n} warna (sesuai ukuran papan)"

            return board, n, "Berhasil input file"

    except FileNotFoundError:
        return None, 0, "File tidak ditemukan"
```

Program membaca file .txt dan melakukan proses *parsing* menjadi *array of array of alphabet* sebagai papan soal tersebut. Program memvalidasi apakah papan tersebut memiliki ukuran yang sesuai (panjang dan lebar sama), memiliki jumlah daerah warna sesuai dengan ukurannya, tidak mengandung karakter ilegal '#', juga apakah file yang diminta ada di folder test/ dan tidak kosong.

2. Penyusunan Kombinasi

```
def solve(row, curr, board, n, show=None):
    global iter

    if row == n:
        iter += 1
        if show and iter % 1000 == 0: show(row, -1, curr, board, n)
        return IsItLegal(curr, board)

    for col in range(n):
        curr.append((row, col))
        if show and iter % 1000 == 0: show(row, col, curr, board, n)
        if solve(row + 1, curr, board, n, show): return True
        curr.pop()
        if show and iter % 1000 == 0: show(row, col, curr, board, n)

    return False
```

Program menempatkan bidak ratu di tiap baris papan dan menyusun seluruh kombinasi agar bisa dicek apakah papan solusi tersebut legal atau tidak. Kombinasi yang dihasilkan belum tentu legal secara peraturan permainan, sehingga belum tentu bidak tidak saling bentrok dengan bidak yang lainnya.

3. Validasi

```
def IsItLegal(curr, board):
    for i in range(len(curr)):
        r1, c1 = curr[i]
        color1 = board[r1][c1]
        for j in range(i + 1, len(curr)):
            r2, c2 = curr[j]
            color2 = board[r2][c2]
            if c1 == c2 or color1 == color2 or (abs(r1 - r2) <= 1 and
abs(c1 - c2) <= 1): return False
    return True
```

Setelah dihasilkan susunan baru, setiap bidak dicek satu persatu apakah melanggar aturan atau tidak, dengan aturan setiap bidak sebagai berikut.

a. Tiap baris hanya diisi oleh 1 ratu (ini otomatis terpenuhi karena program hanya meletakkan 1 ratu di setiap baris),
b. Tiap kolom hanya diisi oleh 1 ratu,
c. Tiap daerah warna hanya diisi oleh 1 ratu, dan
d. Tidak ada ratu yang bertetanggaan (walau secara diagonal).

4. Iterasi
   Jika ada salah satu bidak yang melanggar aturan tersebut, maka program akan mengecek ke susunan selanjutnya. Oleh karena itu, program akan menghasilkan n^n atau sampai menemukan solusi sebelum kombinasi terakhir.

BAB 2

Implementasi Program

Bahasa Pemrograman: Python 3.12
Library Eksternal: Pillow (PIL) untuk ekspor hasil solusi ke format gambar (PNG).
File Program:

main.py: Logika penyelesaian rekursif, fungsi validasi, dan pembacaan file.
gui.py: *Graphic interface* dengan Tkinter dan visualisasi kombinasi papan.

BAB 3

Hasil Test Case

Berikut merupakan hasil *Test Case* untuk program tersebut.

Test Case	Ukuran	Total Iterasi	Waktu (ms)	Solusi
AABB AABB CCDD CCDD	4x4	115	0.97	Ditemukan
AABBB AABCC DDECC DDEEE DDEEE	5x5	359	1.39	Ditemukan
AAABBCCCD ABBBBCECD ABBBDCECD AAABDCCCD BBBBDDDDD FGGGDDHDD FGIGDDHDD FGIGDDHDD FGGGDDHHH	9x9	323741637	1019466.37	Ditemukan
AAAAAAAA BBBBBBBB CCCCCCCC DDDDDDDD EEEEEEEE FFFFFFFF GGGGGGGG HHHHHHHH	8x8	16777216	44355.05	Tidak Ditemukan
File Kosong	-	-	-	File Tidak Valid

LAMPIRAN

Berikut merupakan *link Repository* kode program.

https://github.com/DeSince29/Tucil1_13524096

Berikut merupakan Pernyataan Tidak Melakukan Kecurangan.

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

(Moreno Syawali Ganda Sugita)