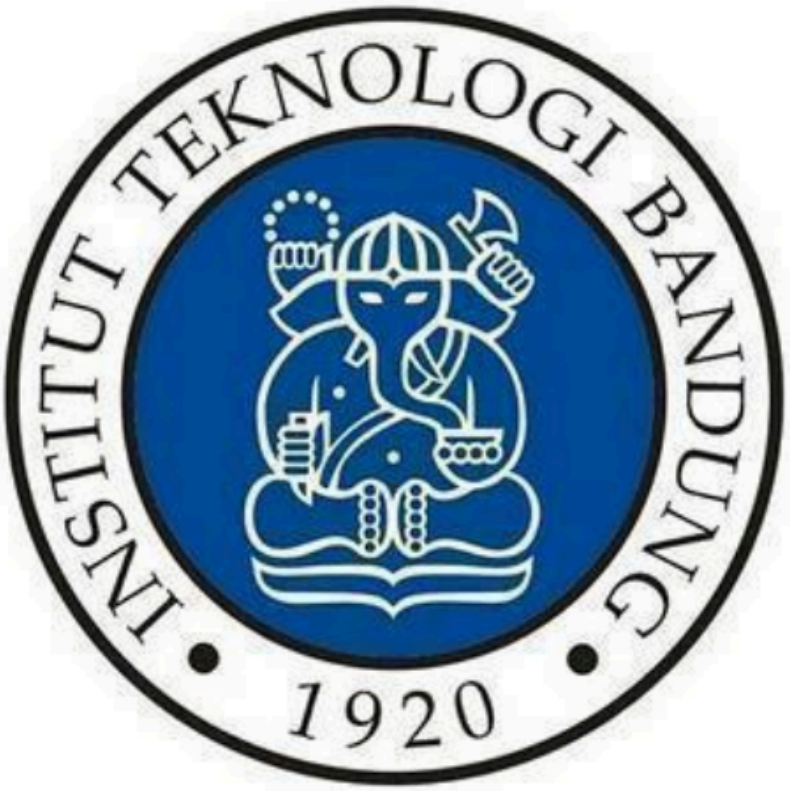
Laporan Tugas Kecil 1
IF2211 Strategi Algoritma
Penyelesaian Permainan Queens Linkedin

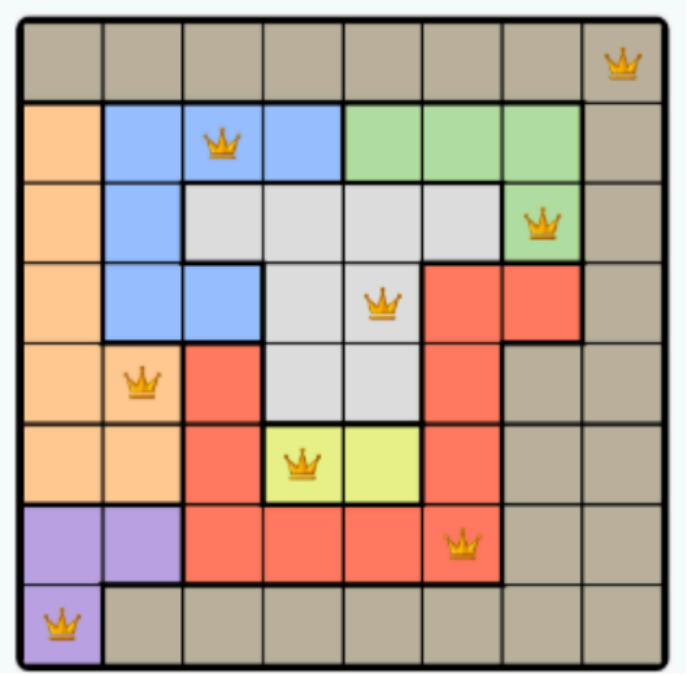


Disusun oleh:
Mirza Tsabita Wafa'Ana (13524114)
K-03

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung 2026

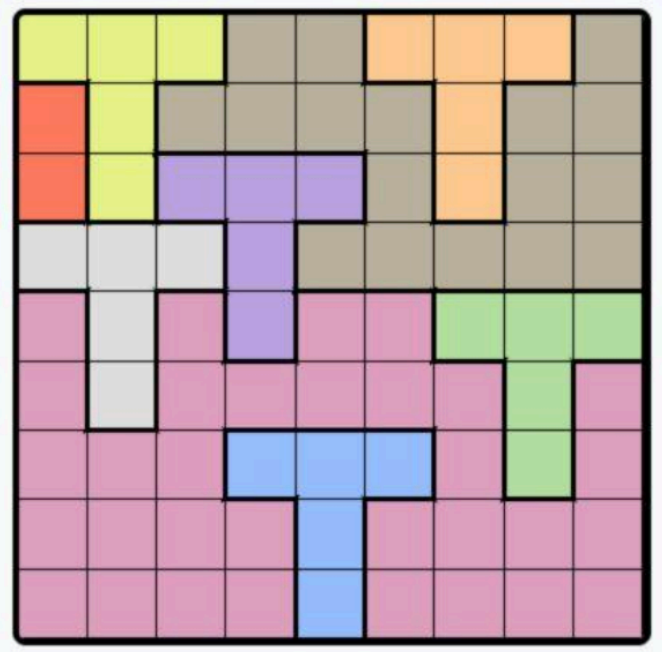
Daftar Isi

BAB 1 DESKRIPSI TUGAS

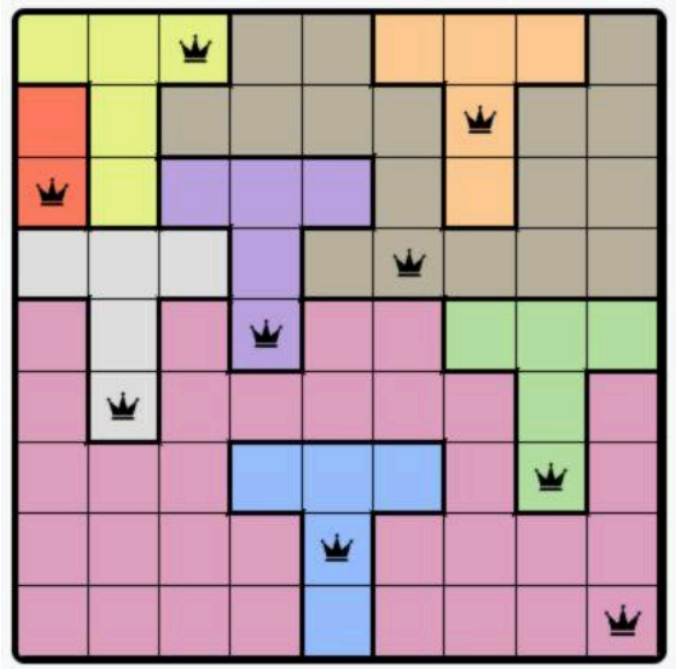


Queens adalah gim logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari gim ini adalah menempatkan queen pada sebuah papan persegi berwarna sehingga terdapat hanya satu queen pada tiap baris, kolom, dan daerah warna. Selain itu, satu queen tidak dapat ditempatkan bersebelahan dengan queen lainnya, termasuk secara diagonal. Tugas anda adalah membuat program yang dapat menemukan satu solusi penempatan queen pada suatu papan berwarna yang diberikan, atau menampilkan bahwa tidak ada solusi yang valid. Program melakukan pencarian solusi menggunakan algoritma brute force.

Ilustrasi Kasus
Diberikan papan sebagai berikut. Untuk tugas ini, papan selalu dimulai kosong.



Di bawah adalah satu-satunya solusi valid. Perhatikan bahwa tiap baris, kolom, dan daerah warna sudah memiliki satu ditempati satu queen.



BAB 2 PENYELESAIAN ALGORITMA

2.1. Algoritma Brute Force yang Digunakan

Pada tugas ini, penyelesaian permainan Queens dilakukan menggunakan algoritma brute force (exhaustive search) berbasis permutasi kolom. Algoritma bekerja dengan mencoba seluruh kemungkinan penempatan Queen yang mungkin tanpa melakukan pruning atau pemotongan jalur pencarian di tengah proses. Validasi aturan permainan dilakukan setelah seluruh kandidat posisi Queen terbentuk.

Algoritma tidak menggunakan heuristik maupun teknik backtracking bertahap. Seluruh kombinasi kandidat dihasilkan menggunakan metode permutasi leksikografis dan diuji satu per satu hingga ditemukan solusi atau seluruh kombinasi habis diperiksa.

Langkah-langkah algoritma yang digunakan adalah sebagai berikut:

1. Pembacaan dan Validasi Input
   Program meminta pengguna memasukkan path file test case (.txt). File dibaca dan direpresentasikan sebagai matriks dua dimensi (board). Program memastikan bahwa papan berbentuk persegi (N×N) dan jumlah wilayah unik sama dengan N.
2. Inisialisasi Permutasi Awal
   Program membuat array permutasi awal perm = [0,1,2,...,N-1] yang merepresentasikan posisi Queen:
   perm[r] = c berarti Queen pada baris r ditempatkan di kolom c.
   Karena menggunakan permutasi, otomatis setiap baris dan kolom hanya memiliki satu Queen.
3. Generasi Kandidat Solusi
   Program menggunakan fungsi permut() untuk menghasilkan seluruh permutasi kolom secara berurutan (next permutation). Setiap permutasi merepresentasikan satu kandidat konfigurasi papan.
4. Validasi Wilayah (Region Check)
   Untuk setiap kandidat, program memeriksa apakah setiap Queen berada pada wilayah yang berbeda. Jika ada dua Queen dalam wilayah yang sama, kandidat langsung dianggap tidak valid.
5. Validasi Adjacency (Ketetanggaan)
   Program memeriksa apakah ada dua Queen yang saling bersebelahan dalam 8 arah (atas, bawah, kiri, kanan, dan diagonal dengan jarak 1). Jika ada pelanggaran, kandidat ditolak.
6. Penghentian Pencarian
   - Jika ditemukan konfigurasi valid, program berhenti dan menampilkan hasil.
   - Jika seluruh permutasi telah diperiksa dan tidak ada solusi, program menampilkan pesan “Tak ada solusi”.
7. Output dan Penyimpanan
   Jika solusi ditemukan, papan akan ditampilkan dengan simbol # menggantikan posisi

Queen. Program juga menampilkan waktu pencarian dan jumlah kasus yang ditinjau. Pengguna dapat memilih untuk menyimpan hasil ke file output.

2.2. Implementasi

Nama	Jenis	Deskripsi
permut(a)	Fungsi	Menghasilkan *permutasi berikutnya* (next lexicographic permutation) dari array a. Mengembalikan False jika sudah tidak ada permutasi lagi.
baca(path)	Fungsi	Membaca file .txt, menghapus spasi/baris kosong, memastikan papan persegi (N×N), lalu mengubahnya menjadi np.ndarray karakter.
validasi(board)	Prosedur	Memastikan board 2D, persegi, dan jumlah wilayah unik = N (sesuai aturan "1 queen per wilayah"). Jika tidak valid → raise error.
wilayahyah(perm, board)	Fungsi	Validasi wilayah: mengambil wilayah di posisi (r, perm[r]) untuk semua baris, memastikan semua wilayah unik.
valadj(perm)	Fungsi	Validasi adjacency: memastikan tidak ada dua queen yang bersebelahan pada jarak 1 (8 arah). Jika ada → False.
SolBF(board)	Fungsi	Inti brute force: mencoba semua permutasi perm (total maksimal N!), mengecek wilayahyah dan valadj. Mengembalikan (perm_solusi/None, cases, ms).
solusi(board, perm)	Fungsi	Membuat salinan papan lalu menandai posisi queen dengan # sesuai perm.
print_board(board)	Prosedur	Mencetak papan ke terminal (tiap baris digabung jadi string).
main()	Prosedur	Alur program: minta path → baca + validasi → jalankan brute force → tampilkan hasil/waktu/kasus → opsi simpan ke file.
if __name__ == "__main__": main()	Entry Point	Menjalankan program saat file dieksekusi langsung (python Queen.py).
board	np.ndarray (N×N, char)	Menyimpan papan permainan (matriks karakter wilayah/warna) hasil pembacaan file .txt.
path	str	Menyimpan path file test case yang

		dimasukkan pengguna lewat input().
n	int	Menyimpan ukuran papan (N) yang didapat dari board.shape[0].
perm	np.ndarray (int)	Kandidat solusi berbentuk permutasi kolom. perm[r] = c artinya Queen di baris r diletakkan pada kolom c.
cases	int	Menghitung berapa banyak kandidat permutasi (kasus) yang sudah diperiksa.
t0, t1	float	Waktu awal dan akhir untuk mengukur durasi pencarian (menggunakan time.perf_counter()).
ms	float	Lama pencarian dalam milidetik.
found	bool	Penanda apakah solusi sudah ditemukan atau belum (di SolBF).
best	np.ndarray (int) / None	Menyimpan permutasi terbaik/solusi saat ditemukan. None jika tidak ada solusi.
wilayah	set	Himpunan karakter wilayah unik yang ditemukan pada papan (untuk validasi jumlah wilayah).

2.3 Source Code

```
import numpy as np
import time


def permut(a: np.ndarray) -> bool:
    i = a.size - 2
    while i >= 0 and a[i] >= a[i + 1]:
        i -= 1
    if i < 0:
        return False
```

```
    j = a.size - 1
    while a[j] <= a[i]:
        j -= 1

    a[i], a[j] = a[j], a[i]
    a[i + 1:] = a[i + 1:][::-1]
    return True

def baca(path: str) -> np.ndarray:
    f = open(path, "r", encoding="utf-8")
    lines = []
    for ln in f:
        ln = ln.strip()
        ln = ln.replace(" ", "")
        if ln != "":
            lines.append(ln)

    f.close()

    if len(lines) == 0:
        raise ValueError(" Ga ada file")

    n = len(lines)
    i = 0
```

```
    while i < n:
        if len(lines[i]) != n:
            raise ValueError("data ndak persegi lo")
        i += 1

    board_list = []
    i = 0
    while i < n:
        row = []
        j = 0
        while j < n:
            row.append(lines[i][j])
            j += 1
        board_list.append(row)
        i += 1

    board = np.array(board_list, dtype="<U1")
    return board

def validasi(board: np.ndarray) -> None:
    if len(board.shape) != 2:
        raise ValueError("2D kan")

    n_baris = board.shape[0]
```

```
    n_kolom = board.shape[1]
    if n_baris != n_kolom:
        raise ValueError("Persegi kan")

    n = n_baris

    wilayah = set()
    i = 0
    while i < n:
        j = 0
        while j < n:
            wilayah.add(board[i][j])
            j += 1
        i += 1

    if len(wilayah) != n:
        raise ValueError("Input ndak sama")

def wilayahyah(perm: np.ndarray, board: np.ndarray) -> bool:
    n = len(perm)
    dipakai = set()

    for r in range(n):
        c = perm[r]
```

```
        reg = board[r][c]
        if reg in dipakai:
            return False
        dipakai.add(reg)

    return True

def valadj(perm: np.ndarray) -> bool:
    n = len(perm)

    for r1 in range(n):
        c1 = perm[r1]
        for r2 in range(r1 + 1, n):
            c2 = perm[r2]

            if abs(r1 - r2) <= 1 and abs(c1 - c2) <= 1:
                return False

    return True

import numpy as np
import time

def SolBF(board: np.ndarray):
```

```
    n = board.shape[0]
    perm = np.arange(n, dtype=int)
    cases = 0
    t0 = time.perf_counter()
    found = False
    best = None
    while True:
        cases = cases + 1
        if wilayahyah(perm, board) and valadj(perm):
            found = True
            best = perm.copy()
            break
        ok = permut(perm)
        if ok == False:
            break
    t1 = time.perf_counter()
    ms = (t1 - t0) * 1000.0
    if found:
        return best, cases, ms
    else:
        return None, cases, ms


def solusi(board: np.ndarray, perm: np.ndarray) -> np.ndarray:
    out = board.copy()
```

```
    n = len(perm)
    for r in range(n):
        c = perm[r]
        out[r][c] = "#"
    return out

def print_board(board: np.ndarray) -> None:
    for r in range(board.shape[0]):
        print("".join(board[r].tolist()))

def main():
    # sesuai spesifikasi: program memberi arahan pilih file
    path = input("Masukkan path file test case (.txt): ").strip().strip('"').strip("'")

    try:
        board = baca(path)
        validasi(board)
    except Exception as e:
        print(f"Input tidak valid: {e}")
        return

    perm, cases, ms = SolBF(board)
    if perm is None:
        print("Tak ada solusi")
```

```
        print(f"Waktunya: {ms:.0f} ms")
        print(f"Jumlah : {cases} kasus")
        return
    solved = solusi(board, perm)
    # Output papan + info seperti contoh
    print_board(solved)
    print()
    print(f"Waktu pencarian: {ms:.0f} ms")
    print(f"Banyak kasus yang ditinjau: {cases} kasus")

    ans = input("Apakah Anda ingin menyimpan solusi? (Ya/Tidak):
").strip().lower()
    if ans in ("ya", "y", "yes"):
        out_path = input("Masukkan nama file output (misal: solusi.txt): ").strip()
        if out_path == "":
            out_path = "solusi.txt"
        with open(out_path, "w", encoding="utf-8") as f:
            for r in range(solved.shape[0]):
                f.write("".join(solved[r].tolist()) + "\n")
            f.write("\n")
            f.write(f"Waktu pencarian: {ms:.0f} ms\n")
            f.write(f"Banyak kasus yang ditinjau: {cases} kasus\n")
        print(f"Solusi disimpan ke: {out_path}")
```

```
if __name__ == "__main__":
    main()
```

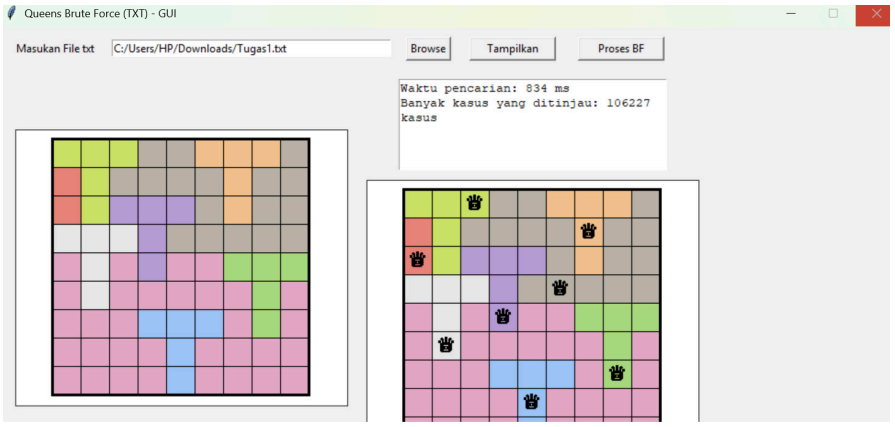
BAB 3 EKSPERIMEN

3.1. TestCase 1

Input:

```
AAABBCCCB
DABBBBCBB
DAEEEBCBB
FFFEBBBBB
GFGEGGIII
GFGGGGGIG
GGGHHHGIG
GGGGHGGGG
GGGGHGGGG
```

Output:



BAB 4 LAMPIRAN

4.1. Link Repository Program

Program dapat diakses pada link sebagai berikut:
https://github.com/Mirza114/Tucil1_13524114.git

4.2. Pernyataan Tidak Melakukan Kecurangan

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

Mirza Tsabita W.

4.3. Tabel Checklist

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	✓	