

**LAPORAN TUGAS KECIL 1**  
**IF2211 STRATEGI ALGORITMA**  
**PENYELESAIAN PERMAINAN *QUEENS LINKEDIN* DENGAN**  
**ALGORITMA *BRUTE FORCE***



**Oleh:**  
**Fazri Arrashyi Putra**  
**13524127**

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

## DAFTAR ISI

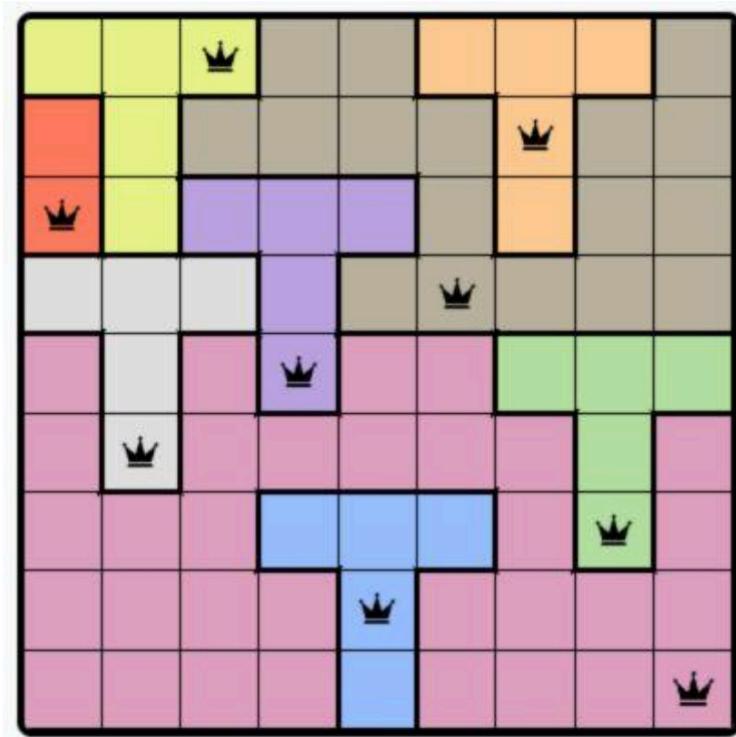
<b>DAFTAR ISI</b>	<b>2</b>
<b>BAB I</b>	<b>3</b>
<b>DESKRIPSI PROGRAM</b>	<b>3</b>
<b>BAB II</b>	<b>4</b>
<b>ALGORITMA PEMROGRAMAN</b>	<b>4</b>
A. Konsep Brute Force	5
B. Implementasi Khusus untuk N-Queens	5
<b>BAB III</b>	<b>6</b>
<b>KODE SUMBER</b>	<b>6</b>
A. Algoritma Brute Force yang digunakan	6
B. Kode Sumber Lengkap	9
<b>BAB IV</b>	<b>15</b>
<b>MASUKKAN dan LUARAN PROGRAM</b>	<b>15</b>
A. Test Case 1	15
B. Test Case 2	16
C. Test Case 3	18
D. Test Case 4	19
E. Test Case 5	20
<b>BAB V</b>	<b>22</b>
<b>LAMPIRAN</b>	<b>22</b>
<b>BAB VI</b>	<b>23</b>
<b>TAUTAN dan REFERENSI</b>	<b>23</b>
A. Pranala Github	23
B. Referensi	23

## BAB I

### DESKRIPSI PROGRAM

Permainan queens merupakan permainan yang terdapat pada platform profesional yaitu LinkedIn, permainan ini melatih logika dan cara berpikir bagaimana menempatkan sejumlah ratu (queens) pada papan catur berukuran  $N \times N$  sehingga tidak ada dua ratu pun yang saling menyerang satu sama lain. Tujuan utama dari permainan ini adalah menemukan semua kemungkinan konfigurasi penempatan ratu yang valid pada papan.

Program ini bertujuan untuk menyelesaikan permainan Queens LinkedIn, yang pada dasarnya adalah masalah N-Queens, dengan menerapkan Algoritma Brute Force. Algoritma Brute Force akan mengecek secara sistematis setiap kemungkinan penempatan ratu pada papan catur untuk mencari solusi yang memenuhi syarat, yaitu tidak ada ratu yang berada pada baris, kolom, atau diagonal yang sama dengan ratu lainnya. Pendekatan ini dipilih untuk menganalisis dan membandingkan efektivitasnya dalam menyelesaikan masalah ini.



## BAB II

### ALGORITMA PEMROGRAMAN

Algoritma yang digunakan untuk menyelesaikan permainan *Queens LinkedIn* (masalah N-Queens) dalam tugas ini adalah **Algoritma Brute Force**.

#### A. Konsep Brute Force

Algoritma Brute Force adalah algoritma yang menyelesaikan persoalan secara lempang (straightforward), yang memecahkan persoalan secara sederhana, langsung, jelas caranya dan langkah-langkahnya mudah dipahami. Dalam konteks masalah N-Queens, ini berarti program akan:

1. Mencoba semua kemungkinan penempatan ratu yang mungkin pada papan sebanyak  $N \times N$ .
2. Memverifikasi penempatan, memastikan hanya satu ratu yang ditempatkan pada baris, kolom dan warna daerahnya.
3. Jika konfigurasi valid dan semua  $N$  ratu telah ditempatkan maka konfigurasi tersebut dicatat sebagai salah satu solusi.
4. Jika verifikasi penempatan tidak valid, maka akan mencoba kombinasi penempatan yang lainnya.

#### B. Implementasi Khusus untuk N-Queens

Untuk mengimplementasikan Brute Force pada masalah Queens Puzzle secara efisien, program menggunakan rekursi dan backtracking untuk membangun solusi secara berurutan, baris per baris.

1. **Representasi Papan:** Papan direpresentasikan sebagai array dua dimensi (list of lists), di mana setiap elemen menyimpan karakter yang mewakili daerah (region) pada posisi tersebut. Posisi ratu yang sudah ditempatkan disimpan dalam list **daftar\_queen** yang berisi tuple (baris, kolom).
2. **Fungsi Rekursif:** Fungsi **\_brute\_force\_rekursif()** mencoba menempatkan ratu pada baris saat ini. Untuk setiap kolom yang mungkin di baris tersebut, fungsi akan:
  - a. **Pengecekan Konflik:** Memeriksa apakah penempatan pada kolom tersebut valid dengan memanggil **\_cek\_penempatan\_valid()**. Fungsi ini memeriksa tiga aturan:
    - Tidak ada ratu lain pada baris yang sama
    - Tidak ada ratu lain pada kolom yang sama
    - Tidak ada ratu lain pada daerah (region) yang sama
  - b. **Maju:** Jika penempatan valid, ratu ditambahkan ke **daftar\_queen** dengan **append((baris, kolom))**, dan fungsi rekursif dipanggil untuk mencoba menempatkan ratu pada baris berikutnya (baris + 1).

- c. **Mundur (Backtracking)**: Jika penempatan pada baris berikutnya gagal menghasilkan solusi (fungsi rekursif mengembalikan False), ratu pada baris saat ini dihapus dari **daftar\_queen** menggunakan **pop()** untuk mencoba kolom lain.

Meskipun algoritma Brute Force menjamin penemuan solusi karena mengecek setiap kemungkinan, efisiensinya rendah (kompleksitas waktu eksponensial,  $O(N^N)$  untuk papan berukuran  $N \times N$ ), sehingga tidak praktis untuk nilai  $N$  yang besar.

## BAB III

### KODE SUMBER

#### A. Algoritma Brute Force yang digunakan

Berikut adalah Algoritma Brute Force yang digunakan, penjelasan ada dalam komentar kode berikut:



```
 1 def selesaikan(self, visualisasi=True):
 2     # Algoritma Brute Force: Fungsi utama untuk memulai pencarian solusi
 3     # Inisialisasi: Reset counter dan daftar ratu
 4     self.kasus_ditinjau = 0
 5     daftar_queen = []
 6
 7     if visualisasi:
 8         print("\nMemulai pencarian solusi...")
 9         time.sleep(1)
10
11     # Mulai brute force dari baris 0
12     waktu_mulai = time.time()
13     hasil = self._brute_force_rekursif(daftar_queen, 0, visualisasi)
14     waktu_selesai = time.time()
15
16     # Hitung waktu eksekusi (I/O diabaikan)
17     self.waktu_pencarian_ms = int((waktu_selesai - waktu_mulai) * 1000)
18
19     if visualisasi and hasil:
20         self._tampilkan_papan(self.solusi, bersihkan_layar=True)
21
22     return hasil
```

```
● ● ●
1 def _brute_force_rekursif(self, daftar_queen, baris, visualisasi=True):
2     # Base case: Jika semua baris sudah diproses
3     if baris == self.ukuran:
4         # Cek apakah semua ratu sudah ditempatkan
5         if len(daftar_queen) == self.ukuran:
6             self.solusi = daftar_queen[:]
7             return True
8         return False
9
10    # Mencoba semua kolom untuk baris saat ini
11    for kolom in range(self.ukuran):
12        self.kasus_ditinjau += 1
13
14        # Cek apakah penempatan ratu valid
15        if self._cek_penempatan_valid(daftar_queen, baris, kolom):
16            # MAJU: Tambahkan ratu ke daftar
17            daftar_queen.append((baris, kolom))
18
19            # Visualisasi progress
20            if visualisasi and (self.kasus_ditinjau % 1000 == 0 or len(daftar_queen) <= 3):
21                self._tampilkan_papan(daftar_queen)
22                time.sleep(0.1)
23
24            # Rekursi: Coba menempatkan ratu pada baris berikutnya
25            if self._brute_force_rekursif(daftar_queen, baris + 1, visualisasi):
26                return True
27            # BACKTRACK: untuk mencoba kolom lain
28            daftar_queen.pop()
29
30    # Jika semua kolom sudah dicoba dan tidak ada yang valid, return False
31    return False
```

```
● ● ●
1 def _cek_penempatan_valid(self, daftar_queen, baris, kolom):
2     # Aturan 1: Cek apakah baris sudah ada ratu
3     for b, k in daftar_queen:
4         if b == baris:
5             return False
6     # Aturan 2: Cek apakah kolom sudah ada ratu
7     for b, k in daftar_queen:
8         if k == kolom:
9             return False
10    # Aturan 3: Cek apakah daerah (region) sudah ada ratu
11    karakter_daerah = self.papan[baris][kolom]
12    for b, k in daftar_queen:
13        if self.papan[b][k] == karakter_daerah:
14            return False
15
16    return True
```

## B. Kode Sumber Lengkap

```
● ● ●

1 import os
2 import time
3
4 class PuzzleQueens:
5     def __init__(self, papan):
6         self.papan = papan
7         self.ukuran = len(papan)
8         self.daerah = self._parse_daerah()
9         self.solusi = None
10        self.kasus_ditinjau = 0
11
12    def _parse_daerah(self):
13        daerah = {}
14        for i in range(self.ukuran):
15            for j in range(self.ukuran):
16                karakter_daerah = self.papan[i][j]
17                if karakter_daerah not in daerah:
18                    daerah[karakter_daerah] = []
19                    daerah[karakter_daerah].append((i, j))
20        return daerah
21
22    def _cek_penempatan_valid(self, daftar_queen, baris, kolom):
23        # Aturan 1: Cek apakah baris sudah ada ratu
24        for b, k in daftar_queen:
25            if b == baris:
26                return False
27        # Aturan 2: Cek apakah kolom sudah ada ratu
28        for b, k in daftar_queen:
29            if k == kolom:
30                return False
31        # Aturan 3: Cek apakah daerah (region) sudah ada ratu
32        karakter_daerah = self.papan[baris][kolom]
33        for b, k in daftar_queen:
34            if self.papan[b][k] == karakter_daerah:
35                return False
36
37    return True
```

```
 1 def _tampilkan_papan(self, daftar_queen, bersihkan_layar=True):
 2     if bersihkan_layar:
 3         os.system('clear' if os.name != 'nt' else 'cls')
 4
 5     print("\n" + "*50)
 6     print("Pencarian Solusi (Live Update)")
 7     print("*50 + "\n")
 8     papan_tampilan = [baris[:] for baris in self.papan]
 9     for b, k in daftar_queen:
10         papan_tampilan[b][k] = '#'
11
12     for baris in papan_tampilan:
13         print(''.join(baris))
14
15     print(f"\nRatu ditempatkan: {len(daftar_queen)}/{self.ukuran}")
16     print(f"Kasus yang ditinjau: {self.kasus_ditinjau}")
17     print("*50 + "\n")
18
19 def _brute_force_rekursif(self, daftar_queen, baris, visualisasi=True):
20     # Base case: Jika semua baris sudah diproses
21     if baris == self.ukuran:
22         # Cek apakah semua ratu sudah ditempatkan
23         if len(daftar_queen) == self.ukuran:
24             self.solusi = daftar_queen[:]
25             return True
26         return False
27
28     # Mencoba semua kolom untuk baris saat ini
29     for kolom in range(self.ukuran):
30         self.kasus_ditinjau += 1
31
32         # Cek apakah penempatan ratu valid
33         if self._cek_penempatan_valid(daftar_queen, baris, kolom):
34             # MAJU: Tambahkan ratu ke daftar
35             daftar_queen.append((baris, kolom))
36
37             # Visualisasi progress
38             if visualisasi and (self.kasus_ditinjau % 1000 == 0 or len(daftar_queen) <= 3):
39                 self._tampilkan_papan(daftar_queen)
40                 time.sleep(0.1)
41
42             # Rekursi: Coba menempatkan ratu pada baris berikutnya
43             if self._brute_force_rekursif(daftar_queen, baris + 1, visualisasi):
44                 return True
45             # BACKTRACK: untuk mencoba kolom lain
46             daftar_queen.pop()
47
48     # Jika semua kolom sudah dicoba dan tidak ada yang valid, return False
49     return False
```

```

● ● ●

1 def selesaikan(self, visualisasi=True):
2     # Algoritma Brute Force: Fungsi utama untuk memulai pencarian solusi
3     # Inisialisasi: Reset counter dan daftar ratu
4     self.kasus_ditinjau = 0
5     daftar_queen = []
6
7     if visualisasi:
8         print("\nMemulai pencarian solusi...")
9         time.sleep(1)
10
11    # Mulai brute force dari baris 0
12    waktu_mulai = time.time()
13    hasil = self._brute_force_rekursif(daftar_queen, 0, visualisasi)
14    waktu_selesai = time.time()
15
16    # Hitung waktu eksekusi (I/O diabaikan)
17    self.waktu_pencarian_ms = int((waktu_selesai - waktu_mulai) * 1000)
18
19    if visualisasi and hasil:
20        self._tampilkan_papan(self.solusi, bersihkan_layar=True)
21
22    return hasil
23
24 def dapatkan_papan_solusi(self):
25     if not self.solusi:
26         return None
27
28     papan_solusi = [baris[:] for baris in self.papan]
29     for b, k in self.solusi:
30         papan_solusi[b][k] = '#'
31
32     return papan_solusi
33
34 def cetak_solusi(self):
35     if not self.solusi:
36         print("Tidak ada solusi ditemukan!")
37         return
38
39     papan_solusi = self.dapatkan_papan_solusi()
40     print("\n" + "*50)
41     print("SOLUSI FINAL")
42     print("*50 + "\n")
43
44     for baris in papan_solusi:
45         print(''.join(baris))
46
47     print(f"\nWaktu pencarian: {self.waktu_pencarian_ms} ms")
48     print(f"\"Banyak kasus yang ditinjau: {self.kasus_ditinjau} kasus")
49     print("*50 + "\n")
50

```

```

1  def baca_file_papan(nama_file):
2      try:
3          if not os.path.isabs(nama_file):
4              base_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
5              test_case_dir = os.path.join(base_dir, 'test', 'case')
6              nama_file = os.path.join(test_case_dir, nama_file)
7
8          with open(nama_file, 'r') as file:
9              baris_baris = file.readlines()
10
11
12         papan = []
13         for baris in baris_baris:
14             baris = baris.strip()
15             if baris:
16                 papan.append(list(baris))
17
18     return papan
19 except FileNotFoundError:
20     print(f"Error: File '{nama_file}' tidak ditemukan!")
21     return None
22 except Exception as e:
23     print(f"Error membaca file: {e}")
24     return None
25
26
27 def validasi_papan(papan):
28     if not papan:
29         return False, "Papan kosong!"
30
31     ukuran = len(papan)
32
33     for i, baris in enumerate(papan):
34         if len(baris) != ukuran:
35             return False, f"Baris {i+1} memiliki panjang {len(baris)}, seharusnya {ukuran}"
36
37     set_daerah = set()
38     for baris in papan:
39         for karakter in baris:
40             set_daerah.add(karakter)
41
42     if len(set_daerah) < ukuran:
43         return False, f"Jumlah daerah ({len(set_daerah)}) kurang dari ukuran papan ({ukuran})"
44
45     jumlah_daerah = {}
46     for baris in papan:
47         for karakter in baris:
48             if karakter not in jumlah_daerah:
49                 jumlah_daerah[karakter] = 0
50                 jumlah_daerah[karakter] += 1
51
52     for daerah, jumlah in jumlah_daerah.items():
53         if jumlah == 0:
54             return False, f"Daerah '{daerah}' tidak memiliki cell"
55
56     return True, "Valid"
57
58

```



Library yang digunakan pada program ini ada dua yakni OS untuk input/output file dan time untuk menghitung waktu eksekusi serta fungsi sleep.

## **BAB IV**

### **MASUKKAN dan LUARAN PROGRAM**

#### **A. Test Case 1**

<b>Isi Test Case 1</b>
AAABBCCCD ABBBBCECD ABBBDCECD AAABDCCCD BBBBDDDDD FGGGDDHDD FGIGDDHDD FGGGDDHHH

<b>Program dijalankan dan memilih test case 1</b>
\$ ./ratu ratu.txt ===== Aturan: Setiap baris, kolom, dan daerah harus memiliki tepat satu ratu. Masukkan nama file test case (.txt): test_case_1 Papan berhasil dibaca! Ukuran: 9x9 Tampilkan live update saat pencarian? (Ya/Tidak): █

## Output dan hasil .txt

```
=====
SOLUSI FINAL
=====

A#ABCDD
ABBB#CECD
ABBBDC#CD
AAABD#CCD
BBBBDDD#D
#GGGDDHDD
FG#GDDHDD
FGI#DDHDD
FGGGDDHH#


Waktu pencarian: 7249 ms
Banyak kasus yang ditinjau: 5652 kasus
=====

Apakah Anda ingin menyimpan solusi? (Ya/Tidak): Ya
Solusi berhasil disimpan ke '/home/fazridep/Documents/Kuliah/Strategi Algoritma/Tucili/test/solutions/test_case_1_solution.txt'
~/Documents/Kuliah/Strategi Algoritma/Tucili  ✓ 53s Tucili Py 17:02:50
```

```
st > solutions > test
1 A#ABCDD
2 ABBB#CECD
3 ABBBDC#CD
4 AAABD#CCD
5 BBBBDDD#D
6 #GGGDDHDD
7 FG#GDDHDD
8 FGI#DDHDD
9 FGGGDDHH#
```

## B. Test Case 2

### Isi Test Case 2

```
AAABBCCCD
ABBBBCECD
ABBDCECD
AAABDCCCD
BBBBDDDD
FGGGDDHDD
FGIGDDHDD
FGGGDDHHH
```

## Program dijalankan dan memilih test case 2

```
=====
QUEENS PUZZLE SOLVER - ALGORITMA BRUTE FORCE
Oleh: Fazri Arrashyi Putra - 13524127
=====
Aturan: Setiap baris, kolom, dan daerah harus memiliki tepat satu ratu.
Masukkan nama file test case (.txt): test_case_2
Papan berhasil dibaca! Ukuran: 9x9
Tampilkan live update saat pencarian? (Ya/Tidak): █
```

## Output dan hasil .txt

```
=====
SOLUSI FINAL
=====

FGGGDDH#H
#GIGDDHDD
FG#GDDHDD
FGG#DDHDD
BBBBDDDD#
A#ABDCCCD
ABBB#ECD
ABBBB#CD
AAAB#CCCD

Waktu pencarian: 27435 ms
Banyak kasus yang ditinjau: 46242 kasus
=====

Apakah Anda ingin menyimpan solusi? (Ya/Tidak): █
```

solutions > test\_case\_2.txt

```
FGGGDDH#H
#GIGDDHDD
FG#GDDHDD
FGG#DDHDD
BBBBDDDD#
A#ABDCCCD
ABBB#ECD
ABBBB#CD
AAAB#CCCD
```

### C. Test Case 3

Isi Test Case 3
AAABBBCCC AAABBBCCC DDDEEEFFF DDDEEEFFF GGGHHHIII GGGHHHIII DDDEEEFFF GGGHHHIII DDDEEEFFF

Program dijalankan dan memilih test case 7
===== QUEENS PUZZLE SOLVER - ALGORITMA BRUTE FORCE Oleh: Fazri Arrashyi Putra - 13524127 ===== Aturan: Setiap baris, kolom, dan daerah harus memiliki tepat satu ratu.  Masukkan nama file test case (.txt): test_case_7  Papan berhasil dibaca! Ukuran: 9x9  Tampilkan live update saat pencarian? (Ya/Tidak): █  Ctrl+K to generate command

Output dan hasil .txt
Ratu ditempatkan: 6/9 Kasus yang ditinjau: 1105000 =====  ===== TIDAK ADA SOLUSI DITEMUKAN ===== Waktu pencarian: 58182 ms Banyak kasus yang ditinjau: 1108170 kasus =====  ~/Doc/Ku/S/Tucill 1m 21s Tucill Py 17:23:20

#### D. Test Case 4

##### Isi Test Case 4

```
DCCCBBAAA  
DCECBBBBA  
DCECDBBBA  
DCCCDBAAA  
DDDDDBBBB  
DDHDDGGGF  
DDHDDGIGF  
HHHDDGGGF
```

##### Program dijalankan dan memilih test case 3

```
○ > python3 src/queens_puzzle.py  
=====  
QUEENS PUZZLE SOLVER - ALGORITMA BRUTE FORCE  
Oleh: Fazri Arrashyi Putra - 13524127  
=====  
Aturan: Setiap baris, kolom, dan daerah harus memiliki tepat satu ratu  
  
Masukkan nama file test case (.txt): test_case_3  
  
Papan berhasil dibaca! Ukuran: 9x9  
  
Tampilkan live update saat pencarian? (Ya/Tidak): █
```

##### Output dan hasil .txt

```
=====  
SOLUSI FINAL  
=====  
  
D#CCBAAA  
DCEC#BBBA  
DC#CDBBBA  
DCCCDBA#A  
DDD#DBBBB  
DDHDD#GGF  
DDHDDG#GF  
DDHDDGIG#  
#HHDDGGGF  
  
Waktu pencarian: 6683 ms  
Banyak kasus yang ditinjau: 6165 kasus  
=====  
  
Apakah Anda ingin menyimpan solusi? (Ya/Tidak): █
```

```
D#CCBAA  
DCEC#BBBA  
DC#CDBBBA  
DCCCDBA#A  
DDD#DBBBB  
DDHDD#GGF  
DDHDDG#GF  
DDHDDGIG#  
#HHDDGGGF
```

#### E. Test Case 5

##### Isi Test Case 5

```
AAABBBCCC  
AAABBBCCC  
DDDDAAABB  
EEEFFFGGG  
EEEFFFHHH  
IIIEEEFFF  
GGGHHHIII  
GGGHHHIII  
EEEFFFIII
```

##### Program dijalankan dan memilih test case 8

```
Oleh: Fazri Arrashyi Putra - 13524127
```

```
=====
```

```
Aturan: Setiap baris, kolom, dan daerah harus memiliki tepat satu ratu.
```

```
Masukkan nama file test case (.txt): test_case_8
```

```
Papan berhasil dibaca! Ukuran: 9x9
```

```
Tampilkan live update saat pencarian? (Ya/Tidak): █
```

```
Ctrl+K to generate command
```

### Output dan hasil .txt

```
AAABBBCC#
AAABB#CCC
DDDDAA#BB
EE#FFFGGG
EEEFFFH#H
#IIEEEFFF
G#GHHHIII
GGGHHHIII
EEEFFFIII

Ratu ditempatkan: 7/9
Kasus yang ditinjau: 369000
=====
=====

TIDAK ADA SOLUSI DITEMUKAN
=====
=====

Waktu pencarian: 35362 ms
Banyak kasus yang ditinjau: 370260 kasus
=====

~/Doc/Ku/Strategi Algoritma/Tucill 41s Tucill Py ]
```

## **BAB V**

### **LAMPIRAN**

No	Poin	Ya	Tidak
1	Program berhasil di kompilasi tanpa kesalahan	V	
2	Program berhasil di jalankan	V	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	V	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	V	
5	Program memiliki Graphical User Interface (GUI)		V
6	Program dapat menyimpan solusi dalam bentuk file gambar		V

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



Fazri Arrashyi Putra

## **BAB VI**

### **TAUTAN dan REFERENSI**

#### **A. Pranala Github**

**Pranala Repository Github: xxx**

#### **B. Referensi:**

- [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/02-Algoritma-Brute-Force-\(2026\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/02-Algoritma-Brute-Force-(2026)-Bag1.pdf)
- <https://medium.com/@jeremy.ockenden/solving-linkedins-queens-puzzle-in-python-3f0d53925842>