LAPORAN TUGAS KECIL 1
IF2211 Strategi Algoritma

"Penyelesaian Permainan Queens LinkedIn dengan Algoritma Brute Force"

Disusun oleh:

Rhenaldy Cahyadi

13524039

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2026

1 Pendahuluan

Pada tugas kecil ini dikembangkan sebuah program untuk menyelesaikan permainan *Queens LinkedIn* menggunakan pendekatan algoritma *brute force*. Permainan dimainkan pada papan berukuran $N \times N$ yang terdiri atas beberapa blok berwarna. Tujuan permainan adalah menempatkan tepat satu ratu pada setiap baris, setiap kolom, dan setiap blok warna dengan batasan tambahan bahwa tidak boleh ada dua ratu yang saling bersebelahan, termasuk secara diagonal.

Berbeda dengan peraturan permainan catur klasik, pada permainan ini dua ratu diperbolehkan berada pada diagonal yang sama selama tidak menempati kotak yang saling bersentuhan langsung. Program membaca konfigurasi papan dari sebuah file teks (.txt) dan kemudian menghasilkan solusi dengan mencoba seluruh kemungkinan konfigurasi posisi ratu hingga ditemukan susunan yang memenuhi seluruh aturan permainan.

2 Aturan Permainan

Aturan permainan adalah sebagai berikut:

1. Setiap blok berwarna memiliki tepat satu ratu.
2. Setiap baris memiliki tepat satu ratu.
3. Setiap kolom memiliki tepat satu ratu.
4. Tidak ada dua ratu yang boleh saling bersebelahan, termasuk secara diagonal.
5. Dua ratu diperbolehkan berada pada diagonal yang sama selama tidak menempati kotak yang bersebelahan.

3 Penjelasan Algoritma

Koordinat ratu disimpan dalam sebuah array satu dimensi:

$$arr[col] = row$$

Indeks array merepresentasikan kolom, sedangkan nilai yang tersimpan pada indeks tersebut menyatakan baris tempat ratu berada. Dengan representasi ini, setiap kolom secara otomatis memiliki tepat satu ratu karena setiap indeks hanya memiliki satu nilai.

Agar setiap baris juga memiliki tepat satu ratu, digunakan permutasi dari himpunan $\{0, 1, 2, \ldots, N-1\}$. Dengan demikian, tidak ada dua kolom yang memiliki nilai baris yang sama. Semua kemungkinan konfigurasi yang memenuhi syarat satu ratu per baris dan satu ratu per kolom dapat direpresentasikan sebagai seluruh permutasi dari himpunan tersebut. Sebagai contoh untuk $N = 4$, beberapa permutasi yang mungkin adalah $[0, 1, 2, 3]$, $[0, 1, 3, 2]$, $[0, 2, 1, 3]$, dan $[0, 2, 3, 1]$. Setiap permutasi menyatakan satu kandidat solusi.

Langkah-Langkah Algoritma

Proses penyelesaian dimulai dengan membaca file konfigurasi warna dan menyimpannya dalam array dua dimensi $color[row][col]$. Nilai N ditentukan berdasarkan jumlah baris pada papan. Selanjutnya, seluruh permutasi dari angka 0 hingga $N-1$ dibangkitkan.

Untuk setiap permutasi yang dihasilkan, dilakukan proses validasi. Pertama, dilakukan validasi warna untuk memastikan tidak ada dua ratu yang berada pada blok warna yang sama. Kemudian dilakukan validasi diagonal dengan memastikan tidak terdapat dua ratu pada kolom bersebelahan yang memiliki selisih baris sebesar satu (yang bersentuhan). Jika kedua validasi terpenuhi, konfigurasi tersebut dinyatakan sebagai solusi yang valid dan proses pencarian dihentikan. Program juga mencatat jumlah iterasi serta waktu eksekusi pencarian.

4 Analisis Kompleksitas

Pada setiap permutasi, dilakukan dua jenis validasi yang masing-masing memiliki kompleksitas $O(N)$. Validasi warna dilakukan dengan menyimpan warna setiap posisi ratu ke dalam sebuah struktur data set dan memeriksa apakah terdapat warna yang muncul lebih dari satu kali. Validasi diagonal dilakukan dengan membandingkan nilai $|arr[col] - arr[col+1]|$ untuk setiap pasangan kolom yang bersebelahan.

Jumlah seluruh kemungkinan konfigurasi adalah $N!$ karena seluruh permutasi dibangkitkan. Oleh karena itu, kompleksitas waktu keseluruhan algoritma adalah:

$$O(N! \cdot N)$$

Pertumbuhan faktorial menyebabkan algoritma brute force menjadi tidak efisien untuk nilai N yang besar. Namun, untuk ukuran papan kecil hingga menengah, pendekatan ini masih dapat digunakan dengan waktu eksekusi yang dapat diterima.

5 Dokumentasi Pengujian

Pada tahap pengujian, program diuji menggunakan lima buah test case valid serta dua test case tidak valid. Pengujian ini bertujuan untuk memastikan bahwa program mampu menghasilkan solusi yang benar untuk masukan yang sesuai spesifikasi, serta mampu menangani kesalahan input dengan tepat.

Test Case 1

Input:

```
UKKBHHHP
UUKBBHHP
MUUUUHHP
MMUUUUHP
OOUUUUHU
COUUUUUU
CCCUUUUU
CCCCCUUU
```

```
Pilih nomor test case yang ingin dijalankan: 1
UK#BHHHP
UUKB#HHP
MUUUUHH#
#MUUUUHP
OOUUUU#U
C#UUUUUU
CCCUU#UU
CCC#CUUU
Waktu pencarian: 22 ms
Banyak kasus yang ditinjau: 18102 kasus
Apakah Anda ingin menyimpan solusi? (Ya/Tidak): Ya
Solusi tersimpan di 'solusi_case1.txt'.
```

Figure 1: Hasil eksekusi Test Case 1

Test Case 2

Input:

```
PPPBMGG
PBBBMGG
PYYYMMG
PYYMMGG
```

```
PYYMMGG
PPPMRGG
RRRRRRT
```

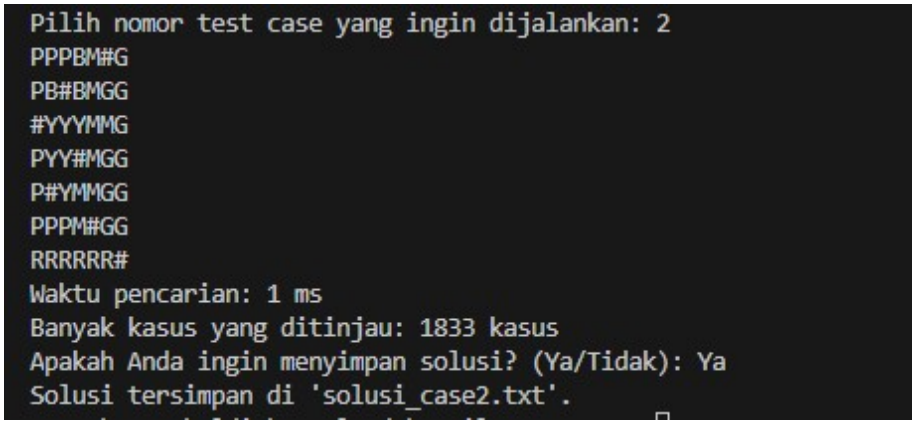

Figure 2: Hasil eksekusi Test Case 2

Test Case 3

Input:

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

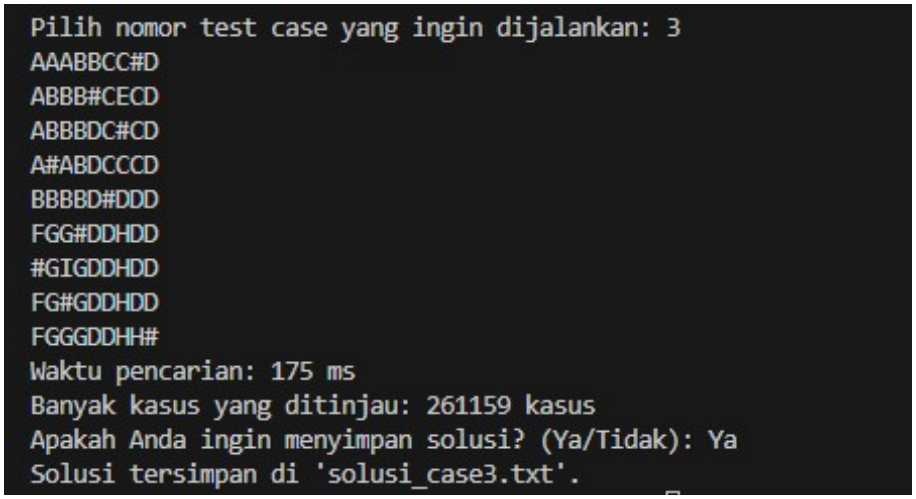

Figure 3: Hasil eksekusi Test Case 3

Test Case 4

Input:

```
UUUWWWWWWWW
UWUWBWBBBWW
UWUWBBBBYWW
UWWWBBBYYYW
UUWCCBBYYOW
UUWCCCBYYOW
UUWBBBMYOOR
UUWWWBMYOOR
ULLWWBMYOOR
ULLGGBMYPPR
ULLGGBMYYPR
```

```
Pilih nomor test case yang ingin dijalankan: 4
#UUWWWWWWW
UWUW#WBBBWW
U#UWBBBBYWW
UWWWBBB#YYW
UUWCCBBYY#W
UUWCC#BYYOW
UUWBBBMYOO#
UUWWWB#YOOR
UL#WWBMYOOR
ULLGGBMY#PR
ULL#GBMYYPR
Waktu pencarian: 419 ms
Banyak kasus yang ditinjau: 640397 kasus
Apakah Anda ingin menyimpan solusi? (Ya/Tidak): Ya
Solusi tersimpan di 'solusi_case4.txt'.
```

Figure 4: Hasil eksekusi Test Case 4

Test Case 5

Input:

```
PPPPPPPPP
POOOOOOOY
PMMMMMBOY
PMCCCCBOY
PMRRRGBOY
PMRRRGBOY
PMRTRGBOY
PBBBBBBOY
PYYYYYYYY
```

```
Pilih nomor test case yang ingin dijalankan: 5
#PPPPPPPP
POOOOOO#Y
P#MMMMBOY
PMCC#CBOY
PM#RRGBOY
PMRRR#BOY
PMR#RGBOY
PBBBBB#OY
PYYYYYYY#
Waktu pencarian: 10 ms
Banyak kasus yang ditinjau: 6873 kasus
Apakah Anda ingin menyimpan solusi? (Ya/Tidak): Ya
Solusi tersimpan di 'solusi_case5.txt'.
```

Figure 5: Hasil eksekusi Test Case 5

Test Case Tidak Valid

Selain pengujian dengan input yang valid, dilakukan pula pengujian terhadap dua kondisi kesalahan input untuk memastikan program dapat melakukan validasi dengan benar.

Case Gagal 1: Ukuran Tidak Persegi

Kasus ini menggunakan file dengan jumlah baris dan kolom yang tidak sama sehingga tidak membentuk matriks persegi $N \times N$. Program menolak input dan menampilkan pesan kesalahan.

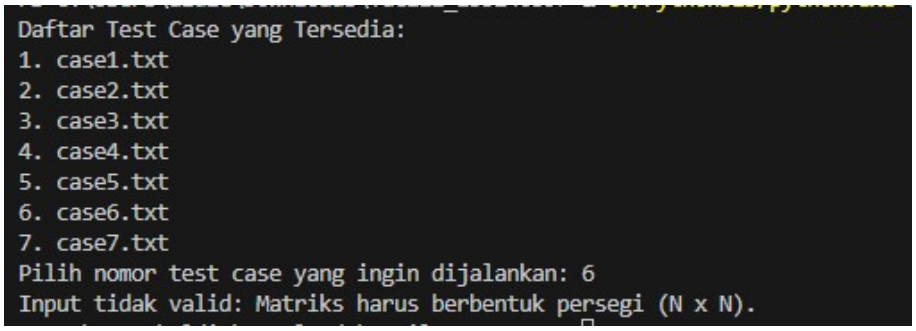

Figure 6: Penanganan kesalahan untuk input tidak persegi

Case Gagal 2: File Kosong

Kasus ini menggunakan file masukan kosong. Program mendeteksi bahwa tidak terdapat data yang dapat diproses dan menampilkan pesan kesalahan yang sesuai.

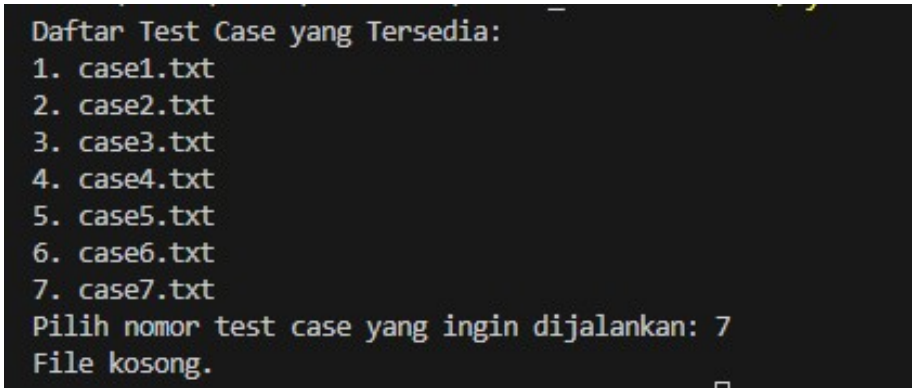

Figure 7: Penanganan kesalahan untuk file kosong

6 Kesimpulan

Pendekatan brute force mampu menyelesaikan permainan Queens LinkedIn dengan membangkitkan seluruh kemungkinan konfigurasi dan memvalidasinya satu per satu. Metode ini memiliki keunggulan dalam kesederhanaan implementasi dan jaminan menemukan solusi apabila solusi tersebut memang ada.

Meskipun demikian, kompleksitas waktu yang bersifat faktorial menyebabkan metode ini tidak efisien untuk ukuran papan yang besar. Untuk kasus dengan nilai N yang relatif kecil, pendekatan ini tetap layak digunakan.

Link Repository

Source code dapat diakses melalui tautan berikut:
`https://github.com/rhenprojects/Tucil1_13524039`

Pernyataan Penggunaan AI

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

Rhenaldy Cahyadi
13524039

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		✓