Laporan Tugas Kecil 1

IF2211 Strategi Algoritma

Penyelesaian Permainan Queens

Semester II Tahun 2025/2026

Disusun oleh:

Jason Edward Salim - 13524034

Laboratorium Ilmu dan Rekayasa Komputasi

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Daftar Isi

1 Penerapan Algoritma Brute Force

1.1 Pendahuluan

Algoritma *Brute force* merupakan metode penyelesaian permasalahan secara *straightforward*. Pada program penyelesaian permainan Queens ini, algoritma *Brute force* digunakan secara murni dengan *exhaustive search*.

1.2 Alur Berpikir

Queens memiliki area bujur sangkar berukuran dinamis NxN dimana kita menempatkan ratu pada tiap warna dimana setiap ratu yang ditempatkan dilarang berada pada baris dan kolom yang sama, serta dilarang berada di 8 sel di sekitar ratu yang lain.
Untuk mendapatkan solusi dari Queens, *Brute force* murni diterapkan dengan menggunakan *exhaustive search* pada tiap baris secara terus-menerus hingga ditemukan solusi yang tepat. Pada setiap baris, penempatan ratu akan dilakukan pada tiap kolom hingga baris terakhir dan dilakukan pengecekan apakah penempatan ratu sekarang sesuai dengan aturan yang ada atau tidak. Jika belum sesuai, maka penempatan berikutnya akan dilakukan hingga ditemukan solusi yang sesuai atau tidak ditemukan solusi setelah seluruh penempatan dicoba. Solusi ini akan memiliki kompleksitas waktu sekitar $O(N^N)$ karena pada tiap N baris terdapat N kolom yang dapat dicoba penempatannya.

1.3 Implementasi Penyelesaian Brute Force

Pada penyelesaian ini, algoritma akan terbagi menjadi:

1. Fungsi utama yang akan melakukan *exhaustive search* untuk penempatan ratu pada tiap baris dan kolom.
2. Fungsi validasi penempatan yang akan mengecek apakah penempatan ratu pada suatu posisi sudah sesuai dengan aturan yang ada.
3. Fungsi yang akan menyimpan dan mengembalikan solusi yang didapatkan.

1.3.1 Exhaustive Search

Untuk setiap baris, kita akan menempatkan ratu pada setiap kolom dan melakukan validasi setelah tiap baris memiliki satu ratu. Langkah ini dapat diimplementasikan sebagai berikut.

1. Iterasi tiap baris dengan kombinasi tiap kolom.

   Contoh: (0, 0) hingga (0, N-1), (1, 0) hingga (1, N-1), dan seterusnya.

```
1 func TryPosition(area *TArea) (queensLocation []TPosition) {
2     n := area.n
3     cols := make([]int, n)
4     for {
5         temp := make([]TPosition, n)
6         for row := 0; row < n; row++ {
7             temp[row] = TPosition{row, cols[row]}
8         }
```

```
9
10          if CheckPosition(*area, temp) {
11              area.queensLocation = temp
12              return temp
13          }
14
15          var i int
16          for i = n - 1; i >= 0; i-- {
17              cols[i]++
18              if cols[i] < n {
19                  break
20              }
21              cols[i] = 0
22          }
23          if i < 0 {
24              break
25          }
26      }
27
28      return nil
29  }
```

2. Setelah menempatkan ratu di tiap baris, validasi apakah penempatan sudah memenuhi aturan. Jika belum, iterasi ke penempatan selanjutnya.

3. Jika solusi ditemukan, solusi akan dikembalikan dan jika tidak maka akhiri iterasi.

   Pada kode di atas, fungsi *CheckPosition* memvalidasi penempatan dan jika valid maka solusi akan tersimpan pada *area.queensLocation* dan dikembalikan. Sebaliknya, kode akan berlanjut jika *CheckPosition* mengembalikan false.

1.3.2 Validasi Penempatan

Validasi yang harus dilakukan pada setiap masukan dan penempatan ratu antara lain:

1. Cek apakah area yang dimasukkan memiliki ukuran NxN dengan N > 0.

2. Cek apakah area NxN memiliki N representasi huruf.

3. Cek apakah tiap representasi huruf selalu terhubung dan tidak terputus.

4. Cek apakah penempatan ratu sesuai dengan aturan permainan: tidak berada pada baris dan kolom yang sama serta tiap ratu tidak terletak di 8 sel sekitar ratu lainnya.

Langkah di atas dapat dilihat implementasinya pada folder ***src/solution*** di *repository* GitHub yang dapat diakses pada lampiran.

2 Implementasi GUI

3 Kasus Uji

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		