Algoritma Brute Force untuk Menyelesaikan Permainan Queens di LinkedIn

Laporan Tugas Kecil 1

Dibuat sebagai Laporan Tugas Kecil 1 Mata Kuliah

IF2211 Strategi Algoritma

Disusun Oleh:

Irvin Tandiarrang Sumual

13524030

K-02

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2025

1. Implementasi Algoritma

Queens Game di LinkedIn bertujuan untuk menempatkan seluruh *queen* yang berjumlah *n* untuk papan berukuran *n x n* di kolom dan baris yang berbeda serta tidak boleh berada di daerah warna yang sama dan juga *queen* tidak boleh saling bertetangga secara diagonal. Algoritma brute force diterapkan dalam bahasa pemrograman Go (Golang).

1.1 Algoritma Brute Force

Algoritma *brute force* yang diimplementasikan bersifat murni tanpa menggunakan heuristik apapun. Algoritma yang diimplementasikan akan menjelajahi semua permutasi posisi queen yang menjamin tidak ada dua *queen* pada baris dan kolom yang sama. Hal ini dapat dimungkinan dengan penggunaan suatu *array*, misalnya *queenPositions*, yang di mana *queenPositions*[i] = j berarti queen pada baris *i* berada di kolom *j*. Selain itu, dalam algoritma yang diterapkan, pengecekan apakah suatu permutasi valid atau tidak dilakukan hanya ketika permutasi telah selesai dibangun terlebih dahulu sehingga tidak ada *pruning* di dalamnya.

Berikut ini Langkah-langkah dari *brute force* yang diimplementasikan.

1. Papan berukuran *n* x *n* direpresentasikan dengan struktur *Board* yang terdiri dari *RowLength*, *ColLength,* dan juga *Grid* yang merupakan matriks dua dimensi berisikan *Cell* yang terdiri dari *Region* dan juga *IsOccupied* yang menyatakan apakah sel sudah ditempati atau belum
2. Dalam menjelajahi permutasi dari posisi *queens*, digunakan *1D array* queenPositions dengan Panjang n, di mana *queenPositions*[i] = j melambangkan *queen* pada baris ke-i ditempatkan di kolom ke-j. Dengan demikian, tidak mungkin ada queen yang berada di baris maupun kolom yang sama.
3. Fungsi *generatePermutation*() akan digunakan untuk menghasilkan seluruh permutasi yang mungkin.
4. Dalam menjelajahi semua permutasi yang mungkin, setiap kali sebuah permutasi lengkap terbentuk, fungsi *checkPermutation*() akan dipanggil dan permutasi akan dikonversi dalam representasi papan dengan memakai *permutationToBoard*().
5. Setelah papan terbentuk, fungsi IsSolution() akan memeriksa apakah permutasi tersebut merupakan solusi yang valid atau tidak.

6. Jika telah ditemukan solusi valid, Solusi akan disimpan dan pencarian dihentikan. Jika tidak, akan kembali ditelusuri permutasi berikutnya

Kompleksitas dasar algoritma ini ialah O(n!) karena seluruh permutasi posisi queen dihasilkan. Akan tetapi dikarenakan terdapat proses konversi dari *1D array* ke *2D array* serta validasi konfigurasi papan maka kompleksitas waktunya ialah $O(n! * n^2)$

Berikut ini adalah contoh visualisasi dari algoritma *brute force* yang diimplementasikan untuk input board 4 x 4 dengan konfigurasi :

AABB

AABB

CCDD

CCDD

Urutan Permutasi	Array Permutasi	Hasil	Status
1	[0 1 2spe 3]	#ABB A#BB CC#D CCD#	Belum Selesai
2	[0 1 3 2]	#ABB A#BB CCD# CC#D	Belum Selesai
3	[0 3 1 2]	#ABB AAB# C#DD CC#D	Belum Selesai
4	[3 0 1 2]	AAB# #ABB C#DD CC#D	Belum Selesai
5	[0 2 1 3]	#ABB AA#B C#DD CCD#	Belum Selesai
6	[0 2 3 1]	#ABB AA#B CCD# C#DD	Belum Selesai
7	[0 3 2 1]	#ABB AAB# CC#D C#DD	Belum Selesai

8	[3 0 2 1]	AAB# #ABB CC#D C#DD	Belum Selesai
9	[2 0 1 3]	AA#B #ABB C#DD CCD#	Belum Selesai
10	[2 0 3 1]	AA#B #ABB CCD# C#DD	Solusi telah ditemukan, berhenti

2. Source Code

Berikut ini struktur dari program yang dibuat.

```
bin/
├── linux/
│   ├── queens-linux
├── windows/
│   ├── queens-windows.exe
doc/
│   Laporan_Tucil1_Stima_13524030.pdf
src/
│
├── board/
│   ├── io.go
│   └── model.go
│
├── gui/
│   ├── app.go
│   ├── buttons.go
│   ├── grid.go
│   ├── state.go
│   └── utils.go
│
```

```
├── solver/
│   └── solver.go
│
├── main.go
├── go.mod
└── go.sum
│
test/
├── 1x1.txt
├── 1x1-result.txt
├── 2x2.txt
├── 3x3.txt
├── 4x4.txt
├── 4x4-result.txt
├── 7x7.txt
├── 7x7-result.txt
├── 9x9.txt
├── 9x9-result.txt
├── 10x10.txt
├── 10x10-result.txt
└── README.md
```

2.1 Folder bin/

Folder ini berisikan *executable file* (*queens.exe)* dari hasil *compile* program yang dapat dijalankan secara langsung baik untuk windows di dalam folder windows maupun untuk linux di dalam folder linux

2.2 Folder doc/

Folder ini berisikan laporan dari tugas ini

2.3 Folder src

Folder ini merupakan bagian utama yang berisikan *source code* dari progam yang dibuat. Folder ini terdiri dari sejumlah bagian.

2.3.1 Folder board/

Folder ini berisikan modul yang merepresentasikan strutkur dari *board* dan juga operasi yang dapat dilakukan.

2.3.1.1 io.go

func TxtToBoard(filename string) (Board, error)	Mengubah input txt file ke dalam bentuk Board, dipakai ketika akan melakukan load txt file
func (board Board) BoardToTxt(filename string) error	Mengubah Board ke dalam bentuk TXT file, dipakai ketika akan menyimpan solusi

2.3.1.2 model.go

type Cell struct {
  Region rune
  IsOccupied bool
}
	Tipe data Cell

type Board struct {
  RowLength int
  ColLength int
  Grid [][]Cell
}
	Tipe data Board

func (board Board) PrintBoard()	Menampilkan isi dari suatu Board di terminal
func (board *Board) DeepCopy() *Board	Membuat objek Board baru berdasarkan Board yang dimasukkan ke dalam fungsi
func (board Board) IsSolution() bool	Memeriksa apakah suatu Board memiliki konfigurasi yang sesuai untuk menjadi Solusi dari Queens LinkedIn game dan mengembalikan true jika valid

func (board Board) checkDiagonalConstraint(row int, col int) bool	Membantu melakukan pengecekan constraint diagonal apakah terpenuhi atau tidak pada suatu konfigurasi papan

2.3.2 Folder gui /

Folder ini berisikan modul yang membangun gui dari program ini

2.3.2.1 app.go

func StartApp()	Memulai menjalankan GUI program yang dipanggil di main.go pada folder src

2.3.2.2 buttons.go

func MakeLoadButton(state *AppState, label *widget.Label) *widget.Button	Membuat button “Load txt File” yang akan membuat user dapat memilih file yang akan di-upload
func MakeSaveToTXTButton(state *AppState) *widget.Button	Membuat button “Save Solution to TXT” yang akan melakukan penyimpanan board solusi ke dalam bentuk txt file
func MakeSolveButton(state *AppState, label *widget.Label) *widget.Button	Membuat button “Solve” yang akan melakukan pemanggilan terhadap algoritma brute force yang dilakukan. Selain itu terdapat penggunaan Go Routine untuk mencegah UI freeze dan memungkinkan live update

2.3.2.3 grid.go

func RefreshGrid(state *AppState)	Menampilkan grid di GUI

2.3.2.4 state.go

type AppState struct {
LoadedBoard board.Board
SolutionBoard board.Board
FoundSolution bool
GridDisplay *fyne.Container
Window fyne.Window
}
	Tipe data merepresentasikan state dari program

2.3.2.5 state.go

var colorMap = map[rune]color.NRGBA	Mapping character dari A – Z ke warna tertentu
func GetColor(char rune) color.Color	Fungsi yang dipanggil untuk mendapatkan hasil warna dari mapping karakter tertentu

2.3.3 Folder solver/

Folder ini berisikan modul yang menjadi pembangun algoritma utama dari algoritma *brute force* yang diimplementasikan

2.3.2.1 solver.go

func Solve(b board.Board, onUpdate func(board.Board)) (board.Board, int, int)	Fungsi yang akan dipanggil ketika Solve button ditekan untuk memulai *brute force* dengan menginisasi permutasi pertama sembari memulai perhitungan waktu dan juga perhitungan jumlah permutasi yang dilakukan
func firstPermutation(dimension int) []int	Memberikan permutasi pertama ketika *brute force* akan dilakukan, di mana inputnya merupakan dimensi dan akan menghasilkan array dari 0

	… n – 1 yang melambangkan posisi queen di kolom
func permutationToBoard(queenPositions []int, template board.Board) board.Board	Mengubah permutasi ke dalam representasi board
func checkPermutation(queenPositions []int, template board.Board, solution *board.Board, countPermutations int, onUpdate func(board.Board)) bool	Mengecek apakah suatu permutasi merupakan solusi yang valid atau bukan dengan memanggil fungsi isSolution()
func generatePermutation(queenPositions []int, template board.Board, solution *board.Board, countPermutations *int, onUpdate func(board.Board)) bool	Melakukan proses permutasi kemungkinan-kemungkinan dari queens position di papan. Fungsi ini akan berhenti lebih awal jika solusi sudah ditemukan

2.4 Folder test/

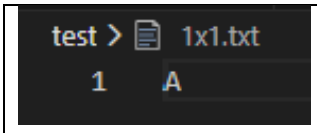
Folder ini berisikan file pengujian beserta dengan hasil dari pengujian-pengujian tersebut

3. Kasus Uji

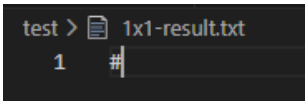
Berikut ini hasil dari pengujian yang telah dilakukan.

3.1 Kasus Uji 1

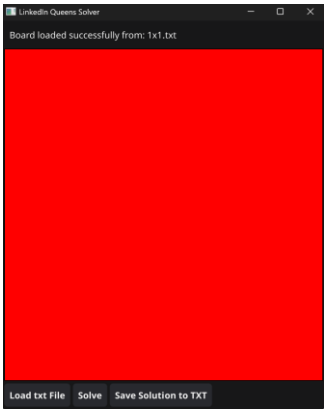
Input .txt



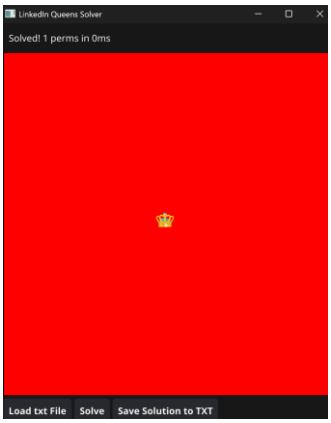
Output .txt



Sebelum Solve

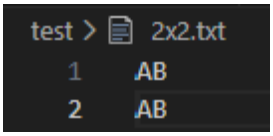


Setelah Solve



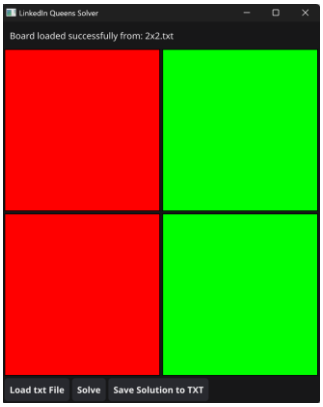
3.2 Kasus Uji 2

Input .txt

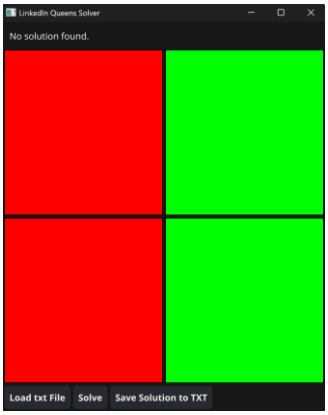


Output .txt

Sebelum Solve

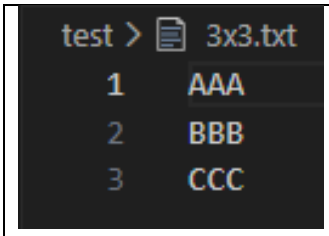


Setelah Solve

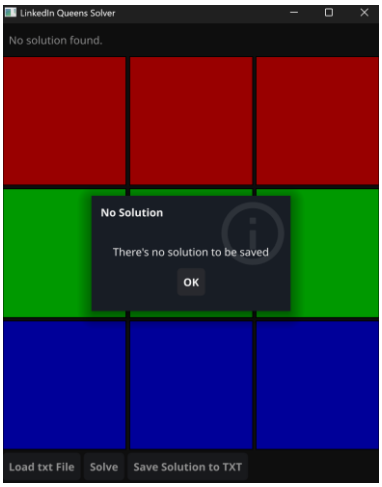


3.3 Kasus Uji 3

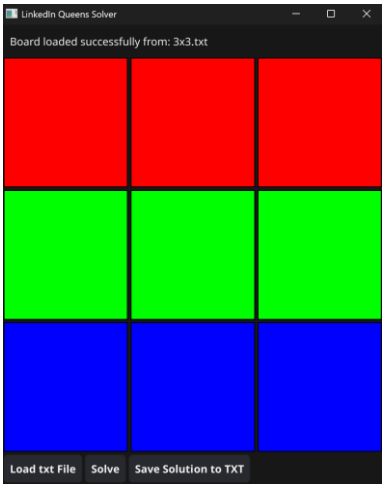
Input .txt



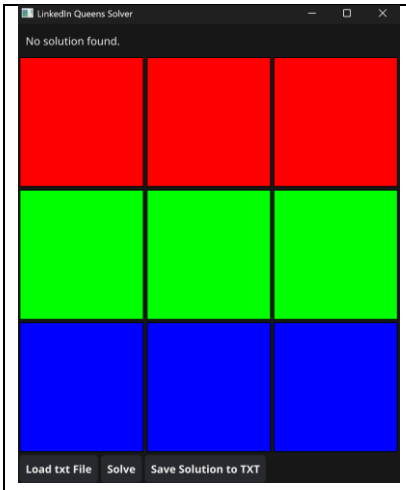
Output .txt



Sebelum Solve

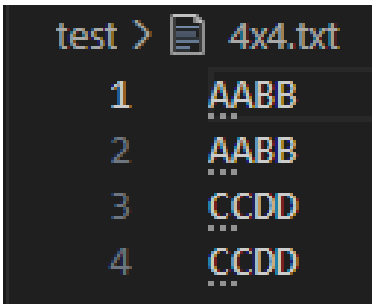


Setelah Solve

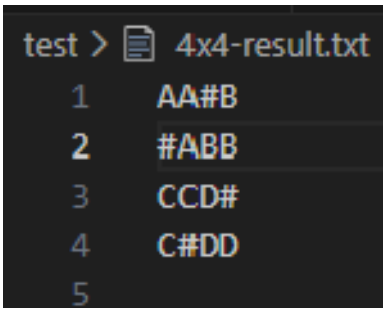


3.4 Kasus Uji 4

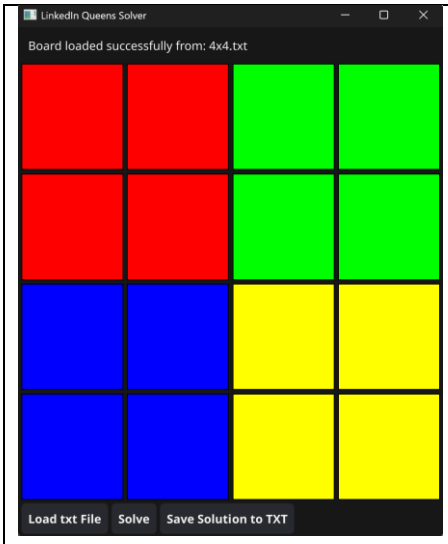
Input .txt



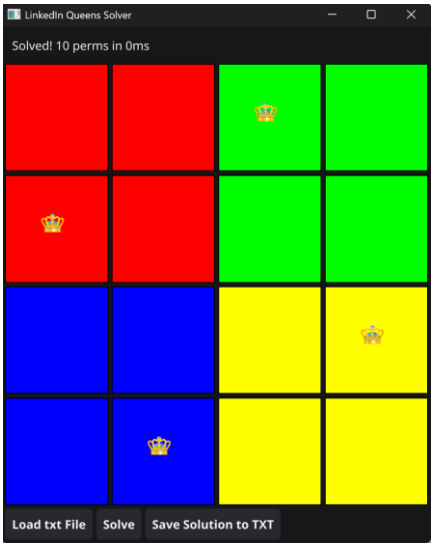
Output .txt



Sebelum Solve

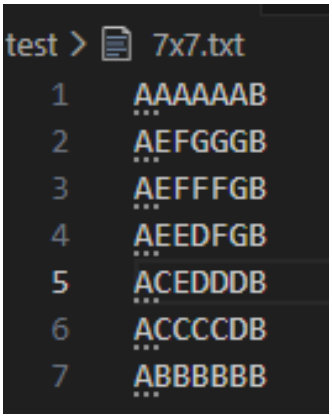


Setelah Solve



3.5 Kasus Uji 5

Input .txt

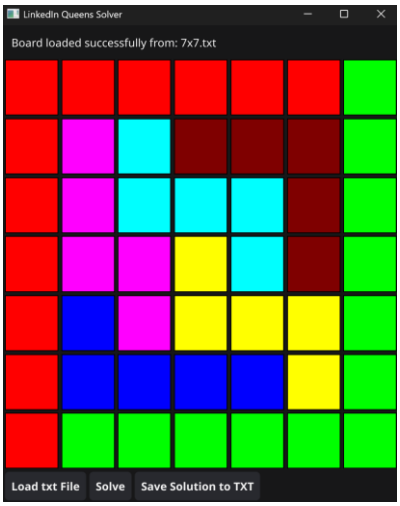


Output .txt

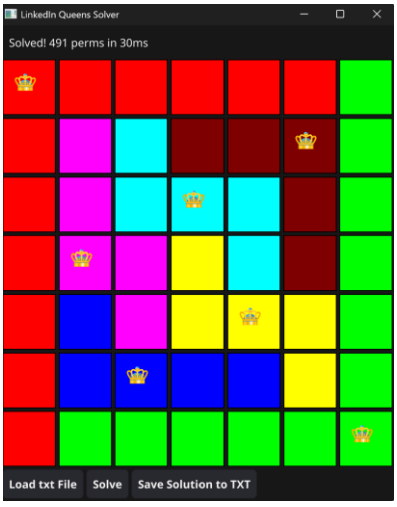
test > 7x7-result.txt

```
1  #AAAAAB
2  AEFGG#B
3  AEF#FGB
4  A#EDFGB
5  ACED#DB
6  AC#CCDB
7  ABBBBB#
```

Sebelum Solve



Setelah Solve



3.6 Kasus Uji 6

Input .txt

test > 9x9.txt

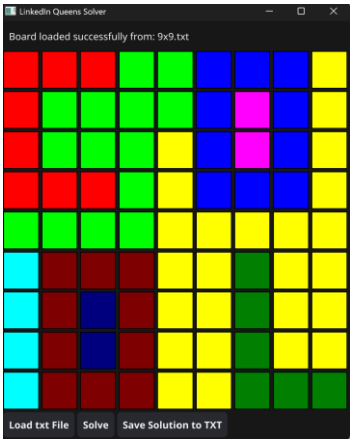
```
1  AAABBCCCD
2  ABBBBCECD
3  ABBBDCECD
4  AAABDCCCD
5  BBBBDDDDD
6  FGGGDDHDD
7  FGIGDDHDD
8  FGIGDDHDD
9  FGGGDDHHH
```

Output .txt

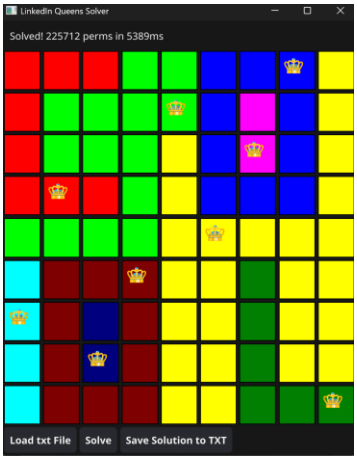
test > 9x9-result.txt

```
1  AAABBCC#D
2  ABBB#CECD
3  ABBBDC#CD
4  A#ABDCCCD
5  BBBBD#DDD
6  FGG#DDHDD
7  #GIGDDHDD
8  FG#GDDHDD
9  FGGGDDHH#
```

Sebelum Solve



Setelah Solve



3.7 Kasus Uji 7

Input .txt

test > 10x10.txt

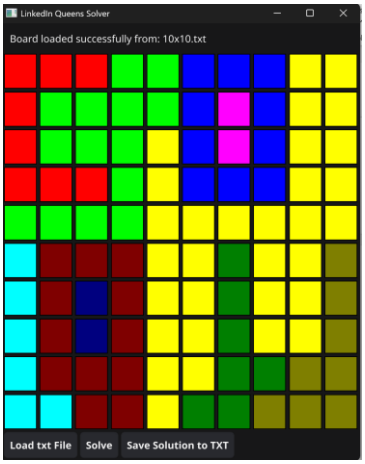
```
AAABBCCCDD
ABBBBCECDD
ABBBDCECDD
AAABDCCCDD
BBBBDDDDDD
FGGGDDHDDJ
FGIGDDHDDJ
FGIGDDHDDJ
FGGGDDHHJJ
FFGGDHHJJJ
```

Output .txt

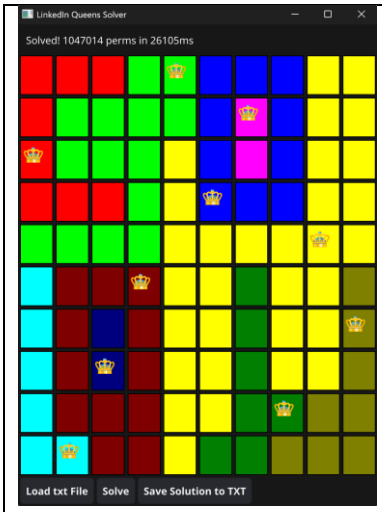
test > 10x10-result.txt

```
AAAB#CCCDD
ABBBBC#CDD
#BBBDCECDD
AAABD#CCDD
BBBBDDDD#D
FGG#DDHDDJ
FGIGDDHDD#
FG#GDDHDDJ
FGGGDDH#JJ
F#GGDHHJJJ
```

Sebelum Solve



Setelah Solve



4. Daftar Pustaka

[1] Solving LinkedIn's Queens puzzle in Python | by Jeremy Ockenden | Feb, 2026 | Medium

[2] Solving the Next Permutation Problem: From Brute Force to Optimal O(N) Solution | by Mahalakshmi Veeraraj | Medium

5. Lampiran

Source code dapat diakses pada : Irvin-Tandiarrang-Sumual/Tucil1_13524030

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		✓

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

Irvin Tandiarrang Sumual