

LAPORAN TUGAS KECIL I
IF2211 STRATEGI ALGORITMA



Dibuat oleh:
Nathaniel Christian - 13524122

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganesha 10, Bandung 40132
2026

BAB I

Algoritma

I. Algoritma Brute Force Exhaustive

Misalkan board berukuran $n = 5$. Kita harus menjawab “queen baris ke- i ditaruh di kolom ke berapa?”. Hal ini akan dijawab dalam ‘rowPlaced := make([]int, b.size)’

Dalam loop utama, akan dimulai dengan *preparation*, yaitu clear board dengan remove semua queen di dalam board. Kemudian kita akan mencoba untuk menaruh queen sesuai rowPlaced. Berikut ini adalah contoh rowPlaced yang di increment:

$[0,0,0,0,0] \rightarrow [0,0,0,0,1] \rightarrow \dots \rightarrow [0,0,0,0,4]$ ($n = 5$, maka $n-1 = 4$) $\rightarrow [0,0,0,1,0] \rightarrow [0,0,0,1,1]$.

Penempatan ini akan di cek menggunakan fungsi isPlaceable, kalau kotaknya valid maka taruh queen, sedangkan kalau tidak valid maka akan dilakukan break sehingga baris di bawahnya tidak perlu di cek. Kemudian, kita cek hasil, apabila valid bernilai true, berarti semua queen berhasil diletakkan di dalam board, yang artinya hasil sudah ditemukan. Apabila valid bernilai false, kombinasi tersebut gagal dan kita lanjut ke iterasi/loop selanjutnya.

Dalam rowPlaced, kita akan melakukan increment dari row paling belakang dulu. Variabel carry akan melakukan increment ke row sebelumnya. Logikanya mirip seperti kalau melakukan pertambahan terhadap angka $9 + 1$. Indeks ke-1 akan bernilai 1 dan indeks ke-2 bernilai 0, yang dihasilkan dari angka 10. Jika variabel carry sudah melewati row pertama, artinya semua sudah di cek, yang artinya tidak ada solusi yang ditemukan.

II. Algoritma Brute Force dengan Backtracking

Dalam solusi ini, kita melakukan penempatan queen pada masing-masing row. Untuk setiap row, kita mengecek setiap kolom, mulai dari kiri ke kanan. Pengecekan setiap row dapat dimulai dari indeks ke-0 atau dari indeks terakhir sebelum kita melakukan backtracking. Penempatan queen pada (row,col) tetap di cek dengan isPlaceable seperti solusi I. Apabila penempatan queen valid, maka kita lanjut ke row selanjutnya dan mencari kolom yang sesuai disitu.

Backtracking dimulai apabila kita tidak menemukan satupun queen valid di sebuah row tertentu. Jika hal ini terjadi, kita harus decrement row dan melanjutkan pencarian queen valid (yang lain) di row sebelumnya.

Jika semua queen berhasil ditaruh ke dalam board, maka kita menemukan solusi yang benar. Sebaliknya, apabila kita menyentuh $row < 0$, maka kita tidak menemukan solusi yang benar, dan berarti board tidak memiliki solusi.

BAB II

Source Code

Struktur program (hanya source code (/src), exclude /bin, /doc, /test):

```
root/
├── README.md
├── go.mod
└── src/
    ├── board.go
    ├── io.go
    ├── main.go
    └── solver.go
```

Isi file (comment di-*remove*):

1. main.go

```
package main

import (
    "fmt"
    "time"
)

func main() {
    fmt.Println("Queens Solver Brute Force Approach")

    var filename string
    fmt.Print("Input nama file (no whitespace, put inside test/ folder): ")
    fmt.Scan(&filename)

    filepath := "../test/" + filename
    board, err := constructBoard(filepath)
    if err != nil {
        fmt.Println("Error: ", err)
        return
    }

    var counter int

    if !board.isValid() {
        fmt.Println("Board tidak valid.")
        return
    }
```

```

startTime := time.Now()
found := Solve(board, &counter)

duration := time.Since(startTime)

clearScreen()
if found {
    printBoard(board)
} else {
    fmt.Println("Tidak ada solusi yang ditemukan.")
}

fmt.Printf("Waktu pencarian: %.3f ms\n", float64(duration.Microseconds())/1000.0)
fmt.Println("Banyak kasus yang ditinjau:", counter)

var save string
fmt.Print("Apakah Anda ingin menyimpan solusi? (Y/N): ")
fmt.Scan(&save)

filepath = "../test/solution/" + filename
if save == "Y" {
    err := saveBoard(board, filepath)
    if err != nil {
        fmt.Println("Error:", err)
    } else {
        fmt.Println("File berhasil disimpan di:", filepath)
    }
}
}
}

```

2. io.go

```

package main

import (
    "bufio"
    "fmt"
    "os"
    "time"
)

func constructBoard(filename string) (*Board, error) {
    file, err := os.Open(filename)
    if err != nil {
        return nil, err
    }
}

```

```

    }
    defer file.Close()

    scanner := bufio.NewScanner(file)

    var colorMatrix [][]rune

    for scanner.Scan() {
        line := scanner.Text()
        row := []rune(line)
        colorMatrix = append(colorMatrix, row)
    }

    if err := scanner.Err(); err != nil {
        return nil, err
    }

    return newBoard(colorMatrix), nil
}

func saveBoard(b *Board, filename string) error {
    file, err := os.Create(filename)
    if err != nil {
        return err
    }
    defer file.Close()

    for i := 0; i < b.size; i++ {
        for j := 0; j < b.size; j++ {
            if b.queen[i][j] {
                _, err = file.WriteString("#")
            } else {
                _, err = file.WriteString(string(b.color[i][j]))
            }
        }

        _, err = file.WriteString("\n")
        if err != nil {
            return err
        }
    }

    return nil
}

func printBoard(b *Board) {

```

```

    for i := 0; i < b.size; i++ {
        for j := 0; j < b.size; j++ {
            if b.queen[i][j] {
                fmt.Printf("#")
            } else {
                fmt.Print(string(b.color[i][j]))
            }
        }
        fmt.Println()
    }
}

func liveBoard(b *Board, counter int) {
    if counter%100 != 0 {
        return
    }

    clearScreen()

    for i := 0; i < b.size; i++ {
        for j := 0; j < b.size; j++ {
            if b.queen[i][j] {
                fmt.Printf("#")
            } else {
                fmt.Print(string(b.color[i][j]))
            }
        }
        fmt.Println()
    }

    time.Sleep(100 * time.Millisecond)
}

func clearScreen() {
    fmt.Print("\033[H\033[2J")
}

```

3. board.go

```

package main

type Board struct {
    size int
    color [][]rune
    queen [][]bool
}

```

```

}

func newBoard(colorMatrix [][]rune) *Board {
    size := len(colorMatrix)

    queenMatrix := make([][]bool, size)
    for i := 0; i < size; i++ {
        queenMatrix[i] = make([]bool, size)
    }

    return &Board {
        size: size,
        color: colorMatrix,
        queen: queenMatrix,
    }
}

func (b *Board) isValid() bool {
    if b.size == 0 {
        return false
    }

    if len(b.color) != b.size {
        return false
    }

    for k:=0; k < b.size; k++ {
        if len(b.color[k]) != b.size {
            return false
        }
    }

    unique := make(map[rune]bool)
    for i:=0; i < b.size; i++ {
        for j:=0; j < b.size; j++ {
            unique[b.color[i][j]] = true
        }
    }
    count := len(unique)

    if count == b.size {
        return true
    }

    return true
}

```

```

}

func (b *Board) isPlaceable(row, col int) bool {
    if b.queen[row][col] == true {
        return false
    }

    for i:=0; i < b.size; i++ {
        if b.queen[row][i] == true {
            return false
        }
    }

    for j:=0; j < b.size; j++ {
        if b.queen[j][col] == true {
            return false
        }
    }

    sides := [8][2]int {
        {-1,-1},{-1,0},{-1,1},{0,-1},{0,1},{1,-1},{1,0},{1,1},
    }

    for _, val := range sides {
        newRow := row + val[0]
        newCol := col + val[1]

        if newRow >= 0 && newCol >= 0 && newRow < b.size && newCol < b.size {
            if b.queen[newRow][newCol] == true {
                return false
            }
        }
    }

    currentColor := b.color[row][col]
    for i:=0; i < b.size; i++ {
        for j:=0; j < b.size; j++ {
            if b.queen[i][j] == true && b.color[i][j] == currentColor {
                return false
            }
        }
    }

    return true
}

```

```

func (b *Board) addQueen(row, col int) {
    b.queen[row][col] = true
}

func (b *Board) rmvQueen(row, col int) {
    b.queen[row][col] = false
}

```

4. solver.go

```

package main

func Solve(b *Board, counter *int) bool {
    rowPlaced := make([]int, b.size)

    for {
        (*counter)++

        for i := 0; i < b.size; i++ {
            for j := 0; j < b.size; j++ {
                if b.queen[i][j] {
                    b.rmvQueen(i, j)
                }
            }
        }

        valid := true
        for row := 0; row < b.size; row++ {
            col := rowPlaced[row]
            if b.isPlaceable(row, col) {
                b.addQueen(row, col)
            } else {
                valid = false
                break
            }
        }

        liveBoard(b, *counter)

        if valid {
            return true
        }

        carry := true
    }
}

```

```

    for row := b.size - 1; row >= 0 && carry; row-- {
        rowPlaced[row]++
        if rowPlaced[row] < b.size {
            carry = false
        } else {
            rowPlaced[row] = 0
        }
    }

    if carry {
        break
    }
}

for i := 0; i < b.size; i++ {
    for j := 0; j < b.size; j++ {
        if b.queen[i][j] {
            b.rmvQueen(i, j)
        }
    }
}
return false
}

```

```

func SolveEffective(b *Board, counter *int) bool {
    row := 0;
    rowPlaced := make([]int, b.size)

    for i:=0; i < len(rowPlaced); i++ {
        rowPlaced[i] = -1
    }

    for row >= 0 {
        var placedOnThisRow bool
        startCol := 0

        if rowPlaced[row] != -1 {
            startCol = rowPlaced[row] + 1
            b.rmvQueen(row, rowPlaced[row])
            liveBoard(b, *counter)
        }

        for i:=startCol; i < b.size; i++ {
            (*counter)++

            if b.isPlaceable(row, i) {

```

```
        b.addQueen(row, i)
        rowPlaced[row] = i

        liveBoard(b, *counter)

        row = row + 1
        placedOnThisRow = true

        break
    }

}

if !placedOnThisRow {
    rowPlaced[row] = -1
    row = row - 1
}

if row == b.size {
    return true
}

}

return false
}
```

BAB III

Input dan Output

Berikut adalah dokumentasi input dan output. Penggunaan algoritma backtracking sepenuhnya hanya untuk solving time, karena untuk n besar, exhaustive search sangat lama.

1. File case-1.txt:

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

Input:

```
nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
Queens Solver Brute Force Approach
Input nama file (no whitespace, put inside test/ folder): case-1.txt
Pakai algoritma exhaustive (1) atau backtracking (2)? Input 1 atau 2: 2
```

Output:

```
nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
AAABBCC#D
ABBB#CECD
ABBBDC#CD
A#ABDCCCD
BBBBD#DDD
FGG#DDHDD
#GIGDDHDD
FG#GDDHDD
FGGGDDHH#

Waktu pencarian: 1215.842 ms
Banyak kasus yang ditinjau: 7659
Apakah Anda ingin menyimpan solusi? (Y/N): Y
File berhasil disimpan di: ../test/solution/case-1.txt
❖ nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$
```

2. File case-2.txt:

```
AABBB
AABCB
ADCCB
DDDCE
DDEEE
```

Input:

```
nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
Queens Solver Brute Force Approach
Input nama file (no whitespace, put inside test/ folder): case-2.txt
Pakai algoritma exhaustive (1) atau backtracking (2)? Input 1 atau 2: 1
```

Output:

```
nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
A#BBB
AABC#
AD#CB
#DDCE
DDE#E

Waktu pencarian: 1106.970 ms
Banyak kasus yang ditinjau: 1179
Apakah Anda ingin menyimpan solusi? (Y/N): Y
File berhasil disimpan di: ../test/solution/case-2.txt
❖ nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$
```

3. File case-3.txt:

```
AABBBBCC
AADDBCCC
AADDBBCC
AADDEECC
FFDDEEGG
FFDDHHHG
FFFFHHGG
FFFFFGGG
```

Input:

```
nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
Queens Solver Brute Force Approach
Input nama file (no whitespace, put inside test/ folder): case-3.txt
Pakai algoritma exhaustive (1) atau backtracking (2)? Input 1 atau 2: 2
```

Output:

```
nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
AA#BBBCC
#ADDBCCC
AAD#BBCC
AADDEE#C
FFDD#EGG
F#DDHHHG
FFFFH#GG
FFFFFG#

Waktu pencarian: 1411.970 ms
Banyak kasus yang ditinjau: 4676
Apakah Anda ingin menyimpan solusi? (Y/N): Y
File berhasil disimpan di: ../test/solution/case-3.txt
❖ nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$
```

4. File case-4.txt:

```
AAAABBBCCCC
AAADBBCCCCC
ADDDDBCCCCC
ADDDDBBEEEE
ADDDFFFEEEE
AGGDDFFFEEEE
AGGGFFHHEEEE
AGGGIIHHHJJ
KKGIIHHHJJ
KKKIIHHHJJ
KKKKIIHHJJ
```

Input:

```
o nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
Queens Solver Brute Force Approach
Input nama file (no whitespace, put inside test/ folder): case-4.txt
Pakai algoritma exhaustive (1) atau backtracking (2)? Input 1 atau 2: 2
```

Output:

```
nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
AAA#BBCCCC
ADDDDBC#CCC
ADDD#BEEEE
ADDDFFFE#E
A#GDDFFFE
AGGG#FHHHEE
AGGGIIHH#JJ
KKGIIHHHJ#
KK#IIHHJJ
KKKKII#HJJ

Waktu pencarian: 1116.091 ms
Banyak kasus yang ditinjau: 5544
Apakah Anda ingin menyimpan solusi? (Y/N): Y
File berhasil disimpan di: ../test/solution/case-4.txt
❖ nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$
```

5. File case-5.txt:

```
AABC
ABBC
ADCC
DDDC
```

Input:

```
o nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
Queens Solver Brute Force Approach
Input nama file (no whitespace, put inside test/ folder): case-5.txt
Pakai algoritma exhaustive (1) atau backtracking (2)? Input 1 atau 2: 1
```

Output:

```
nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
AA#C
#BBC
ADC#
D#DC

Waktu pencarian: 100.456 ms
Banyak kasus yang ditinjau: 142
Apakah Anda ingin menyimpan solusi? (Y/N): Y
File berhasil disimpan di: ../test/solution/case-5.txt
❖ nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$
```

6. File case-6.txt:

```
AABBCCCCCEEEFFGG
ABBBCCCCCEEEFFGG
ADBCCCCCEEEFFGG
DDDCCCCCEEEFFGG
DDDCCCCCEEEFFGG
```

```
DDDHCCCEEEFFFGG
DDHHHHHHEEEFFGGG
HHHHHHHJJJEFFIGG
HHHHHKJJJJIIII
HHHHKKJJJJIIIL
KKKKKKKKJJJJIIIL
KKKKKKMMJJNLLLL
MMMMMMMMMMNNNLLL
MMMMMMMONNNNNLL
OOOOOOOONNNNLL
```

Input:

```
nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
Queens Solver Brute Force Approach
Input nama file (no whitespace, put inside test/ folder): case-6.txt
Pakai algoritma exhaustive (1) atau backtracking (2)? Input 1 atau 2: 2
```

Output:

```
nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$ ./solver
#ABCCCCCEEEFFGG
AB#BCCCCCEEEFFGG
ADBC#CCCEEEFFGG
D#DCCCCCEEEFFGG
DDDCC#EEEEFFGG
DDD#CCCEEEFFGG
DDHHHHHEEE#FGGG
HHHHHHHJJJEFFI#G
HHHHH#JJJJIIIII
HHHHKKJJJJIIIL
KKKKKKKJJJJ#ILL
KKKKKKM#JJNLLLL
MMMMMMMMMMNNNLL#
MMMMMMMONNNN#LL
OOOOOOO#NNNNLL

Waktu pencarian: 710.837 ms
Banyak kasus yang ditinjau: 3180
Apakah Anda ingin menyimpan solusi? (Y/N): Y
File berhasil disimpan di: ../test/solution/case-6.txt
❖ nathan@nathan-vivobook:~/Documents/code/Tucil1_13524122/bin$
```

Lampiran

Link repository Github: https://github.com/Vearance/Tucil1_13524122

No.	Poin	Ya	Tidak
1	Program berhasil di kompilasi 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.



Nathaniel Christian