

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL 5



**DISUSUN OLEH:
RIZKINA AZIZAH
103112400082
S1 IF-12-01**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

DASAR TEORI

Apa itu Benchmark Tests?

Dalam Go, Benchmark Tests digunakan untuk mengukur kinerja (kecepatan dan penggunaan memori) fungsi atau blok kode. Pengujian ini merupakan bagian dari kerangka pengujian Go dan ditulis dalam berkas yang sama dengan pengujian unit, tetapi pengujian ini khusus untuk analisis kinerja.

Contoh Kasus Penggunaan: Urutan Fibonacci

Ada beberapa cara untuk mengimplementasikan kode ini, dan saya akan memilih dua di antaranya untuk pengujian patokan: metode rekursif dan iteratif. Tujuan utama fungsi ini adalah untuk memberikan *posisi* dan mengembalikan angka Fibonacci pada posisi tersebut.

a. Metode Rekursif

Urutan langkah penulisannya:

1. Fungsi

```
func fibRecursive(n uint) uint
```

- `func`: Kata kunci ini mendefinisikan fungsi dalam Go.
- `fibRecursive`: Ini adalah nama fungsinya. Disebut demikian `fibRecursive` karena menghitung angka Fibonacci menggunakan rekursi.
- `n uint`: Fungsi ini mengambil satu argumen, `n`, yang bertipe `uint` (bilangan bulat tak bertanda). Ini mewakili posisi deret Fibonacci yang ingin kita hitung.
- `uint`: Fungsi ini mengembalikan `uint` (bilangan bulat tak bertanda) karena bilangan Fibonacci merupakan bilangan bulat non-negatif.

2. Tahap Dasar

```
if n <= 2 {  
    return 1  
}
```

- Pernyataan tersebut memeriksa apakah `n` kurang dari atau sama dengan 2.
- Dalam deret Fibonacci, angka ke-1 dan ke-2 sama-sama bernilai 1. Jadi, jika `n` bernilai 1 atau 2, fungsi tersebut akan menghasilkan nilai 1.

- Inilah yang disebut **tahap dasar**, dan menghentikan rekurensi agar tidak berlanjut terlalu dalam.

3. Tahap Rekursif

```
return fibRecursive(n-1) + fibRecursive(n-2)
```

- Jika n lebih besar dari 2, fungsi tersebut memanggil dirinya sendiri dua kali:
 - fibRecursive(n-1): Ini akan menghitung angka Fibonacci untuk posisi tepat sebelum n.
 - fibRecursive(n-2): Ini akan menghitung angka Fibonacci untuk dua posisi sebelumnya n.
- Fungsi tersebut kemudian menambahkan kedua hasil tersebut, karena setiap angka Fibonacci adalah jumlah dari dua angka sebelumnya.

b. Metode Iteratif

Urutan Langkah penulisannya:

1. Fungsi

```
func fibIterative(position uint) uint
```

- func: Kata kunci ini mendeklarasikan suatu fungsi dalam Go.
- fibIterative: Nama fungsi tersebut menunjukkan bahwa ia menghitung angka Fibonacci menggunakan iterasi (perulangan).
- position uint: Fungsi ini mengambil satu argumen, position yaitu bilangan bulat tak bertanda (uint). Ini mewakili posisi deret Fibonacci yang ingin Anda hitung.
- uint: Fungsi ini mengembalikan bilangan integer tak bertanda (uint), yang akan menjadi angka Fibonacci pada posisi yang ditentukan.

2. Membuat Slice (Struktur seperti Array)

```
slc := make([]uint, position)
```

slc adalah irisan (array dinamis dalam Go) yang dibuat dengan panjang position.

Irisan ini akan menyimpan angka Fibonacci pada setiap indeks.

3. Nilai Awal dari Deret Fibonacci

```
slc[0] = 1
```

```
slc[1] = 1
```

Dua angka Fibonacci pertama keduanya 1, jadi dua posisi pertama pada irisan (slc[0] dan slc[1]) ditetapkan menjadi 1.

4. Pengembalian Awal Posisi Kecil

```
if position <= 2 {
```

```
    return 1
}
```

Jika inputnya position adalah 1 atau 2, fungsi langsung mengembalikan 1, karena dua angka Fibonacci pertama selalu 1.

5. Loop Iteratif

```
var result, i uint
for i = 2; i < position; i++ {
    result = slc[i-1] + slc[i-2]
    slc[i] = result
}
```

- Perulangan dimulai dari $i = 2$ dan berjalan hingga mencapai position.
- Dalam setiap iterasi, angka Fibonacci pada indeks i dihitung sebagai jumlah dari dua angka Fibonacci sebelumnya ($slc[i-1]$ dan $slc[i-2]$).
- Hasilnya disimpan di dalam result dan di dalam irisan $slc[i]$ untuk perhitungan selanjutnya.

6. Mengembalikan Hasil

```
return result
```

Setelah perulangan selesai, variabel result menampung angka Fibonacci pada posisi yang diinginkan, dan fungsi mengembalikannya.

GUIDED

1. Latihan1

Source Code:

```
package main

import "fmt"

// Fungsi Iteratif untuk menghitung pangkat (base^exp)
func PangkatIteratif(base, exp int) int {
    hasil := 1
    for i := 0; i < exp; i++ {
        hasil *= base
    }
    return hasil
}

//Fungsi iteratif untuk menghitung faktorial (n!)
func faktorialIteratif(n int) int {
    hasil := 1
    for i := 2; i <= n; i++ {
        hasil*= i
    }
    return hasil
}

func main() {
    var base, exp, n int

    // Input Pangkat
    fmt.Print("Masukkan Bilangan: ")
    fmt.Scanln(&base)
    fmt.Print("Masukkan Pangkat: ")
    fmt.Scanln(&exp)
```

```

fmt.Printf("%d^%d = %d\n", base, exp, PangkatIteratif(base, exp))

//Input faktorial
fmt.Print("Masukkan angka untuk faktorial: ")
fmt.Scanln(&n)

fmt.Printf("%d! = %d\n", n, faktorialIteratif(n))
}

```

Deskripsi Program:

- Program ini digunakan untuk perhitungan pangkat dan faktorial tanpa menggunakan rekursi.
- Metode iteratif digunakan pada *func PangkatIteratif* dan *func faktorialIteratif*
- *func PangkatIteratif* berfungsi untuk mengalikan base sebanyak exp dan mengembalikan hasilnya
- *func faktorialIteratif* berfungsi untuk mengalikan 2 hingga n dan mengembalikan hasil faktorial n
- *func main* berfungsi untuk:
 - input base dan exp
 - Memanggil *func PangkatIteratif*
 - Input n
 - Memanggil *func faktorialIteratif*

2. Latihan2

Source Code:

```

package main

import "fmt"

func pangkatRekursif(base, exp int) int {
    if exp == 0 {

```

```

        return 1
    }
    return base * pangkatRekursif(base, exp-1)
}
func faktorialRekursif(n int) int {
    if n == 0 || n == 1 {
        return 1
    }
    return n * faktorialRekursif(n-1)
}
func main() {
    var base, exp, n int
    // Input pangkat
    fmt.Print("Masukkan bilangan: ")
    fmt.Scanln(&base)
    fmt.Print("Masukkan pangkat: ")
    fmt.Scanln(&exp)

    fmt.Printf("%d^%d = %d\n", base, exp, pangkatRekursif(base, exp))

    // Input faktorial
    fmt.Print("Masukkan angka untuk faktorial: ")
    fmt.Scanln(&n)

    fmt.Printf("%d! = %d\n", n, faktorialRekursif(n))
}

```

Deskripsi Program:

- Program ini digunakan untuk perhitungan pangkat dan faktorial tanpa menggunakan rekursi.
- Metode rekursif digunakan pada *func PangkatIteratif* dan *func faktorialIteratif*
- *func PangkatIteratif* berfungsi untuk menentukan :

- Jika `exp == 0` maka mengembalikan 1
- Jika bukan maka akan mengembalikan `base * pangkatRekursif(base,exp-1)`, yang berarti pangkat dihitung dengan cara mengalikan base berulang kali hingga `exp` mencapai 0
- *func faktorialIteratif* berfungsi untuk menentukan:
 - Jika `n == 0` atau `n == 1` maka mengembalikan 1
 - Jika bukan maka akan mengembalikan `n*faktorialRekursif(n-1)`, yang berarti faktorial dihitung dengan cara mengalikan n dengan faktorial dari n-1 hingga mencapai basis rekursi (`n==1`)
- *func main* berfungsi untuk:
 - input base dan exp
 - Memanggil *func PangkatIteratif*
 - Input n
 - Memanggil *func faktorialIteratif*

UNGUIDED

1. Latihan 1

Source Code:

```
package main

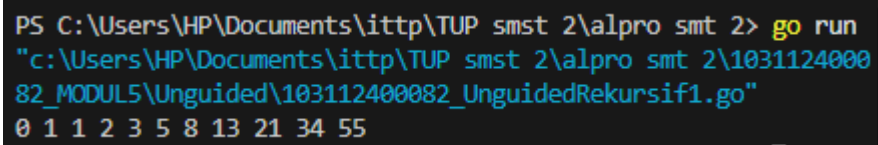
import "fmt"

func fibonacci(n int) int {
    if n == 0 {
        return 0
    } else if n == 1 {
        return 1
    }
    return fibonacci(n-1) + fibonacci(n-2)
}
```



```
func main() {
    for i := 0; i <= 10; i++ {
        fmt.Print(fibonacci(i), " ")
    }
    fmt.Println()
}
```

Output:



```
PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run
"c:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2\10311240082
82_MODUL5\Unguided\10311240082_UnguidedRekursif1.go"
0 1 1 2 3 5 8 13 21 34 55
```

Deskripsi Program:

- Program ini digunakan untuk menampilkan deret fibonacci
- *Func fibonacci* berfungsi menentukan:
 - Basis pertama jika $n == 0$, maka mengembalikan 0
 - Basis kedua jika $n == 1$, maka mengembalikan 1
 - Jika bukan, maka mengembalikan hasil dari penjumlahan dua angka sebelumnya dengan rumus $\text{fibonacci}(n-1) + \text{fibonacci}(n-2)$
- *Func main* berfungsi untuk mencetak 11 angka deret fibonacci menggunakan paradigma perulangan
- *fmt.Print(fibonacci(i), " ")* untuk memisahkan setiap angka deret menggunakan spasi

2. Latihan 2

Source Code:

```
package main

import "fmt"

func bintang(j int) {
    if j <= 0 {
        return
    }
}
```

```
    }  
    fmt.Print("*")  
    bintang(j - 1)  
}  
  
func baris(i, tinggi int) {  
    if i > tinggi {  
        return  
    }  
    bintang(i)  
    fmt.Println()  
    baris(i + 1, tinggi)  
}  
  
func main() {  
    var tinggi int  
    fmt.Scan(&tinggi)  
  
    baris(1, tinggi)  
}
```

Output:

```

PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run
"c:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2\1031124000
82_MODUL5\Unguided\103112400082_UnguidedRekursif2.go"
5
*
**
***
****
*****
PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run
"c:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2\1031124000
82_MODUL5\Unguided\103112400082_UnguidedRekursif2.go"
1
*
PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run
"c:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2\1031124000
82_MODUL5\Unguided\103112400082_UnguidedRekursif2.go"
3
*
**
***

```

Deskripsi program:

- Program ini digunakan untuk mencetak pola segitiga bintang (*).
- *Func bintang* berfungsi untuk:
 - Mencetak j bintang (*) secara rekursif.
 - Basis rekursi: Jika $j \leq 0$ maka fungsi berhenti.
 - Rekursi: Memanggil dirinya sendiri dengan $j-1$, sehingga setiap pemanggilan akan mencetak satu bintang hingga j habis.
- *Func baris* berfungsi untuk:
 - Mencetak baris
 - Basis rekursi: Jika $i > \text{tinggi}$ maka fungsi berhenti.
 - Rekursi: Memanggil dirinya sendiri dengan $i + 1$, sehingga setiap pemanggilan akan mencetak satu baris bintang dengan jumlah bintang bertambah.
- *Func main* berfungsi untuk:
 - Input tinggi segitiga
 - Memanggil *baris(1, tinggi)* untuk mencetak segitiga bintang dari 1 hingga tinggi.

3. Latihan 3

Source Code:

```

package main

import "fmt"

func faktor(n, i int) {
    if i > n {
        return
    }

    if n%i == 0 {
        fmt.Print(i, " ")
    }

    faktor(n, i+1)
}

func main() {
    var n int
    fmt.Scan(&n)

    faktor(n, 1)
    fmt.Println()
}

```

Output:

```

PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run
"c:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2\1031124000
82_MODUL5\Unguided\103112400082_UnguidedRekursif3.go"
5
1 5
PS C:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2> go run
"c:\Users\HP\Documents\ittp\TUP smst 2\alpro smt 2\1031124000
82_MODUL5\Unguided\103112400082_UnguidedRekursif3.go"
12
1 2 3 4 6 12

```

Deskripsi Program:

- Program ini digunakan untuk mencetak faktor
- *Func faktor* berfungsi untuk menentukan:
 - Basis rekursi: Jika $i > n$ maka fungsi berhenti.
 - Pengecekan faktor: Jika n habis dibagi I , maka i adalah faktor dari n dan akan dicetak
 - Rekursi: Memanggil dirinya sendiri dengan $i+1$ untuk mengecek angka berikutnya hingga $i > n$
- *Func main* berfungsi untuk:
 - Input n
 - Memanggil *func faktor*($n, 1$) untuk mencari faktor mulai dari 1 hingga n
 - Menampilkan output dengan spasi

DAFTAR PUSTAKA

Pedro Bertao(2024), How to Write Benchmark Tests for Your Golang Functions
(<https://medium.com/@felipedutraine/iterative-vs-recursive-vs-tail-recursive-in-golang-c196ca5fd489> ,2024)