

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
PENGURUTAN DATA
MODUL XII DAN XIII**



Disusun Oleh :

Rakha Arbiyandanu / 2311102263

IF-11-6

Dosen Pengampu :

ABEDNEGO DWI SEPTIADI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data adalah proses menyusun elemen-elemen dalam suatu koleksi, seperti array atau slice, berdasarkan urutan tertentu, baik secara ascending (menaik) maupun descending (menurun). Pengurutan mempermudah pengolahan data, pencarian elemen, analisis, dan presentasi informasi. Dalam pemrograman, pengurutan yang efisien sangat penting untuk memastikan kinerja aplikasi yang optimal, terutama saat menangani data dalam jumlah besar. Golang, sebagai bahasa pemrograman modern, menyediakan berbagai fasilitas untuk mendukung pengurutan data melalui paket bawaan `sort`.

Paket `sort` dalam Golang mempermudah pengurutan data untuk tipe-tipe dasar seperti integer, float, dan string. Fungsi-fungsi bawaan seperti `sort.Ints()`, `sort.Float64s()`, dan `sort.Strings()` memungkinkan pengurutan slice dengan cepat tanpa memerlukan logika tambahan. Misalnya, `sort.Ints()` akan mengurutkan slice integer dalam urutan menaik. Fungsi-fungsi ini dirancang untuk efisiensi dan kemudahan penggunaan, sehingga ideal untuk kebutuhan pengurutan sederhana.

Selain itu, untuk pengurutan yang lebih kompleks, Golang menyediakan mekanisme kustomisasi melalui interface `sort.Interface`. Interface ini memerlukan implementasi tiga metode: `Len()` untuk mendapatkan panjang koleksi, `Less(i, j int)` untuk menentukan aturan pengurutan (misalnya, elemen mana yang lebih kecil), dan `Swap(i, j int)` untuk menukar posisi elemen. Dengan interface ini, pengembang dapat mengurutkan data berdasarkan kriteria tertentu, seperti mengurutkan struct berdasarkan atribut seperti nama, nilai, atau tanggal.

Kemampuan kustomisasi ini memungkinkan pengurutan berbagai jenis data secara fleksibel. Misalnya, jika Anda memiliki slice berisi struct siswa dengan atribut `Name` dan `Grade`, Anda dapat menggunakan interface `sort.Interface` untuk mengurutkan siswa berdasarkan nilai mereka, baik secara ascending maupun descending. Proses ini memberikan kontrol penuh kepada pengembang untuk menentukan logika pengurutan yang sesuai dengan kebutuhan aplikasi.

Keunggulan utama pengurutan data di Golang terletak pada efisiensi, fleksibilitas, dan integrasi yang baik dengan struktur data bawaan bahasa tersebut. Paket `sort` menggunakan algoritma pengurutan yang dirancang untuk performa tinggi, menjadikannya andal untuk pengolahan data dalam skala besar. Dengan fungsi bawaan yang mudah digunakan serta dukungan untuk pengurutan kustom melalui interface, Golang menawarkan solusi yang kuat dan efisien untuk berbagai skenario pengurutan data, menjadikannya alat yang esensial dalam pengembangan aplikasi modern.

I. GUIDED

Soal Studi Case

Hercules, preman terkenal seantero ibukota, memiliki kerabat di banyak daerah. Tentunya Hercules sangat suka mengunjungi semua kerabatnya itu. Diberikan masukan nomor rumah dari semua kerabatnya di suatu daerah, buatlah program rumahkerabat yang akan menyusun nomor-nomor rumah kerabatnya secara terurut membesar menggunakan algoritma selection sort.

Sourcecode

```
package main

import (
    "fmt"
)

func selectionSort(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        idxMin := i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}

func main() {
    var n int
    fmt.Print("Masukkan jumlah daerah kerabat (n): ")
    fmt.Scan(&n)

    for daerah := 1; daerah <= n; daerah++ {
        var m int
        fmt.Printf("Masukkan jumlah nomor rumah kerabat\nuntuk daerah %d: ", daerah)
        fmt.Scan(&m)

        arr := make([]int, m)
        fmt.Printf("Masukkan %d nomor rumah kerabat: ", m)
        for i := 0; i < m; i++ {
            fmt.Scan(&arr[i])
        }

        selectionSort(arr, m)

        fmt.Printf("Nomor rumah terurut untuk daerah %d:\n", daerah)
        for _, num := range arr {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}
```

```
}  
}
```

Screenshot Output

```
PS D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\23111802263_Rakha Arbiyandaru_modul 12> go run "D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\23111802263_Rakha Arbiyandaru_modul 12\guided1\guided1.go"  
Masukkan jumlah daerah kerabat (n): 3  
Masukkan jumlah nomor rumah kerabat untuk daerah 1: 6  
Masukkan 6 nomor rumah kerabat: 5 2 1 7 9 13  
Nomor rumah terurut untuk daerah 1: 1 2 5 7 9 13  
Masukkan jumlah nomor rumah kerabat untuk daerah 2: 7  
Masukkan 7 nomor rumah kerabat: 6 189 15 27 39 75 133  
Nomor rumah terurut untuk daerah 2: 6 15 27 39 75 133 189  
Masukkan jumlah nomor rumah kerabat untuk daerah 3: 4  
Masukkan 4 nomor rumah kerabat: 3 4 9 1  
Nomor rumah terurut untuk daerah 3: 3 4 9  
PS D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\23111802263_Rakha Arbiyandaru_modul 12>
```

Deskripsi Program

Program di atas digunakan buat ngurutin nomor rumah kerabat di beberapa daerah biar rapi. Pertama, dimasukkan jumlah daerah, lalu untuk tiap daerah diminta jumlah nomor rumahnya dan nomor-nomornya. Setelah itu, nomor-nomor tadi diurutin pakai Selection Sort—metode yang nyari nomor paling kecil terus ditaruh di tempat yang pas. Hasil akhirnya, nomor rumah di tiap daerah ditampilkan dalam urutan yang rapi dan gampang dicek.

II. GUIDED

Soal Studi Case

Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metoda insertion sort), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Insertion Sort
func insertionSort(arr []int, n int) {
    for i := 1; i < n; i++ {
        key := arr[i]
        j := i - 1

        // Geser elemen yang lebih besar dari key ke kanan
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

// Fungsi untuk memeriksa apakah selisih elemen array tetap
func isConstantDifference(arr []int, n int) (bool, int) {
    if n < 2 {
        return true, 0
    }

    difference := arr[1] - arr[0]
    for i := 1; i < n-1; i++ {
        if arr[i+1]-arr[i] != difference {
            return false, 0
        }
    }
    return true, difference
}

func main() {
    var arr []int
```

```

var num int

// Input data hingga bilangan negatif ditemukan
fmt.Println("Masukkan data integer (akhiri dengan
bilangan negatif):")
for {
    fmt.Scan(&num)
    if num < 0 {
        break
    }
    arr = append(arr, num)
}

n := len(arr)

// Urutkan array menggunakan Insertion Sort
insertionSort(arr, n)

// Periksa apakah selisih elemen tetap
isConstant, difference := isConstantDifference(arr, n)

// Tampilkan hasil pengurutan
fmt.Println("Array setelah diurutkan:")
for _, val := range arr {
    fmt.Printf("%d ", val)
}
fmt.Println()

// Tampilkan status jarak
if isConstant {
    fmt.Printf("Data berjarak %d\n", difference)
} else {
    fmt.Println("Data berjarak tidak tetap")
}
}

```

Screenshoot Output

```

PS D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311102263_Rakha Arbiyandaru_modul 12> go run "D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311102263_Rakha Arbiyandaru_modul 12\guides2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
4 40 14 8 26 1 39 2 32 -31
Array setelah diurutkan:
1 2 4 8 14 26 32 38 40
Data berjarak tidak tetap
PS D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311102263_Rakha Arbiyandaru_modul 12>

```

Deskripsi Program

Program ini digunakan buat ngurutin angka yang dimasukkan menggunakan algoritma *Insertion Sort* dan mengecek apakah selisih antar angkanya tetap atau nggak setelah diurutkan. Angka-angka dimasukkan satu per satu (proses berhenti kalau angka negatif dimasukkan), lalu diurutkan biar rapi. Setelah itu, program akan memeriksa selisih antara angka pertama ke kedua, kedua ke ketiga, dan seterusnya, untuk melihat apakah selisihnya sama.

Kalau iya, hasilnya ditampilkan dengan jaraknya, kalau nggak, akan dibilang selisihnya nggak tetap.

I. UNGUIDED

Soal Studi Case

Belakangan diketahui ternyata Hercules itu tidak berani menyeberang jalan, maka selalu diusahakan agar hanya menyeberang jalan sesedikit mungkin, hanya diujung jalan. Karena nomor rumah sisi kiri jalan selalu ganjil dan sisi kanan jalan selalu genap, maka buatlah program kerabat dekat yang akan menampilkan nomor rumah mulai dari nomor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

func main() {
    var t int
    fmt.Print("Masukkan jumlah daerah kerabat: ")
    fmt.Scan(&t)

    for daerah := 1; daerah <= t; daerah++ {
        var n int
        fmt.Printf("Masukkan jumlah nomor rumah untuk daerah %d: ", daerah)
        fmt.Scan(&n)

        arr := make([]int, n)
        fmt.Printf("Masukkan %d nomor rumah: ", n)
        for i := 0; i < n; i++ {
            fmt.Scan(&arr[i])
        }

        ganjil := []int{}
        genap := []int{}
        for _, num := range arr {
            if num%2 == 0 {
                genap = append(genap, num)
            } else {
                ganjil = append(ganjil, num)
            }
        }

        sort.Slice(ganjil, func(i, j int) bool {
            return ganjil[i] > ganjil[j]
        })
    }
}
```

```

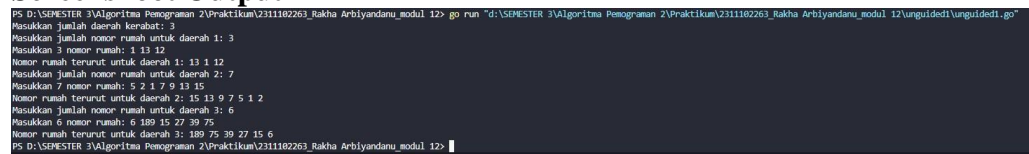
    })

    sort.Ints(genap)

    // Cetak hasil
    fmt.Printf("Nomor rumah terurut untuk daerah %d:
", daerah)
    for _, num := range ganjil {
        fmt.Printf("%d ", num)
    }
    for _, num := range genap {
        fmt.Printf("%d ", num)
    }
    fmt.Println()
}
}

```

Screenshoot Output



```

PS D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311102263_Rakha Arbiyandaru_modul 12> go run "d:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311102263_Rakha Arbiyandaru_modul 12\unguided\unguided.go"
Masukkan jumlah daerah kerabat: 3
Masukkan jumlah nomor rumah untuk daerah 1: 3
Masukkan 3 nomor rumah: 1 13 12
Nomor rumah terurut untuk daerah 1: 13 1 12
Masukkan jumlah nomor rumah untuk daerah 2: 7
Masukkan 7 nomor rumah: 5 2 1 7 9 13 15
Nomor rumah terurut untuk daerah 2: 15 13 9 7 5 1 2
Masukkan jumlah nomor rumah untuk daerah 3: 6
Masukkan 6 nomor rumah: 6 189 15 27 39 75
Nomor rumah terurut untuk daerah 3: 189 75 39 27 15 6
PS D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311102263_Rakha Arbiyandaru_modul 12>

```

Deskripsi Program

Program ini buat ngatur nomor rumah dari beberapa daerah biar rapi sesuai aturan. Jadi, setiap nomor rumah yang dimasukkan bakal dipisah jadi dua kelompok: ganjil dan genap. Nomor ganjil diurutin dari yang paling gede ke yang paling kecil, sementara nomor genap diurutin dari yang paling kecil ke yang paling gede. Setelah itu, hasilnya ditampilkan, dimulai dari semua angka ganjil dulu baru angka genap. Proses ini diulang untuk tiap daerah, dan hasil akhirnya ditampilin satu per satu sesuai urutan daerahnya.

II. UNGUIDED

Soal Studi Case

Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi temama. Dalam kompetisi tersebut, setiap tim berlomba untuk menyelesaikan sebanyak mungkin problem yang diberikan. Dari 13 problem yang diberikan, ada satu problem yang menarik. Problem tersebut mudah dipahami, hampir semua tim mencoba untuk menyelesaikannya, tetapi hanya 3 tim yang berhasil. Apa sih problemnya

Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

func hitungMedian(data []int) float64 {
    jumlah := len(data)
    if jumlah%2 == 1 {
        return float64(data[jumlah/2])
    }

    return float64(data[jumlah/2-1]+data[jumlah/2]) / 2.0
}

func main() {
    var data []int
    var angka int

    fmt.Println("Masukkan data bilangan bulat (akhiri dengan -5313):")
    for {
        fmt.Scan(&angka)
        if angka == -5313 {
            break
        }
        if angka != 0 {
            data = append(data, angka)
        } else {
            if len(data) == 0 {
                fmt.Println("Median: Tidak ada data.")
            } else {
```

```

        sort.Ints(data)
        median := hitungMedian(data)
        fmt.Printf("Median: %.0f\n", median)
    }
    data = []int{}
}
}
}

```

Screenshoot Output

```

PS D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311180263_Rakha Arbiyandaru_nodul 12> go run "D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311180263_Rakha Arbiyandaru_nodul 12\unguided\tempCodeRunnerFile.go"
Masukkan data bilangan bulat (akhiri dengan -5313):
2 23 11 0 5 19 2 3 13 17 29 0 -5313
Median: 11
Median: 13
PS D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311180263_Rakha Arbiyandaru_nodul 12>

```

Deskripsi Program

Program ini bakal minta input deretan angka, dan setiap baris diakhiri dengan angka 0. Setelah angka 0 dimasukkan, program bakal urutin angka-angka yang sudah dimasukkan, lalu menghitung median. Kalau jumlah angka ganjil, median adalah angka yang ada di tengah, sedangkan kalau jumlahnya genap, median dihitung dari rata-rata dua angka di tengah. Proses ini diulang terus sampai angka -5313 dimasukkan untuk berhenti. Hasil median bakal ditampilkan setiap kali perhitungan selesai, dan data akan direset untuk baris berikutnya.

III. UNGUIDED

Soal Studi Case

Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metoda insertion sort), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.

Sourcecode

```
package main

import (
    "fmt"
)

const nMax = 7919

type Buku struct {
    id    int
    judul string
    penulis string
    penerbit string
    eksemplar int
    rating int
}

type DataBuku struct {
    Buku    []Buku
    Pustaka string
    nPustaka int
}

func main() {
    var N int
    var pustaka DataBuku

    fmt.Print("Masukkan jumlah buku yang ada di dalam perpustakaan: ")
    fmt.Scan(&N)

    for i := 0; i < N; i++ {
        var buku Buku
        fmt.Printf("Masukkan data buku ke-%d:\n", i+1)
```

```

        fmt.Print("ID Buku: ")
        fmt.Scan(&buku.id)

        fmt.Print("Judul Buku: ")
        fmt.Scan(&buku.judul)

        fmt.Print("Penulis Buku: ")
        fmt.Scan(&buku.penulis)

        fmt.Print("Penerbit Buku: ")
        fmt.Scan(&buku.penerbit)

        fmt.Print("Jumlah Eksemplar: ")
        fmt.Scan(&buku.eksemplar)

        fmt.Print("Rating Buku (1-5): ")
        fmt.Scan(&buku.rating)

        pustaka.Buku = append(pustaka.Buku, buku)
    }

    fmt.Println("Data Buku di Perpustakaan:")
    for _, buku := range pustaka.Buku {
        fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s,
Eksemplar: %d, Rating: %d\n",
            buku.id, buku.judul, buku.penulis, buku.penerbit,
            buku.eksemplar, buku.rating)
    }
}

```

Screenshot Output

```

PS D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311182263_Rakha Arbiyandani_modul 12> go run "d:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311182263_Rakha Arbiyandani_modul 12\ungulides\ungulides.go"
Masukkan jumlah buku yang ada di dalam perpustakaan: 2
Masukkan data buku ke-1:
ID Buku: 1
Judul Buku: Pemrograman
Penulis Buku: John
Penerbit Buku: Telkom
Jumlah Eksemplar: 4
Rating Buku (1-5): 2
Masukkan data buku ke-2:
ID Buku: 2
Judul Buku: Paman
Penulis Buku: Rakha
Penerbit Buku: Satria
Jumlah Eksemplar: 2
Rating Buku (1-5): 5
Data Buku di Perpustakaan (diurutkan berdasarkan rating):
ID: 1, Judul: Pemrograman, Penulis: John, Penerbit: Telkom, Eksemplar: 4, Rating: 2
ID: 2, Judul: Paman, Penulis: Rakha, Penerbit: Satria, Eksemplar: 2, Rating: 5
PS D:\SEMESTER 3\Algoritma Pemrograman 2\Praktikum\2311182263_Rakha Arbiyandani_modul 12>

```

Deskripsi Program

Program ini terdiri dari dua bagian utama. Pertama, program menerima input angka, lalu mengurutkannya pakai metode **Insertion Sort**. Setelah itu, program memeriksa apakah selisih antar angka tetap sama atau enggak. Kalau selisihnya konsisten, bakal muncul pesan "Data berjarak tetap", kalau enggak, muncul "Data berjarak tidak tetap". Bagian kedua, program digunakan buat mengelola data buku di perpustakaan. Setiap buku punya ID, judul, penulis, penerbit, jumlah eksemplar, dan rating. Program akan mengurutkan buku-buku berdasarkan rating menggunakan Insertion Sort dan menampilkan hasilnya.