

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 12 & 13  
PENGURUTAN DASAR**



**Disusun Oleh :**

**M. Faleno Albar Firjatulloh / 2311102297**

**S1-IF-11-05**

**Dosen Pengampu :**

**Arif Amrulloh, S.Kom.,M.Kom**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## **I. DASAR TEORI**

### **A. Penjelasan Pengurutan Data**

Pengurutan data merupakan proses menyusun elemen-elemen dalam sebuah daftar sesuai dengan urutan tertentu, seperti dari yang terkecil ke terbesar atau sebaliknya. Metode ini sering digunakan di berbagai bidang untuk mempercepat pencarian, analisis data, serta pemrosesan lainnya.

#### **1. Ide Algoritma Selection Sort**

Selection Sort adalah salah satu algoritma pengurutan yang bekerja dengan cara mencari nilai ekstrem (terkecil atau terbesar) dari sekumpulan data yang belum diurutkan, kemudian menempatkannya pada posisi yang sesuai. Misalnya, jika pengurutan dilakukan secara menaik (ascending), algoritma akan mencari nilai terkecil dari data yang tersisa, lalu menukarnya dengan data yang berada pada posisi paling kiri dari rentang tersebut. Proses ini diulang secara bertahap dengan mempersempit rentang data yang belum diurutkan hingga hanya tersisa satu elemen. Dalam Selection Sort, ada dua langkah utama, yaitu mencari nilai ekstrem dan melakukan pertukaran (swap) posisi dengan elemen tertentu. Algoritma ini memiliki keunggulan dalam kesederhanaannya, tetapi kurang efisien untuk dataset yang besar karena kompleksitas waktu yang dimilikinya adalah  $O(n^2)$ .

#### **2. Algoritma Selection Sort**

Algoritma Selection Sort bekerja dengan langkah-langkah berikut: Pertama, mulai dari indeks awal, carilah elemen terkecil dari bagian daftar yang belum diurutkan. Setelah itu, tukarkan elemen terkecil tersebut dengan elemen yang berada di posisi awal dari bagian yang belum terurut. Selanjutnya, geser batas daftar yang belum diurutkan ke kanan sebanyak satu elemen. Proses ini diulangi hingga seluruh elemen dalam daftar telah tersusun rapi. Kompleksitas waktu algoritma ini, baik dalam kasus terbaik, rata-rata, maupun terburuk, adalah  $O(n^2)$ , di mana  $n$  adalah jumlah elemen dalam daftar. Meskipun algoritma ini sederhana untuk diimplementasikan, namun tidak efisien untuk digunakan pada daftar dengan ukuran besar.

#### **3. Ide Algoritma Insertion Sort**

Insertion Sort adalah salah satu algoritma pengurutan yang bekerja dengan cara menyisipkan elemen ke dalam posisi yang tepat pada bagian daftar yang sudah terurut. Tidak seperti Selection Sort yang

mencari nilai ekstrem terlebih dahulu, Insertion Sort memproses elemen satu per satu dengan membandingkan elemen yang belum terurut dengan elemen-elemen di bagian yang sudah terurut. Jika diperlukan, elemen-elemen yang sudah terurut akan digeser ke kanan untuk memberikan ruang bagi elemen yang sedang diproses agar dapat disisipkan ke posisi yang sesuai. Proses ini diulang untuk setiap elemen yang belum terurut hingga seluruh elemen berada dalam urutan yang benar. Dalam penjelasan pada gambar, data diurutkan secara menurun (descending), sehingga elemen terbesar berada di sisi kiri. Algoritma ini sederhana dan efisien untuk daftar kecil atau yang hampir terurut, tetapi memiliki kompleksitas waktu  $O(n^2)$  dalam kasus rata-rata dan terburuk, sehingga kurang cocok untuk daftar berukuran besar.

#### 4. Algoritma Insertion Sort

Algoritma Insertion Sort memiliki langkah-langkah sebagai berikut: Dimulai dari elemen kedua dalam daftar, dengan asumsi bahwa elemen pertama sudah dalam keadaan terurut. Selanjutnya, elemen tersebut dibandingkan dengan elemen-elemen dalam bagian daftar yang sudah terurut. Jika ada elemen yang lebih besar, geser elemen-elemen tersebut ke kanan untuk memberikan ruang bagi elemen yang sedang diproses. Setelah itu, masukkan elemen tersebut ke posisi yang sesuai dalam bagian yang sudah terurut. Langkah ini diulangi untuk setiap elemen dalam daftar hingga seluruhnya terurut. Kompleksitas waktu algoritma ini adalah  $O(n)$  pada kasus terbaik (ketika daftar hampir terurut) dan  $O(n^2)$  pada kasus terburuk (ketika daftar terbalik). Algoritma ini sangat cocok untuk daftar berukuran kecil atau daftar yang hampir terurut.

## II. GUIDED

### 1. Guided 1

#### Soal Studi Case

Pengurutan nomor rumah kerabat dengan selection sort

#### Sourcecode

```
package main

import (
    "fmt"
)

func selectionSort(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        idxMin := i
        for j := 0; 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("\n Masukkan jumlah nomor kerabat untuk 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("Monor rumah terurut untuk daerah %d: ", daerah)
```

```

        for _, num := range arr {
            fmt.Printf("%d", num)
        }
        fmt.Println()
    }

}

```

### Screenshoot Output

```

PS D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\Modul 12 & 1
3\guided.go"
Masukkan jumlah daerah kerabat (n): 3

Masukkan jumlah nomor kerabat untuk daerah 1: 5 2 1 7 9 13
Masukkan 5 nomor rumah kerabat: Monor rumah terurut untuk daerah 1: 279113

Masukkan jumlah nomor kerabat untuk daerah 2: 6 189 15c27c39c75 133
Masukkan 6 nomor rumah kerabat: Monor rumah terurut untuk daerah 2: 18927397515133

Masukkan jumlah nomor kerabat untuk daerah 3: 3 4 9 1
Masukkan 3 nomor rumah kerabat: Monor rumah terurut untuk daerah 3: 914
PS D:\smt 3\Praktikum Alpro 2>

```

### Deskripsi Program :

Program ini adalah sebuah aplikasi sederhana yang dirancang untuk mengelola dan mengurutkan nomor rumah kerabat dalam berbagai daerah menggunakan algoritma selection sort. Program dimulai dengan meminta pengguna memasukkan jumlah daerah kerabat yang akan diproses. Untuk setiap daerah, pengguna diharuskan memasukkan jumlah nomor rumah kerabat yang ada di daerah tersebut, kemudian memasukkan daftar nomor rumah tersebut.

Setelah nomor rumah dimasukkan, program menggunakan fungsi selection sort untuk mengurutkan nomor rumah dari yang terkecil hingga terbesar. Algoritma selection sort bekerja dengan cara mencari elemen terkecil dalam array yang belum terurut dan menukarnya dengan elemen pertama yang belum terurut. Proses ini diulangi untuk setiap bagian array yang belum terurut, sehingga pada akhirnya menghasilkan array yang terurut secara ascending.

Setelah pengurutan selesai, program akan menampilkan nomor rumah yang sudah terurut untuk setiap daerah. Hal ini membantu pengguna dengan cepat melihat nomor rumah kerabat dalam urutan yang sistematis, yang dapat berguna untuk berbagai keperluan seperti pendataan, perencanaan, atau sekadar organisasi informasi. Program ini memberikan fleksibilitas dengan memungkinkan pengguna memproses beberapa daerah sekaligus dalam satu sesi eksekusi.

## 2. Guided 2

### Soal Studi Case

Pengurutan dan analisis pola selisih elemen array

#### 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:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\Modul 12 & 13\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Array setelah diurutkan:
1 7 13 19 25 31 37 43
Data berjarak 6
PS D:\smt 3\Praktikum Alpro 2>

```

**Deskripsi Program :**

Program ini adalah sebuah aplikasi dalam bahasa pemrograman Go (Golang) yang dirancang untuk melakukan pengelolaan dan analisis sebuah deret bilangan integer. Program dimulai dengan memungkinkan pengguna untuk memasukkan sejumlah bilangan bulat ke dalam sebuah array, dengan proses input berlangsung hingga pengguna memasukkan bilangan negatif sebagai tanda berakhirnya input.

Setelah proses input selesai, program menggunakan algoritma Insertion Sort untuk mengurutkan array tersebut dari nilai terkecil hingga terbesar. Algoritma Insertion Sort bekerja dengan cara membandingkan dan memindahkan elemen-elemen array secara berurutan, sehingga menghasilkan susunan elemen yang terurut dengan efisien.

Selanjutnya, program melakukan pemeriksaan terhadap perbedaan antar elemen dalam array yang telah diurutkan. Melalui fungsi `isConstantDifference()`, program menghitung dan memeriksa apakah selisih antara setiap pasang elemen berurutan konstan atau tetap. Jika selisih antar elemen sama, program akan menampilkan informasi jarak tetap beserta nilai perbedaannya. Namun jika selisih tidak konstan, program akan memberitahukan bahwa data memiliki jarak yang tidak tetap.

Terakhir, program menampilkan array yang telah diurutkan dan memberikan informasi status tentang pola jarak antar elemen, memberikan wawasan tambahan kepada pengguna tentang karakteristik data yang dimasukkan.x

### III. UNGUIDED

#### 1. Unguided 1

**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 normor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

**Sourcecode**

```
package main

import (
    "fmt"
    "sort"
```



```

)

// Struct untuk menyimpan data daerah
type Daerah struct {
    ganjil []int
    genap  []int
}

// Fungsi untuk memisahkan dan mengurutkan nomor
rumah
func prosesNomorRumah(nomor []int) Daerah {
    var hasil Daerah

    // Pemisahan nomor ganjil dan genap
    for _, n := range nomor {
        if n%2 == 0 {
            hasil.genap = append(hasil.genap, n)
        } else {
            hasil.ganjil = append(hasil.ganjil,
n)
        }
    }

    // Pengurutan menggunakan package sort
    sort.Ints(hasil.ganjil)
    // Urutkan ganjil naik
    sort.Sort(sort.Reverse(sort.IntSlice(hasil.
genap))) // Urutkan genap turun

    return hasil
}

func main() {
    var jumlahDaerah int

    // Input jumlah daerah
    fmt.Print("Masukkan jumlah daerah: ")
    fmt.Scan(&jumlahDaerah)

    // Proses untuk setiap daerah
    for i := 1; i <= jumlahDaerah; i++ {
        var jumlahRumah int

        fmt.Printf("\nJumlah rumah di daerah %d:
", i)
        fmt.Scan(&jumlahRumah)
    }
}

```

```

        // Membuat slice untuk nomor rumah
        rumah := make([]int, 0, jumlahRumah)
        fmt.Printf("Masukkan %d nomor rumah: ",
jumlahRumah)

        // Input nomor rumah
        for j := 0; j < jumlahRumah; j++ {
            var nomor int
            fmt.Scan(&nomor)
            rumah = append(rumah, nomor)
        }

        // Proses pengurutan
        hasil := prosesNomorRumah(rumah)

        // Output hasil
        fmt.Printf("\nHasil pengurutan daerah
%d:\n", i)
        // Cetak nomor ganjil
        for _, n := range hasil.ganjil {
            fmt.Printf("%d ", n)
        }
        // Cetak nomor genap
        for _, n := range hasil.genap {
            fmt.Printf("%d ", n)
        }
        fmt.Println()
    }
}

```

## Screenshoot Output

```

PS D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\Modul 12 & 13\unguieded1.go"
unguieded1.go
Masukkan jumlah daerah: 3

Jumlah rumah di daerah 1: 6
Masukkan 6 nomor rumah: 5 2 1 7 9 13

Hasil pengurutan daerah 1:
1 5 7 9 13 2

Jumlah rumah di daerah 2: 6 189 15 27 39 75 133
Masukkan 6 nomor rumah:
Hasil pengurutan daerah 2:
15 27 39 75 133 189

Jumlah rumah di daerah 3: 3 4 9 1
Masukkan 3 nomor rumah:
1 9 4
PS D:\smt 3\Praktikum Alpro 2>

```

### **Deskripsi Program :**

Program ini adalah sebuah aplikasi pengolahan data nomor rumah yang dikembangkan menggunakan bahasa pemrograman Go (Golang). Dirancang untuk mengelola dan mengatur nomor rumah dalam berbagai daerah, program ini memiliki kemampuan unik dalam mengklasifikasi dan mengurutkan nomor-nomor rumah berdasarkan kategori genap dan ganjil.

Inti fungsionalitas program terletak pada fungsi `prosesNomorRumah()`, yang bertugas memisahkan dan mengurutkan nomor rumah. Proses dimulai dengan pengguna memasukkan jumlah daerah yang akan diproses. Untuk setiap daerah, pengguna diminta memasukkan jumlah rumah dan merinci nomor-nomor rumah tersebut. Setelah data dimasukkan, program mengklasifikasikan setiap nomor rumah ke dalam dua kelompok: nomor genap dan nomor ganjil.

Pengurutan data dilakukan dengan strategi khusus menggunakan fungsi bawaan package `sort` Go. Nomor ganjil diurutkan dari terkecil ke terbesar, sementara nomor genap diurutkan dari terbesar ke terkecil. Pendekatan ini menciptakan pola pengurutan yang unik dan terstruktur. Keluaran program menampilkan hasil pengurutan untuk setiap daerah dengan cara sistematis: pertama menampilkan nomor rumah ganjil yang telah diurutkan dari terkecil, kemudian diikuti dengan nomor rumah genap yang diurutkan dari terbesar.

Keunikan program ini terletak pada kemampuannya untuk mengolah data dengan cara yang tidak konvensional, memberikan fleksibilitas dalam pengorganisasian dan presentasi nomor rumah sesuai kriteria tertentu. Dengan dukungan struktur data dan algoritma pengurutan yang efisien, program ini mampu menangani berbagai skenario input nomor rumah dalam jumlah dan variasi yang berbeda-beda.

## **2. Unguided 2**

### **Soal Studi Case**

Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi ternama. 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?

"Median adalah nilai tengah dari suatu koleksi data yang sudah terurut. Jika jumlah data genap, maka nilai median adalah rerata dari kedua nilai tengahnya. Pada problem ini, semua data merupakan bilangan bulat positif, dan karenanya rerata nilai tengah dibulatkan ke bawah." Buatlah program

median yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah 0.

Buatlah program **median** yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah 0.

**Masukan** berbentuk rangkaian bilangan bulat. Masukan tidak akan berisi lebih dari 1000000 data, tidak termasuk bilangan 0. Data 0 merupakan tanda bahwa median harus dicetak, tidak termasuk data yang dicari mediannya. Data masukan diakhiri dengan bilangan bulat -5313.

**Keluaran** adalah median yang diminta, satu data per baris.

No	Masukan	Keluaran
1	7 23 11 0 5 19 2 29 3 13 17 0 -5313	11 12

### Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk melakukan insertion sort
func insertionSort(arr []int) {
    for i := 1; i < len(arr); i++ {
        key := arr[i]
        j := i - 1
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

// Fungsi untuk menghitung median
func hitungMedian(arr []int) float64 {
    n := len(arr)
    if n == 0 {
        return 0
    }

    // Jika jumlah data ganjil
    if n%2 != 0 {
        return float64(arr[n/2])
    }

    // Jika jumlah data genap
```

```

        return float64(arr[(n-1)/2]+arr[n/2]) / 2.0
    }

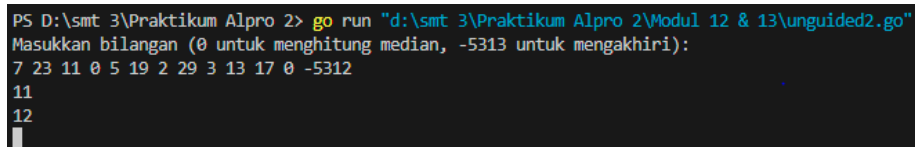
    func main() {
        var data []int
        var input int

        // Membaca input sampai bertemu angka 0 atau
        -5313
        fmt.Println("Masukkan bilangan (0 untuk
menghitung median, -5313 untuk mengakhiri):")
        for {
            fmt.Scan(&input)

            if input == 0 {
                // Urutkan data yang ada dan hitung
median
                if len(data) > 0 {
                    tempData := make([]int, len(data))
                    copy(tempData, data)
                    insertionSort(tempData)
                    median := hitungMedian(tempData)
                    fmt.Printf("%.0f\n", median)
                }
            } else if input == -5313 {
                break
            } else {
                // Tambahkan input ke slice data
                data = append(data, input)
            }
        }
    }
}

```

## Screenshoot Output



```

PS D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\Modul 12 & 13\unguided2.go"
Masukkan bilangan (0 untuk menghitung median, -5313 untuk mengakhiri):
7 23 11 0 5 19 2 29 3 13 17 0 -5312
11
12

```

## Deskripsi Program :

Program Go ini memungkinkan pengguna memasukkan serangkaian bilangan dan menghitung median dengan fitur khusus. Pengguna dapat memasukkan bilangan berapa pun, menekan 0 untuk menghitung median saat itu, dan -5313 untuk mengakhiri program. Menggunakan algoritma

insertion sort untuk mengurutkan data, program menghitung median secara akurat baik untuk jumlah data ganjil maupun genap, kemudian mencetak hasilnya.

### 3. Unguided 3

#### Soal Studi Case

Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Misalnya terdefinisi struct dan array seperti berikut ini:

```
const nMax : integer = 7919
type Buku = <
    id, judul, penulis, penerbit : string
    eksemplar, tahun, rating : integer >

type DaftarBuku = array [ 1..nMax ] of Buku
Pustaka : DaftarBuku
nPustaka: integer
```

**Masukan** terdiri dari beberapa baris. Baris pertama adalah bilangan bulat N yang menyatakan banyaknya data buku yang ada di dalam perpustakaan. N baris berikutnya, masing-masingnya adalah data buku sesuai dengan atribut atau field pada struct. Baris terakhir adalah bilangan bulat yang menyatakan rating buku yang akan dicari.

**Keluaran** terdiri dari beberapa baris. Baris pertama adalah data buku terfavorit, baris kedua adalah lima judul buku dengan rating tertinggi, selanjutnya baris terakhir adalah data buku yang dicari sesuai rating yang diberikan pada masukan baris terakhir.

Lengkapi subprogram-subprogram dibawah ini, sesuai dengan I.S. dan F.S yang diberikan.

```
procedure DaftarkanBuku(in/out pustaka : DaftarBuku, n : integer)
{I.S. sejumlah n data buku telah siap para piranti masukan
 F.S. n berisi sebuah nilai, dan pustaka berisi sejumlah n data buku}

procedure CetakTerfavorit(in pustaka : DaftarBuku, in n : integer)
{I.S. array pustaka berisi n buah data buku dan belum terurut
 F.S. Tampilan data buku (judul, penulis, penerbit, tahun)
 terfavorit, yaitu memiliki rating tertinggi}

procedure UrutBuku( in/out pustaka : DaftarBuku, n : integer )
{I.S. Array pustaka berisi n data buku
 F.S. Array pustaka terurut menurun/mengecil terhadap rating.
 Catatan: Gunakan metoda Insertion sort}

procedure Cetak5Terbaru( in pustaka : DaftarBuku, n integer )
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
 F.S. Laporan 5 judul buku dengan rating tertinggi
 Catatan: Isi pustaka mungkin saja kurang dari 5}

procedure CariBuku(in pustaka : DaftarBuku, n : integer, r : integer )
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
 F.S. Laporan salah satu buku (judul, penulis, penerbit, tahun,
 eksemplar, rating) dengan rating yang diberikan. Jika tidak ada buku
 dengan rating yang ditanyakan, cukup tuliskan "Tidak ada buku dengan
 rating seperti itu". Catatan: Gunakan pencarian biner/belah dua.}
```

### Sourcecode

```
package main

import (
    "fmt"
)

const nMax = 7919

// Struct untuk menyimpan data buku
type Buku struct {
    id          string
    judul       string
    penulis     string
    penerbit    string
    eksemplar   int
    tahun       int
    rating      int
}

// Struct untuk menyimpan daftar buku
type DaftarBuku []Buku
```

```

// Fungsi untuk mendaftarkan buku baru
func DaftarkanBuku(pustaka *DaftarBuku, n *int)
{
    var buku Buku
    fmt.Print("Masukkan ID: ")
    fmt.Scan(&buku.id)
    fmt.Print("Masukkan Judul: ")
    fmt.Scan(&buku.judul)
    fmt.Print("Masukkan Penulis: ")
    fmt.Scan(&buku.penulis)
    fmt.Print("Masukkan Penerbit: ")
    fmt.Scan(&buku.penerbit)
    fmt.Print("Masukkan Jumlah Eksemplar: ")
    fmt.Scan(&buku.eksemplar)
    fmt.Print("Masukkan Tahun: ")
    fmt.Scan(&buku.tahun)
    fmt.Print("Masukkan Rating: ")
    fmt.Scan(&buku.rating)

    *pustaka = append(*pustaka, buku)
    *n++
}

// Fungsi untuk menampilkan buku terfavorit
func CetakTerfavorit(pustaka DaftarBuku, n int)
{
    if n == 0 {
        fmt.Println("Tidak ada buku dalam perpustakaan")
        return
    }

    maxRating := pustaka[0].rating
    maxIndex := 0
    for i := 1; i < n; i++ {
        if pustaka[i].rating > maxRating {
            maxRating = pustaka[i].rating
            maxIndex = i
        }
    }

    fmt.Println("\nBuku Terfavorit:")
    fmt.Printf("Judul:      %s\nPenulis: %s\nPenerbit: %s\nTahun: %d\nRating: %d\n",
        pustaka[maxIndex].judul,
        pustaka[maxIndex].penulis,

```



```

        pustaka[maxIndex].penerbit,
pustaka[maxIndex].tahun,
        pustaka[maxIndex].rating)
    }

// Fungsi untuk mengurutkan buku berdasarkan
rating (Insertion Sort)
func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := (*pustaka)[i]
        j := i - 1
        for j >= 0 && (*pustaka)[j].rating <
key.rating {
            (*pustaka)[j+1] = (*pustaka)[j]
            j--
        }
        (*pustaka)[j+1] = key
    }
}

// Fungsi untuk mencetak 5 buku dengan rating
tertinggi
func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    fmt.Println("\n5 Buku dengan Rating
Tertinggi:")
    limit := 5
    if n < 5 {
        limit = n
    }
    for i := 0; i < limit; i++ {
        fmt.Printf("%d. %s (Rating: %d)\n", i+1,
pustaka[i].judul, pustaka[i].rating)
    }
}

// Fungsi untuk mencari buku berdasarkan rating
(Binary Search)
func CariBuku(pustaka DaftarBuku, n int,
targetRating int) {
    left := 0
    right := n - 1
    found := false

    for left <= right {
        mid := (left + right) / 2
        if pustaka[mid].rating == targetRating {

```

```

        fmt.Printf("\nBuku dengan rating %d
ditemukan:\n", targetRating)
        fmt.Printf("Judul: %s\nPenulis:
%s\nPenerbit: %s\nTahun: %d\nEksemplar: %d\n",
                    pustaka[mid].judul,
pustaka[mid].penulis,
                    pustaka[mid].penerbit,
pustaka[mid].tahun,
                    pustaka[mid].eksemplar)
        found = true
        break
    }
    if pustaka[mid].rating < targetRating {
        right = mid - 1
    } else {
        left = mid + 1
    }
}

if !found {
    fmt.Printf("\nTidak ada buku dengan
rating %d\n", targetRating)
}

func main() {
    var pustaka DaftarBuku
    var n int
    var pilihan int
    var targetRating int

    for {
        fmt.Println("\n=== Menu Perpustakaan
===")
        fmt.Println("1. Daftarkan Buku")
        fmt.Println("2. Tampilkan Buku
Terfavorit")
        fmt.Println("3. Urutkan Buku berdasarkan
Rating")
        fmt.Println("4. Tampilkan 5 Buku Rating
Tertinggi")
        fmt.Println("5. Cari Buku berdasarkan
Rating")
        fmt.Println("6. Keluar")
        fmt.Print("Pilih menu (1-6): ")
        fmt.Scan(&pilihan)
    }
}

```

```

switch pilihan {
case 1:
    DaftarkanBuku(&pustaka, &n)
case 2:
    CetakTerfavorit(pustaka, n)
case 3:
    UrutBuku(&pustaka, n)
    fmt.Println("Buku telah diurutkan
berdasarkan rating")
case 4:
    Cetak5Terbaru(pustaka, n)
case 5:
    fmt.Print("Masukkan rating yang
dicari: ")
    fmt.Scan(&targetRating)
    CariBuku(pustaka, n, targetRating)
case 6:
    fmt.Println("Terima kasih telah
menggunakan program perpustakaan")
    return
default:
    fmt.Println("Pilihan tidak valid")
}
}
}

```

## Screenshot Output

```
PS D:\smt 3\Praktikum Alpro 2> go run "d:\smt 3\Praktikum Alpro 2\Modul 12 & 13\unguided3.go"

=== Manajemen Perpustakaan ===
1. Daftarkan Buku Baru
2. Tampilkan Buku Terfavorit
3. Urutkan Buku berdasarkan Rating
4. Tampilkan 5 Buku Teratas
5. Cari Buku berdasarkan Rating
6. Keluar
Pilih menu (1-6): 1
Masukkan ID Buku: 11
Masukkan Judul: Cinta
Masukkan Penulis: Hamba
Masukkan Penerbit: Penerbit1
Masukkan Jumlah Eksemplar: 4
Masukkan Tahun: 2022
Masukkan Rating: 8

=== Manajemen Perpustakaan ===
1. Daftarkan Buku Baru
2. Tampilkan Buku Terfavorit
3. Urutkan Buku berdasarkan Rating
4. Tampilkan 5 Buku Teratas
5. Cari Buku berdasarkan Rating
6. Keluar
Pilih menu (1-6): 1
Masukkan ID Buku: 22
Masukkan Judul: Dia
Masukkan Penulis: Aku
Masukkan Penerbit: Penerbit3
Masukkan Jumlah Eksemplar: 8
Masukkan Tahun: 2021
Masukkan Rating: 7

=== Manajemen Perpustakaan ===
1. Daftarkan Buku Baru
2. Tampilkan Buku Terfavorit
3. Urutkan Buku berdasarkan Rating
4. Tampilkan 5 Buku Teratas
5. Cari Buku berdasarkan Rating
6. Keluar
Pilih menu (1-6): 2

Buku Terfavorit:
```

Judul: Cinta  
Penulis: Hamba  
Penerbit: Penerbit1  
Tahun: 2022  
Rating: 8

=== Manajemen Perpustakaan ===

1. Daftarkan Buku Baru
2. Tampilkan Buku Terfavorit
3. Urutkan Buku berdasarkan Rating
4. Tampilkan 5 Buku Teratas
5. Cari Buku berdasarkan Rating
6. Keluar

Pilih menu (1-6): 3

Buku telah diurutkan berdasarkan rating

=== Manajemen Perpustakaan ===

1. Daftarkan Buku Baru
2. Tampilkan Buku Terfavorit
3. Urutkan Buku berdasarkan Rating
4. Tampilkan 5 Buku Teratas
5. Cari Buku berdasarkan Rating
6. Keluar

Pilih menu (1-6): 4

5 Buku dengan Rating Tertinggi:

1. Cinta (Rating: 8)
2. Dia (Rating: 7)

=== Manajemen Perpustakaan ===

1. Daftarkan Buku Baru
2. Tampilkan Buku Terfavorit
3. Urutkan Buku berdasarkan Rating
4. Tampilkan 5 Buku Teratas
5. Cari Buku berdasarkan Rating
6. Keluar

Pilih menu (1-6): 5

Masukkan rating yang dicari: 8

Buku dengan rating 8 ditemukan:

Judul: Cinta  
Penulis: Hamba  
Penerbit: Penerbit1  
Tahun: 2022  
Eksemplar: 4

```
=== Manajemen Perpustakaan ===  
1. Daftarkan Buku Baru  
2. Tampilkan Buku Terfavorit  
3. Urutkan Buku berdasarkan Rating  
4. Tampilkan 5 Buku Teratas  
5. Cari Buku berdasarkan Rating  
6. Keluar  
Pilih menu (1-6): 
```

**Deskripsi Program :**

Program manajemen perpustakaan berbasis Go ini memungkinkan pengguna mendaftarkan, mengurutkan, dan mencari buku menggunakan berbagai algoritma seperti insertion sort dan binary search. Melalui menu interaktif, pengguna dapat menambah buku, melihat buku terfavorit, menampilkan 5 buku teratas, dan melakukan pencarian berdasarkan rating, dengan antarmuka yang sederhana dan fungsional.

#### **IV. KESIMPULAN**

Dari laporan praktikum tentang pengurutan dasar menggunakan algoritma Selection Sort dan Insertion Sort, dapat disimpulkan bahwa kedua algoritma ini merupakan metode pengurutan yang sederhana dan mudah dipahami, namun memiliki perbedaan dalam cara kerja dan efisiensinya.

Selection Sort bekerja dengan cara mencari elemen terkecil dalam daftar yang belum terurut dan menukarnya dengan elemen yang ada pada posisi awal bagian tersebut. Proses ini diulang hingga seluruh daftar terurut. Meskipun mudah diterapkan, Selection Sort memiliki kompleksitas waktu  $O(n^2)$  yang menjadikannya tidak efisien untuk dataset yang besar.

Insertion Sort bekerja dengan cara menyisipkan elemen ke dalam posisi yang tepat pada bagian yang sudah terurut. Setiap elemen dibandingkan dengan elemen-elemen sebelumnya dan dipindahkan ke posisi yang sesuai. Insertion Sort lebih efisien daripada Selection Sort pada data yang sudah hampir terurut, dengan kompleksitas waktu  $O(n)$  pada kasus terbaik, namun tetap memiliki kompleksitas  $O(n^2)$  pada kasus terburuk.

Secara keseluruhan, meskipun kedua algoritma ini memiliki kekurangan dalam hal efisiensi untuk dataset besar, keduanya tetap berguna dalam pengurutan data kecil atau data yang hampir terurut. Pemilihan algoritma pengurutan yang tepat sangat bergantung pada karakteristik data yang akan diurutkan dan kebutuhan aplikasi yang bersangkutan.

#### **V. REFERENSI**

[1] Modul 12 & 13 Praktikum Algoritma 2