

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 12 & 13
PENGURUTAN DASAR**



Disusun Oleh :

Shiva Indah Kurnia

2311102035

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 adalah proses mengatur elemen-elemen dalam sebuah daftar berdasarkan urutan tertentu, seperti dari nilai terkecil ke terbesar atau sebaliknya. Proses ini sering dimanfaatkan di berbagai bidang untuk mempercepat proses pencarian, analisis data, dan pengolahan informasi lainnya.

1. Ide Algoritma Selection Sort

Selection Sort adalah algoritma pengurutan yang bekerja dengan menemukan nilai ekstrem (terkecil atau terbesar) dari data yang belum diurutkan, lalu menempatkannya di posisi yang sesuai. Proses ini diulang dengan mempersempit rentang data hingga semua elemen terurut. Algoritma ini sederhana, namun kurang efisien untuk dataset besar karena memiliki kompleksitas waktu $O(n^2)$.

2. Algoritma Selection Sort

Algoritma Selection Sort bekerja dengan mencari elemen terkecil dari bagian yang belum diurutkan, lalu menukarnya dengan elemen di posisi awal. Batas daftar yang belum diurutkan kemudian digeser, dan proses ini diulang hingga semua elemen terurut. Dengan kompleksitas waktu $O(n^2)$ untuk semua kasus, algoritma ini sederhana namun kurang efisien untuk daftar besar.

3. Ide Algoritma Insertion Sort

Insertion Sort adalah algoritma pengurutan yang menyisipkan setiap elemen ke posisi yang tepat dalam bagian yang sudah terurut. Berbeda dengan Selection Sort, algoritma ini membandingkan elemen yang belum terurut dengan elemen di bagian terurut, lalu menggeser elemen jika diperlukan untuk memberikan ruang. Proses ini diulang hingga semua elemen terurut. Algoritma ini sederhana dan efisien untuk daftar kecil atau hampir terurut, namun dengan kompleksitas waktu $O(n^2)$, kurang ideal untuk daftar besar.

4. Algoritma Insertion Sort

Insertion Sort bekerja dengan memproses elemen kedua hingga terakhir dalam daftar, mengasumsikan elemen pertama sudah terurut. Setiap elemen dibandingkan dengan bagian yang sudah

terurut, dan elemen yang lebih besar digeser ke kanan untuk memberi ruang. Elemen yang sedang diproses kemudian ditempatkan di posisi yang sesuai. Langkah ini diulangi hingga seluruh daftar terurut. Algoritma ini efisien untuk daftar kecil atau hampir terurut, dengan kompleksitas waktu $O(n)$ pada kasus terbaik dan $O(n^2)$ pada kasus terburuk.

II. GUIDED

1. Guided 1 Soal Studi Case

Pengurutan nomor rumah kerabat dengan selection sort

```
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

```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu12\guided1.go"
Masukkan jumlah daerah kerabat (n): 3

Masukkan jumlah nomor kerabat untuk daerah 1: 3 2 1
Masukkan 3 nomor rumah kerabat: 5 6 4
Nomor rumah terurut untuk daerah 1: 215

Masukkan jumlah nomor kerabat untuk daerah 2: Masukkan 6 nomor rumah kerabat: 6 5 3 1 4 2
Nomor rumah terurut untuk daerah 2: 653414

Masukkan jumlah nomor kerabat untuk daerah 3: Masukkan 2 nomor rumah kerabat: 7
5
Nomor rumah terurut untuk daerah 3: 57
PS C:\Alpro sem 2> 

```

Deskripsi Program :

Program ini mengimplementasikan algoritma Selection Sort dalam bahasa Go untuk mengurutkan nomor rumah berdasarkan daerah. Pengguna memasukkan jumlah daerah (nnn) dan untuk setiap daerah, jumlah nomor rumah (mmm) beserta daftarnya.

Fungsi selectionSort digunakan untuk mengurutkan daftar dengan memilih elemen terkecil dari bagian yang belum terurut dan menukarnya ke posisi yang sesuai, hingga seluruh elemen terurut. Program memproses setiap daerah secara terpisah dan menampilkan daftar nomor rumah yang sudah diurutkan untuk masing-masing wilayah.

Program ini mempermudah pengurutan data berdasarkan wilayah, cocok untuk kasus dengan banyak daerah dan daftar elemen berbeda.

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)

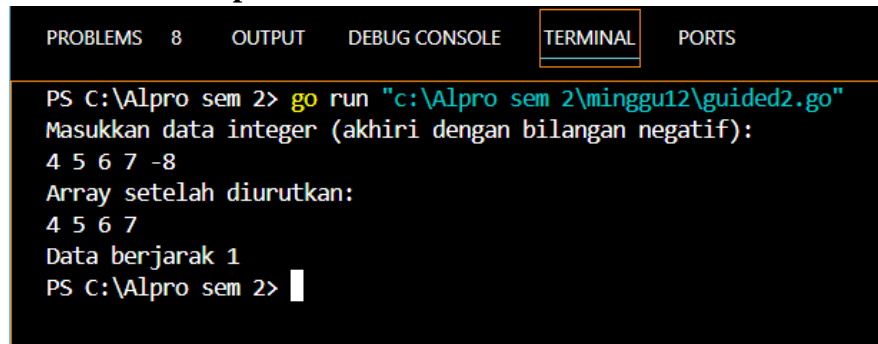
// Periksa apakah selisih elemen tetap
isConstant, difference := isConstantDifference(arr,
n)

// 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 C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu12\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
4 5 6 7 -8
Array setelah diurutkan:
4 5 6 7
Data berjarak 1
PS C:\Alpro sem 2> 
```

Deskripsi Program :

Program ini mengimplementasikan algoritma Insertion Sort dalam Go untuk mengurutkan array integer dan memeriksa apakah selisih antar elemen konstan. Pengguna memasukkan bilangan bulat hingga memasukkan bilangan negatif sebagai tanda akhir input. Data yang dimasukkan akan disimpan dalam array.

Program menggunakan fungsi `insertionSort` untuk mengurutkan array dengan menyisipkan elemen ke posisi yang sesuai. Setelah itu, fungsi `isConstantDifference` memeriksa apakah selisih antar elemen dalam array tetap. Jika konsisten, fungsi mengembalikan `true` beserta nilai selisihnya, jika tidak, mengembalikan `false`.

Program menampilkan array yang sudah diurutkan dan status selisih elemen. Jika selisih tetap, nilai selisih dicetak; jika tidak, diinformasikan bahwa selisihnya tidak konstan. Program ini cocok untuk menganalisis pola data dengan interval tetap.

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.

Format **Masukan** masih persis sama seperti sebelumnya.

Keluaran terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap, di masing-masing daerah.

No	Masukan	Keluaran
1	3 5 2 1 7 9 13 6 189 15 27 39 75 133 3 4 9 1	1 13 12 8 2 15 27 39 75 133 189 8 4 2

Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

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)

        // Input nomor rumah
        rumah := make([]int, jumlahRumah)
        fmt.Printf("Masukkan %d nomor rumah: ",
jumlahRumah)
        for j := 0; j < jumlahRumah; j++ {
            fmt.Scan(&rumah[j])
        }

        // Pemisahan nomor ganjil dan genap
        ganjil, genap := []int{}, []int{}
        for _, n := range rumah {
            if n%2 == 0 {
                genap = append(genap, n)
            }
        }
    }
}
```



```

        } else {
            ganjil = append(ganjil, n)
        }
    }

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

    // Output hasil
    fmt.Printf("\nHasil pengurutan daerah %d:\n", i)
    fmt.Println(append(ganjil, genap...)) // Gabung
    dan cetak ganjil diikuti genap
}
}

```

Screenshoot Output

```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu12\unguided1.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:
Hasil pengurutan daerah 3:
[1 9 4]
PS C:\Alpro sem 2>

```

Deskripsi Program :

Program ini mengimplementasikan pemisahan dan pengurutan nomor rumah berdasarkan sifat bilangan (ganjil atau genap) dalam bahasa Go.

Pengguna memasukkan jumlah daerah, diikuti dengan jumlah dan daftar nomor rumah untuk setiap daerah.

Fungsi proses NomorRumah memisahkan nomor rumah ganjil dan genap, lalu mengurutkan nomor ganjil secara menaik dan nomor genap secara menurun. Hasil pengurutan disimpan dalam struct Daerah, yang memiliki dua atribut: daftar nomor ganjil dan genap.

Program kemudian menampilkan nomor rumah ganjil terlebih dahulu, diikuti nomor rumah genap, sesuai dengan urutannya. Program ini efektif untuk mengorganisasi data berdasarkan sifat bilangan dan kriteria pengurutan tertentu.

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"
```

```

"sort"
)

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

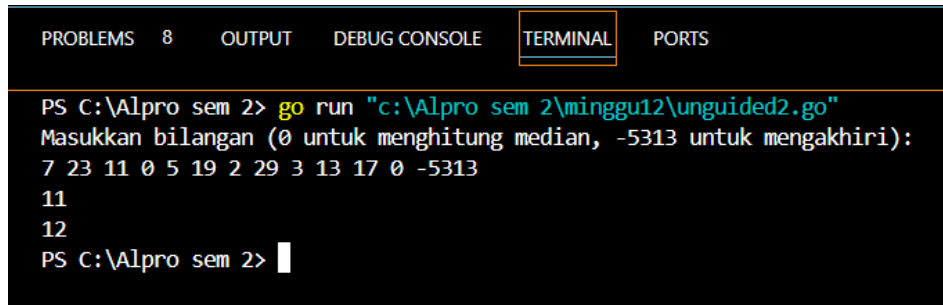
func main() {
    var data []int
    var input int

    // 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 dan hitung median
            if len(data) > 0 {
                sort.Ints(data) // Urutkan data menggunakan `sort.Ints`
                fmt.Printf("%.0f\n", hitungMedian(data))
            }
        } else if input == -5313 {
            break
        } else {
            // Tambahkan input ke slice data
            data = append(data, input)
        }
    }
}

```

Screenshoot Output

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', '8', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is highlighted with a yellow border), and 'PORTS'. The terminal shows the command 'go run "c:\Alpro sem 2\minggu12\unguided2.go"' being executed. The program prompts the user to 'Masukkan bilangan (0 untuk menghitung median, -5313 untuk mengakhiri):'. The user enters the numbers '7 23 11 0 5 19 2 29 3 13 17 0 -5313'. The program then outputs '11' and '12' on separate lines. The prompt 'PS C:\Alpro sem 2>' is visible at the bottom with a cursor.

```
PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu12\unguided2.go"
Masukkan bilangan (0 untuk menghitung median, -5313 untuk mengakhiri):
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
PS C:\Alpro sem 2> █
```

Deskripsi Program :

Program ini dalam bahasa Go memisahkan dan mengurutkan nomor rumah berdasarkan sifat bilangan (ganjil atau genap) di beberapa daerah. Pengguna memasukkan jumlah daerah dan daftar nomor rumah untuk setiap daerah.

Fungsi prosesNomorRumah memisahkan nomor ganjil dan genap, lalu mengurutkan nomor ganjil secara menaik dan nomor genap secara menurun. Hasil pengurutan disimpan dalam struct Daerah dengan dua atribut: nomor ganjil dan genap.

Program menampilkan nomor ganjil terlebih dahulu dalam urutan menaik, diikuti nomor genap dalam urutan menurun. Program ini membantu mengorganisir data berdasarkan sifat bilangan dan kriteria pengurutan tertentu.

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"
    "sort"
)

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

// Fungsi untuk mendaftarkan buku baru
func DaftarkanBuku(pustaka *[]Buku) {
    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)
}

// Fungsi untuk menampilkan buku terfavorit
func CetakTerfavorit(pustaka []Buku) {
    if len(pustaka) == 0 {
        fmt.Println("Tidak ada buku dalam perpustakaan")
        return
    }
}
```

```

    }

    terfavorit := pustaka[0]
    for _, buku := range pustaka {
        if buku.rating > terfavorit.rating {
            terfavorit = buku
        }
    }

    fmt.Println("\nBuku Terfavorit:")
    fmt.Printf("Judul: %s\nPenulis: %s\nPenerbit: %s\nTahun: %d\nRating: %d\n",
        terfavorit.judul, terfavorit.penulis,
        terfavorit.penerbit,
        terfavorit.tahun, terfavorit.rating)
}

// Fungsi untuk mengurutkan buku berdasarkan rating
func UrutBuku(pustaka []*Buku) {
    sort.Slice(*pustaka, func(i, j int) bool {
        return (*pustaka)[i].rating > (*pustaka)[j].rating
    })
}

// Fungsi untuk mencetak 5 buku dengan rating tertinggi
func Cetak5Terbaru(pustaka []Buku) {
    if len(pustaka) == 0 {
        fmt.Println("Tidak ada buku dalam perpustakaan")
        return
    }

    fmt.Println("\n5 Buku dengan Rating Tertinggi:")
    for i, buku := range pustaka {
        if i == 5 {
            break
        }
        fmt.Printf("%d. %s (Rating: %d)\n", i+1, buku.judul,
            buku.rating)
    }
}

// Fungsi untuk mencari buku berdasarkan rating
func CariBuku(pustaka []Buku, targetRating int) {
    idx := sort.Search(len(pustaka), func(i int) bool {
        return pustaka[i].rating <= targetRating
    })
}

```

```

    })

    if idx < len(pustaka) && pustaka[idx].rating ==
targetRating {
        buku := pustaka[idx]
        fmt.Printf("\nBuku dengan rating %d ditemukan:\n",
targetRating)
        fmt.Printf("Judul: %s\nPenulis: %s\nPenerbit:
%s\nTahun: %d\nEksemplar: %d\n",
            buku.judul, buku.penulis, buku.penerbit,
buku.tahun, buku.eksemplar)
    } else {
        fmt.Printf("\nTidak ada buku dengan rating %d\n",
targetRating)
    }
}

func main() {
    var pustaka []Buku
    var pilihan 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)
        case 2:
            CetakTerfavorit(pustaka)
        case 3:
            UrutBuku(&pustaka)
            fmt.Println("Buku telah diurutkan berdasarkan
rating")
        case 4:
            Cetak5Terbaru(pustaka)
        case 5:
            if len(pustaka) == 0 {

```



```

        fmt.Println("Tidak ada buku dalam
perpustakaan")
        continue
    }
    fmt.Print("Masukkan rating yang dicari: ")
    var targetRating int
    fmt.Scan(&targetRating)
    UrutBuku(&pustaka) // Pastikan buku terurut
sebelum pencarian
    CariBuku(pustaka, targetRating)
case 6:
    fmt.Println("Terima kasih telah menggunakan
program perpustakaan")
    return
default:
    fmt.Println("Pilihan tidak valid")
}
}
}

```

Screenshoot Output :

```

PS C:\Alpro sem 2> go run "c:\Alpro sem 2\minggu12\unguided3.go"

=== Menu Perpustakaan ===
1. Daftarkan Buku
2. Tampilkan Buku Terfavorit
3. Urutkan Buku berdasarkan Rating
4. Tampilkan 5 Buku Rating Tertinggi
5. Cari Buku berdasarkan Rating
6. Keluar
Pilih menu (1-6): 1
Masukkan ID: 123
Masukkan Judul: Aku
Masukkan Penulis: penulis1
Masukkan Penerbit: penerbit1
Masukkan Jumlah Eksemplar: 5
Masukkan Tahun: 2016
Masukkan Rating: 2

=== Menu Perpustakaan ===
1. Daftarkan Buku
2. Tampilkan Buku Terfavorit
3. Urutkan Buku berdasarkan Rating
4. Tampilkan 5 Buku Rating Tertinggi
5. Cari Buku berdasarkan Rating
6. Keluar
Pilih menu (1-6): 1
Masukkan ID: 234
Masukkan Judul: Kamu
Masukkan Penulis: penulis2
Masukkan Penerbit: penerbit2
Masukkan Jumlah Eksemplar: 5
Masukkan Tahun: 2017
Masukkan Rating: 1

```

=== Menu Perpustakaan ===

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

Pilih menu (1-6): 2

Buku Terfavorit:

Judul: Aku

Penulis: penulis1

Penerbit: penerbit1

Tahun: 2016

Rating: 2

=== Menu Perpustakaan ===

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

Pilih menu (1-6): 3

Buku telah diurutkan berdasarkan rating

=== Menu Perpustakaan ===

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

Pilih menu (1-6): 4

5 Buku dengan Rating Tertinggi:

1. Aku (Rating: 2)
2. Kamu (Rating: 1)

=== Menu Perpustakaan ===

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

Pilih menu (1-6): 5

Masukkan rating yang dicari: 1

Buku dengan rating 1 ditemukan:

Judul: Kamu

Penulis: penulis2

Penerbit: penerbit2

Tahun: 2017

Eksemplar: 5

Deskripsi Program :

Program di atas merupakan sebuah aplikasi perpustakaan sederhana yang memungkinkan pengguna untuk mengelola data buku, seperti mendaftarkan buku, mengurutkan buku berdasarkan rating, menampilkan buku terfavorit, dan mencari buku berdasarkan rating. Program ini menggunakan beberapa fungsi untuk memanipulasi data buku yang disimpan dalam sebuah struktur `DaftarBuku` yang merupakan slice dari struktur `Buku`. Setiap buku memiliki atribut seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun penerbitan, dan rating.

Pengguna dapat memilih berbagai opsi melalui menu yang disediakan, seperti mendaftarkan buku baru, menampilkan buku dengan rating tertinggi, atau mencari buku berdasarkan rating tertentu menggunakan pencarian biner. Buku juga dapat diurutkan berdasarkan rating menggunakan algoritma Insertion Sort, dan program akan menampilkan 5 buku dengan rating tertinggi jika diminta. Fungsi `CetakTerfavorit` digunakan untuk menampilkan buku dengan rating tertinggi, sementara fungsi `Cek5Terbaru` menampilkan 5 buku terbaik dari yang telah diurutkan.

Program ini berjalan dalam sebuah loop yang akan terus meminta input pengguna sampai pengguna memilih opsi untuk keluar. Setiap buku yang didaftarkan akan disimpan dalam sebuah slice, dan berbagai operasi pengurutan atau pencarian dilakukan secara dinamis. Program ini sangat berguna untuk manajemen koleksi buku di perpustakaan kecil dan memberikan cara yang mudah untuk mengorganisir buku berdasarkan rating.

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