LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII

SORTING

Disusun Oleh :

Ghilbran Alfaries Pryma

\2311102267 S1IF-11-05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Dasar Teori

Pengertian Pengurutan Data

Pengurutan data adalah proses mengorganisasi elemen-elemen dalam suatu struktur data berdasarkan urutan tertentu, seperti urutan menaik (ascending) atau menurun (descending). Dalam ilmu komputer, pengurutan merupakan operasi dasar yang sering digunakan untuk meningkatkan efisiensi pencarian, penyajian, atau manipulasi data.

Algoritma Pengurutan yang Umum Digunakan

Beberapa algoritma pengurutan yang populer meliputi:

Bubble Sort: Mengurutkan elemen dengan membandingkan pasangan elemen bertetangga dan menukarnya jika tidak sesuai urutan.

Selection Sort: Menemukan elemen terkecil atau terbesar dan menempatkannya pada posisi yang tepat.

Insertion Sort: Menyisipkan elemen ke posisi yang benar dalam bagian yang telah diurutkan.

Merge Sort: Membagi data menjadi sub-kumpulan, mengurutkan masing-masing, lalu menggabungkannya kembali.

Quick Sort: Memilih pivot dan membagi data menjadi dua bagian berdasarkan pivot.

Implementasi Pengurutan di Golang

Beberapa fungsi yang sering digunakan adalah:

sort.Ints(slice []int): Mengurutkan slice integer dalam urutan menaik.

sort.Strings(slice []string): Mengurutkan slice string dalam urutan menaik.

sort.Float64s(slice []float64): Mengurutkan slice float dalam urutan menaik.

sort.Sort(interface{}): Untuk pengurutan kustom, pengguna dapat mengimplementasikan interface sort.Interface.

Keuntungan Pengurutan

- Mempermudah pencarian data.
- Memungkinkan penggunaan algoritma pencarian yang lebih cepat seperti binary search.
- Membantu dalam analisis data untuk visualisasi atau laporan.

II. GUIDED

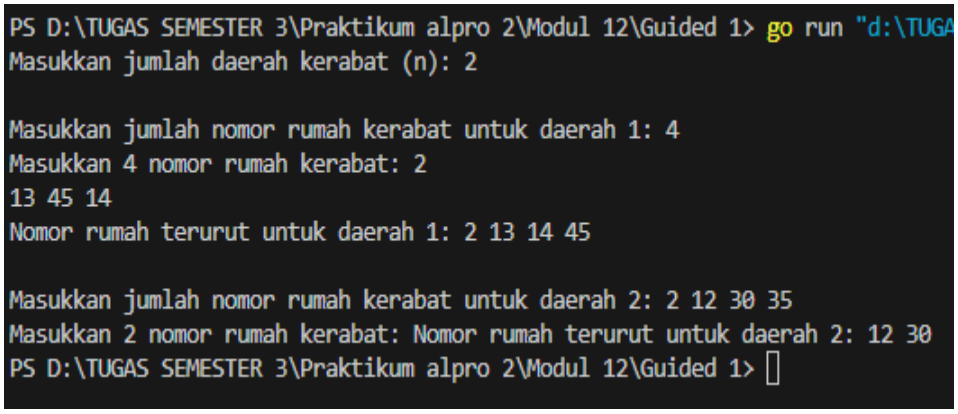
Guided I

Source code

```go
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("\nMasukkan jumlah nomor rumah 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])
        }

        // Sort the house numbers
        selectionsort(arr, m)

        // Print sorted house numbers for the current daerah
        fmt.Printf("Nomor rumah terurut untuk daerah %d: ",
daerah)
        for _, num := range arr {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}
```

Screenshoot Output



Deskripsi Program

Program ini ditulis dalam bahasa pemrograman Go dan berfungsi untuk mengurutkan nomor rumah kerabat berdasarkan daerah yang dimasukkan oleh pengguna. Berikut adalah penjelasan rinci tentang setiap bagian dari program ini.

Fungsi `selectionsort`

Fungsi ini mengimplementasikan algoritma selection sort untuk mengurutkan elemen dalam array:

- Fungsi menerima dua parameter: `arr` (array yang akan diurutkan) dan `n` (jumlah elemen dalam array).
- Di dalam fungsi, terdapat dua loop bersarang. Loop luar berjalan dari indeks 0 hingga n-1, dan loop dalam mencari indeks elemen terkecil di antara elemen yang belum diurutkan.
- Setelah menemukan indeks elemen terkecil (`idxmin`), fungsi menukar elemen tersebut dengan elemen pada posisi saat ini (i).

Fungsi `main`

Fungsi utama program melakukan hal berikut:

- Meminta pengguna untuk memasukkan jumlah daerah kerabat (n).
- Menggunakan loop untuk iterasi melalui setiap daerah. Untuk setiap daerah, program meminta pengguna untuk memasukkan jumlah nomor rumah kerabat (m).
- Program kemudian membuat array dengan panjang m dan meminta pengguna untuk memasukkan nomor rumah ke dalam array.
- Setelah semua nomor rumah dimasukkan, fungsi `selectionsort` dipanggil untuk mengurutkan nomor-nomor tersebut.
- Setelah pengurutan, program mencetak nomor rumah yang telah terurut untuk daerah yang bersangkutan.

Alur Kerja Program

- Pengguna diminta untuk memasukkan jumlah daerah kerabat.
- Untuk setiap daerah, pengguna memasukkan jumlah nomor rumah dan kemudian memasukkan nomor-nomor tersebut.
- Program mengurutkan nomor rumah menggunakan algoritma selection sort dan mencetak hasilnya.

Guided II

Source code

```
package main

import (
    "fmt"
    "math"
)

// Fungsi insertion sort untuk mengurutkan array
func insertionSort(arr []int) {
    n := len(arr)
    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 data berjarak tetap
func isDataConsistentlySpaced(arr []int) (bool, int) {
    if len(arr) < 2 {
        return true, 0 // Array dengan kurang dari 2 elemen
dianggap berjarak tetap
    }
    // Hitung selisih awal
    diff := int(math.Abs(float64(arr[1] - arr[0])))
    for i := 1; i < len(arr)-1; i++ {
        currentDiff := int(math.Abs(float64(arr[i+1] -
            arr[i])))
        if currentDiff != diff {
            return false, 0 // Jika ada selisih yang berbeda,
tidak berjarak tetap
        }
    }
    return true, diff
}
func main() {
    var data []int
```

```
    var input int
    fmt.Println("Masukkan data (akhiri dengan bilangan negatif):")
    for {
        fmt.Scan(&input)
        if input < 0 {
            break
        }
        data = append(data, input)
    }
    // Urutkan data menggunakan insertion sort
    insertionSort(data)
    // Periksa apakah data berjarak tetap
    isConsistent, diff := isDataConsistentlySpaced(data)
    // Cetak hasil
    fmt.Println("Hasil pengurutan:", data)
    if isConsistent {
        fmt.Printf("Data berjarak %d\n", diff)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}
```

Screenshoot Output

```
PS D:\TUGAS SEMESTER 3\Praktikum alpro 2\Modul 12\Guided 2> go run
Masukkan data (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6
PS D:\TUGAS SEMESTER 3\Praktikum alpro 2\Modul 12\Guided 2>
```

Deskripsi Program

Program ini memiliki dua fungsi utama: untuk mengurutkan array menggunakan algoritma insertion sort dan untuk memeriksa apakah elemen-elemen dalam array memiliki jarak yang tetap.

Fungsi pertama, **insertionSort** menerima sebuah array sebagai parameter. Fungsi ini bekerja dengan cara menelusuri elemen dari indeks 1 hingga akhir array. Untuk setiap elemen yang disebut `key`, fungsi membandingkannya dengan elemen sebelumnya dan memindahkan elemen yang lebih besar ke

kanan. Setelah menemukan posisi yang tepat untuk `key`, elemen tersebut dimasukkan ke dalam posisi yang sesuai.

Fungsi kedua, **isDataConsistentlySpaced** memeriksa apakah jarak antara elemen-elemen dalam array adalah tetap. Jika panjang array kurang dari dua, fungsi menganggap bahwa jarak antar elemen adalah tetap. Fungsi ini menghitung selisih antara dua elemen pertama dan kemudian membandingkan selisih ini dengan selisih antara elemen berturut-turut lainnya. Jika semua selisih sama, fungsi mengembalikan nilai true dan nilai selisih; jika ada perbedaan, fungsi mengembalikan false.

Fungsi utama program, **main**, meminta pengguna untuk memasukkan angka satu per satu hingga pengguna memasukkan angka negatif sebagai sinyal untuk berhenti. Angka-angka yang dimasukkan kemudian diurutkan menggunakan fungsi insertion sort. Setelah itu, program memeriksa apakah jarak antar elemen adalah tetap menggunakan fungsi isDataConsistentlySpaced. Hasil pengurutan dan informasi mengenai konsistensi jarak antar elemen kemudian dicetak ke layar.

Secara keseluruhan, program ini memungkinkan pengguna untuk memasukkan serangkaian angka, mengurutkannya, dan memeriksa apakah angka-angka tersebut memiliki jarak yang konsisten.

III. UNGUIDED

Unguided I

1.Soal

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.

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

Keluaran terdiri dari n baris, yaitu rangkalan 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

Keterangan: Terdapat 3 daerah dalam contoh masukan. Baris kedua berisi campuran bilangan ganjil dan genap. Baris berikutnya hanya berisi bilangan ganjil, dan baris terakhir hanya berisi bilangan genap.

Petunjuk:

- Waktu pembacaan data, bilangan ganjil dan genap dipisahkan ke dalam dua array yang berbeda, untuk kemudian masing-masing diurutkan tersendiri.

- Atau, tetap disimpan dalam satu array, diurutkan secara keseluruhan. Tetapi pada waktu pencetakan, mulal dengan mencetak semua nilal ganjil lebih dulu, kemudian setelah selesai cetaklah semua nilai genapnya.

Source code

```go
package main

import (
    "bufio"
    "fmt"
    "os"
    "sort"
    "strconv"
    "strings"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin)

    // Meminta jumlah daerah
    fmt.Print("Masukkan jumlah daerah: ")
    scanner.Scan()
    jumlahDaerah, _ := strconv.Atoi(scanner.Text())

    for i := 1; i <= jumlahDaerah; i++ {
        // Meminta jumlah rumah kerabat
        fmt.Printf("Masukkan jumlah rumah kerabat daerah %d: ", i)
        scanner.Scan()
        jumlahRumah, _ := strconv.Atoi(scanner.Text())

        // Meminta nomor rumah
        fmt.Printf("Masukkan nomor rumah daerah %d: ", i)
        scanner.Scan()
        baris := scanner.Text()

        // Memisahkan angka-angka dari input
        angkaStr := strings.Fields(baris)
        if len(angkaStr) != jumlahRumah {
            fmt.Println("Jumlah rumah harus sesuai dengan input!")
            continue
        }

        // Memisahkan angka menjadi ganjil dan genap
        var rumahGanjil, rumahGenap []int
        for _, angka := range angkaStr {
            nomor, _ := strconv.Atoi(angka)
            if nomor%2 == 0 {
```

```
                rumahGenap = append(rumahGenap, nomor)
            } else {
                rumahGanjil = append(rumahGanjil, nomor)
            }
        }

        // Mengurutkan ganjil secara menaik
        sort.Ints(rumahGanjil)

        // Mengurutkan genap secara menurun
        sort.Sort(sort.Reverse(sort.IntSlice(rumahGenap)))

        // Mencetak output sesuai format
        fmt.Print("Urutan rumah: ")
        for _, nomor := range rumahGanjil {
            fmt.Printf("%d ", nomor)
        }
        for _, nomor := range rumahGenap {
            fmt.Printf("%d ", nomor)
        }
        fmt.Println()
    }
}
```

Screenshoot Output

```
go run "d:\TUGAS SEMESTER 3\Praktikum alpro 2\Modul 12\Unguided 1\main.go"
Masukkan jumlah daerah: 3
Masukkan jumlah rumah kerabat daerah 1: 5
Masukkan nomor rumah daerah 1: 2 1 7 9 13
Urutan rumah: 1 7 9 13 2
Masukkan jumlah rumah kerabat daerah 2: 6
Masukkan nomor rumah daerah 2: 189 15 27 39 75 133
Urutan rumah: 15 27 39 75 133 189
Masukkan jumlah rumah kerabat daerah 3: 3
Masukkan nomor rumah daerah 3: 4 9 1
Urutan rumah: 1 9 4
PS D:\TUGAS SEMESTER 3\Praktikum alpro 2\Modul 12\Unguided 1>
```

Deskripsi Program

Program ini bertujuan untuk mengelola dan mengurutkan nomor rumah kerabat berdasarkan daerah. Program ini memisahkan nomor rumah menjadi dua kategori: ganjil dan genap, lalu mengurutkannya dengan cara yang berbeda. Berikut adalah penjelasan rinci tentang setiap bagian dari program ini.

Alur Kerja Program

- Input Jumlah Daerah:
- Program dimulai dengan meminta pengguna untuk memasukkan jumlah daerah. Input ini dibaca menggunakan bufio.Scanner, yang memungkinkan pembacaan input dari konsol.
- Iterasi Melalui Setiap Daerah:
- Program menggunakan loop untuk iterasi melalui setiap daerah berdasarkan jumlah yang dimasukkan oleh pengguna.
- Input Jumlah Rumah Kerabat:
- Untuk setiap daerah, program meminta pengguna untuk memasukkan jumlah rumah kerabat yang akan diinput.
- Input Nomor Rumah:
- Pengguna diminta untuk memasukkan nomor rumah dalam satu baris. Nomor-nomor ini kemudian dipisahkan menjadi elemen-elemen individual menggunakan fungsi strings.Fields, yang memecah string berdasarkan spasi.
- Validasi Jumlah Input:
- Program memeriksa apakah jumlah nomor rumah yang dimasukkan sesuai dengan jumlah yang telah ditentukan sebelumnya. Jika tidak sesuai, program akan mencetak pesan kesalahan dan melanjutkan ke iterasi berikutnya.
- Pemrosesan Nomor Rumah:
- Program memisahkan nomor rumah menjadi dua kategori: ganjil dan genap. Ini dilakukan dengan menggunakan loop yang memeriksa setiap nomor.
- Nomor genap disimpan dalam satu slice, sedangkan nomor ganjil disimpan dalam slice lain.
- Pengurutan:
- Nomor rumah ganjil diurutkan secara menaik menggunakan fungsi sort.Ints.

- Nomor rumah genap diurutkan secara menurun dengan menggunakan sort.Sort dan sort.Reverse.
- Output Hasil:
  Setelah pengurutan, program mencetak urutan nomor rumah dengan format yang diinginkan, di mana nomor ganjil dicetak terlebih dahulu diikuti oleh nomor genap.

Unguided II

2.Soal

Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagal 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 nilal tengah dibulatkan ke bawah."

Buatlah program **median** yang mencetak nilal 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 O 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

Keterangan:

Sampai bilangan O yang pertama, data terbaca adalah 7 23 11, setelah tersusun: 7 11 23, maka median saat itu adalah 11.

Sampai bilangan O yang kedua, data adalah 7 23 11 5 19 2 29 3 13 17, setelah tersusun diperoleh: 2 3 5 7 11 13 17 19 23 29. Karena ada 10 data, genap, maka median adalah (11+13)/2=12.

Petunjuk:

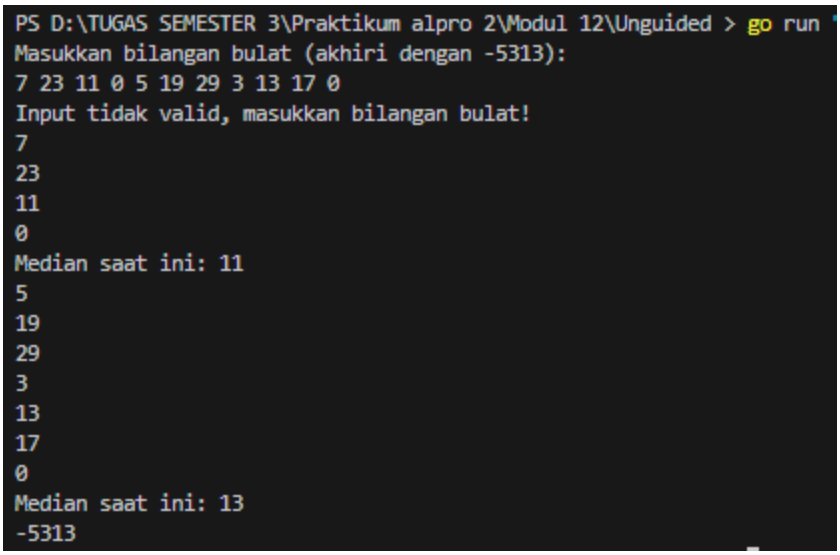
Untuk setiap data bukan 0 (dan bukan marker -5313541) simpan ke dalam array. Dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metode insertion sort dan ambil mediannya.

Source code

```go
package main

import (
    "bufio"
    "fmt"
    "os"
    "sort"
    "strconv"
)

func main() {
    var data []int // Array untuk menyimpan data yang dibaca

    scanner := bufio.NewScanner(os.Stdin)
    fmt.Println("Masukkan bilangan bulat (akhiri dengan -5313):")

    for scanner.Scan() {
        input := scanner.Text()
        // Konversi input menjadi bilangan bulat
        num, err := strconv.Atoi(input)
        if err != nil {
            fmt.Println("Input tidak valid, masukkan bilangan
bulat!")
```

```
            continue
        }

        if num == -5313 { // Tanda akhir input
            break
        }

        if num == 0 {
            if len(data) == 0 {
                fmt.Println("Tidak ada data untuk dihitung
median.")
                continue
            }

            // Salin dan urutkan data menggunakan metode insertion
sort
            sortedData := make([]int, len(data))
            copy(sortedData, data)
            sort.Slice(sortedData, func(i, j int) bool { return
sortedData[i] < sortedData[j] })

            // Hitung median
            median := 0
            length := len(sortedData)
            if length%2 == 0 {
                median = (sortedData[length/2-1] +
sortedData[length/2]) / 2
            } else {
                median = sortedData[length/2]
            }
            fmt.Println("Median saat ini:", median)
        } else {
            // Tambahkan bilangan ke array data
            data = append(data, num)
        }
    }

    if err := scanner.Err(); err != nil {
        fmt.Println("Error saat membaca input:", err)
    }
}
```

Screenshoot Output



Deskripsi Program

Program ini bertujuan untuk menghitung median dari sekumpulan bilangan bulat positif yang dimasukkan oleh pengguna. Data disimpan dalam array `data`, dan program membaca input secara bertahap menggunakan `bufio.Scanner`. Ketika angka **0** dimasukkan, program mengurutkan array menggunakan algoritma sorting (fungsi `sort.Slice`) dan menghitung median berdasarkan jumlah elemen. Jika jumlah elemen ganjil, median adalah elemen tengah; jika genap, median dihitung sebagai rata-rata dua elemen tengah, dibulatkan ke bawah. Median ini kemudian dicetak ke layar. Angka positif yang dimasukkan ditambahkan ke array `data`, sementara angka `-5313` digunakan sebagai tanda akhir input. Sebagai contoh, jika input berupa `7 23 11 0`, maka data `[7, 23, 11]` akan diurutkan menjadi `[7, 11, 23]` dan median adalah `11`. Jika dilanjutkan dengan input `5 19 2 29 3 13 17 0`, data lengkapnya akan diurutkan menjadi `[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]`, dan median dihitung sebagai `(11 + 13) / 2 = 12`. Program dapat menangani hingga 1.000.000 data dan berhenti saat input `-5313` diberikan. Program ini efisien dan memastikan median dihitung hanya saat dibutuhkan.

Unguided III

3.Soal

Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu

perpustakaan. Misalnya terdefinisi struct dan array seperti berikut ini

```
const nMax integer 7919
id, judul, penulis, penerbit: string eksemplar, tahun, rating integer >
type Buku = <
type DaftarBuku array [1..nMax) of Buku
Pustaka Daftar Buku
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 bafis. 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 L.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 Cetak Terfavorit (in pustaka DaftarBuku, inn: 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 berisin data buku
F.S. Array pustaka terurut menurun/mengecil terhadap rating.
Catatan: Gunakan metoda Insertion sort)

procedure Cetak5Terbaru 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, 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.

Source code

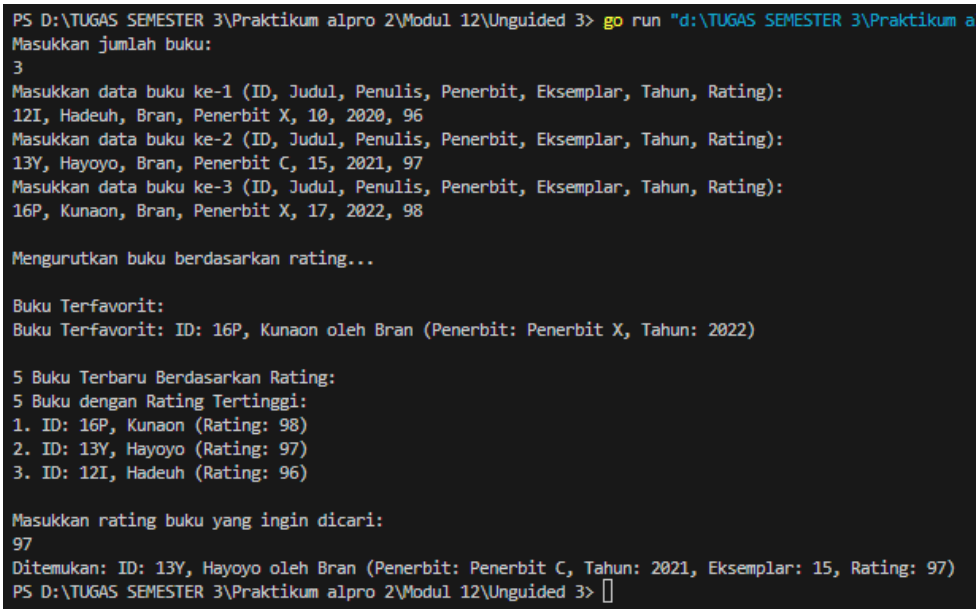
```go
package main

import (
    "bufio"
    "fmt"
    "os"
    "sort"
    "strconv"
    "strings"
)

type Buku struct {
    ID        string
    Judul     string
    Penulis   string
    Penerbit  string
    Eksemplar int
    Tahun     int
    Rating    int
}
```

```
type DaftarBuku []Buku

// Prosedur untuk mendaftarkan buku
func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    scanner := bufio.NewScanner(os.Stdin)
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan data buku ke-%d (ID, Judul, Penulis,
Penerbit, Eksemplar, Tahun, Rating):\n", i+1)
        scanner.Scan()
        input := strings.Split(scanner.Text(), ",")
        if len(input) != 7 {
            fmt.Println("Data buku tidak lengkap, masukkan
ulang!")
            i--
            continue
        }
        eksemplar, _ := strconv.Atoi(strings.TrimSpace(input[4]))
        tahun, _ := strconv.Atoi(strings.TrimSpace(input[5]))
        rating, _ := strconv.Atoi(strings.TrimSpace(input[6]))
        *pustaka = append(*pustaka, Buku{
            ID:        strings.TrimSpace(input[0]),
            Judul:     strings.TrimSpace(input[1]),
            Penulis:   strings.TrimSpace(input[2]),
            Penerbit:  strings.TrimSpace(input[3]),
            Eksemplar: eksemplar,
            Tahun:     tahun,
            Rating:    rating,
        })
    }
}

// Prosedur untuk mencetak buku terfavorit
func CetakTerfavorit(pustaka DaftarBuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada data buku.")
        return
    }
    maxRating := pustaka[0].Rating
    terfavorit := pustaka[0]
    for _, buku := range pustaka {
        if buku.Rating > maxRating {
            maxRating = buku.Rating
            terfavorit = buku
        }
```

```
    }
    fmt.Printf("Buku Terfavorit: ID: %s, %s oleh %s (Penerbit: %s,
Tahun: %d)\n", terfavorit.ID, terfavorit.Judul,
terfavorit.Penulis, terfavorit.Penerbit, terfavorit.Tahun)
}

// Prosedur untuk mengurutkan buku berdasarkan rating (descending)
func UrutBuku(pustaka *DaftarBuku, n int) {
    if n < 2 {
        return
    }
    sort.Slice(*pustaka, func(i, j int) bool {
        return (*pustaka)[i].Rating > (*pustaka)[j].Rating
    })
}

// Prosedur untuk mencetak 5 buku terbaru
func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    fmt.Println("5 Buku dengan Rating Tertinggi:")
    for i := 0; i < n && i < 5; i++ {
        fmt.Printf("%d. ID: %s, %s (Rating: %d)\n", i+1,
pustaka[i].ID, pustaka[i].Judul, pustaka[i].Rating)
    }
}

// Prosedur untuk mencari buku berdasarkan rating menggunakan
pencarian biner
func CariBuku(pustaka DaftarBuku, n, targetRating int) {
    low, high := 0, n-1
    for low <= high {
        mid := (low + high) / 2
        if pustaka[mid].Rating == targetRating {
            fmt.Printf("Ditemukan: ID: %s, %s oleh %s (Penerbit:
%s, Tahun: %d, Eksemplar: %d, Rating: %d)\n",
                pustaka[mid].ID, pustaka[mid].Judul,
pustaka[mid].Penulis, pustaka[mid].Penerbit, pustaka[mid].Tahun,
pustaka[mid].Eksemplar, pustaka[mid].Rating)
            return
        } else if pustaka[mid].Rating > targetRating {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
    fmt.Println("Tidak ada buku dengan rating seperti itu.")
```

```
}

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

    fmt.Println("Masukkan jumlah buku:")
    fmt.Scanln(&n)
    DaftarkanBuku(&pustaka, n)

    // Mengurutkan buku berdasarkan rating
    fmt.Println("\nMengurutkan buku berdasarkan rating...")
    UrutBuku(&pustaka, n)

    // Menampilkan buku terfavorit
    fmt.Println("\nBuku Terfavorit:")
    CetakTerfavorit(pustaka, n)

    // Menampilkan 5 buku terbaru berdasarkan rating
    fmt.Println("\n5 Buku Terbaru Berdasarkan Rating:")
    Cetak5Terbaru(pustaka, n)

    // Mencari buku berdasarkan rating
    fmt.Println("\nMasukkan rating buku yang ingin dicari:")
    fmt.Scanln(&targetRating)
    CariBuku(pustaka, n, targetRating)
}
```

Screenshoot Output



Deskripsi Program

Struktur Program

Tipe Data Buku:

- Program mendefinisikan tipe data Buku yang berisi informasi tentang buku, termasuk ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating.

Tipe Data DaftarBuku:

- DaftarBuku didefinisikan sebagai slice dari tipe Buku, yang akan digunakan untuk menyimpan koleksi buku.

Fungsi DaftarkanBuku:

- Fungsi ini bertanggung jawab untuk mendaftarkan buku ke dalam pustaka. Pengguna diminta untuk memasukkan data buku dalam format tertentu (ID, judul, penulis, penerbit, eksemplar, tahun, rating) yang dipisahkan oleh koma.
- Fungsi ini memvalidasi input dan memastikan bahwa semua data yang diperlukan telah dimasukkan.
- Jika input tidak lengkap, pengguna diminta untuk memasukkan ulang data tersebut.
- Data yang valid kemudian ditambahkan ke dalam slice pustaka.

Fungsi CetakTerfavorit:

- Fungsi ini mencetak buku dengan rating tertinggi dari daftar buku. Jika tidak ada data buku, fungsi akan memberi tahu pengguna bahwa tidak ada data yang tersedia.

Fungsi UrutBuku:

- Fungsi ini mengurutkan daftar buku berdasarkan rating secara menurun menggunakan fungsi sort.Slice. Jika jumlah buku kurang dari dua, fungsi tidak melakukan pengurutan.

Fungsi Cetak5Terbaru:

- Fungsi ini mencetak lima buku teratas berdasarkan rating tertinggi. Jika jumlah buku kurang dari lima, fungsi hanya mencetak sebanyak mungkin yang tersedia.

Fungsi CariBuku:

- Fungsi ini mencari buku berdasarkan rating tertentu menggunakan metode pencarian biner. Jika buku dengan rating yang dicari ditemukan, informasi lengkap tentang buku tersebut akan dicetak; jika tidak ditemukan, program akan memberi tahu pengguna bahwa tidak ada buku dengan rating tersebut.

Fungsi main:

- Fungsi utama program mengatur alur kerja keseluruhan:
- Meminta pengguna untuk memasukkan jumlah buku.
- Memanggil fungsi DaftarkanBuku untuk mendaftarkan buku-buku tersebut.
- Mengurutkan daftar buku berdasarkan rating.
- Menampilkan informasi tentang buku terfavorit.
- Menampilkan lima buku terbaru berdasarkan rating.
- Meminta pengguna untuk memasukkan rating yang ingin dicari dan memanggil fungsi pencarian.

IV. DAFTAR PUSTAKA

1. A. A. Donovan and B. W. Kernighan, *The Go Programming Language*. Upper Saddle River, NJ, USA: Addison-Wesley Professional, 2015.

2. "math Package," Golang Documentation. [Online]. Available: https://pkg.go.dev/math. [Accessed: 24-Nov-2024].

3. "Effective Go," Golang Documentation. [Online]. Available: https://go.dev/doc/effective_go. [Accessed: 24-Nov-2024].

4. R. Sedgewick and K. Wayne, *Algorithms*, 4th ed. Upper Saddle River, NJ, USA: Addison-Wesley Professional, 2011.

5. "Go Programming Language Tutorial," Tutorialspoint. [Online]. Available: https://www.tutorialspoint.com/go/index.htm. [Accessed: 24-Nov-2024].

6. "Find Maximum and Minimum in an Array," GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/. [Accessed: 24-Nov-2024].