

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 12 & 13
PENGURUTAN DATA**



Oleh:

NAUFAL THORIQ MUZHAFAR

2311102078

IF-11-02

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

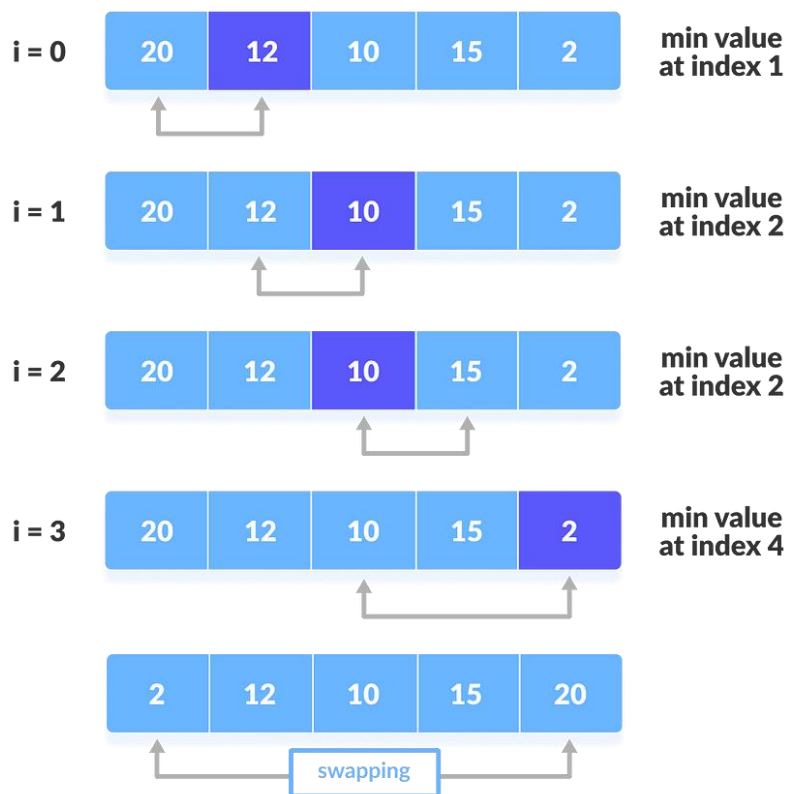
2024

I. DASAR TEORI

Selection Sort

Selection Sort: Memilih elemen terkecil dari bagian daftar yang belum diurutkan dan menukarnya dengan elemen yang belum diurutkan di paling kiri. Proses ini diulang hingga seluruh daftar diurutkan.

step = 0



```
func selection(array []int, size int) {  
    for i := 0; i < size; i++ {  
        min := i  
        for j := i + 1; j < size; j++ {  
            if array[min] > array[j] {  
                min = j  
            }  
        }  
        // swapping  
    }  
}
```

```

        }

    }

    temp := array[min]

    array[min] = array[i]

    array[i] = temp

}

fmt.Println("The selection sorted array is : ")

fmt.Println(array)

}

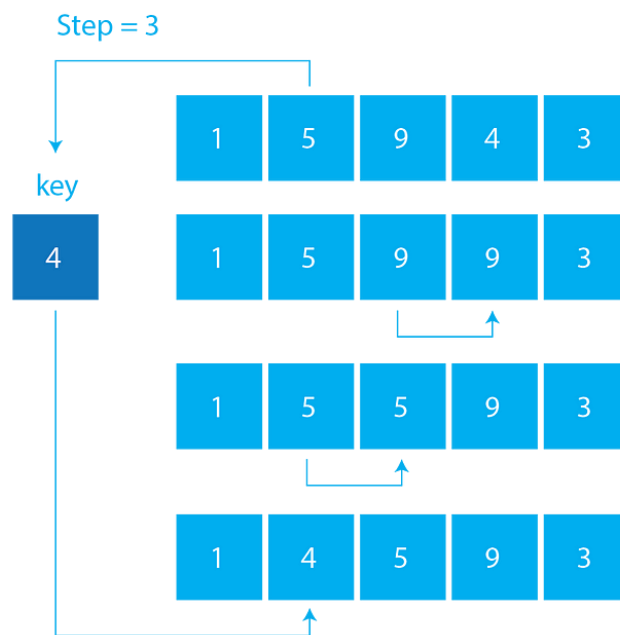
```

- Fungsi selection sort mengambil dua parameter: array, yang merupakan irisan bilangan bulat yang akan diurutkan, dan size, yang mewakili ukuran array.
- Perulangan for luar beriterasi dari 0 hingga size-1 untuk memilih elemen minimum saat ini.
- Di dalam perulangan luar, variabel min diinisialisasi dengan nilai i, yang menunjukkan indeks elemen minimum saat ini.
- Perulangan for dalam dimulai dari i+1 dan berlanjut hingga size-1. Perulangan ini membandingkan setiap elemen di bagian array yang belum diurutkan dengan elemen minimum saat ini (array[min]).
- Jika elemen pada indeks j ditemukan lebih kecil dari elemen minimum saat ini (array[min]), indeks min diperbarui menjadi j. Proses ini menemukan elemen terkecil di bagian array yang belum diurutkan.
- Setelah perulangan dalam selesai, elemen minimum (array[min]) ditukar dengan elemen pada indeks saat ini (array[i]). Ini menempatkan elemen terkecil pada posisi yang diurutkan dengan benar. Proses ini diulang untuk iterasi berikutnya dari loop luar, memilih elemen minimum berikutnya dan menempatkannya pada posisi yang sesuai.
- Setelah loop luar selesai, seluruh array diurutkan dalam urutan menaik.
- Akhirnya, array yang diurutkan dicetak menggunakan fungsi fmt.Println.

Selection Sort memiliki kompleksitas waktu $O(n^2)$ dalam kasus terburuk dan rata-rata, di mana n adalah jumlah elemen dalam daftar. Ia berkinerja lebih baik daripada Bubble Sort dalam hal jumlah pertukaran tetapi masih memiliki keterbatasan untuk daftar yang lebih besar.

Insertion Sort

Insertion Sort: Menyusun daftar akhir yang diurutkan satu per satu. Mengambil elemen dari bagian yang tidak diurutkan dan menyisipkannya ke posisi yang benar di bagian yang diurutkan, menggeser elemen yang lebih besar ke kanan.



```
func insertion(array []int, size int) {  
    for i, v := range array {  
        j := i - 1  
        if i >= 1 {  
            for j >= 0 && array[j] > v {  
                array[j+1] = array[j]  
                j--  
            }  
            array[j+1] = v  
        }  
    }  
    fmt.Println("The insertion sorted array is : ")  
}
```

```
        fmt.Println(array)
    }
```

- Fungsi insertion sort mengambil dua parameter: array, yang merupakan irisan bilangan bulat yang akan diurutkan, dan size, yang mewakili ukuran array.
- Perulangan for luar mengiterasi setiap elemen array menggunakan kata kunci range. Untuk setiap iterasi, ia menetapkan indeks saat ini ke i dan nilai elemen yang sesuai ke v.
- Di dalam perulangan, variabel j diinisialisasi dengan nilai i - 1. Variabel ini melacak posisi di bagian yang diurutkan tempat elemen saat ini v perlu disisipkan.
- Kondisi if i >= 1 memeriksa apakah elemen saat ini bukan elemen pertama array. Kondisi ini memastikan bahwa ada posisi yang valid untuk menyisipkan elemen.
- Di dalam perulangan for dalam, ia membandingkan elemen saat ini v dengan elemen di sebelah kirinya (array[j]). Ia terus menggeser elemen yang lebih besar ke kanan dengan satu posisi hingga menemukan posisi yang benar untuk menyisipkan elemen saat ini. Perulangan berlanjut hingga mencapai awal array (j >= 0) atau hingga menemukan elemen yang lebih kecil atau sama dengan elemen saat ini (array[j] <= v).
- Setelah menemukan posisi yang benar, elemen saat ini v dimasukkan ke dalam bagian array yang diurutkan pada indeks j+1.
- Perulangan luar melanjutkan proses ini untuk setiap elemen array, secara bertahap membangun bagian yang diurutkan.
- Akhirnya, array yang diurutkan dicetak menggunakan fungsi fmt.Println.

Insertion sort memiliki kompleksitas waktu $O(n^2)$, yang membuatnya efisien untuk array kecil dan array yang diurutkan sebagian. Namun, untuk array besar, algoritma pengurutan lain seperti merge sort atau quicksort umumnya lebih efisien.

II. GUIDED

GUIDED 1 (SELECTION SORT)

Source Code

```
package main
import "fmt"

// Fungsi untuk mengurutkan array menggunakan selection
sort
func selectionSort(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        maxIdx := i
        for j := i + 1; j < n; j++ {
            if arr[j] > arr[maxIdx] { // Cari elemen
terbesar
                                maxIdx = j
                            }
                        }
            arr[i], arr[maxIdx] = arr[maxIdx], arr[i] //
Tukar elemen
        }
    }
}

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

    if n <= 0 || n >= 1000 {
        fmt.Println("n harus lebih besar dari 0 dan
kurang dari 1000.")
        return
    }

    for i := 0; i < n; i++ {
        var m int
        fmt.Printf("Masukkan jumlah rumah kerabat
untuk daerah ke-%d: ", i+1)
        fmt.Scan(&m)

        if m <= 0 || m >= 1000000 {
            fmt.Println("m harus lebih besar dari 0
dan kurang dari 1000000.")
        }
    }
}
```

```

        return
    }

    // Masukkan nomor rumah
    houses := make([]int, m)
    fmt.Printf("Masukkan nomor rumah kerabat
untuk daerah ke-%d: ", i+1)
    for j := 0; j < m; j++ {
        fmt.Scan(&houses[j])
    }

    // Urutkan dengan selection sort
    selectionSort(houses)

    // Cetak hasil
    fmt.Printf("Hasil urutan rumah untuk daerah
ke-%d: ", i+1)
    for _, house := range houses {
        fmt.Printf("%d ", house)
    }
    fmt.Println()
}
}

```

Screenshoot Output

```

[lauraneval@arco-kun laprak_8]$ go run guided_1.go
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5
Masukkan nomor rumah kerabat untuk daerah ke-1: 21 33 42 10 32
Hasil urutan rumah untuk daerah ke-1: 42 33 32 21 10
Masukkan jumlah rumah kerabat untuk daerah ke-2: 5
Masukkan nomor rumah kerabat untuk daerah ke-2: 121 434 324 12 453
Hasil urutan rumah untuk daerah ke-2: 453 434 324 121 12
Masukkan jumlah rumah kerabat untuk daerah ke-3: 5
Masukkan nomor rumah kerabat untuk daerah ke-3: 1213 324 52 23 1
Hasil urutan rumah untuk daerah ke-3: 1213 324 52 23 1
[lauraneval@arco-kun laprak_8]$

```

Penjelasan Program

Program ini meminta pengguna memasukkan jumlah daerah n (harus antara 1 dan 999). Untuk setiap daerah, pengguna memasukkan jumlah rumah m (harus antara 1 dan 999.999) dan daftar nomor rumah sebanyak m . Program kemudian mengurutkan nomor rumah di setiap daerah dari yang terbesar ke yang terkecil menggunakan algoritma **Selection Sort**. Hasilnya outputnya akan ditampilkan setelah pengurutan. Jika input n atau m tidak valid, program langsung memberi pesan kesalahan dan berhenti.

GUDED 2 (INSERTION SORT)

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

```
[lauraneval@arco-kun laprak_8]$ go run guided_2.go
Masukkan data (akhiri dengan bilangan negatif):
4 10 6 12 2 8 20 14 18 16 -1
Hasil pengurutan: [2 4 6 8 10 12 14 16 18 20]
Data berjarak 2
[lauraneval@arco-kun laprak_8]$ _
```

Penjelasan Program

Program ini meminta pengguna untuk memasukkan serangkaian angka yang diakhiri dengan bilangan negatif. Setelah input selesai, program mengurutkan angka-angka tersebut secara menaik menggunakan algoritma Insertion Sort. Selanjutnya, program memeriksa apakah data yang telah diurutkan memiliki selisih tetap antar elemen (jarak konsisten). Jika data berjarak tetap, program mencetak selisih tersebut; jika tidak, program menyatakan bahwa jarak antar elemen tidak konsisten.

III. UNGUIDED

UNGUIDED 1 (SELECTION SORT)

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 rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap, di masing-masing daerah.

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

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, mulai dengan mencetak semua nilai ganjil lebih dulu, kemudian setelah selesai cetaklah semua nilai genapnya.

Source Code

```
package main

import "fmt"

func selectionSortAsc(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        minIdx := i
        for j := i + 1; j < n; j++ {
```

```

        if arr[j] < arr[minIdx] {
            minIdx = j
        }
    }
    arr[i], arr[minIdx] = arr[minIdx], arr[i]
}
}

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

    if n <= 0 || n >= 1000 {
        fmt.Println("Jumlah daerah harus di antara 1
dan 999.")
        return
    }

    masukan := make([][]int, n)

    for i := 0; i < n; i++ {
        var m int
        fmt.Scan(&m)

        if m <= 0 || m >= 1000000 {
            fmt.Println("Jumlah rumah harus di
antara 1 dan 999999.")

```

```

        return
    }

    masukan[i] = make([]int, m)
    for j := 0; j < m; j++ {
        fmt.Scan(&masukan[i][j])
    }
}

fmt.Println("\nOutput:")
for _, daerah := range masukan {
    var ganjil []int
    var genap []int

    for _, num := range daerah {
        if num%2 == 0 {
            genap = append(genap, num)
        } else {
            ganjil = append(ganjil, num)
        }
    }

    selectionSortAsc(ganjil)
    selectionSortAsc(genap)
    hasil := append(ganjil, genap...)
}

```

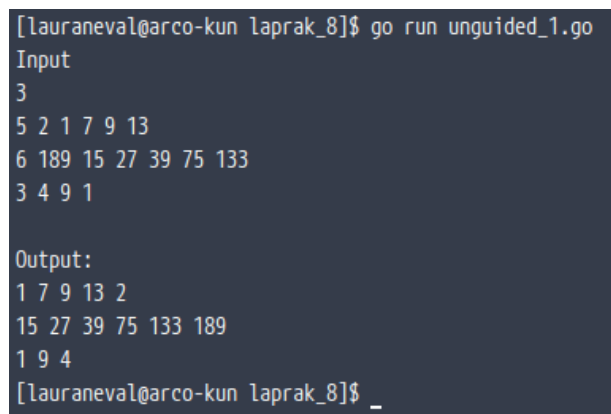
```

        for _, val := range hasil {
            fmt.Print(val, " ")
        }

        fmt.Println()
    }
}

```

Screenshot Output



```

[lauraneval@arco-kun laprak_8]$ go run unguided_1.go
Input
3
5 2 1 7 9 13
6 189 15 27 39 75 133
3 4 9 1

Output:
1 7 9 13 2
15 27 39 75 133 189
1 9 4
[lauraneval@arco-kun laprak_8]$ _

```

Deskripsi Program

Program ini membaca jumlah daerah beserta nomor rumah di masing-masing daerah, memisahkan nomor-nomor tersebut ke dalam dua kelompok: ganjil dan genap. Nomor ganjil diurutkan terlebih dahulu secara menaik menggunakan algoritma selection sort, diikuti oleh pengurutan nomor genap dengan cara yang sama. Setelah proses pengurutan selesai, kedua kelompok digabungkan menjadi satu urutan, dengan nomor ganjil muncul lebih dulu, diikuti nomor genap. Hasil pengurutan untuk setiap daerah ditampilkan dalam baris terpisah. Program juga memastikan bahwa jumlah daerah dan rumah yang dimasukkan berada dalam batasan tertentu (1-999 untuk jumlah daerah dan 1-999999 untuk jumlah rumah) sebelum melanjutkan proses.

UNGUIDED 2 (SELECTION SORT)

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.

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

Keterangan:

Sampai bilangan 0 yang pertama, data terbaca adalah 7 23 11, setelah tersusun: 7 11 23, maka median saat itu adalah 11.

Sampai bilangan 0 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

```
package main  
  
import "fmt"
```



```

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

func calculateMedian(arr []int) int {
    n := len(arr)
    if n%2 == 1 {
        return arr[n/2]
    }
    return (arr[n/2-1] + arr[n/2]) / 2
}

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

    for {

```

```

        fmt.Scan(&input)

        if input == -5313 {

            break

        }

        if input == 0 {

            if len(data) == 0 {

                continue

            }

            selectionSort(data)

            median := calculateMedian(data)

            fmt.Println(median)

        } else {

            data = append(data, input)

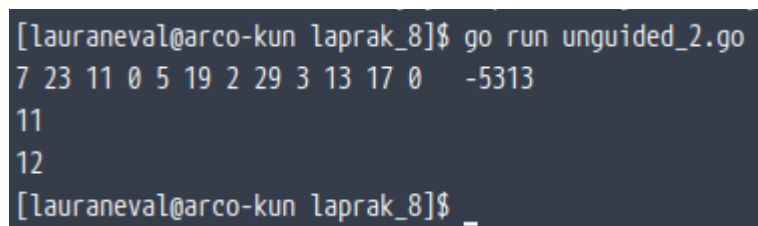
        }

    }

}

```

Sreenshoot Output



```

[lauraneval@arco-kun laprak_8]$ go run unguided_2.go
7 23 11 0 5 19 2 29 3 13 17 0  -5313
11
12
[lauraneval@arco-kun laprak_8]$ _

```

Penjelasan Program

Program ini menerima input angka dari pengguna secara bertahap dan menambahkan angka tersebut ke dalam daftar hingga pengguna memasukkan angka 0. Saat 0 dimasukkan, program mengurutkan daftar angka menggunakan **Selection Sort** secara menaik, menghitung median (elemen tengah untuk jumlah

angka ganjil, atau rata-rata dua elemen tengah untuk jumlah angka genap), dan mencetak nilai median tersebut. Program terus berjalan, memungkinkan pengguna menambahkan angka baru atau memproses median lagi hingga angka khusus -5313 dimasukkan untuk mengakhiri program. Jika daftar kosong, input 0 tidak akan menghasilkan tindakan

UNGUIDED 3 (INSERTION SORT)

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.}
```

Source Code

```
package main

import (
    "fmt"
)

const nMax = 7919

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

type DaftarBuku struct {
    pustaka []Buku
}

func DaftarkanBuku(pustaka *DaftarBuku, buku Buku) {
    if len(pustaka.pustaka) < nMax {
        pustaka.pustaka = append(pustaka.pustaka,
buku)
    }
}
```

```

}

func CetakTerfavorit(pustaka DaftarBuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada buku dalam pustaka.")
        return
    }

    terfavorit := pustaka.pustaka[0]

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

    fmt.Println("Buku Terfavorit:")

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

func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := pustaka.pustaka[i]

```

```

        j := i - 1
        for j >= 0 && pustaka.pustaka[j].rating <
key.rating {
            pustaka.pustaka[j+1] =
pustaka.pustaka[j]
            j--
        }
        pustaka.pustaka[j+1] = key
    }
}

func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada buku dalam pustaka.")
        return
    }

    fmt.Println("5 Buku Dengan Rating Tertinggi:")
    for i := 0; i < 5 && i < n; i++ {
        buku := pustaka.pustaka[i]
        fmt.Printf("Judul      : %s\nRating    : %d\n",
buku.judul, buku.rating)
    }
}

func CariBuku(pustaka DaftarBuku, n, r int) {
    UrutBuku(&pustaka, n)
}

```

```

        low, high := 0, n-1

        for low <= high {

            mid := (low + high) / 2

            if pustaka.pustaka[mid].rating == r {

                buku := pustaka.pustaka[mid]

                fmt.Printf("Buku
ditemukan:\nJudul          : %s\nPenulis      : %s\nPenerb
it      : %s\nTahun        : %d\nEksemplar    : %d\nRating
: %d\n",

                        buku.judul,          buku.penulis,
buku.penerbit, buku.tahun, buku.eksemplar, buku.rating)

                return

            } else if pustaka.pustaka[mid].rating < r {

                high = mid - 1

            } else {

                low = mid + 1

            }

        }

        fmt.Println("Tidak ada buku dengan rating seperti
itu.")
    }

func main() {

    var pustaka DaftarBuku

    var n int

    fmt.Print("Masukkan jumlah buku: ")

    fmt.Scanln(&n)

```



```

        for i := 0; i < n; i++ {
            var id, eksemplar, tahun, rating int
            var judul, penulis, penerbit string

            fmt.Printf("Masukkan data buku ke-%d:\n",
i+1)

            fmt.Print("ID          : ")
            fmt.Scanln(&id)

            fmt.Print("Judul       : ")
            fmt.Scanln(&judul)

            fmt.Print("Penulis     : ")
            fmt.Scanln(&penulis)

            fmt.Print("Penerbit    : ")
            fmt.Scanln(&penerbit)

            fmt.Print("Eksemplar   : ")
            fmt.Scanln(&eksemplar)

            fmt.Print("Tahun       : ")
            fmt.Scanln(&tahun)

            fmt.Print("Rating      : ")
            fmt.Scanln(&rating)

            buku := Buku{id, judul, penulis, penerbit,
eksemplar, tahun, rating}

            DaftarkanBuku(&pustaka, buku)

            fmt.Println()
        }

```

```

        CetakTerfavorit(pustaka, n)

        UrutBuku(&pustaka, n)

        Cetak5Terbaru(pustaka, n)

        var targetRating int

        fmt.Print("\nMasukkan rating buku yang ingin
dicari: ")

        fmt.Scanln(&targetRating)

        CariBuku(pustaka, n, targetRating)
    }

```

Screenshoot Output

```

[lauraneval@arco-kun laprak_8]$ go run unguided_3.go
Masukkan jumlah buku: 2
Masukkan data buku ke-1:
ID      : 9786231028778
Judul   : [LN]Classroom-of-The-Elite-01
Penulis : Shougo_Kinugasa
Penerbit : Phoenix-Gramedia-Indonesia
Eksemplar : 352
Tahun   : 2024
Rating  : 9

Masukkan data buku ke-2:
ID      : 9786230997259
Judul   : [LN]Alya-Sometimes-Hides-Her-Feeling-in-Russian-1
Penulis : SunSunSun
Penerbit : Phoenix-Gramedia-Indonesia
Eksemplar : 264
Tahun   : 2024
Rating  : 8

Buku Terfavorit:
Judul   : [LN]Classroom-of-The-Elite-01
Penulis : Shougo_Kinugasa
Penerbit : Phoenix-Gramedia-Indonesia
Tahun   : 2024
Rating  : 9

5 Buku Dengan Rating Tertinggi:
Judul   : [LN]Classroom-of-The-Elite-01
Rating  : 9
Judul   : [LN]Alya-Sometimes-Hides-Her-Feeling-in-Russian-1
Rating  : 8

Masukkan rating buku yang ingin dicari: 8
Buku ditemukan:
Judul   : [LN]Alya-Sometimes-Hides-Her-Feeling-in-Russian-1
Penulis : SunSunSun
Penerbit : Phoenix-Gramedia-Indonesia
Tahun   : 2024
Eksemplar : 264
Rating  : 8
[lauraneval@arco-kun laprak_8]$ _

```

Penjelasan Program

Program ini adalah aplikasi manajemen data buku berbasis terminal yang memungkinkan pengguna untuk memasukkan data buku, termasuk ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Setelah data dimasukkan, program menampilkan buku dengan rating tertinggi, mengurutkan daftar buku berdasarkan rating secara menurun menggunakan algoritma insertion sort, dan menampilkan hingga lima buku dengan rating tertinggi. Selain itu, program juga memungkinkan pengguna untuk mencari buku berdasarkan rating tertentu menggunakan pencarian biner, di mana hasil pencarian yang cocok akan menampilkan informasi lengkap buku, sedangkan jika tidak ditemukan, program akan memberikan pesan bahwa buku dengan rating tersebut tidak ada.