

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



Disusun Oleh :

Erwin Rivaldo Silaban/2311102248

Dosen Pengampu :

Abednego Dwi Septiadi, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data merupakan proses pengorganisasian elemen-elemen dalam suatu kumpulan data agar teratur sesuai dengan kriteria tertentu, seperti menaik (ascending) atau menurun (descending). Tujuan utama pengurutan adalah memudahkan pencarian, analisis, dan pengelolaan data. Dalam komputer, pengurutan data sering diterapkan pada array, daftar, atau struktur data lainnya.

Pengurutan dilakukan menggunakan algoritma tertentu yang dirancang untuk mengatur elemen-elemen data. Beberapa algoritma pengurutan yang umum digunakan adalah:

1. **Selection Sort**

Algoritma ini bekerja dengan memilih elemen terkecil atau terbesar dari sisa elemen yang belum diurutkan, lalu menukarnya dengan elemen di posisi saat ini. Proses ini diulangi hingga seluruh elemen terurut. Selection Sort sederhana dan mudah diimplementasikan, tetapi kurang efisien untuk data besar.

2. **Insertion Sort**

Algoritma ini mengurutkan data dengan cara menyisipkan elemen ke posisi yang sesuai dalam bagian array yang sudah diurutkan. Elemen-elemen yang lebih besar digeser untuk memberikan ruang bagi elemen yang akan disisipkan. Insertion Sort efektif untuk data kecil atau hampir terurut.

3. **Bubble Sort**

Algoritma ini membandingkan elemen-elemen yang berdekatan dan menukarnya jika mereka tidak berada dalam urutan yang benar. Proses ini diulangi hingga tidak ada lagi pertukaran yang diperlukan. Bubble Sort mudah dipahami tetapi memiliki efisiensi rendah untuk data besar.

4. **Quick Sort**

Algoritma ini membagi data ke dalam dua bagian berdasarkan pivot, kemudian mengurutkan masing-masing bagian secara rekursif. Quick Sort dikenal karena kecepatannya, meskipun performanya bisa menurun jika pivot tidak dipilih dengan baik.

5. **Merge Sort**

Algoritma ini membagi data menjadi bagian-bagian kecil, mengurutkannya secara terpisah, dan kemudian menggabungkan kembali bagian-bagian tersebut menjadi array yang terurut. Merge Sort stabil dan memiliki efisiensi yang konsisten.

II. GUIDED

1. Guided 1

Soal Studi Case

XXXXXXXXXXXXXXXXXXXX

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Selection Sort
func selectionSort(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        idxMin := i
        for j := i + 1; j < n; j++ {
            // Cari elemen terkecil
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
        // Tukar elemen terkecil dengan elemen di
        // posisi i
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}

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

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

        // Membaca nomor rumah untuk daerah ini
        arr := make([]int, m)
        fmt.Printf("Masukkan %d nomor rumah kerabat:
", m)

        for i := 0; i < m; i++ {
            fmt.Scan(&arr[i])
        }

        // Urutkan array dari terkecil ke terbesar
    }
}
```

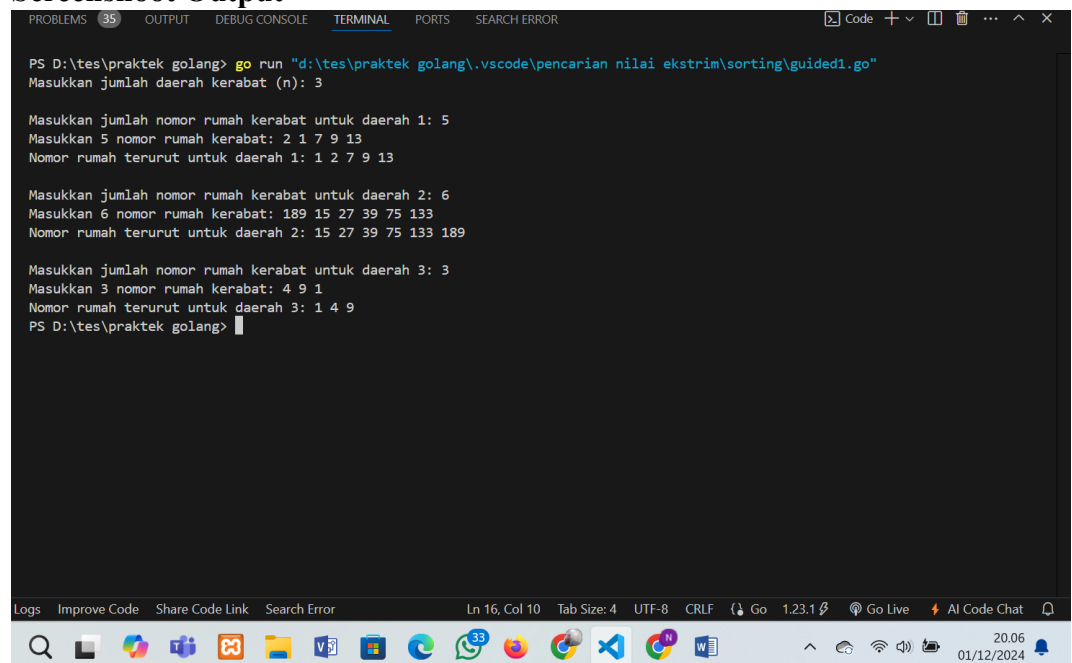
```

        selectionSort(arr, m)

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

```

Screenshoot Output



```

PS D:\tes\praktek golang> go run "d:\tes\praktek golang\.vscode\pencarian nilai ekstrim\sorting\guided1.go"
Masukkan jumlah daerah kerabat (n): 3

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

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 6
Masukkan 6 nomor rumah kerabat: 189 15 27 39 75 133
Nomor rumah terurut untuk daerah 2: 15 27 39 75 133 189

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 3
Masukkan 3 nomor rumah kerabat: 4 9 1
Nomor rumah terurut untuk daerah 3: 1 4 9
PS D:\tes\praktek golang>

```

Deskripsi Program

Pertama, pengguna diminta memasukkan jumlah daerah yang akan diproses. Untuk setiap daerah, program meminta jumlah nomor rumah yang akan diinputkan, lalu meminta pengguna memasukkan nomor rumah tersebut. Setelah data nomor rumah untuk daerah tertentu dimasukkan, program mengurutkannya dari yang terkecil hingga terbesar menggunakan algoritma Selection Sort, di mana elemen terkecil dipindahkan ke posisi awal secara bertahap hingga semua elemen terurut. Setelah pengurutan selesai, nomor rumah yang sudah terurut ditampilkan ke layar. Proses ini diulangi untuk semua daerah yang telah ditentukan, sehingga setiap daerah

memiliki daftar nomor rumah yang terurut secara independen. Program ini sederhana dan efektif untuk mengelola data dengan struktur berulang.

2. Guided 2

Soal Studi Case

XXXXXXXXXXXXXXXXXX

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)

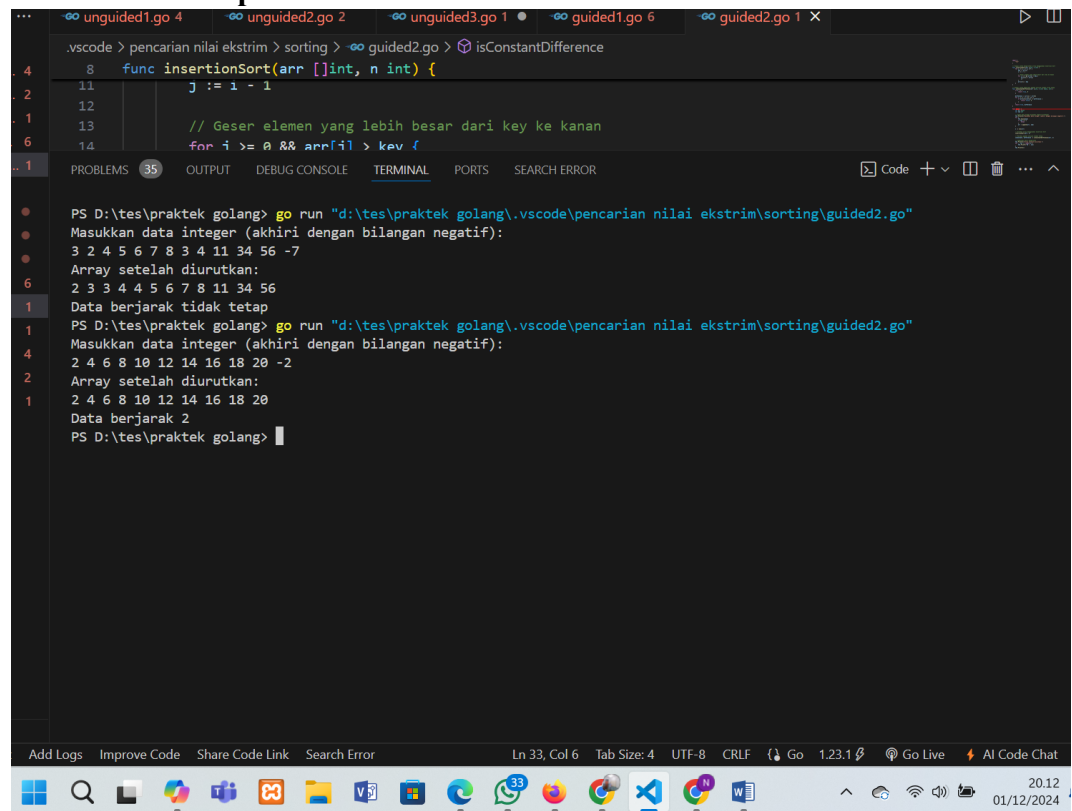
// Periksa apakah selisih elemen tetap
isConstant, difference := isConstantDifference(arr,
n)

// 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



The screenshot shows a VS Code editor with a Go file named `guided2.go`. The code implements an `insertionSort` function. The terminal output shows the program being run twice. In the first run, the input is `3 2 4 5 6 7 8 3 4 11 34 56 -7`, and the output is `2 3 3 4 4 5 6 7 8 11 34 56` with the message "Data berjarak tidak tetap". In the second run, the input is `2 4 6 8 10 12 14 16 18 20 -2`, and the output is `2 4 6 8 10 12 14 16 18 20` with the message "Data berjarak 2".

```
.vscode > pencarian nilai ekstrim > sorting > - guided2.go > isConstantDifference
4      func insertionSort(arr []int, n int) {
2      j := 1 - 1
1      // Geser elemen yang lebih besar dari key ke kanan
6      for i := 0 && arr[i] > key {
1      PROBLEMS 35 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

PS D:\tes\praktek golang> go run "d:\tes\praktek golang\.vscode\pencarian nilai ekstrim\sorting\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
3 2 4 5 6 7 8 3 4 11 34 56 -7
Array setelah diurutkan:
2 3 3 4 4 5 6 7 8 11 34 56
Data berjarak tidak tetap
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\.vscode\pencarian nilai ekstrim\sorting\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
2 4 6 8 10 12 14 16 18 20 -2
Array setelah diurutkan:
2 4 6 8 10 12 14 16 18 20
Data berjarak 2
PS D:\tes\praktek golang>
```

Deskripsi Program

Program ini membaca serangkaian angka dari pengguna, mengurutkannya, dan memeriksa apakah selisih antara elemen-elemen yang sudah diurutkan adalah konstan. Pengguna diminta untuk memasukkan angka-angka integer satu per satu, dan proses input akan berhenti ketika pengguna memasukkan angka negatif. Angka-angka tersebut kemudian diurutkan menggunakan metode *Insertion Sort*, di mana elemen-elemen diatur secara bertahap dengan membandingkan dan memindahkan elemen-elemen yang lebih besar ke kanan hingga mencapai posisi yang benar.

Setelah array diurutkan, program memeriksa apakah selisih antara elemen-elemen yang berurutan tetap konstan. Jika jaraknya tetap, program mencetak nilai selisih tersebut; jika tidak, program menyatakan bahwa data tidak memiliki jarak tetap. Selain itu, array yang telah diurutkan juga ditampilkan ke layar. Dengan cara ini, program membantu mengelola data, memeriksa pola, dan memberikan informasi apakah angka-angka tersebut memiliki jarak tetap.

III. UNGUIDED

1. Soal Studi Case 1

XXXXXXXXXXXXXXXXXX

Sourcecode

```
package main

import "fmt"
//Erwin Rivaldo Silaban/2311102248

func selectionSort(arr []int, asc bool) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        idx := i
        for j := i + 1; j < n; j++ {
            // Kondisi pengurutan berdasarkan
parameter asc
            if (asc && arr[j] < arr[idx]) ||
(!asc && arr[j] > arr[idx]) {
                idx = j
            }
        }
        // Tukar elemen jika diperlukan
        arr[i], arr[idx] = arr[idx], arr[i]
    }
}

// processDaerah memproses rumah dalam satu
daerah
func processDaerah(i, m int) ([]int, []int) {
    // Buat slice untuk menampung nomor rumah
    var arr = make([]int, m)

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

    // Pisahkan nomor rumah genap dan ganjil
    var odd, even []int
    for _, num := range arr {
```



```

        if num%2 == 0 {
            even = append(even, num)
        } else {
            odd = append(odd, num)
        }
    }
    return odd, even
}

// printSortedResults mencetak hasil pengurutan
func printSortedResults(i int, odd, even []int)
{
    selectionSort(odd, true)
    selectionSort(even, false)

    // Cetak header
    fmt.Printf("\nNomor Rumah Setelah di
Urutkan di daerah ke-%d:\n", i+1)

    // Cetak nomor ganjil
    for _, num := range odd {
        fmt.Printf("%d ", num)
    }
    // Cetak nomor genap
    for _, num := range even {
        fmt.Printf("%d ", num)
    }
    fmt.Println()
}

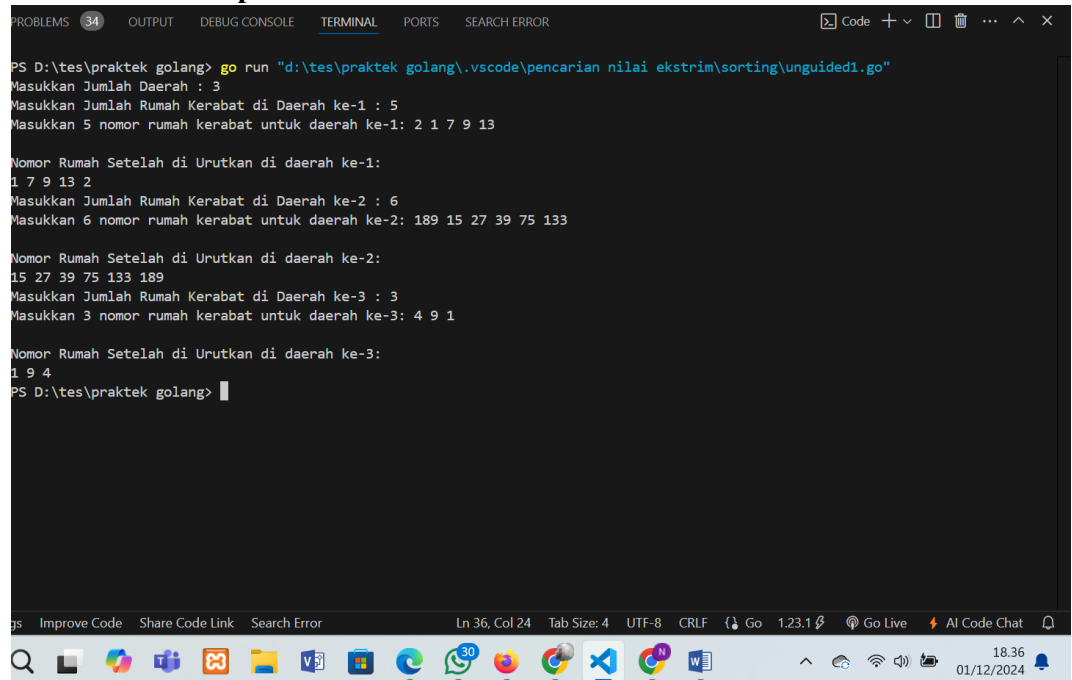
func main() {
    // Variabel untuk jumlah daerah
    var n int
    fmt.Print("Masukkan Jumlah Daerah : ")
    fmt.Scan(&n)

    // Proses setiap daerah
    for i := 0; i < n; i++ {
        // Variabel untuk jumlah rumah di
        setiap daerah
        var m int
        fmt.Printf("Masukkan Jumlah Rumah
Kerabat di Daerah ke-%d : ", i+1)
    }
}

```

```
        fmt.Scan(&m)
        odd, even := processDaerah(i, m)
        printSortedResults(i, odd, even)
    }
}
```

Screenshoot Output



```
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\.vscode\pencarian nilai ekstrim\sorting\unguided1.go"
Masukkan Jumlah Daerah : 3
Masukkan Jumlah Rumah Kerabat di Daerah ke-1 : 5
Masukkan 5 nomor rumah kerabat untuk daerah ke-1: 2 1 7 9 13

Nomor Rumah Setelah di Urutkan di daerah ke-1:
1 7 9 13 2
Masukkan Jumlah Rumah Kerabat di Daerah ke-2 : 6
Masukkan 6 nomor rumah kerabat untuk daerah ke-2: 189 15 27 39 75 133

Nomor Rumah Setelah di Urutkan di daerah ke-2:
15 27 39 75 133 189
Masukkan Jumlah Rumah Kerabat di Daerah ke-3 : 3
Masukkan 3 nomor rumah kerabat untuk daerah ke-3: 4 9 1

Nomor Rumah Setelah di Urutkan di daerah ke-3:
1 9 4
PS D:\tes\praktek golang> 
```

Deskripsi Program

Program akan meminta pengguna memasukkan jumlah daerah yang akan diproses, lalu untuk setiap daerah, pengguna memasukkan jumlah rumah kerabat dan nomor-nomor rumah tersebut. Program memisahkan nomor rumah menjadi dua kelompok, yaitu nomor ganjil dan nomor genap. Nomor ganjil diurutkan secara menaik (ascending), sedangkan nomor genap diurutkan secara menurun (descending) menggunakan algoritma Selection Sort. Setelah proses pengurutan selesai, hasilnya dicetak, dimulai dari nomor ganjil yang sudah diurutkan diikuti dengan nomor genap yang juga telah diurutkan. Langkah ini diulang untuk setiap daerah hingga semua data selesai diproses, memberikan hasil nomor rumah yang terorganisir untuk setiap daerah.

2. Soal Studi Case 2

XXXXXXXXXXXXXXXXXXXX

Sourcecode

```
package main

import (
    "fmt"
)

// Erwin Rivaldo Silaban 2311102248
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
            }
        }
        // untuk menukar elemen terkecil dengan elemen
        // di posisi i
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}

func median(arr []int, n int) int {
    if n%2 == 0 {
        return (arr[n/2-1] + arr[n/2]) / 2
    } else {
        return arr[(n-1)/2]
    }
}

func main() {
    var arr [1000000]int
    var arrMedian [1000000]int
    var n, nMed, datum int

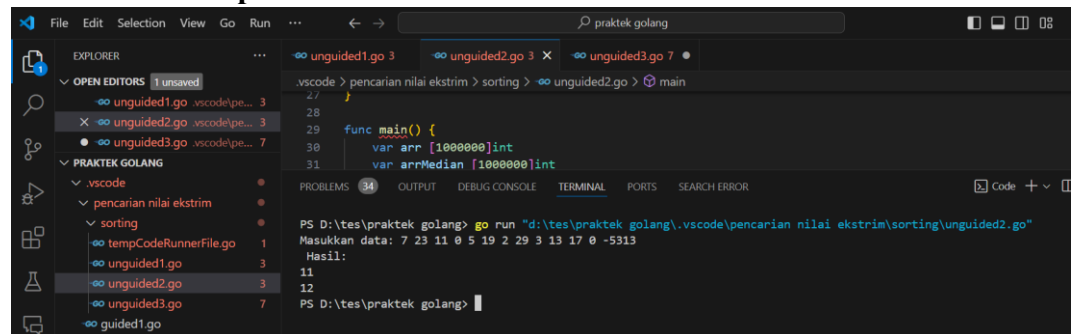
    fmt.Printf("Masukkan data: ")
    for datum != -5313 && n < 1000000 {
        fmt.Scan(&datum)
        if datum == 0 {
            selectionSort(arr[:n], n)
            arrMedian[nMed] = median(arr[:n], n)
            nMed++
        } else {
            arr[n] = datum
            n++
        }
    }
}
```

```

    fmt.Println(" Hasil: ")
    for i := 0; i < nMed; i++ {
        fmt.Println(arrMedian[i], " ")
    }
}

```

Screenshoot Output



Deskripsi Program

Program ini dapat meminta pengguna memasukkan angka satu per satu, di mana angka 0 digunakan sebagai tanda akhir dari sebuah kelompok, dan angka -5313 digunakan untuk menghentikan proses input sepenuhnya. Setiap kali sebuah kelompok selesai, angka-angka dalam kelompok tersebut diurutkan menggunakan algoritma Selection Sort. Setelah diurutkan, program menghitung median dari kelompok tersebut, di mana jika jumlah angka dalam kelompok ganjil, median adalah angka di posisi tengah, dan jika genap, median adalah rata-rata dari dua angka tengah. Median dari setiap kelompok disimpan dalam sebuah array, dan ketika pengguna menghentikan input, program mencetak seluruh median yang telah dihitung. Program ini membantu menganalisis data dalam kelompok-kelompok terpisah dengan efisien, terutama untuk menghitung nilai tengah dari setiap kelompok data.

3. Soal Studi Case 3

XXXXXXXXXXXXXXXXXXXX

Sourcecode

```

package main

import (
    "fmt"
)

// Erwin Rivaldo Silaban/2311102248

```

```

const nMax = 7919

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

type DaftarBuku struct {
    Pustaka [nMax]Buku
    nPustaka int
}

// Fungsi untuk menambahkan buku ke daftar
func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    for i := 0; i < n; i++ {
        fmt.Println("Masukkan data buku (id, judul,
penulis, penerbit, eksemplar, tahun, rating):")
        fmt.Scan(&pustaka.Pustaka[i].id,
&pustaka.Pustaka[i].judul, &pustaka.Pustaka[i].penulis,
&pustaka.Pustaka[i].penerbit,
&pustaka.Pustaka[i].eksemplar,
&pustaka.Pustaka[i].tahun, &pustaka.Pustaka[i].rating)
    }
    pustaka.nPustaka = n
}

// Fungsi untuk mencetak buku dengan rating tertinggi
func CetakTerfavorit(pustaka DaftarBuku) {
    if pustaka.nPustaka == 0 {
        fmt.Println("Tidak ada buku dalam pustaka.")
        return
    }

    terfavorit := pustaka.Pustaka[0]
    for i := 1; i < pustaka.nPustaka; i++ {
        if pustaka.Pustaka[i].rating > terfavorit.rating
{
            terfavorit = pustaka.Pustaka[i]
        }
    }

    fmt.Println("Buku terfavorit:")
    fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s,
Tahun: %d, Rating: %d\n",
        terfavorit.judul, terfavorit.penulis,
terfavorit.penerbit, terfavorit.tahun,
terfavorit.rating)

```

```

}

// Fungsi untuk mengurutkan buku berdasarkan rating
(Insertion Sort)
func UrutBuku(pustaka *DaftarBuku) {
    for i := 1; i < pustaka.nPustaka; 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
    }
}

// Fungsi untuk mencetak 5 buku dengan rating tertinggi
func Cetak5Terbaru(pustaka DaftarBuku) {
    fmt.Println("5 Buku dengan rating tertinggi:")
    for i := 0; i < 5 && i < pustaka.nPustaka; i++ {
        buku := pustaka.Pustaka[i]
        fmt.Printf("Judul: %s, Penulis: %s, Penerbit:
%s, Tahun: %d, Rating: %d\n",
            buku.judul, buku.penulis, buku.penerbit,
buku.tahun, buku.rating)
    }
}

// Fungsi untuk mencari buku berdasarkan rating (Binary
Search)
func CariBuku(pustaka DaftarBuku, rating int) {
    low, high := 0, pustaka.nPustaka-1
    for low <= high {
        mid := (low + high) / 2
        if pustaka.Pustaka[mid].rating == rating {
            buku := pustaka.Pustaka[mid]
            fmt.Printf("Buku ditemukan: Judul: %s,
Penulis: %s, Penerbit: %s, Tahun: %d, Rating: %d\n",
                buku.judul, buku.penulis, buku.penerbit,
buku.tahun, buku.rating)
            return
        } else if pustaka.Pustaka[mid].rating < rating {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
    fmt.Println("Tidak ada buku dengan rating seperti
itu.")
}

```

```

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

    // Input jumlah buku
    fmt.Print("Masukkan jumlah buku: ")
    fmt.Scan(&n)
    DaftarkanBuku(&pustaka, n)
    CetakTerfavorit(pustaka)
    UrutBuku(&pustaka)
    Cetak5Terbaru(pustaka)
    fmt.Print("Masukkan rating buku yang ingin dicari: ")

    fmt.Scan(&rating)
    CariBuku(pustaka, rating)
}

```

Screenshoot Output

```

PS D:\tes\praktek golang> go run "d:\tes\praktek golang\.vscode\pencarian nilai ekstrim\sorting\unguided3.go"
Masukkan jumlah buku: 3
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 buku01 penulis01 penerbit01 500 1999 4
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 buku02 penulis02 penerbit02 300 2000 5
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
3 buku03 penulis03 penerbit03 700 2022 4
Buku terfavorit:
Judul: buku02, Penulis: penulis02, Penerbit: penerbit02, Tahun: 2000, Rating: 5
5 Buku dengan rating tertinggi:
Judul: buku02, Penulis: penulis02, Penerbit: penerbit02, Tahun: 2000, Rating: 5
Judul: buku01, Penulis: penulis01, Penerbit: penerbit01, Tahun: 1999, Rating: 4
Judul: buku03, Penulis: penulis03, Penerbit: penerbit03, Tahun: 2022, Rating: 4
Masukkan rating buku yang ingin dicari: go run "d:\tes\praktek golang\.vscode\pencarian nilai ekstrim\sorting\unguided3.go"
Tidak ada buku dengan rating seperti itu.
PS D:\tes\praktek golang>

```

Deskripsi Program

Pada program kali ini pengguna diminta memasukkan detail buku, seperti jumlah buku yang akan didaftarkan, dan data setiap buku disimpan dalam daftar. Program kemudian mencari buku dengan rating tertinggi untuk ditampilkan, mengurutkan buku berdasarkan rating menggunakan metode Insertion Sort, dan menampilkan lima buku terbaik dalam daftar. Selain itu, pengguna dapat mencari buku tertentu berdasarkan rating yang dimasukkan

dengan menggunakan algoritma Binary Search. Jika ditemukan, detail buku akan ditampilkan; jika tidak, program akan memberi tahu bahwa buku dengan rating tersebut tidak tersedia. Dengan alur yang sistematis, program ini memungkinkan pengelolaan data pustaka secara efisien, termasuk pencatatan, pengurutan, dan pencarian buku berdasarkan kriteria tertentu.