LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII & XIII

PENGURUTAN DATA

Disusun Oleh :

Tegar Aji Pangestu / 2311102021

IF-11-06

Dosen Pengampu :

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan memiliki peran penting dalam pengelolaan data. Data yang terurut memudahkan dalam pencarian, pengolahan, dan analisis lebih lanjut. Dalam Golang, pengurutan dapat diimplementasikan secara manual menggunakan algoritma seperti Selection Sort, Insertion Sort, dan Bubble Sort, atau memanfaatkan pustaka bawaan seperti sort.

Algoritma pengurutan adalah langkah-langkah sistematis untuk mengatur data. Beberapa algoritma umum yang dapat diimplementasikan dalam Golang adalah:

1. **SelectionSort**
   Selection Sort bekerja dengan mencari elemen terkecil (atau terbesar) dari bagian data yang belum terurut, lalu menempatkannya di posisi yang sesuai. Proses ini diulang hingga semua data terurut.

2. **InsertionSort**
   Insertion Sort mengatur elemen satu per satu dengan cara menyisipkannya ke dalam posisi yang sesuai dalam bagian data yang sudah terurut.

3. **MergeSort dan quick short**
   Algoritma ini lebih kompleks namun lebih efisien pada data berukuran besar, karena memiliki waktu eksekusi yang lebih baik dibanding algoritma sederhana seperti Selection dan Insertion Sort.

II. GUIDED

1. Guided 1

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

```
Masukkan jumlah daerah kerabat (n): 3

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

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

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 3 4 9 1
Masukkan 3 nomor rumah kerabat: Nomor rumah terurut untuk daerah 3: 1 4 9
```

Deskripsi Program

Program ini dirancang untuk membaca data berupa nomor rumah kerabat dari beberapa daerah, kemudian mengurutkan nomor rumah tersebut menggunakan algoritma Selection Sort dan menampilkannya.

2. Guided 2

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

```
Masukkan data integer (akhiri dengan bilangan negatif):
21 22 23 24 25 26 -27
Array setelah diurutkan:
21 22 23 24 25 26
Data berjarak 1
```

Deskripsi Program

Program ini menggunakan Insertion Sort untuk mengurutkan data, yang cukup efisien untuk array dengan jumlah elemen kecil hingga menengah. Selain pengurutan, program juga memeriksa apakah data yang diurutkan memiliki selisih yang tetap antar elemen, yang merupakan properti dari urutan aritmatika.

III. UNGUIDED

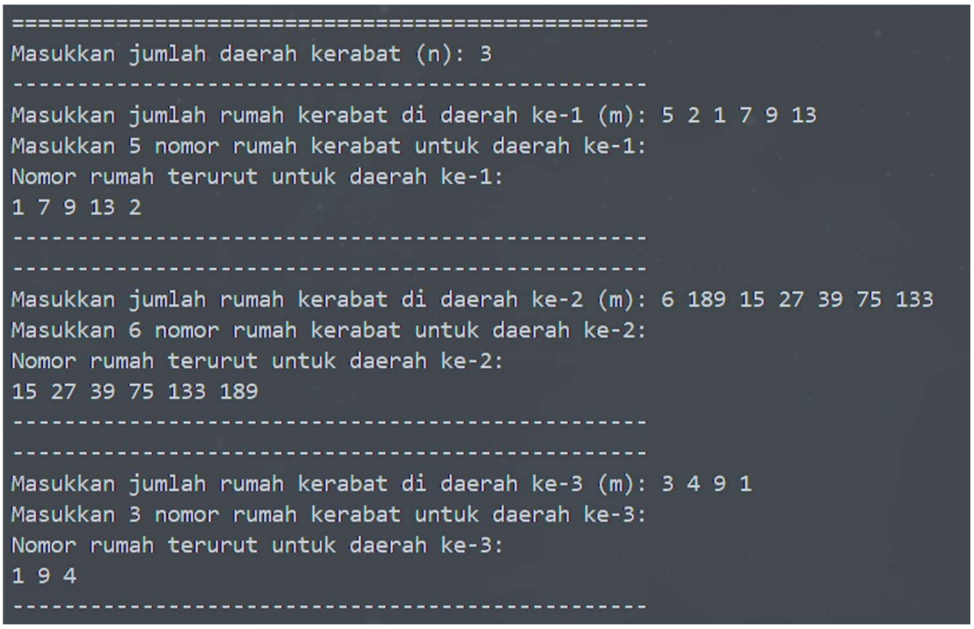
Unguided 1

Sourcecode

```
package main
//Tegar Aji Pangestu 2311102021 IF 06//
import "fmt"
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++ {
              if (asc && arr[j] < arr[idx]) || (!asc &&
arr[j] > arr[idx]) {
                  idx = j
            }
        }
        arr[i], arr[idx] = arr[idx], arr[i]
    }
}

func printSeparator() {
     fmt.Println("\n=================================
============")
}

func printDashedSeparator() {
    fmt.Println("-----------------------------------
---------")
}

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

    var odd, even []int
    for _, num := range arr {
        if num%2 == 0 {
            even = append(even, num)
        } else {
            odd = append(odd, num)
        }
    }
    return odd, even
}

func printSortedResults(i int, odd, even []int) {
```

```
    selectionSort(odd, true)
    selectionSort(even, false)

     fmt.Printf("\nNomor rumah terurut untuk daerah ke-
%d:\n", i+1)

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

    for _, num := range even {
        fmt.Printf("%d ", num)
    }
    fmt.Println()
}
func main() {
    var n int
    printSeparator()
    fmt.Print("Masukkan jumlah daerah kerabat (n): ")
    fmt.Scan(&n)

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

        odd, even := processDaerah(i, m)

        printSortedResults(i, odd, even)
        printDashedSeparator()
    }
}
```

Screenshoot Output



Deskripsi Program

Program ini memberikan solusi untuk memisahkan dan mengurutkan nomor rumah kerabat berdasarkan kategori ganjil dan genap menggunakan algoritma Selection Sort. Hasil yang ditampilkan terstruktur dengan jelas, memudahkan pengguna untuk melihat nomor rumah yang terurut sesuai dengan ketentuan yang diinginkan.

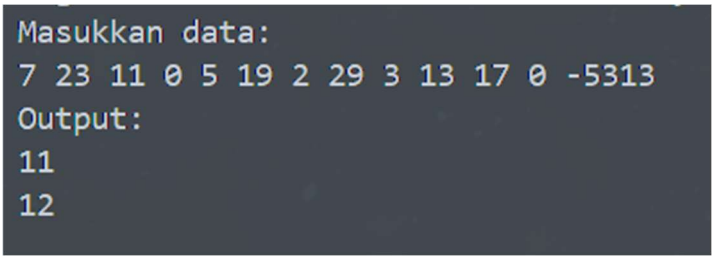
Unguided 2

Sourcecode

```
package main
//Tegar Aji Pangestu 2311102021 IF 06//
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 median(arr []int, n int) int {
    if (n % 2) == 0 {
        return int((arr[n/2 - 1] + arr[n/2])/2)
    } else {
        return arr[(n-1)/2]
    }
}

func main() {

    arr := make([]int, 1000000)
    arrMedian := make([]int, 1000000)
    n := 0
    nMed := 0
    datum := 0

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

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

}
```

Screenshoot Output



Deskripsi Program

Program ini berfungsi untuk menerima input data angka, mengurutkan angka yang dimasukkan menggunakan Selection Sort, dan menghitung median setiap kali angka 0 dimasukkan. Setelah menghitung median, hasilnya disimpan dan ditampilkan pada akhir program ketika input -5313 dimasukkan.

Unguided 3

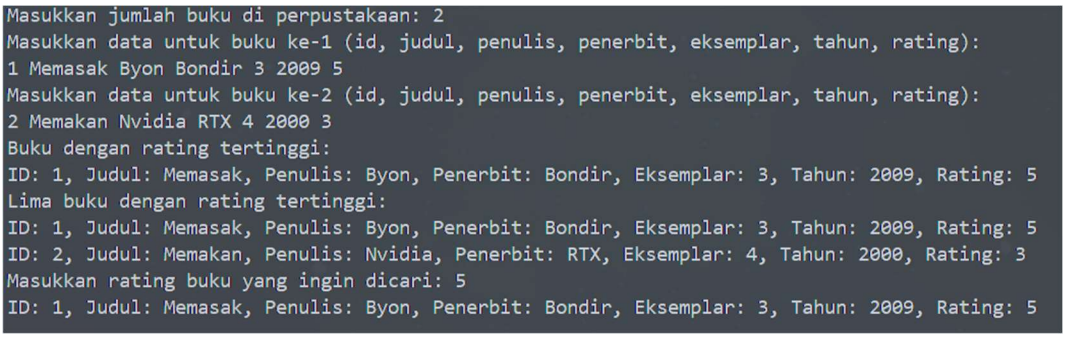
Sourcecode

```
package main
//Tegar Aji Pangestu 2311102021 IF 06//
import (
    "fmt"
)
type Buku struct {
    id int
    eksemplar int
    tahun int
    rating int
    judul,penerbit, penulis string
}

func DaftarkanBuku(pustaka *[]Buku, n int) {
    for i := 0; i < n; i++ {
        var buku Buku
         fmt.Printf("Masukkan data untuk buku ke-%d (id,
judul, penulis, penerbit, eksemplar, tahun, rating):\n",
i+1)
          fmt.Scan(&buku.id, &buku.judul, &buku.penulis,
&buku.penerbit,       &buku.eksemplar,       &buku.tahun,
&buku.rating)
        *pustaka = append(*pustaka, buku)
    }
}

func CetakFavorit(pustaka []Buku, n int) {
    if len(pustaka) == 0 {
        fmt.Println("Pustaka kosong.")
        return
    }
    terfavorit := pustaka[0]
    for _, buku := range pustaka {
        if buku.rating > terfavorit.rating {
            terfavorit = buku
        }
    }
```

```
    fmt.Println("Buku dengan rating tertinggi:")
    fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit:
%s, Eksemplar: %d, Tahun: %d, Rating: %d\n",
                      terfavorit.id,    terfavorit.judul,
terfavorit.penulis,                 terfavorit.penerbit,
terfavorit.eksemplar,                  terfavorit.tahun,
terfavorit.rating)
}

func UrutkanBuku(pustaka *[]Buku, n int) {
    for i := 1; i < len(*pustaka); i++ {
        key := (*pustaka)[i]
        j := i - 1
        for j >= 0 && (*pustaka)[j].rating < key.rating {
            (*pustaka)[j+1] = (*pustaka)[j]
            j--
        }
        (*pustaka)[j+1] = key
    }
}

func Cetak5Terbaik(pustaka []Buku, n int) {
    fmt.Println("Lima buku dengan rating tertinggi:")
    for i := 0; i < 5 && i < len(pustaka); i++ {
        buku := pustaka[i]
           fmt.Printf("ID:  %d,  Judul:  %s,  Penulis:  %s,
Penerbit: %s, Eksemplar: %d, Tahun: %d, Rating: %d\n",
                    buku.id,  buku.judul,  buku.penulis,
buku.penerbit, buku.eksemplar, buku.tahun, buku.rating)
    }
}

func CariBuku(pustaka []Buku, n int, r int) {
    ditemukan := false
    for _, buku := range pustaka {
        if buku.rating == r {
            ditemukan = true
             fmt.Printf("ID: %d, Judul: %s, Penulis: %s,
Penerbit: %s, Eksemplar: %d, Tahun: %d, Rating: %d\n",
                     buku.id,  buku.judul,  buku.penulis,
buku.penerbit, buku.eksemplar, buku.tahun, buku.rating)
        }
    }
    if !ditemukan {
            fmt.Println("Tidak  ada  buku  dengan  rating
tersebut.")
    }
}
func main() {
    var n int
    fmt.Print("Masukkan jumlah buku di perpustakaan: ")
    fmt.Scan(&n)
```

```
    if n <= 0 || n > 7919 {
         fmt.Println("Jumlah buku harus antara 1 hingga
7919.")
        return
    }

    var pustaka []Buku

    DaftarkanBuku(&pustaka, n)
    CetakFavorit(pustaka, n)
    UrutkanBuku(&pustaka, n)

    Cetak5Terbaik(pustaka, n)
    var rating int
    fmt.Print("Masukkan rating buku yang ingin dicari: ")
    fmt.Scan(&rating)
    CariBuku(pustaka, n, rating)
}
```

Screenshoot Output



Deskripsi Program

Program ini adalah aplikasi manajemen buku yang menyediakan berbagai fitur seperti pendaftaran buku, pengurutan berdasarkan rating, pencarian buku berdasarkan rating, dan menampilkan buku dengan rating tertinggi. Penggunaan algoritma **Insertion Sort** membantu dalam pengurutan data buku berdasarkan rating, serta memberikan fleksibilitas bagi pengguna untuk berinteraksi dengan data buku melalui pencarian dan pengurutan.