

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



**Disusun Oleh :**

**Reza Alvonzo / 2311102026**

**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 data merupakan proses mengatur elemen-elemen dalam suatu kumpulan data (seperti array atau slice) agar tersusun berdasarkan aturan tertentu, baik secara **menaik (ascending)** maupun **menurun (descending)**. Proses ini penting untuk mendukung berbagai operasi seperti pencarian, analisis data, dan pemrosesan yang lebih efisien.

Pengurutan dapat dilakukan secara internal (dalam memori) atau eksternal (melibatkan penyimpanan eksternal untuk dataset besar). Teknik pengurutan biasanya ditentukan oleh jenis data yang digunakan, ukuran dataset, dan kebutuhan efisiensi waktu maupun memori.

Paket `sort` dalam Golang adalah pustaka yang dirancang untuk mempermudah proses pengurutan. Fungsi-fungsi utama dalam paket ini meliputi:

- `sort.Ints([]int)`: Mengurutkan elemen `int` dalam slice secara menaik.
- `sort.Strings([]string)`: Mengurutkan elemen `string` dalam slice secara menaik.
- `sort.Float64s([]float64)`: Mengurutkan elemen `float64` secara menaik.

Untuk pengurutan yang lebih kompleks, paket ini menyediakan interface `sort.Interface` yang memungkinkan pengembang mendefinisikan cara pengurutan dengan metode:

- `Len() int`: Mengembalikan panjang slice.
- `Less(i, j int) bool`: Menentukan apakah elemen `i` lebih kecil dari elemen `j`.
- `Swap(i, j int)`: Menukar elemen pada indeks `i` dan `j`.

## 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 menggunakan algoritma sederhana **Selection Sort** untuk mengurutkan nomor rumah kerabat per daerah. Setiap daerah diproses secara independen, memungkinkan hasil pengurutan ditampilkan satu per satu. Program ini cocok untuk pengurutan dengan jumlah data kecil hingga sedang.

## 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)

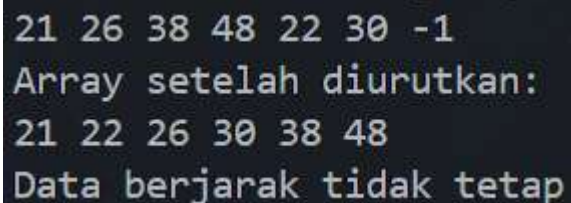
// Periksa apakah selisih elemen tetap
isConstant, difference := isConstantDifference(arr, n)

// 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



```

21 26 38 48 22 30 -1
Array setelah diurutkan:
21 22 26 30 38 48
Data berjarak tidak tetap

```

### Deskripsi Program

Program ini menggabungkan algoritma pengurutan dan pemeriksaan pola aritmatika dalam array. Insertion Sort digunakan untuk mengurutkan data dengan efisien. Fungsi `isConstantDifference` memeriksa apakah data membentuk deret aritmatika. Program ini cocok untuk menganalisis data dengan kebutuhan pola tertentu, seperti deret aritmatika.

### III. UNGUIDED

#### Unguided 1

##### Sourcecode

```
package main
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()
    }
    //Reza Alvonzo 2311102026 IF 11 06
    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

```
Masukkan jumlah daerah kerabat (n): 3
-----
Masukkan jumlah rumah kerabat di daerah ke-1 (m): 5 2 1 7 9 13
Masukkan 5 nomor rumah kerabat untuk daerah ke-1:
Nomor rumah terurut untuk daerah ke-1:
1 7 9 13 2
-----
-----
Masukkan jumlah rumah kerabat di daerah ke-2 (m): 6 189 15 27 39 75 133
Masukkan 6 nomor rumah kerabat untuk daerah ke-2:
Nomor rumah terurut untuk daerah ke-2:
15 27 39 75 133 189
-----
-----
Masukkan jumlah rumah kerabat di daerah ke-3 (m): 3 4 9 1
Masukkan 3 nomor rumah kerabat untuk daerah ke-3:
Nomor rumah terurut untuk daerah ke-3:
1 9 4
-----
```

## Deskripsi Program

Program ini memanfaatkan Selection Sort untuk mengelola pengurutan nomor rumah dengan fleksibilitas arah pengurutan (ganjil naik, genap turun). Dengan memisahkan nomor rumah berdasarkan ganjil dan genap, program ini dapat mempermudah analisis dan pelaporan data.

## Unguided 2

### Sourcecode

```
package main
import (
    "bufio"
    "fmt"
    "os"
    "strconv"
    "strings"
)

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]
    }
}
```

```

    }
    }
    arr[i], arr[minIdx] = arr[minIdx], arr[i]
}

func calculateMedian(arr []int) float64 {
    n := len(arr)
    if n%2 == 0 {
        return float64(arr[n/2-1]+arr[n/2]) / 2.0
    }
    return float64(arr[n/2])
}

//Reza Alvonzo 2311102026 IF 11 06
func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Println("Masukkan angka yang dipisahkan dengan spasi (akhiri dengan -5313):")
    scanner.Scan()
    line := scanner.Text()
    parts := strings.Split(line, " ")
    var data []int

    for _, part := range parts {
        num, err := strconv.Atoi(part)
        if err != nil {
            fmt.Println("Input tidak valid, harap masukkan angka bulat saja.")
            return
        }

        if num == -5313 {
            break
        } else if num == 0 {
            selectionSort(data)
            median := calculateMedian(data)
            fmt.Printf("%.0f\n", median)
        } else {
            data = append(data, num)
        }
    }
}

```

## Screenshoot Output

```
Masukkan angka yang dipisahkan dengan spasi (akhiri dengan -5313):  
7 23 11 0 5 19 2 29 3 13 17 0 -5313  
11  
12
```

## Deskripsi Program

Program ini memungkinkan pengguna untuk memasukkan data angka secara dinamis dan menghitung median setiap kali diminta melalui angka **0**. Penggunaan algoritma **Selection Sort** memastikan bahwa data selalu terurut sebelum median dihitung, sehingga hasilnya akurat. Program juga dilengkapi dengan validasi input untuk mencegah kesalahan.

## Unguided 3

### Sourcecode

```
package main  
import (  
    "fmt"  
)  
type Buku struct {  
    id,eksemplar,tahun,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.")
    }
}

//Reza Alvonzo 2311102026 IF 11 06

```

```

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

```

Masukkan jumlah buku di perpustakaan: 3
Masukkan data untuk buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 Buku Reza PT 2 2004 4
Masukkan data untuk buku ke-2 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 Rendang Reza PT 3 2005 5
Masukkan data untuk buku ke-3 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
3 Semangka Reza PT 4 2006 3
Buku dengan rating tertinggi:
ID: 2, Judul: Rendang, Penulis: Reza, Penerbit: PT, Eksemplar: 3, Tahun: 2005, Rating: 5
Lima buku dengan rating tertinggi:
ID: 2, Judul: Rendang, Penulis: Reza, Penerbit: PT, Eksemplar: 3, Tahun: 2005, Rating: 5
ID: 1, Judul: Buku, Penulis: Reza, Penerbit: PT, Eksemplar: 2, Tahun: 2004, Rating: 4
ID: 3, Judul: Semangka, Penulis: Reza, Penerbit: PT, Eksemplar: 4, Tahun: 2006, Rating: 3
Masukkan rating buku yang ingin dicari: 4
ID: 1, Judul: Buku, Penulis: Reza, Penerbit: PT, Eksemplar: 2, Tahun: 2004, Rating: 4

```

## Deskripsi Program

Program ini dirancang untuk mendukung kebutuhan perpustakaan kecil hingga menengah, dengan batas maksimum 7919 buku. Dengan algoritma sederhana seperti **Insertion Sort** dan struktur data yang terorganisasi,

program ini memastikan efisiensi pengelolaan data buku, sekaligus mempermudah akses informasi bagi pengguna.