

**LAPORAN PRAKTIKUM**  
**Algoritma Pemrograman**  
**MODUL XII & XIII**  
**PENGURUTAN DATA**



Oleh:

Ilhan Sahal Mansiz

2311102029

IF-11-02

**S1 TEKNIK INFORMATIKA**  
**TELKOM UNIVERSITY PURWOKERTO**  
**2024**

## I. DASAR TEORI

**Selection Sort dan Insertion Sort** adalah dua algoritma pengurutan sederhana yang sering digunakan dalam pemrograman. **Selection Sort** bekerja dengan mencari elemen terkecil dari bagian yang belum diurutkan pada array dan menukarnya dengan elemen di posisi awal bagian tersebut. Proses ini diulang hingga seluruh elemen berada dalam urutan yang benar. Algoritma ini memiliki kompleksitas waktu  $O(n^2)$  dalam semua kasus, baik terbaik, rata-rata, maupun terburuk, sehingga lebih cocok untuk dataset kecil. Dalam bahasa Go, Selection Sort dapat diimplementasikan dengan menggunakan loop bersarang dan operasi pertukaran elemen untuk mengurutkan array.

**Insertion Sort**, di sisi lain, membangun array yang sudah terurut secara bertahap dengan memasukkan elemen baru ke posisi yang sesuai. Algoritma ini membandingkan elemen saat ini dengan elemen-elemen sebelumnya dan menggeser elemen-elemen yang lebih besar untuk memberikan ruang bagi elemen yang sedang dimasukkan. Dengan kompleksitas waktu  $O(n)$  untuk array yang hampir terurut dan  $O(n^2)$  untuk array acak, Insertion Sort lebih efisien dibandingkan Selection Sort dalam kasus tertentu, terutama ketika data sudah hampir terurut. Implementasi algoritma ini dalam bahasa Go menggunakan loop bertingkat dengan pemeriksaan kondisi untuk menyisipkan elemen ke posisi yang tepat.

Kedua algoritma ini memiliki kelebihan dalam hal kesederhanaan implementasi, namun kurang efisien untuk dataset yang besar. **Selection Sort** tidak stabil karena elemen dengan nilai yang sama dapat tertukar posisinya, sedangkan **Insertion Sort** merupakan algoritma yang stabil. Penggunaan algoritma-algoritma ini di dunia nyata biasanya terbatas pada skenario-skenario yang memerlukan pengurutan sederhana dan cepat untuk jumlah data yang kecil. Implementasinya dalam bahasa Go sangat cocok untuk kebutuhan pembelajaran dan eksplorasi dasar-dasar algoritma pengurutan.

## II. GUIDED

1.

```
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()
}
}

```

Screenshot Program :

```

Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5
Masukkan nomor rumah kerabat untuk daerah ke-1: 12 45 7 89 23
Hasil urutan rumah untuk daerah ke-1: 89 45 23 12 7
Masukkan jumlah rumah kerabat untuk daerah ke-2: 4
Masukkan nomor rumah kerabat untuk daerah ke-2: 3 67 45 20
Hasil urutan rumah untuk daerah ke-2: 67 45 20 3
PS C:\Users\ACER\Downloads\Semester 3\LapraK Alpro smstr3\Ilhan

```

Deskripsi Program :

Program ini adalah aplikasi berbasis teks untuk mengurutkan nomor rumah kerabat di beberapa daerah menggunakan algoritma **Selection Sort**. Pengguna diminta memasukkan jumlah daerah (n) dan jumlah rumah di setiap daerah (m), dengan validasi agar nilainya berada dalam batas tertentu. Selanjutnya, program meminta input nomor rumah untuk setiap daerah dan mengurutkannya dari yang terbesar ke yang terkecil menggunakan algoritma Selection Sort yang dimodifikasi untuk mencari elemen terbesar. Hasil pengurutan nomor rumah ditampilkan untuk setiap daerah, memberikan solusi sederhana untuk menyusun data secara teratur sesuai kebutuhan pengguna.

2.

```
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

```

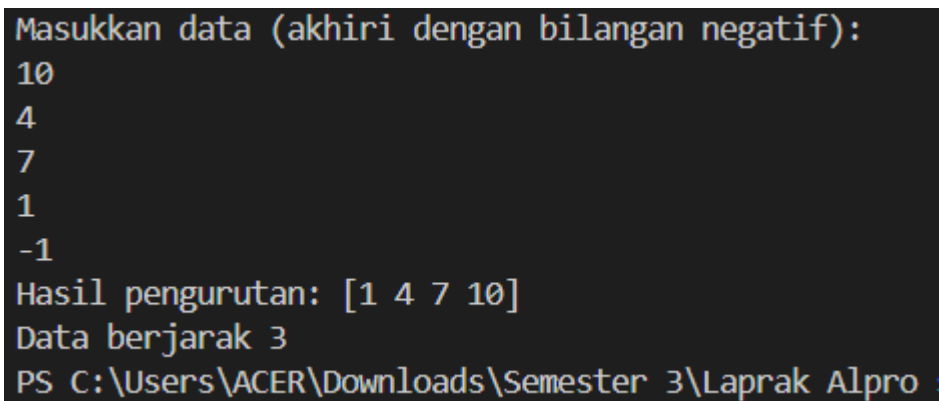
```

        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")
        }
    }
}

```

Screenshot Program :



```

Masukkan data (akhiri dengan bilangan negatif):
10
4
7
1
-1
Hasil pengurutan: [1 4 7 10]
Data berjarak 3
PS C:\Users\ACER\Downloads\Semester 3\Laprak Alpro

```

Deskripsi Program :

Program ini menerima serangkaian angka positif dari pengguna, mengurutkannya menggunakan algoritma **Insertion Sort**, lalu memeriksa apakah data memiliki selisih antar elemen yang tetap. Jika selisih antar elemen konsisten, program mencetak nilai selisih tersebut; jika tidak, program menyatakan bahwa data tidak berjarak tetap. Program ini membantu mengidentifikasi pola jarak pada data setelah pengurutan.

### III. UNGUIDED

1.

- 2) Belakang 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 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

Source Code :

```
package main

import (
    "fmt"
    "sort"
)

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])
    }

    // Pisahkan nomor ganjil dan genap
    var odd, even []int
    for _, house := range houses {
        if house%2 == 0 {
            even = append(even, house)
        } else {
            odd = append(odd, house)
        }
    }

    // Urutkan ganjil membesar
    sort.Ints(odd)
    // Urutkan genap mengecil
    sort.Sort(sort.Reverse(sort.IntSlice(even)))

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

```

```

    }
    for _, house := range even {
        fmt.Printf("%d ", house)
    }
    fmt.Println()
}
}

```

Screenshot Program :

```

Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 6
Masukkan nomor rumah kerabat untuk daerah ke-1: 5 2 1 7 9 13
Hasil urutan rumah untuk daerah ke-1: 1 5 7 9 13 2
Masukkan jumlah rumah kerabat untuk daerah ke-2: 7
Masukkan nomor rumah kerabat untuk daerah ke-2: 6 189 15 27 39 75 133
Hasil urutan rumah untuk daerah ke-2: 15 27 39 75 133 189 6
Masukkan jumlah rumah kerabat untuk daerah ke-3: 4
Masukkan nomor rumah kerabat untuk daerah ke-3: 3 4 9 1
Hasil urutan rumah untuk daerah ke-3: 1 3 9 4
PS C:\Users\ACER\Downloads\Semester 3\Laprak Alpro smstr3\Ilhan Sahal M

```

Deskripsi Program :

Program ini mengelompokkan nomor rumah kerabat menjadi dua kategori, yaitu ganjil dan genap, untuk setiap daerah yang dimasukkan oleh pengguna. Nomor rumah ganjil diurutkan dalam urutan membesar, sedangkan nomor rumah genap diurutkan dalam urutan mengecil. Pengguna diminta memasukkan jumlah daerah (n) dan jumlah rumah di setiap daerah (m), dengan validasi input agar nilainya berada dalam batas tertentu. Setelah nomor rumah dimasukkan, program mencetak hasil pengurutan sesuai kategori untuk setiap daerah.

2.

- 3) 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

Source Code :

```
package main

import (
    "fmt"
)

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
    }
}

func findMedian(arr []int) int {
    n := len(arr)

    // Jika jumlah elemen ganjil, ambil nilai tengah
    if n%2 == 1 {
        return arr[n/2]
    }

    // Jika jumlah elemen genap, ambil rerata kedua nilai tengah
    // (dibulatkan ke bawah)
    return (arr[(n/2)-1] + arr[n/2]) / 2
}

func main() {
    var data []int

    for {
        var num int
        fmt.Scan(&num)

        if num == -5313 { // Marker untuk menghentikan input
            break
        }

        if num == 0 { // Jika 0 ditemukan, hitung median
            // Urutkan data menggunakan insertion sort
            insertionSort(data)

            // Cari median dan cetak hasil
            median := findMedian(data)

```

```
        fmt.Println(median)
    } else {
        // Tambahkan data ke array
        data = append(data, num)
    }
}
}
```

Screenshot Program :

```
PS C:\Users\ACER\Downloads\Semester_2311102029_Modul11\unguided2.go"
7
23
11
0
11
5
19
2
29
3
13
17
0
12
```

Deskripsi Program :

Program ini menerima input angka secara berulang dan menghentikan input ketika angka -5313 dimasukkan. Setiap kali angka 0 dimasukkan, program akan menghitung median dari semua angka yang telah dimasukkan sebelumnya. Program menggunakan algoritma Insertion Sort untuk mengurutkan data sebelum menghitung median. Jika jumlah elemen ganjil, median adalah elemen tengah dari array yang sudah diurutkan, sedangkan jika jumlah elemen genap, median dihitung sebagai rata-rata dari dua elemen tengah (dibulatkan ke bawah). Program ini

memungkinkan pengguna untuk terus menambah data hingga angka -5313 dimasukkan sebagai tanda untuk berhenti.

3.

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.

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 [nMax]Buku

// Prosedur 1: Daftarkan Buku
func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan data buku ke-%d:\n", i+1)

        // Input masing-masing atribut buku
        fmt.Print("Masukkan id buku: ")
        fmt.Scan(&pustaka[i].id)

        fmt.Print("Masukkan judul buku: ")
        fmt.Scan(&pustaka[i].judul)

        fmt.Print("Masukkan penulis buku: ")
        fmt.Scan(&pustaka[i].penulis)

        fmt.Print("Masukkan penerbit buku: ")
        fmt.Scan(&pustaka[i].penerbit)

        fmt.Print("Masukkan jumlah eksemplar buku: ")
        fmt.Scan(&pustaka[i].eksemplar)

        fmt.Print("Masukkan tahun buku: ")
        fmt.Scan(&pustaka[i].tahun)

        fmt.Print("Masukkan rating buku: ")
        fmt.Scan(&pustaka[i].rating)
    }
}

```

```

// Prosedur 2: Cetak Terfavorit
func CetakTerfavorit(pustaka DaftarBuku, n int) {
    maxRating := -1
    var favorit Buku
    for i := 0; i < n; i++ {
        if pustaka[i].rating > maxRating {
            maxRating = pustaka[i].rating
            favorit = pustaka[i]
        }
    }
    fmt.Println("Buku terfavorit:")
    fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d\n",
        favorit.judul, favorit.penulis, favorit.penerbit, favorit.tahun)
}

// Prosedur 3: Urutkan Buku (Insertion Sort)
func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; 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
    }
}

// Prosedur 4: Cetak 5 Terbaru
func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    fmt.Println("5 Buku dengan rating tertinggi:")
    for i := 0; i < 5 && i < n; i++ {
        fmt.Printf("%d.  Judul:  %s,  Rating:  %d\n", i+1,
            pustaka[i].judul, pustaka[i].rating)
    }
}

// Prosedur 5: Cari Buku (Pencarian Biner)

```



```

func CariBuku(pustaka DaftarBuku, n int, r int) {
    low, high := 0, n-1
    for low <= high {
        mid := (low + high) / 2
        if pustaka[mid].rating == r {
            fmt.Printf("Buku dengan rating %d ditemukan:\n", r)
            fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun:
%d, Eksemplar: %d, Rating: %d\n",
                pustaka[mid].judul, pustaka[mid].penulis,
pustaka[mid].penerbit, pustaka[mid].tahun,
pustaka[mid].eksemplar, pustaka[mid].rating)
            return
        } else if 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, ratingCari int

    // Meminta input jumlah buku
    fmt.Print("Masukkan jumlah data buku: ")
    fmt.Scan(&n)

    // Pastikan n > 0 dan tidak melebihi batas maksimum
    if n <= 0 || n > nMax {
        fmt.Println("Jumlah buku harus lebih besar dari 0 dan kurang
dari 7919.")
        return
    }

    // Prosedur Daftarkan Buku
    DaftarkanBuku(&pustaka, n)

```

```
// Prosedur Cetak Terfavorit
CetakTerfavorit(pustaka, n)

// Prosedur Urutkan Buku
UrutBuku(&pustaka, n)

// Prosedur Cetak 5 Terbaru
Cetak5Terbaru(pustaka, n)

// Prosedur Cari Buku
fmt.Print("Masukkan rating buku yang ingin dicari: ")
fmt.Scan(&ratingCari)
CariBuku(pustaka, n, ratingCari)
}
```

Screenshot Program :

```
Masukkan jumlah data buku: 2
Masukkan data buku ke-1:
Masukkan id buku: 101
Masukkan judul buku: Kucing
Masukkan penulis buku: Ilhan
Masukkan penerbit buku: Gramedia
Masukkan jumlah eksemplar buku: 5
Masukkan tahun buku: 2024
Masukkan rating buku: 7
Masukkan data buku ke-2:
Masukkan id buku: 102
Masukkan judul buku: Ayam
Masukkan penulis buku: Padil
Masukkan penerbit buku: Gramedia
Masukkan jumlah eksemplar buku: 3
Masukkan tahun buku: 2021
Masukkan rating buku: 8
Buku terfavorit:
Judul: Ayam, Penulis: Padil, Penerbit: Gramedia, Tahun: 2021
5 Buku dengan rating tertinggi:
1. Judul: Ayam, Rating: 8
2. Judul: Kucing, Rating: 7
Masukkan rating buku yang ingin dicari: 
```

Deskripsi Program :

Program ini merupakan aplikasi manajemen daftar buku yang memungkinkan pengguna untuk memasukkan data buku, mencetak buku terfavorit, mengurutkan buku berdasarkan rating, menampilkan 5 buku dengan rating tertinggi, dan mencari buku berdasarkan rating tertentu. Pengguna dapat memasukkan jumlah buku yang akan didaftarkan, dan kemudian input data setiap buku secara terpisah, termasuk ID, judul, penulis, penerbit, jumlah eksemplar, tahun penerbitan, dan rating. Setelah itu, program akan menampilkan buku dengan rating tertinggi sebagai buku terfavorit, mengurutkan buku berdasarkan rating dari tertinggi ke terendah, dan mencetak 5 buku dengan rating tertinggi. Program juga dilengkapi dengan fitur pencarian untuk menemukan buku berdasarkan rating yang dimasukkan pengguna.