

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 12&13  
PENGURUTAN DATA**



**Disusun Oleh :**

**Egi Umar Ferdhika / 2311102277**

**11-IF-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 (sorting) adalah salah satu operasi dasar dalam ilmu komputer yang bertujuan untuk mengatur elemen-elemen dalam urutan tertentu, seperti urutan menaik (ascending) atau menurun (descending). Operasi ini sangat penting dalam banyak aplikasi, seperti pencarian data, pengelolaan basis data, dan optimisasi algoritma. Dalam bahasa pemrograman Go (GoLang), pengurutan data dapat dilakukan dengan berbagai algoritma yang memiliki karakteristik dan efisiensi berbeda. Beberapa algoritma pengurutan yang umum digunakan di antaranya adalah Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, Merge Sort, serta Binary Search untuk pencarian data setelah pengurutan.

Setelah data diurutkan, pencarian data dapat dilakukan dengan lebih efisien menggunakan Binary Search. Binary Search adalah algoritma pencarian yang mencari elemen dalam array yang sudah terurut dengan membagi array menjadi dua bagian secara berulang hingga elemen yang dicari ditemukan. Prinsip dasar Binary Search adalah membandingkan elemen tengah dengan elemen yang dicari. Jika elemen tengah lebih besar, pencarian dilanjutkan pada bagian kiri, dan jika lebih kecil, pencarian dilanjutkan pada bagian kanan. Proses ini diulang hingga elemen yang dicari ditemukan atau interval pencarian tidak ada lagi. Binary Search memiliki kompleksitas waktu  $O(\log n)$ , yang jauh lebih efisien dibandingkan pencarian linear (linear search) yang memiliki kompleksitas  $O(n)$ . Karena efisiensi ini, Binary Search sangat berguna setelah pengurutan data, karena pencarian elemen dalam array yang terurut dapat dilakukan dengan lebih cepat.

## II. GUIDED

1. Berisi source code dan output dari kegiatan praktikum yang telah dilaksanakan.

Source Code diberi penjelasan maka akan menjadi nilai ++

### 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()
    }
}

```

### Screenshot Output

```

Masukkan jumlah daerah kerabat (n): 2

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 3 6 5 5
Masukkan 3 nomor rumah kerabat: Nomor rumah terurut untuk daerah 1: 5 5 6

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 1
Masukkan 1 nomor rumah kerabat: 1
Nomor rumah terurut untuk daerah 2: 1

```

### Deskripsi Program

Kode diatas berfungsi untuk mengurutkan nomor rumah kerabat di beberapa daerah menggunakan algoritma Selection Sort. Pada fungsi main, program meminta input dari pengguna untuk jumlah daerah (n) yang akan diproses. Kemudian, untuk setiap daerah, program meminta jumlah nomor rumah kerabat dan kemudian menerima input nomor rumah tersebut.

Fungsi selectionSort digunakan untuk mengurutkan array nomor rumah yang diterima dari pengguna. Algoritma ini bekerja dengan cara mencari elemen terkecil dalam array yang belum terurut, lalu menukarnya dengan elemen di posisi yang sesuai. Proses ini diulang hingga seluruh array terurut. Setelah array terurut, nomor rumah yang telah disusun ditampilkan di layar.

## 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

```

```

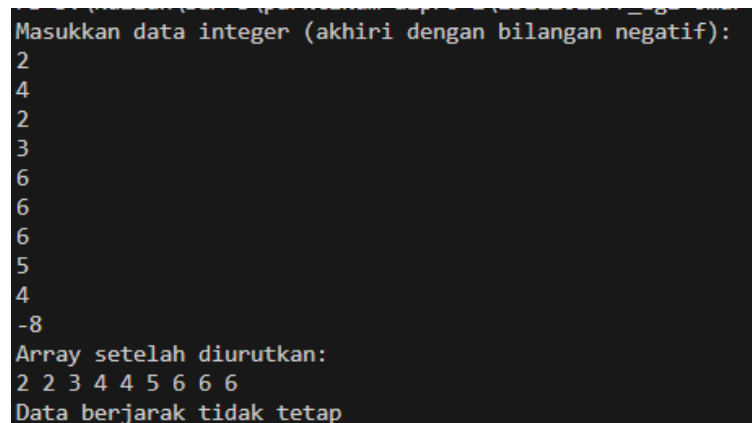
        isConstant, difference := isConstantDifference(arr,
n)

        // 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):
2
4
2
3
6
6
6
5
4
-8
Array setelah diurutkan:
2 2 3 4 4 5 6 6 6
Data berjarak tidak tetap

```

### Deskripsi Program

Kode diatas mengimplementasikan dua fungsi utama: Insertion Sort untuk mengurutkan array dan pemeriksaan apakah selisih antar elemen dalam array tetap konsisten setelah diurutkan. Dalam fungsi main, program meminta pengguna untuk memasukkan serangkaian angka integer hingga angka negatif dimasukkan, yang menandakan akhir input. Angka-angka tersebut kemudian disimpan dalam array dan diurutkan menggunakan algoritma Insertion Sort. Algoritma ini bekerja dengan cara memproses setiap elemen dari array, dimulai dari elemen kedua, dan menempatkannya pada posisi yang sesuai di bagian array yang sudah terurut dengan menggeser elemen yang lebih besar ke kanan. Setelah pengurutan selesai, program melanjutkan untuk memeriksa apakah selisih antara elemen-elemen yang terurut tetap sama, menggunakan fungsi `isConstantDifference`. Fungsi ini mengembalikan nilai `true` jika selisih antar elemen konsisten, bersama dengan nilai

selisihnya, atau false jika tidak. Hasil akhir berupa array yang telah diurutkan dan informasi tentang konsistensi selisih antar elemen kemudian ditampilkan kepada pengguna.

### III. UNGUIDED

1. Berisi source code dan output dari kegiatan praktikum yang telah dilaksanakan.

Source Code diberi penjelasan maka akan menjadi nilai ++

#### UNGUIDED 1

##### Sourcecode

```
package main

import "fmt"
// NAMA : Egi Umar Ferdhika
// NIM : 2311102277

func sortNumbers(arr []int, isAsc bool) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        extremeIdx := i
        for j := i + 1; j < n; j++ {
            if (isAsc && arr[j] < arr[extremeIdx]) ||
(!isAsc && arr[j] > arr[extremeIdx]) {
                extremeIdx = j
            }
        }
        arr[i], arr[extremeIdx] = arr[extremeIdx],
arr[i]
    }
}

func collectHouseNumbers(areaIndex, numHouses int)
([]int, []int) {
    var houseNumbers = make([]int, numHouses)
    fmt.Printf("Masukkan %d nomor rumah untuk daerah ke-
%d: ", numHouses, areaIndex+1)
    for i := 0; i < numHouses; i++ {
        fmt.Scan(&houseNumbers[i])
    }

    oddNumbers, evenNumbers :=
categorizeNumbers(houseNumbers)
    return oddNumbers, evenNumbers
}

func categorizeNumbers(houseNumbers []int) (odd []int,
even []int) {
    for _, num := range houseNumbers {
```

```

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

func displaySortedResults(areaIndex int, odd, even
[]int) {
    sortNumbers(odd, true)
    sortNumbers(even, false)

    fmt.Printf("\nNomor rumah terurut untuk daerah ke-
%d:\n", areaIndex+1)
    printArray(odd)
    printArray(even)
}

func printArray(arr []int) {
    for _, num := range arr {
        fmt.Printf("%d ", num)
    }
    fmt.Println()
}

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

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

        odd, even := collectHouseNumbers(areaIndex,
numHouses)

        displaySortedResults(areaIndex, odd, even)
    }
}

```



## Screenshoot Output

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

## Deskripsi Program

Kode diatas mengelola pengurutan nomor rumah kerabat di beberapa daerah dengan memisahkan nomor rumah menjadi dua kategori: angka ganjil dan genap, kemudian mengurutkan masing-masing kategori secara terpisah dengan cara yang berbeda. Program ini dimulai dengan meminta pengguna untuk memasukkan jumlah daerah, dan untuk setiap daerah, jumlah rumah kerabat yang ada. Setiap nomor rumah kemudian dimasukkan ke dalam array dan dipisahkan menjadi dua array terpisah berdasarkan apakah angka tersebut ganjil atau genap.

Fungsi `sortNumbers` digunakan untuk mengurutkan array berdasarkan parameter `isAsc` yang menentukan apakah pengurutan dilakukan dalam urutan menaik (ascending) atau menurun (descending). Untuk kategori angka ganjil, pengurutan dilakukan dalam urutan menaik, sedangkan untuk angka genap, pengurutan dilakukan dalam urutan menurun. Setelah nomor rumah dibagi dan diurutkan, fungsi `displaySortedResults` digunakan untuk menampilkan hasil pengurutan dengan memanggil fungsi `printArray` untuk menampilkan angka ganjil dan genap.

## UNGUIDED 2

### Sourcecode

```
package main
```

```

import (
    "fmt"
    "sort"
    // NAMA : Egi Umar Ferdhika
    // NIM : 2311102277
)

func main() {
    var number int
    var numbers []int

    fmt.Println("Masukkan bilangan (akhiri dengan -5313):")
    for {
        fmt.Scan(&number)

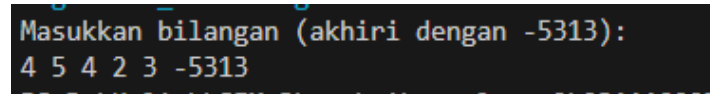
        if number == -5313 {
            break
        }

        if number == 0 {
            if len(numbers) > 0 {
                sort.Ints(numbers)
                median := findMedian(numbers)
                fmt.Printf("Median: %d\n", median)
            }
        } else {
            numbers = append(numbers, number)
        }
    }

    func findMedian(numbers []int) int {
        length := len(numbers)
        if length%2 == 1 {
            return numbers[length/2]
        }
        return (numbers[(length/2)-1] + numbers[length/2]) /
2
    }
}

```

### Screenshoot Output



```

Masukkan bilangan (akhiri dengan -5313):
4 5 4 2 3 -5313

```

### Deskripsi Program

Kode diatas bertujuan untuk mengumpulkan angka yang dimasukkan oleh pengguna, kemudian menghitung dan menampilkan median dari angka-angka tersebut setiap kali pengguna memasukkan angka 0. Pengguna terus diminta untuk

memasukkan angka hingga mereka memasukkan angka -5313, yang menandakan akhir input.

Di dalam fungsi main, program pertama-tama meminta pengguna untuk memasukkan bilangan, kemudian memeriksa apakah angka yang dimasukkan adalah 0. Jika angka yang dimasukkan adalah 0, program akan mengurutkan array numbers menggunakan fungsi sort.Ints dari paket sort dan menghitung median menggunakan fungsi findMedian. Median dihitung berdasarkan panjang array: jika panjangnya ganjil, median adalah elemen di tengah array yang sudah diurutkan, sedangkan jika panjangnya genap, median adalah rata-rata dua elemen tengah.

Jika angka yang dimasukkan bukan 0 atau -5313, angka tersebut akan ditambahkan ke dalam array numbers. Proses ini berulang hingga pengguna memasukkan angka -5313, yang menghentikan input dan menyelesaikan eksekusi program.

### UNGUIDED 3

#### Sourcecode

```
package main

import (
    "fmt"
    "sort"
    // NAMA : Egi Umar Ferdhika
    // NIM : 2311102277
)

type DaftarBuku struct {
    nBuku int
    buku []DataBuku
}

type DataBuku struct {
    id        int
    judul     string
    pengarang string
    tahun     int
    rating    int
}

// DaftarkanBuku initializes a new DaftarBuku with
// specified size
// I.S: sejumlah n data buku telah siap para piranti
// masukan
// F.S: n berisi sebuah nilai, dan pustaka berisi
// sejumlah n data buku
func DaftarkanBuku(n int) *DaftarBuku {
```

```

        return &DaftarBuku{
            nBuku: 0,
            buku:  make([]DataBuku, n),
        }
    }

    // CetakTerfavorit prints book data sorted by rating
    // I.S: array pustaka berisi n buah data buku dan belum
    terurut
    // F.S: Tampilan data buku (judul, penulis, penerbit,
    tahun) terfavorit
    func CetakTerfavorit(pustaka *DaftarBuku, n int) {
        if n == 0 || pustaka.nBuku == 0 {
            return
        }

        // Create temporary slice for sorting
        temp := make([]DataBuku, pustaka.nBuku)
        copy(temp, pustaka.buku[:pustaka.nBuku])

        // Sort by rating (highest first)
        sort.Slice(temp, func(i, j int) bool {
            return temp[i].rating > temp[j].rating
        })

        fmt.Println("Buku Terfavorit:")
        fmt.Printf("Judul: %s, Pengarang: %s, Tahun: %d,
        Rating: %d\n",
            temp[0].judul, temp[0].pengarang, temp[0].tahun,
            temp[0].rating)
    }

    // UrutBuku sorts books by rating using insertion sort
    // I.S: Array pustaka berisi n data buku
    // F.S: Array pustaka terurut menurun/mengecil terhadap
    rating
    func UrutBuku(pustaka *DaftarBuku, n int) {
        for i := 1; i < pustaka.nBuku; i++ {
            key := pustaka.buku[i]
            j := i - 1

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

    // Cetak5Terbaru prints the 5 books with highest rating
    // I.S: pustaka berisi n data buku yang sudah terurut
    menurut rating
    // F.S: Laporan 5 judul buku dengan rating tertinggi
    func Cetak5Terbaru(pustaka *DaftarBuku, n int) {
        fmt.Println("5 Buku Rating Tertinggi:")
    }

```

```

    limit := 5
    if pustaka.nBuku < 5 {
        limit = pustaka.nBuku
    }

    for i := 0; i < limit; i++ {
        fmt.Printf("%d. Judul: %s, Rating: %d\n",
            i+1, pustaka.buku[i].judul,
            pustaka.buku[i].rating)
    }
}

// CariBuku searches for books with specific rating
using binary search
// I.S: pustaka berisi n data buku yang sudah terurut
menurut rating
// F.S: Laporan salah satu buku dengan rating yang
diberikan
func CariBuku(pustaka *DaftarBuku, n int, r int) {
    left := 0
    right := pustaka.nBuku - 1
    found := false

    for left <= right {
        mid := (left + right) / 2
        if pustaka.buku[mid].rating == r {
            fmt.Printf("Buku ditemukan:\nJudul:
%s\nPengarang: %s\nTahun: %d\nRating: %d\n",
                pustaka.buku[mid].judul,
                pustaka.buku[mid].pengarang,
                pustaka.buku[mid].tahun,
                pustaka.buku[mid].rating)
            found = true
            break
        }
        if pustaka.buku[mid].rating < r {
            right = mid - 1
        } else {
            left = mid + 1
        }
    }

    if !found {
        fmt.Printf("Tidak ada buku dengan rating %d\n",
r)
    }
}

func main() {
    perpustakaan := DaftarkanBuku(10)

    perpustakaan.buku[0] = DataBuku{1, "Go Programming",
"John Doe", 2020, 5}
    perpustakaan.buku[1] = DataBuku{2, "Database
Design", "Jane Smith", 2019, 4}

```

```

        perpustakaan.buku[2] = DataBuku{3, "Web
Development", "John Doe", 2021, 3}
        perpustakaan.buku[3] = DataBuku{4, "AI Basics",
"Alice Johnson", 2022, 5}
        perpustakaan.nBuku = 4

        fmt.Println("Original books:")
        for i := 0; i < perpustakaan.nBuku; i++ {
            fmt.Printf("%+v\n", perpustakaan.buku[i])
        }

        fmt.Println("\nTesting CetakTerfavorit:")
        CetakTerfavorit(perpustakaan, perpustakaan.nBuku)

        fmt.Println("\nTesting UrutBuku:")
        UrutBuku(perpustakaan, perpustakaan.nBuku)

        fmt.Println("\nTesting Cetak5Terbaru:")
        Cetak5Terbaru(perpustakaan, perpustakaan.nBuku)

        fmt.Println("\nTesting CariBuku with rating 4:")
        CariBuku(perpustakaan, perpustakaan.nBuku, 4)
    }

```

## Screenshoot Output

```

Original books:
{id:1 judul:Go Programming pengarang:John Doe tahun:2020 rating:5}
{id:2 judul:Database Design pengarang:Jane Smith tahun:2019 rating:4}
{id:3 judul:Web Development pengarang:John Doe tahun:2021 rating:3}
{id:4 judul:AI Basics pengarang:Alice Johnson tahun:2022 rating:5}

Testing CetakTerfavorit:
Buku Terfavorit:
Judul: Go Programming, Pengarang: John Doe, Tahun: 2020, Rating: 5

Testing UrutBuku:

Testing Cetak5Terbaru:
5 Buku Rating Tertinggi:
1. Judul: Go Programming, Rating: 5
2. Judul: AI Basics, Rating: 5
3. Judul: Database Design, Rating: 4
4. Judul: Web Development, Rating: 3

Testing CariBuku with rating 4:
Buku ditemukan:
Judul: Database Design
Pengarang: Jane Smith
Tahun: 2019
Rating: 4

```

## Deskripsi Program

Kode diatas dirancang untuk mengelola sebuah perpustakaan dengan struktur data yang menyimpan informasi buku, termasuk judul, pengarang, tahun terbit, dan rating. Program ini memungkinkan beberapa operasi seperti mencari buku berdasarkan rating, mengurutkan buku berdasarkan rating, menampilkan buku

dengan rating tertinggi, dan menampilkan lima buku teratas. Ada dua struktur data utama: DaftarBuku, yang menyimpan jumlah buku dan array buku, dan DataBuku, yang menyimpan detail buku individu. Beberapa fungsi penting disertakan, seperti DaftarkanBuku untuk menginisialisasi perpustakaan, CetakTerfavorit untuk menampilkan buku dengan rating tertinggi setelah pengurutan, UrutBuku yang mengurutkan buku berdasarkan rating dengan Insertion Sort, Cetak5Terbaru untuk menampilkan lima buku dengan rating tertinggi, dan CariBuku yang menggunakan Binary Search untuk mencari buku dengan rating tertentu. Di dalam fungsi main, program menginisialisasi perpustakaan, menambahkan buku contoh, dan kemudian menguji berbagai fungsi yang ada, seperti pengurutan dan pencarian buku.