

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL XII

PENGURUTAN DATA



Disusun Oleh :

Haposan Felix Marcel Siregar / 2311102210

S1IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

- **Selection Sort:** Algoritma pengurutan yang bekerja dengan cara menemukan elemen terkecil (atau terbesar) dari array dan menukarnya dengan elemen pertama, kemudian mengulangi proses ini untuk elemen kedua, ketiga, dan seterusnya hingga seluruh array terurut.
- **Insertion Sort:** Algoritma pengurutan yang bekerja dengan cara membangun array yang terurut satu per satu. Setiap elemen baru dimasukkan ke posisi yang tepat dalam array yang sudah terurut sebelumnya.
- **Median:** Nilai tengah dari suatu kumpulan data yang telah diurutkan. Jika jumlah data ganjil, median adalah elemen tengah. Jika jumlah data genap, median adalah rata-rata dari dua elemen tengah.
- **Rekursi:** Teknik pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan masalah yang lebih kecil dari masalah asli. Rekursi sering digunakan untuk menyelesaikan masalah yang dapat dipecah menjadi sub-masalah yang lebih kecil dan serupa.
- **Binary Search:** Algoritma pencarian yang efisien untuk menemukan posisi elemen dalam array yang sudah diurutkan. Algoritma ini bekerja dengan cara membagi array menjadi dua bagian dan membandingkan elemen tengah dengan elemen yang dicari, kemudian mengulangi proses ini pada bagian yang relevan hingga elemen ditemukan atau seluruh array telah diperiksa.
- **Struct:** Tipe data di Go yang digunakan untuk mengelompokkan beberapa variabel yang berbeda jenis menjadi satu kesatuan. Struct sering digunakan untuk merepresentasikan objek dengan berbagai atribut, seperti buku dengan atribut id, judul, penulis, penerbit, eksemplar, tahun, dan rating.

II. GUIDED

1.

Soal Guided

Program ini akan mengurutkan nomor rumah di setiap daerah dari yang terkecil hingga terbesar.

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 identitas() {
    fmt.Println("=====")
    fmt.Println("Nama :Haposan Felix Marcel Siregar")
    fmt.Println("NIM:2311102210")
    fmt.Println("=====")
}

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

    // Proses tiap daerah
```

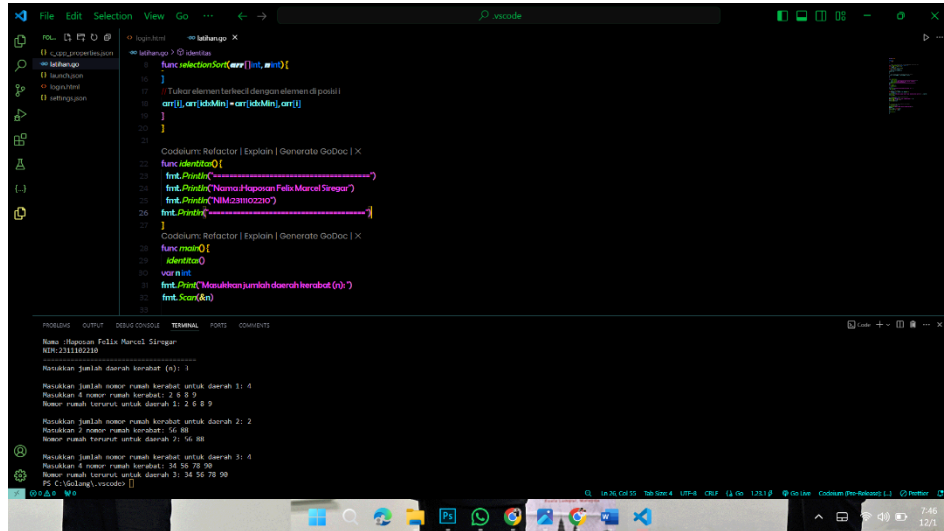
```

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



```
func selectionSort(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        // Tukar elemen terkecil dengan elemen di posisi i
        var min = i
        for j := i+1; j < n; j++ {
            if arr[j] < arr[min] {
                min = j
            }
        }
        // Tukar elemen terkecil dengan elemen di posisi i
        arr[i], arr[min] = arr[min], arr[i]
    }
}

func main() {
    // Masukkan jumlah daerah kerabat (n)
    n := 3
    // Masukkan jumlah nomor rumah kerabat untuk daerah 1: 4
    // Masukkan 4 nomor rumah kerabat: 2 8 8 9
    // Masukkan jumlah nomor rumah kerabat untuk daerah 2: 2
    // Masukkan 2 nomor rumah kerabat: 56 88
    // Masukkan jumlah nomor rumah kerabat untuk daerah 3: 4
    // Masukkan 4 nomor rumah kerabat: 34 56 78 90
    arr := []int{2, 8, 8, 9, 56, 88, 34, 56, 78, 90}
    selectionSort(arr, n)
    for i := 0; i < n; i++ {
        // Cetak nomor rumah kerabat untuk daerah i
        for j := 0; j < arr[i]; j++ {
            fmt.Print(arr[j], " ")
        }
        fmt.Println()
    }
}
```

Output:

```
Memasukkan jumlah daerah kerabat (n): 3
Memasukkan jumlah nomor rumah kerabat untuk daerah 1: 4
Memasukkan 4 nomor rumah kerabat: 2 8 8 9
Nomor rumah kerabat untuk daerah 1: 2 8 8 9
Memasukkan jumlah nomor rumah kerabat untuk daerah 2: 2
Memasukkan 2 nomor rumah kerabat: 56 88
Nomor rumah kerabat untuk daerah 2: 56 88
Memasukkan jumlah nomor rumah kerabat untuk daerah 3: 4
Memasukkan 4 nomor rumah kerabat: 34 56 78 90
Nomor rumah kerabat untuk daerah 3: 34 56 78 90
PS C:\Users\user> go run *.go
```

Deskripsi Program

- Program meminta pengguna memasukkan jumlah daerah kerabat (n).
- Untuk setiap daerah, program meminta jumlah nomor rumah (m) dan nomor rumah itu sendiri.
- Nomor rumah disimpan dalam array dan diurutkan menggunakan algoritma Selection Sort.
- Setelah diurutkan, program mencetak nomor rumah yang telah terurut untuk setiap daerah.

2.

Soal Guided

Program ini akan mengurutkan data tersebut dan memeriksa apakah selisih antara elemen-elemen yang diurutkan tetap.

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
    }
}
```

```

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 identitas() {
fmt.Println("=====")
fmt.Println("Nama :Haposan Felix Marcel Siregar")
fmt.Println("NIM:2311102210")
fmt.Println("=====")
}

func main() {
    identitas()

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

```

File Edit Selection View Go ...
latihan.go
func insertionSort(arr []int, nint) {
    arr[0] = hey
}
// Fungsi untuk memeriksa apakah selisih elemen array tetap
// Codium: Refactor | Explain | X
func isConstantDifference(arr []int, nint) (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
}

```

```

PS C:\Golang\vscode> go run "C:\Golang\vscode\latihan.go"
=====
Nama : Iqbalan Felix Marcel Siregar
NIM : 2311102719
=====
Masukkan data bilangan (akhiri dengan bilangan negatif):
5 6 9 1 14 31 22 7
Array setelah diurutkan:
1 5 6 9 14 22 31 51
Data berjarak tidak tetap
PS C:\Golang\vscode>

```

Deskripsi Program

- ☐ Program meminta pengguna memasukkan data integer hingga bilangan negatif ditemukan.
- ☐ Data yang dimasukkan disimpan dalam array.
- ☐ Array diurutkan menggunakan algoritma Insertion Sort.
- ☐ Program memeriksa apakah selisih antara elemen-elemen yang diurutkan tetap.
- ☐ Hasil pengurutan dan status jarak elemen ditampilkan.

III. UNGUIDED

1.

Soal Unguided

Program ini akan mengurutkan nomor rumah di setiap daerah dari yang terkecil hingga terbesar, dengan aturan khusus untuk nomor ganjil dan genap.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Selection Sort
func selectionSortKunjungan(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        idxEx := i
        for j := i + 1; j < n; j++ {
            // Jika keduanya ganjil, pilih yang lebih kecil
            if arr[j]%2 == 1 && arr[idxEx]%2 == 1 {
                if arr[j] < arr[idxEx] {
                    idxEx = j
                }
            } else if arr[j]%2 == 1 { // Jika arr[j] ganjil dan arr[idxEx]
bukan ganjil
                idxEx = j
            } else if arr[j]%2 == 0 && arr[idxEx]%2 == 0 { // Jika
keduanya genap, pilih yang lebih besar
                if arr[j] > arr[idxEx] {
                    idxEx = j
                }
            }
        }
        // Tukar elemen di posisi i dengan elemen terkecil/terbesar
        arr[i], arr[idxEx] = arr[idxEx], arr[i]
    }
}

func identitas() {
    fmt.Println("=====")
}
```

```

    fmt.Println("Nama :Haposan Felix Marcel Siregar")
    fmt.Println("NIM:2311102210")
    fmt.Println("=====")
}

func main() {
    identitas()
    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
        selectionSortKunjungan(arr, m)

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

```

Screenshoot Output

```
func selectionSortKunjungan(arr []int, n int) {
    // ...
    fmt.Println("Nama Hapusan Felix Marcel Siregar")
    fmt.Println("NIM 23022007")
    // ...
}
```

```
PS C:\Golang\vscode> go run "C:\Golang\vscode\latihan-go"
=====
Nama Hapusan Felix Marcel Siregar
NIM 23022007

Masukkan jumlah daerah kerabat (n): 4
Masukkan jumlah nomor rumah kerabat untuk daerah 1: 2
Masukkan 2 nomor rumah kerabat: 3 6
Nomor rumah terurut untuk dikunjungi di daerah 1: 3 6

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 4
Masukkan 4 nomor rumah kerabat: 11 22 25 37
Nomor rumah terurut untuk dikunjungi di daerah 2: 11 25 37 22

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 3
Masukkan 3 nomor rumah kerabat: 2 6 9
Nomor rumah terurut untuk dikunjungi di daerah 3: 9 6 2

Masukkan jumlah nomor rumah kerabat untuk daerah 4: 8
Masukkan 8 nomor rumah kerabat: 2 8 6 10 3 11 9 7
Nomor rumah terurut untuk dikunjungi di daerah 4: 3 7 9 11 10 8 6 2
PS C:\Golang\vscode>
```

Deskripsi Program

Program ini mengurutkan nomor rumah kerabat berdasarkan aturan khusus. Pengguna memasukkan jumlah daerah kerabat dan nomor rumah untuk setiap daerah. Nomor rumah diurutkan menggunakan Selection Sort dengan aturan: nomor ganjil diurutkan dari yang terkecil, dan nomor genap diurutkan dari yang terbesar. Setelah diurutkan, program mencetak nomor rumah yang telah terurut untuk setiap daerah. Hasil pengurutan ditampilkan untuk setiap daerah. Program ini membantu mengatur kunjungan ke rumah kerabat dengan lebih efisien.

2.

Soal Unguided

Program ini mengurutkan array menggunakan Selection Sort dan menghitung median dari data yang dimasukkan.

Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

// 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]
}
}

// Fungsi untuk menghitung median
func median(arr []int, n int) float64 {
    if n%2 == 0 {
        return float64(arr[n/2-1]+arr[n/2]) / 2.0
    }
    return float64(arr[n/2])
}

func identitas() {
    fmt.Println("=====")
    fmt.Println("Nama :Haposan Felix Marcel Siregar")
    fmt.Println("NIM:2311102210")
    fmt.Println("=====")
}

func main() {
    identitas()

    // Inisialisasi array dan variabel
    arr := make([]int, 0, 1000000)
    arrMedian := make([]float64, 0, 1000000)
    datum := 0

    fmt.Println("Masukkan data (akhiri dengan -5313):")

    // Input data hingga ditemukan -5313
    for {
        fmt.Scan(&datum)
        if datum == -5313 {
            break
        }

        if datum == 0 {

```

```

// Jika data = 0, hitung median dari array
sort.Ints(arr) // Menggunakan sort untuk pengurutan
arrMedian = append(arrMedian, median(arr, len(arr)))
} else {
// Tambahkan data ke array
arr = append(arr, datum)
}
}

// Output hasil
fmt.Println("Output:")
for _, med := range arrMedian {
    fmt.Printf("%0.2f\n", med)
}
}

```

Screenshoot Output

```

// Screenshoot Output
// Program Output:
// Output:
// 9.00

```

Deskripsi Program

Program di atas memuat algoritma yang berfungsi untuk mencetak median dari sekumpulan data yang muncul sebelum setiap angka 0 seperti pada contoh keluaran di atas. Program akan berakhir jika masukan berupa angka -5313.

3.

Soal Unguided

Program ini membantu pengguna dalam mengelola dan menemukan buku berdasarkan rating dengan lebih efisien.

Sourcecode

```
package main

import (
    "fmt"
)

const nMax int = 7919

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

type DaftarBuku [nMax]Buku

func DaftarkanBuku(daftar_buku *DaftarBuku, n int) {
    fmt.Println("Masukkan data buku di setiap baris (id judul\npenulis penerbit eksemplar tahun rating):")
    for i := 0; i < n; i++ {
        fmt.Scan(&daftar_buku[i].id, &daftar_buku[i].judul,
            &daftar_buku[i].penulis, &daftar_buku[i].penerbit,
            &daftar_buku[i].eksemplar, &daftar_buku[i].tahun,
            &daftar_buku[i].rating)
    }
}

func CetakTerfavorit(daftar_buku DaftarBuku, n int) {
    var buku_terfavorit Buku = daftar_buku[0]
    for i := 1; i < n; i++ {
        if buku_terfavorit.rating < daftar_buku[i].rating {
            buku_terfavorit = daftar_buku[i]
        }
    }
    fmt.Println("Buku terfavorit adalah buku dengan judul:",
        buku_terfavorit.judul)
```

```

}

func UrutBukuBerdasarkanRating(daftar_buku *DaftarBuku, n
int) {
    for i := 0; i < n-1; i++ {
        idxMax := i
        for j := i + 1; j < n; j++ {
            if daftar_buku[j].rating > daftar_buku[idxMax].rating {
                idxMax = j
            }
        }
        daftar_buku[i], daftar_buku[idxMax] =
daftar_buku[idxMax], daftar_buku[i]
    }
}

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

func CariBuku(daftar_buku DaftarBuku, n, rating int) {
    ditemukan := false
    bottom, top := 0, n-1

    for bottom <= top {
        middle := (bottom + top) / 2
        if daftar_buku[middle].rating == rating {
            ditemukan = true
            fmt.Println("Buku dengan rating", rating, ":")
            fmt.Println(daftar_buku[middle].judul)

            // Cek buku-buku di sekitarnya dengan rating yang sama
            for i := middle - 1; i >= 0 && daftar_buku[i].rating == rating,
i-- {
                fmt.Println(daftar_buku[i].judul)
            }
        }
    }
}

```

```

        for i := middle + 1; i < n && daftar_buku[i].rating == rating;
i++ {
            fmt.Println(daftar_buku[i].judul)
        }
        break
    } else if daftar_buku[middle].rating < rating {
        bottom = middle + 1
    } else {
        top = middle - 1
    }
}

if !ditemukan {
    fmt.Println("Buku dengan rating", rating, "tidak ditemukan")
}
}

func identitas() {
    fmt.Println("=====")
    fmt.Println("Nama :Haposan Felix Marcel Siregar")
    fmt.Println("NIM:2311102210")
    fmt.Println("=====")
}

func main() {
    identitas()

    var daftar_buku DaftarBuku
    var n int

    fmt.Print("Masukkan jumlah buku (n): ")
    fmt.Scan(&n)

    DaftarkanBuku(&daftar_buku, n)
    UrutBukuBerdasarkanRating(&daftar_buku, n)

    fmt.Println()
    CetakTerfavorit(daftar_buku, n)
    fmt.Println()
    Cetak5Terbaru(daftar_buku, n)

```



```

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

```

Screenshot Output

```

PS C:\Golang\vscode> go run ".\main.go"
=====
Nama : Hapsan Felix Marvel Siregar
IDN: 21111802218

Masukkan jumlah buku (n): 3
Masukkan data buku di setiap baris (id judul penulis penerbit eksemplar tahun rating):
1 Buku1 Penulis1 Penerbit1 10 2020 3
2 Buku2 Penulis2 Penerbit2 15 2024 9
3 Buku3 Penulis3 Penerbit3 5 2013 7

Buku terfavorit adalah buku dengan judul: Buku3

3 Buku dengan rating tertinggi:
1. Buku3, rating: 9
2. Buku2, rating: 7
3. Buku1, rating: 3
Masukkan rating buku yang dicari: 7
Buku3
PS C:\Golang\vscode>

```

Deskripsi Program

Program ini mengelola daftar buku dengan berbagai fungsi seperti pendaftaran, pengurutan berdasarkan rating, pencarian buku berdasarkan rating, dan pencetakan buku terfavorit serta lima buku dengan rating tertinggi. Pengguna memasukkan jumlah buku dan data setiap buku, termasuk id, judul, penulis, penerbit, eksemplar, tahun, dan rating. Program kemudian mengurutkan buku berdasarkan rating tertinggi, mencetak buku terfavorit, dan lima buku dengan rating tertinggi. Pengguna juga dapat mencari buku berdasarkan rating tertentu, dan program akan mencetak buku-buku yang memiliki rating tersebut.