LAPORAN PRAKTIKUM

ALGORITMA DAN PEMOGRAMAN II

MODUL XI

PENGURUTAN ARRAY

Oleh:

NAMA : DWI HESTI ARIANI

NIM : 2311102094

KELAS : 11- IF -02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan array ialah Proses yang melibatkan pengaturan elemen-elemen dalam array ke dalam urutan tertentu, baik itu secara menaik maupun menurun. Pengurutan tidak hanya penting untuk keperluan penyimpanan data yang teratur, tetapi juga untuk meningkatkan efisiensi dalam pencarian dan pengolahan data lebih lanjut. Pengurutan (sorting) adalah proses yang memungkinkan data diatur dalam urutan tertentu. Ini dapat dilakukan menggunakan berbagai algoritma yang memiliki karakteristik dan kompleksitas yang berbeda.

Algoritma pengurutan yang umum digunakan yaitu Bubble Sort, Selection Sort, Quick Sort, Insertion Sort, dan Merge Sort. Masing-masing algoritma ini memiliki cara kerja dan efisiensi yang berbeda, sehingga pemilihan algoritma yang tepat sangat bergantung pada konteks penggunaan dan jenis data yang akan diurutkan.

Algoritma Pengurutan

- **Selection sort :** Algoritma ini memilih elemen terkecil dari array dan menukarnya dengan elemen pertama, kemudian melanjutkan proses ini untuk elemen berikutnya hingga seluruh array terurut.

	Notasi Algoritma	Notasi dalam bahasa Go
1	`i ← 1`	`i = 1`
2	`while i <= n-1 do`	`for i <= n-1 {`
3	`    idx_min ← i - 1`	`    idx_min = i - 1`
4	`    j ← i`	`    j = i`
5	`    while j < n do`	`    for j < n {`
6	`        if a[idx_min] > a[j] then`	`        if a[idx_min] > a[j] {`
7	`            idx_min ← j`	`            idx_min = j`
8	`        endif`	`        }`
9	`        j ← j + 1`	`        j = j + 1`
10	`    endwhile`	`    }`
11	`    t ← a[idx_min]`	`    t = a[idx_min]`
12	`    a[idx_min] ← a[i-1]`	`    a[idx_min] = a[i-1]`
13	`    a[i-1] ← t`	`    a[i-1] = t`
14	`    i ← i + 1`	`    i = i + 1`
15	`endwhile`	`}`

- **Insertion sort** : menyisipkan suatu nilai pada posisi yang seharusnya. Berbeda dengan pengurutan seleksi, yang mana pada pengurutan ini tidak dilakukan pencarian nilai ekstrim terlebih dahulu, cukup memilih suatu nilai tertentu kemudian mencari posisinya secara *sequential search.*

	Notasi Algoritma	Notasi dalam bahasa Go
1	`i ← 1`	`i = 1`
2	`while i <= n-1 do`	`for i <= n-1 {`
3	`    j ← i`	`    j = i`
4	`    temp ← a[j]`	`    temp = a[j]`
5	`    while j > 0 and temp > a[j-1] do`	`    for j > 0 && temp > a[j-1] {`
6	`        a[j] ← a[j-1]`	`        a[j] = a[j-1]`
7	`        j ← j - 1`	`        j = j - 1`
8	`    endwhile`	`    }`
9	`    a[j] ← temp`	`    a[j] = temp`
10	`    i ← i + 1`	`    i = i + 1`
11	`endwhile`	`}`

II. GUIDED

Guided 1

Source Code

```
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()
    }
}
```

Output Program

```
PS C:\Users\dwihe\OneDrive\Documents\SEMESTER 3\PRAKTIKUM ALPRO2\MODUL 11> go
PRO2\MODUL 11\guided1.go"
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: 13 9 7 5 2 1
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: 189 133 75 39 27 15 6
Masukkan jumlah rumah kerabat untuk daerah ke-3: 4
Masukkan nomor rumah kerabat untuk daerah ke-3: 3 4 1 9
Hasil urutan rumah untuk daerah ke-3: 9 4 3 1
PS C:\Users\dwihe\OneDrive\Documents\SEMESTER 3\PRAKTIKUM ALPRO2\MODUL 11>
```

Deskripsi Program

Program diatas ialah program sederhana dalam bahasa Go yang digunakan untuk membaca nomor rumah dari beberapa daerah, mengurutkannya dalam urutan menurun (descending) menggunakan algoritma Selection Sort, dan mencetak hasil urutannya untuk setiap daerah. Program juga memvalidasi input jumlah daerah (n) dan jumlah rumah di setiap daerah (m) agar sesuai dengan

batas yang ditentukan. Input nomor rumah dimasukkan dalam array, diurutkan, dan hasilnya ditampilkan untuk masing-masing daerah.

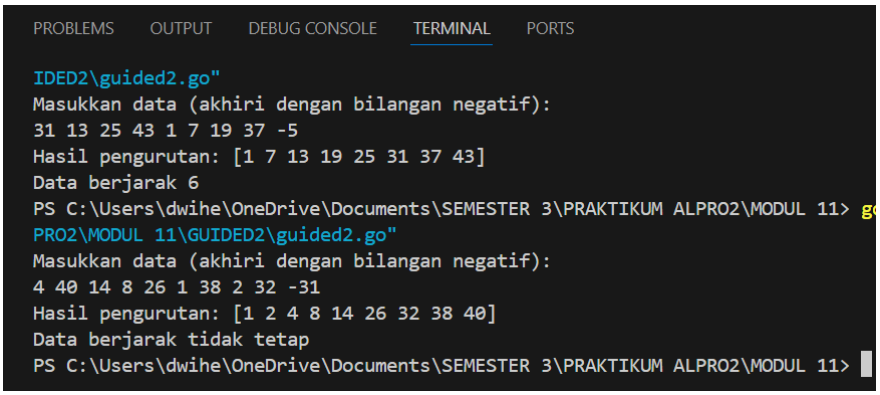
Guided 2

Source Code

```
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")
    }
}
```

Output Program



Deskripsi Program

Program ini merupakan program sederhana Bahasa Golang yang melakukan dua tugas utama mengurutkan data yang dimasukkan oleh pengguna dan memeriksa apakah data tersebut berjarak tetap. Data diurutkan menggunakan algoritma **Insertion Sort**. Setelah itu, program memeriksa apakah selisih antar elemen dalam data tersebut **konsisten (berjarak tetap)**. Jika konsisten, program mencetak jarak tetap tersebut; jika tidak, program mencetak bahwa data tidak berjarak tetap. Hasil pengurutan juga ditampilkan.

III. UNGUIDED

Source Code

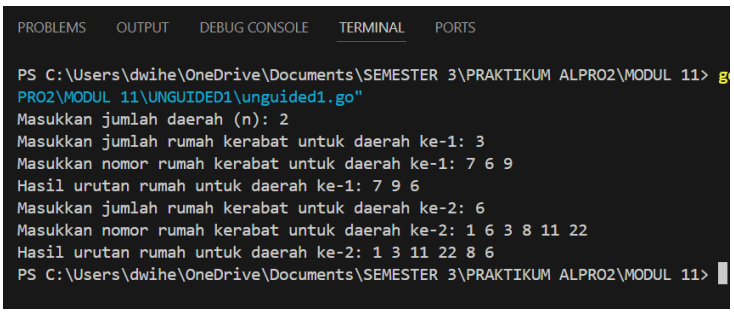
Unguided 1

```
package main

import "fmt"

func selectionSort(arr []int, ascending bool) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        targetIdx := i
        for j := i + 1; j < n; j++ {
            if (ascending && arr[j] < arr[targetIdx]) ||
(!ascending && arr[j] > arr[targetIdx]) {
                targetIdx = j
```

```
            }
        }
        arr[i], arr[targetIdx] = arr[targetIdx], arr[i]
    }
}

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
        }

        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])
        }

        var oddHouses []int
        var evenHouses []int

        for _, house := range houses {
            if house%2 == 1 {
                oddHouses = append(oddHouses, house) // Nomor
ganjil
            } else {
```

```
                evenHouses = append(evenHouses, house) // Nomor
genap
            }
        }

        // Urutkan nomor rumah ganjil membesar
        selectionSort(oddHouses, true)

        // Urutkan nomor rumah genap mengecil
        selectionSort(evenHouses, false)

        // Cetak hasil
        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ",
i+1)

        // Cetak nomor rumah ganjil
        for _, house := range oddHouses {
            fmt.Printf("%d ", house)
        }

        // Cetak nomor rumah genap
        for _, house := range evenHouses {
            fmt.Printf("%d ", house)
        }

        fmt.Println()
    }
}
```

Output Program



Deskripsi Program

Program diatas ialah program sederhana dalam bahasa Go yang berfungsi mengurutkan nomor rumah kerabat berdasarkan kriteria tertentu. Program akan meminta pengguna untuk memasukkan jumlah daerah dan nomor rumah kerabat di setiap daerah. Nomor rumah dipisahkan menjadi dua kelompok :

- Nomor ganjil, yang akan diurutkan secara naik (ascending).
- Nomor genap, yang akan diurutkan secara turun (descending).

Program menggunakan algoritma Selection sort untuk masing-masing kelompok untuk mengurutkan nomor rumah .

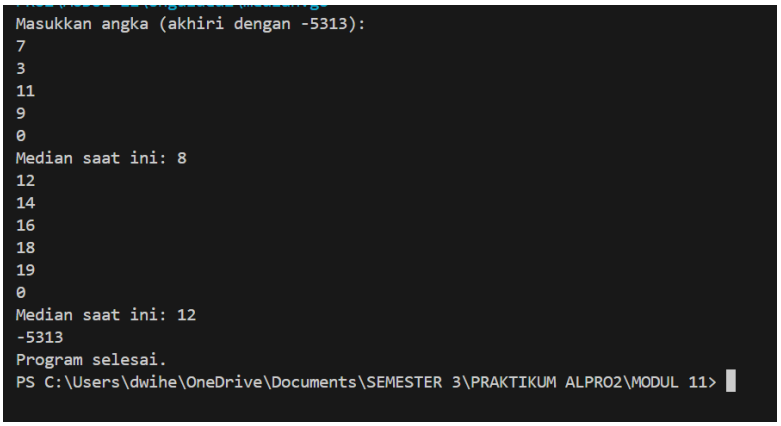
Unguided 2

Source Code

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "sort"
    "strconv"
)

func main() {
    var numbers []int
    scanner := bufio.NewScanner(os.Stdin)

    fmt.Println("Masukkan angka (akhiri dengan -5313):")
    for {
        scanner.Scan()
        input := scanner.Text()

        // Konversi input menjadi angka
        num, err := strconv.Atoi(input)
        if err != nil {
            fmt.Println("Masukkan angka valid.")
```

```
            continue
        }

        // Jika input adalah -5313, hentikan program
        if num == -5313 {
            break
        }

        // Jika input adalah 0, hitung dan cetak median
        if num == 0 {
            if len(numbers) == 0 {
                fmt.Println("Belum ada angka untuk dihitung
median.")
                continue
            }
            // Urutkan data
            sort.Ints(numbers)

            // Hitung median
            mid := len(numbers) / 2
            var median int
            if len(numbers)%2 == 0 {
                median = (numbers[mid-1] + numbers[mid]) / 2
            } else {
                median = numbers[mid]
            }
            fmt.Println("Median saat ini:", median)
        } else {
            // Tambahkan angka ke dalam array
            numbers = append(numbers, num)
        }
    }

    fmt.Println("Program selesai.")
}
```

Output Program



Deskripsi Program

Program diatas ialah program sederhana dalam bahasa Go yang berfungsi untuk menerima input angka dari pengguna, menyimpan angka tersebut, dan menghitung median saat pengguna memasukkan angka 0. Program meminta pengguna untuk memasukkan angka satu per satu. Pengguna dapat mengakhiri input dengan memasukkan angka **-5313**.

Ketika pengguna memasukkan **0**, program akan menghitung dan mencetak median dari angka-angka yang telah dimasukkan. Jika belum ada angka yang dimasukkan sebelumnya, program akan memberi tahu pengguna bahwa tidak ada angka untuk dihitung median. Untuk menghitung median, program mengurutkan angka-angka yang ada dan menentukan nilai tengah. Jika jumlah angka genap, median adalah rata-rata dari dua angka tengah; jika ganjil, median adalah angka tengah.

UNGUIDED 3

Source Code

```
package main

import (
    "fmt"
)

// Definisi konstanta
const nMax = 7919

// Definisi struct Buku
type Buku struct {
    id       int
    judul    string
    penulis  string
    penerbit string
    tahun    int
    exemplar int
    rating   int
}

// Definisi struct DaftarBuku
type DaftarBuku struct {
    buku  [nMax]Buku
    nBuku int
}

// Fungsi untuk mendaftarkan buku
func DaftarkanBuku(perpus *DaftarBuku, n int) {
    for i := 0; i < n; i++ {
        var id, tahun, exemplar, rating int
        var judul, penulis, penerbit string

        fmt.Printf("Masukkan data buku ke-%d:\n", i+1)
        fmt.Print("ID: ")
        fmt.Scan(&id)
        fmt.Print("Judul: ")
        fmt.Scan(&judul)
        fmt.Print("Penulis: ")
        fmt.Scan(&penulis)
        fmt.Print("Penerbit: ")
        fmt.Scan(&penerbit)
```

```
        fmt.Print("Tahun: ")
        fmt.Scan(&tahun)
        fmt.Print("Exemplar: ")
        fmt.Scan(&exemplar)
        fmt.Print("Rating: ")
        fmt.Scan(&rating)

        perpus.buku[i] = Buku{id, judul, penulis, penerbit, tahun,
exemplar, rating}
    }
    perpus.nBuku = n
}

// Fungsi untuk mencetak buku dengan rating tertinggi
func CetakTerfavorit(perpus *DaftarBuku) {
    if perpus.nBuku == 0 {
        fmt.Println("Tidak ada buku dalam perpustakaan.")
        return
    }

    // Cari buku dengan rating tertinggi
    maxRating := -1
    var terbaik Buku
    for i := 0; i < perpus.nBuku; i++ {
        if perpus.buku[i].rating > maxRating {
            maxRating = perpus.buku[i].rating
            terbaik = perpus.buku[i]
        }
    }

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

// Fungsi untuk mengurutkan buku berdasarkan rating menggunakan
Insertion Sort
func UrutkanBuku(perpus *DaftarBuku) {
    for i := 1; i < perpus.nBuku; i++ {
        key := perpus.buku[i]
        j := i - 1

        for j >= 0 && perpus.buku[j].rating < key.rating {
```

```
            perpus.buku[j+1] = perpus.buku[j]
            j--
        }
        perpus.buku[j+1] = key
    }
}

// Fungsi untuk mencari buku berdasarkan rating
func CariBuku(perpus *DaftarBuku, targetRating int) {
    fmt.Printf("Mencari buku dengan rating: %d\n", targetRating)
    ditemukan := false
    for i := 0; i < perpus.nBuku; i++ {
        if perpus.buku[i].rating == targetRating {
            fmt.Printf("Buku ditemukan: Judul: %s, Penulis: %s,
Penerbit: %s, Tahun: %d\n",
                perpus.buku[i].judul, perpus.buku[i].penulis,
perpus.buku[i].penerbit, perpus.buku[i].tahun)
            ditemukan = true
        }
    }

    if !ditemukan {
        fmt.Println("Tidak ada buku dengan rating tersebut.")
    }
}

func main() {
    var perpustakaan DaftarBuku
    var n int

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

    DaftarkanBuku(&perpustakaan, n)
    fmt.Println()

    fmt.Println("Mengurutkan buku berdasarkan rating...")
    UrutkanBuku(&perpustakaan)
    fmt.Println("Daftar buku setelah diurutkan berdasarkan
rating:")
    for i := 0; i < perpustakaan.nBuku; i++ {
        fmt.Printf("Judul: %s, Rating: %d\n",
perpustakaan.buku[i].judul, perpustakaan.buku[i].rating)
    }
    fmt.Println()
```

```
    CetakTerfavorit(&perpustakaan)

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

Output Program

```
PS C:\Users\dwihe\OneDrive\Documents\SEMESTER 3\PRAKTIKUM ALPRO2\MODUL 11> go run "c
PRO2\MODUL 11\Unguided2\UNGUIDED 3\unguided3.go"
Masukkan jumlah buku yang akan didaftarkan: 2
Masukkan data buku ke-1:
ID: 068
Judul: siksaneraka
Penulis: Saya
Penerbit: Gramedia
Tahun: 2023
Exemplar: 4
Rating: 5
Masukkan data buku ke-2:
ID: 074
Judul: Bintang
Penulis: Riria
Penerbit: Gramedia
Tahun: 1945
Exemplar: 6
Rating: 5

Mengurutkan buku berdasarkan rating...
Daftar buku setelah diurutkan berdasarkan rating:
Judul: siksaneraka, Rating: 5
Judul: Bintang, Rating: 5

Buku dengan rating tertinggi:
Judul: siksaneraka, Penulis: Saya, Penerbit: Gramedia, Tahun: 2023, Rating: 5
Masukkan rating buku yang ingin dicari: 5
Mencari buku dengan rating: 5
Buku ditemukan: Judul: siksaneraka, Penulis: Saya, Penerbit: Gramedia, Tahun: 2023
Buku ditemukan: Judul: Bintang, Penulis: Riria, Penerbit: Gramedia, Tahun: 1945
PS C:\Users\dwihe\OneDrive\Documents\SEMESTER 3\PRAKTIKUM ALPRO2\MODUL 11>
```

Deskripsi Program

Program diatas ialah program sederhana dalam bahasa Go yang berfungsi sebagai sistem manajemen perpustakaan sederhana yang memungkinkan pengguna untuk mendaftarkan buku, mengurutkan buku berdasarkan rating, mencetak buku dengan rating tertinggi, dan mencari buku berdasarkan rating tertentu.

Program akan meminta pengguna untuk memasukkan data buku (ID, judul, penulis, penerbit, tahun, exemplar, dan rating) dan menyimpannya dalam daftar buku. Kemudian program akan mencetak informasi tentang buku dengan rating

tertinggi dari daftar buku yang terdaftar, dan daftar buku yang diurutkan berdasarkan rating menggunakan algoritma Insertion sort.