LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL XI

PENGURUTAN DATA

Oleh:

ALIFATUS SHABRINA AMALIA

NIM:

2311102225

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data mencakup berbagai algoritma, di antaranya Insertion Sort, yang bekerja dengan cara menyisipkan setiap elemen ke posisi yang benar dalam bagian array yang sudah terurut, dan Selection Sort, yang memilih elemen terkecil atau terbesar pada setiap iterasi dan menukarnya dengan elemen pada posisi saat ini, keduanya memiliki kompleksitas waktu $O(n^2)$ pada kasus terburuk dan sering digunakan pada dataset kecil atau hampir terurut, menyediakan implementasi yang lebih efisien melalui package sort untuk algoritma seperti Quick Sort dan Heap Sort yang mengoptimalkan pengurutan dengan kompleksitas lebih baik ($O(n \log n)$).

II. GUIDED

1. Guided 1

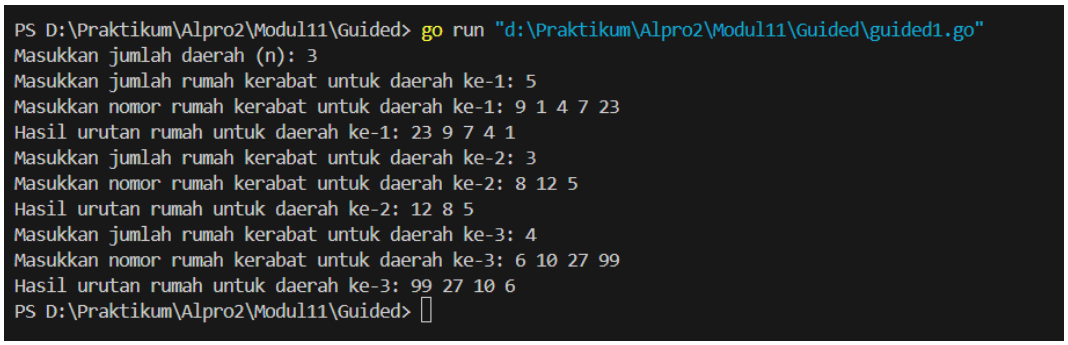
Source code

```
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()

      }

}
```

Screenshoot program



Deskripsi program

Program ini memungkinkan pengguna untuk memasukkan data jumlah daerah dan nomor rumah kerabat di setiap daerah, kemudian mengurutkan nomor rumah tersebut secara menurun menggunakan algoritma selection sort. Setiap daerah akan diproses secara terpisah, dengan input nomor rumah yang divalidasi agar sesuai dengan batasan yang telah ditentukan. Setelah pengurutan selesai, program menampilkan hasil urutan nomor rumah untuk setiap daerah. Program ini dirancang untuk menangani jumlah daerah dan rumah yang cukup besar dengan efisien dan memberikan hasil yang terstruktur.

2. Guided 2

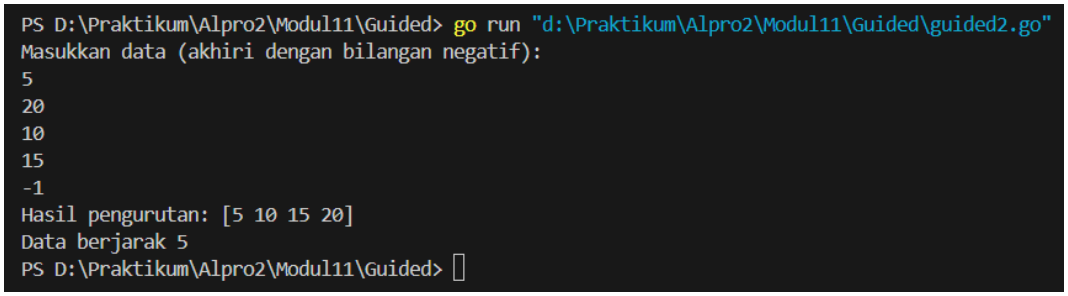
Source code

```
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")
        }
}
```

Screenshoot program



```
PS D:\Praktikum\Alpro2\Modul11\Guided> go run "d:\Praktikum\Alpro2\Modul11\Guided\guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
3
9
2
5
-1
Hasil pengurutan: [2 3 5 9]
Data berjarak tidak tetap
PS D:\Praktikum\Alpro2\Modul11\Guided>
```

Deskripsi program

Program ini menerima input serangkaian angka dari pengguna, mengurutkannya menggunakan insertion sort, dan kemudian memeriksa apakah angka-angka tersebut berjarak tetap (dengan selisih yang sama antar elemen). Prosesnya terdiri dari dua langkah utama: pertama, data diurutkan secara menaik, kemudian diperiksa untuk memastikan bahwa selisih antara elemen-elemen berturut-turut tetap sama. Jika jaraknya tetap, program akan menampilkan selisih antar elemen, jika tidak, program akan memberi tahu bahwa data tersebut tidak berjarak tetap. Program berakhir ketika pengguna memasukkan angka negatif sebagai tanda akhir input.

III. UNGUIDED

1. Unguided 1

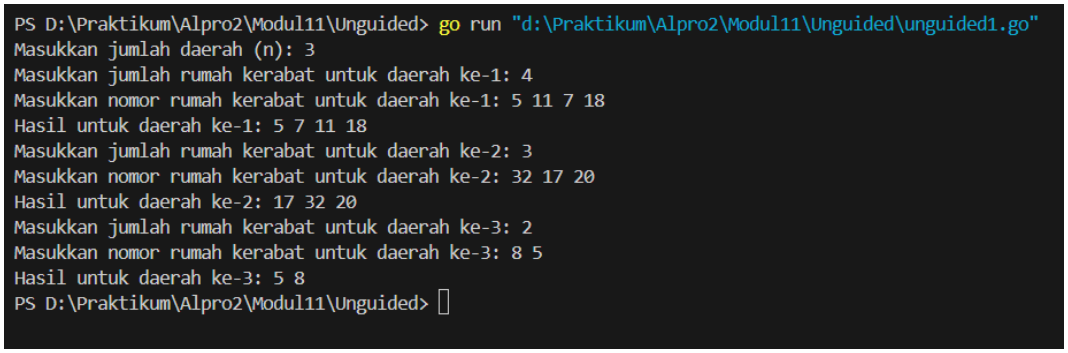
Source code

```
package main


import (
    "fmt"
    "sort"
)


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


    if n <= 0 {
        fmt.Println("Jumlah daerah harus lebih besar dari 0.")
        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 {
            fmt.Println("Jumlah rumah harus lebih besar dari 0.")
```

```
            return
        }

        // Input 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 bilangan ganjil dan genap
        var oddNumbers, evenNumbers []int
        for _, house := range houses {
            if house%2 == 0 {
                evenNumbers = append(evenNumbers, house)
            } else {
                oddNumbers = append(oddNumbers, house)
            }
        }

        // Urutkan ganjil secara menaik dan genap secara menurun
        sort.Ints(oddNumbers)
        sort.Sort(sort.Reverse(sort.IntSlice(evenNumbers)))

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

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

Screenshoot program



Deskripsi program

Program ini mengurutkan nomor rumah kerabat di setiap daerah dengan cara memisahkan bilangan ganjil dan genap, di mana bilangan ganjil diurutkan secara menaik dan dicetak lebih dahulu, kemudian diikuti oleh bilangan genap yang diurutkan secara menurun. Program menerima input berupa jumlah daerah, jumlah rumah di setiap daerah, dan nomor rumah kerabat, lalu mengelompokkan angka-angka tersebut ke dalam dua kategori (ganjil dan genap) sebelum melakukan pengurutan. Hasil akhirnya ditampilkan dalam format terurut untuk setiap daerah sesuai dengan aturan, sehingga mempermudah Hercules yang hanya ingin menyeberang di ujung jalan.

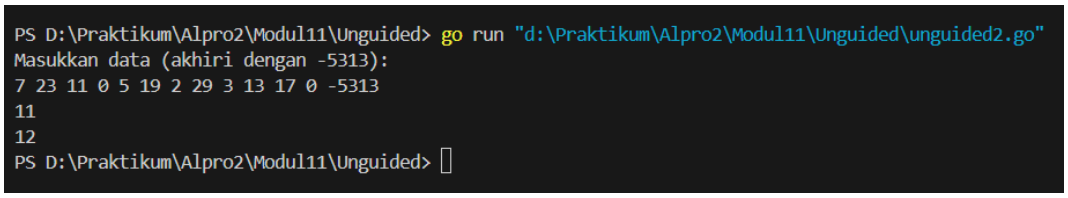
2. Unguided 2

Source code

```
package main


import (
    "fmt"
    "sort"
)


// Fungsi untuk menghitung median dari array
func findMedian(data []int) int {
    n := len(data)
    if n == 0 {
        return 0
    }
    if n%2 == 1 { // Jika jumlah data ganjil
        return data[n/2]
    }
    // Jika jumlah data genap
    return (data[n/2-1] + data[n/2]) / 2
}


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


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

```
    for {
        fmt.Scan(&input)
        if input == -5313 {
            break
        }
        if input == 0 {
            // Jika menemukan 0, urutkan data dan hitung median
            sort.Ints(numbers)
            median := findMedian(numbers)
            fmt.Println(median)
        } else {
            // Tambahkan data ke array
            numbers = append(numbers, input)
        }
    }
}
```

Screenshoot program



Deskripsi program

1. Input Data:
   - Program membaca serangkaian angka dari pengguna.
   - Input angka 0 menandakan bahwa median dari data yang sudah dimasukkan harus dihitung dan ditampilkan.

- Input -5313 menandakan akhir dari input data.

2. Pengurutan Data:
   - Setiap kali angka 0 ditemukan, program mengurutkan semua angka yang telah dimasukkan menggunakan fungsi bawaan sort.Ints.
3. Perhitungan Median:
   - Jika jumlah data ganjil, median adalah elemen di tengah.
   - Jika jumlah data genap, median adalah rata-rata dari dua elemen tengah yang dibulatkan ke bawah.
4. Output:
   - Setelah menghitung median, hasilnya langsung dicetak ke layar.
   - Proses ini diulangi setiap kali angka 0 ditemukan, dengan mempertimbangkan semua data yang telah dimasukkan hingga saat itu.

3. Unguided 3

Source code

```
package main


import (
    "fmt"
    "sort"
)


// Definisi struct Buku
type Buku struct {
    ID        string
    Judul     string
    Penulis   string
    Penerbit  string
    Eksemplar int
    Tahun     int
```

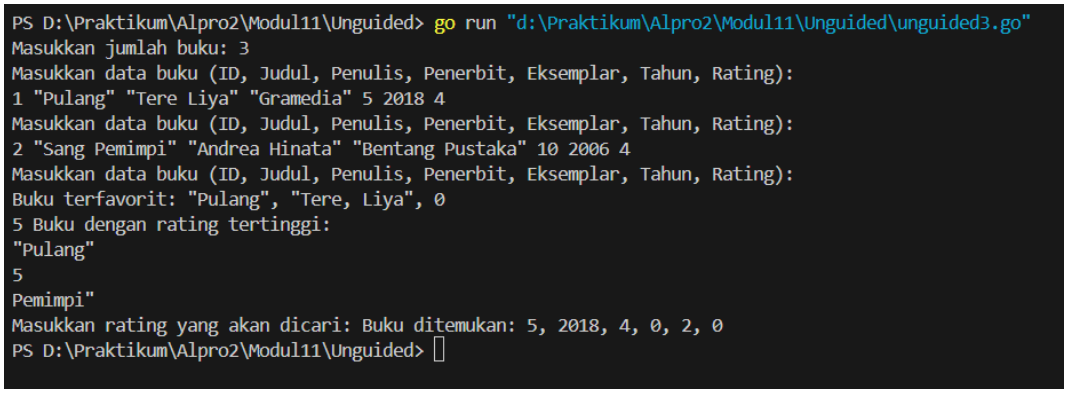
```
    Rating    int
}

// DaftarBuku adalah slice dari Buku
type DaftarBuku []Buku

// Prosedur DaftarkanBuku
func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    for i := 0; i < n; i++ {
        var buku Buku
        fmt.Println("Masukkan data buku (ID, Judul, Penulis,
Penerbit, Eksemplar, Tahun, Rating):")
        fmt.Scan(&buku.ID, &buku.Judul, &buku.Penulis,
&buku.Penerbit, &buku.Eksemplar, &buku.Tahun, &buku.Rating)
        *pustaka = append(*pustaka, buku)
    }
}

// Prosedur CetakTerfavorit
func CetakTerfavorit(pustaka DaftarBuku) {
    if len(pustaka) == 0 {
        fmt.Println("Tidak ada buku.")
        return
    }
    maxRating := pustaka[0]
    for _, buku := range pustaka {
        if buku.Rating > maxRating.Rating {
            maxRating = buku
```

```
        }
    }
    fmt.Printf("Buku terfavorit: %s, %s, %s, %d\n",
maxRating.Judul, maxRating.Penulis, maxRating.Penerbit,
maxRating.Tahun)
}

// Prosedur UrutBuku
func UrutBuku(pustaka *DaftarBuku) {
    sort.Slice(*pustaka, func(i, j int) bool {
        return (*pustaka)[i].Rating > (*pustaka)[j].Rating
    })
}

// Prosedur Cetak5Terbaru
func Cetak5Terbaru(pustaka DaftarBuku) {
    fmt.Println("5 Buku dengan rating tertinggi:")
    for i := 0; i < len(pustaka) && i < 5; i++ {
        fmt.Println(pustaka[i].Judul)
    }
}

// Prosedur CariBuku
func CariBuku(pustaka DaftarBuku, r int) {
    low, high := 0, len(pustaka)-1
    for low <= high {
        mid := (low + high) / 2
        if pustaka[mid].Rating == r {
```

```
            buku := pustaka[mid]
            fmt.Printf("Buku ditemukan: %s, %s, %s, %d, %d, %d\n",
buku.Judul, buku.Penulis, buku.Penerbit, buku.Tahun,
buku.Eksemplar, buku.Rating)
            return
        } else if pustaka[mid].Rating > r {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }
    fmt.Println("Tidak ada buku dengan rating seperti itu.")
}

func main() {
    var pustaka DaftarBuku
    var n int
    fmt.Print("Masukkan jumlah buku: ")
    fmt.Scan(&n)

    // Daftarkan buku
    DaftarkanBuku(&pustaka, n)

    // Urutkan buku berdasarkan rating
    UrutBuku(&pustaka)

    // Cetak buku terfavorit
    CetakTerfavorit(pustaka)
```

```
    // Cetak 5 buku dengan rating tertinggi
    Cetak5Terbaru(pustaka)

    // Cari buku berdasarkan rating
    var rating int
    fmt.Print("Masukkan rating yang akan dicari: ")
    fmt.Scan(&rating)
    CariBuku(pustaka, rating)
}
```

Screenshoot program



Deskripsi program

1. Struct Buku:
    - Menyimpan informasi tentang buku seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating.
2. Prosedur DaftarkanBuku:
    - Membaca dan menyimpan data buku ke dalam slice DaftarBuku.
3. Prosedur CetakTerfavorit:
    - Mencetak informasi buku dengan rating tertinggi dari daftar buku.
4. Prosedur UrutBuku:

- Mengurutkan buku dalam slice berdasarkan rating secara menurun menggunakan fungsi sort.Slice.

5. Prosedur Cetak5Terbaru:
- Menampilkan maksimal 5 buku dengan rating tertinggi.

6. Prosedur CariBuku:
- Mencari buku berdasarkan rating menggunakan pencarian biner. Jika ditemukan, menampilkan informasi buku; jika tidak, mencetak pesan bahwa buku tidak ditemukan.

7. Fungsi main:
- Mengelola alur eksekusi, mulai dari pendaftaran buku, pengurutan, pencetakan buku terfavorit, pencetakan buku dengan rating tertinggi, hingga pencarian buku berdasarkan rating.