LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 12

PENGURUTAN DATA

Oleh:

TRI PANJI UTOMO

2311102213

IF – 11- 02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

A. Ide Algoritma Selection Sort

Pengurutan secara seleksi ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian meletakkan pada posisi yang seharusnya. Pada penjelasan berikut ini data akan diurut membesar (ascending), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

	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`	`}`

B. Algoritma Selection Sort

Selection Sort adalah perbaikan dari algoritma bubble sort, dengan mengurangi jumlah perbandingan. Dikatakan selection sort karena algoritma ini mencoba memilih satu per satu elemen data dari posisi awal, untuk mencari data paling kecil dengan mencatat posisi index-nya saja, lalu dilakukan pertukaran hanya sekali pada akhir setiap tahapan.

Selection sort merupakan metode pengurutan dengan mencari nilai data terkecil dimulai dari data di posisi 0 hingga di posisi N-1. Jika terdapat N data dan data terkoleksi dari urutan 0 sampai dengan N-1 maka algoritma pengurutan dengan metode selection sort adalah sebagai berikut:

1. Cari data terkecil dalam interval j= 0 sampai dengan j= N-1
2. Jika pada posisi pos ditemukan data yang terkecil, tukarkan data di posisi pos dengan data di posisi i jika k.
3. Ulangi langkah 1 dan 2 dengan j= j+i sampai dengan j= N-1, dan seterusnya sampai j = N

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada saat proses perbandingan nilai ekstrim, kemudian tipe data dari variabel `t` sama dengan struct dari arraynya.

```
...
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk float64
}
type arrMhs [2023]mahasiswa
...
func selectionSort2(T * arrMhs, n int){
/* I.S. terdefinisi array T yang berisi n data mahasiswa
   F.S. array T terurut secara asceding atau membesar berdasarkan ipk dengan
menggunakan algoritma SELECTION SORT */
    var i, j, idx_min int
    var t mahasiswa
    i = 1
    for i <= n-1 {
        idx_min = i - 1
        j = i
        for j < n {
            if T[idx_min].ipk > T[j].ipk {
                idx_min = j
            }
            j = j + 1
        }
        t = T[idx_min]
        T[idx_min] = T[i-1]
        T[i-1] = t
        i = i + 1
    }
}
```

II. GUIDED

1. Guided 1

Source Code

```go
package main

import "fmt"

// Fungsi untuk mengurutkan array menggunakan selection sort (ascending)
func selectionSort(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        minIdx := i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[minIdx] { // Cari elemen terkecil
                minIdx = j
            }
        }
        arr[i], arr[minIdx] = arr[minIdx], 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 (setelah semua input selesai)
        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d:
", i+1)
        for _, house := range houses {
            fmt.Printf("%d ", house)
        }
```

```
        fmt.Println()
    }
}
```

Screenshot

```
Masukkan nomor rumah kerabat untuk daerah ke-1: Hasil urutan rumah untuk daerah ke-1: 1 2 7 9 13
Masukkan jumlah rumah kerabat untuk daerah ke-2: 6 189 15 27 39 75 133
Masukkan nomor rumah kerabat untuk daerah ke-2: 15 27 39 75 133 189
Masukkan jumlah rumah kerabat untuk daerah ke-3: 3 4 9 1
Masukkan nomor rumah kerabat untuk daerah ke-3: 1 4 9
PS C:\Users\ASUS\OneDrive - Telkom University\Modul Laprak Alpro SMS 3\2311102213_Tri Panji Utomo_Modul 11>
```

Deskripsi: Program ini meminta pengguna untuk memasukkan sejumlah daerah dan jumlah rumah di setiap daerah. Kemudian, program meminta pengguna untuk memasukkan nomor rumah di setiap daerah tersebut. Setelah menerima semua input, program akan mengurutkan nomor rumah di setiap daerah secara ascending (menaik) menggunakan algoritma selection sort, dan menampilkan hasilnya.

2. **Guided 2**

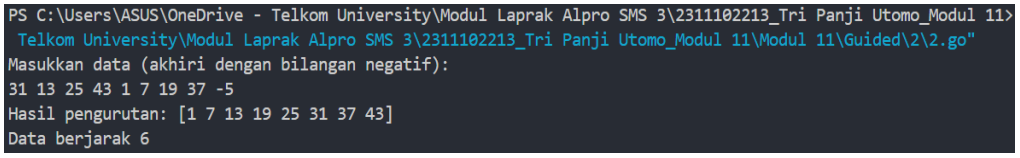
Source Code

```go
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



Deskripsi:

Program ini menerima serangkaian angka, mengurutkannya menggunakan *insertion sort*, lalu memeriksa apakah angka-angka tersebut memiliki jarak yang tetap (selisih yang sama) antara satu sama lain.

III. UNGUIDED

1. Unguided 1

Source Code

```
package main

import "fmt"

func ganjilgenap(arr []int) ([]int, []int) {
    n := len(arr)
    var gnjil213 []int
    var gnp213 []int
    for i := 0; i < n; i++ {
        if arr[i]%2 == 0 {
            gnp213 = append(gnp213, arr[i])
        } else {
            gnjil213 = append(gnjil213, arr[i])
        }
    }
    return gnjil213, gnp213
}

func SSGanjil(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 SSGenap(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 m213 int
        fmt.Printf("Masukkan jumlah rumah kerabat untuk
daerah ke-%d: ", i+1)
        fmt.Scan(&m213)

        if m213 <= 0 || m213 >= 1000000 {
```

```
            fmt.Println("m harus lebih besar dari 0 dan
kurang dari 1000000.")
            return
        }

        houses := make([]int, m213)
        fmt.Printf("Masukkan nomor rumah kerabat untuk
daerah ke-%d: ", i+1)
        for j := 0; j < m213; j++ {
            fmt.Scan(&houses[j])
        }
        gnjil213, gnp213 := ganjilgenap(houses)

        SSGanjil(gnjil213)
        SSGenap(gnp213)

        result := append(gnjil213, gnp213...)

        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d:
", i+1)
        for _, house := range result {
            fmt.Printf("%d ", house)
        }
        fmt.Println()
    }
}
```

Screenshot

```
Laprak Alpro SMS 3\2311102213_Tri Panji Utomo_Modul 11> go run "c:\Users\ASUS\OneDrive - Telkom University\Mod
13_Tri Panji Utomo_Modul 11\Modul 11\Unguided\1\1.go"
Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5 2 1 7 9 13
Masukkan nomor rumah kerabat untuk daerah ke-1: Hasil urutan rumah untuk daerah ke-1: 1 7 9 13 2
Masukkan jumlah rumah kerabat untuk daerah ke-2: 3 4 9 1
Masukkan nomor rumah kerabat untuk daerah ke-2: Hasil urutan rumah untuk daerah ke-2: 1 9 4
PS C:\Users\ASUS\OneDrive - Telkom University\Modul Laprak Alpro SMS 3\2311102213_Tri Panji Utomo_Modul 11>
```

Deskripsi:

Program ini mengurutkan nomor rumah kerabat. Nomor rumah ganjil diurutkan menaik, sedangkan nomor rumah genap diurutkan menurun. Hasilnya, nomor rumah ganjil akan berada di awal, diikuti oleh nomor rumah genap.

2. **Unguided 2**

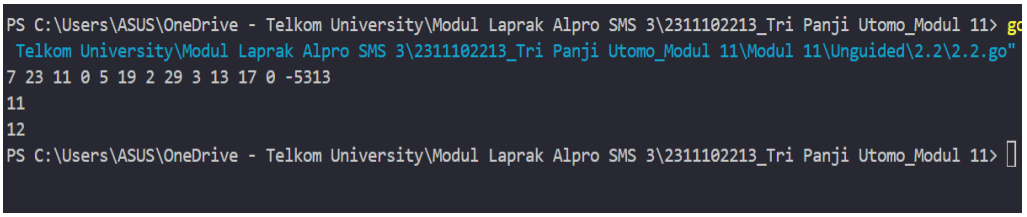
Source Code

```
package main
```

```go
import "fmt"

func selectionSort(arr213 []int) {
    n := len(arr213)
    for i := 0; i < n-1; i++ {
        maxIdx := i
        for j := i + 1; j < n; j++ {
            if arr213[j] < arr213[maxIdx] { // Cari elemen terbesar
                maxIdx = j
            }
        }
        arr213[i], arr213[maxIdx] = arr213[maxIdx], arr213[i] // Tukar elemen
    }
}
func main() {
    var arr213 []int
    var slice213 []int
    var n int

    var median int
    for {
        fmt.Scan(&n)
        arr213 = append(arr213, n)
        if n == -5313 {
            break
        }
    }
    panjang := len(arr213)

    for i := 0; i < panjang; i++ {
        if arr213[i] == 0 {
            selectionSort(slice213)
            if len(slice213)%2 == 0 {

                median = (slice213[(len(slice213)/2)-1] + slice213[len(slice213)/2]) / 2

            } else {
```

```
                median = slice213[len(slice213)/2]
                fmt.Print()
            }
            fmt.Println(median)
        } else {
            slice213 = append(slice213, arr213[i])
        }
    }

}
```

Screenshot



Deskripsi:

Program ini membaca serangkaian angka hingga menemukan angka -5313. Setiap kali menemukan angka 0, program mengurutkan angka-angka sebelumnya dan menghitung mediannya.

3. **Unguided 3**

Source Code

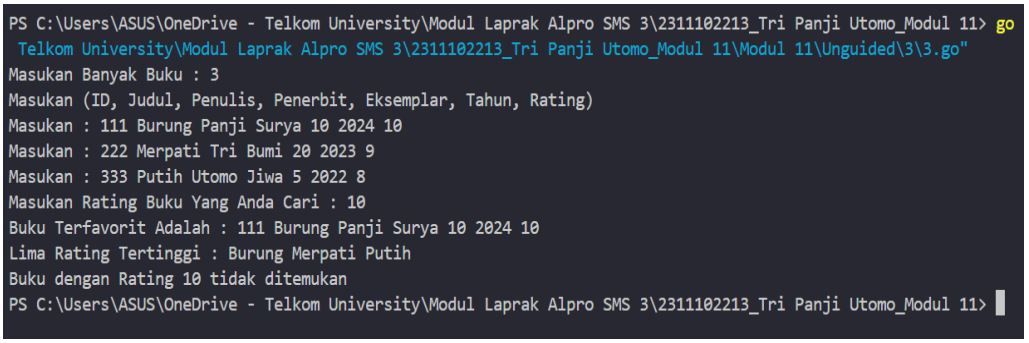
```
package main

import "fmt"

const nmax = 7919

type Buku struct {
    ID213, Judul, Penulis, Penerbit string
    Eksemplar, Tahun, Ranting   int
}
type DaftarBuku [nmax]Buku

func insertionSort(Pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := Pustaka[i]
        j := i - 1

        for j >= 0 && Pustaka[j].Ranting < key.Ranting {
            Pustaka[j+1] = Pustaka[j]
```

```
            j--
        }
        Pustaka[j+1] = key
    }
}

func findfavorit(Pustaka DaftarBuku, n int) {
    max := Pustaka[0].Ranting
    favorit := 0
    for i := 0; i < n; i++ {
        if Pustaka[i].Ranting > max {
            max = Pustaka[i].Ranting
            favorit = i
        }
    }
    // Perbaikan: Tampilkan semua informasi buku favorit
    fmt.Printf("Buku Terfavorit Adalah : %s %s %s %s %v %v
%v\n",
                Pustaka[favorit].ID213,
Pustaka[favorit].Judul,
                Pustaka[favorit].Penulis,
Pustaka[favorit].Penerbit,
                Pustaka[favorit].Eksemplar,
Pustaka[favorit].Tahun,
                Pustaka[favorit].Ranting)
}

func LimaRating(Pustaka DaftarBuku, n int) {
    insertionSort(&Pustaka, n)
    fmt.Print("Lima Rating Tertinggi : ")
    for i := 0; i < 5 && i < n; i++ {
        fmt.Print(Pustaka[i].Judul, " ")
    }
    fmt.Println()
}

func BinarySearch(Pustaka DaftarBuku, n, target int) int {
    low, high := 0, n-1

    for low <= high {
        mid := (low + high) / 2
```

```
        if Pustaka[mid].Ranting == target {
            return mid
        } else if Pustaka[mid].Ranting < target {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }

    return -1
}

func main() {
    var Buku DaftarBuku
    var n, Cari int
    fmt.Print("Masukan Banyak Buku : ")
    fmt.Scan(&n)
    fmt.Println("Masukan (ID, Judul, Penulis, Penerbit,
Eksemplar, Tahun, Rating)")
    for i := 0; i < n; i++ {
        fmt.Print("Masukan : ")
        fmt.Scan(&Buku[i].ID213, &Buku[i].Judul,
&Buku[i].Penulis, &Buku[i].Penerbit, &Buku[i].Eksemplar,
&Buku[i].Tahun, &Buku[i].Ranting)
    }

    fmt.Print("Masukan Rating Buku Yang Anda Cari : ")
    fmt.Scan(&Cari)

    findfavorit(Buku, n)
    LimaRating(Buku, n)

    insertionSort(&Buku, n)
    Temukan := BinarySearch(Buku, n, Cari)

    if Temukan != -1 {
        fmt.Printf("Buku dengan Rating %v : %s %s %s %s %v
%v %v\n", Cari, Buku[Temukan].ID213, Buku[Temukan].Judul,
Buku[Temukan].Penulis, Buku[Temukan].Penerbit,
Buku[Temukan].Eksemplar, Buku[Temukan].Tahun,
Buku[Temukan].Ranting)
    } else {
```

```
        fmt.Printf("Buku dengan Rating %v tidak
ditemukan\n", Cari)
    }
}
```

Screenshot



Deskripsi:

Program ini mengimplementasikan algoritma insertion sort dan binary search untuk mengelola data buku di perpustakaan. program ini mencari buku berdasarkan rating yang dimasukkan oleh user.