LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL XI
PENGURUTAN DATA

Oleh:

NAMA : KARTIKA PRINGGO HUTOMO

NIM : 2311102196

KELAS ; IF-11-02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data, juga dikenal sebagai sorting, adalah proses menyusun kembali himpunan objek berdasarkan aturan tertentu untuk memudahkan pencarian dan pengolahan data. Proses ini sangat penting dalam ilmu komputer dan digunakan dalam banyak aplikasi, seperti pengolahan database dan algoritma pencarian. Metode menyusun elemen data ke dalam urutan tertentu dikenal sebagai "pengurutan data". Pengurutan naik mengurutkan elemen data dari nilai terkecil ke terbesar, sedangkan pengurutan turun mengurutkan elemen data dari nilai terbesar ke terkecil.

Metode Pengurutan data
Ada banyak teknik pengurutan yang berbeda, masing-masing dengan kelebihan dan kekurangan. Beberapa metode yang biasa digunakan termasuk
sortasi bubble: mengurutkan elemen dengan membandingkannya dan mengubahnya jika berada dalam urutan yang salah. Sampai tidak ada lagi pertukaran yang diperlukan, proses ini diulang.
Sorting Selection: Mengurutkan dengan memilih elemen terkecil dari bagian yang belum terurut dan menukarnya dengan elemen pertama dari bagian tersebut; proses ini diulang untuk setiap elemen berikutnya.
Insertion Sort: Mengurutkan dengan menyisipkan elemen baru ke dalam posisi yang tepat di bagian yang sudah terurut. Metode ini serupa dengan mengurutkan kartu Sort Shell. Ini adalah generalisasi dari sort insertion yang membandingkan elemen dengan jarak tertentu. Seiring berjalannya waktu, jarak ini secara bertahap dikurangi hingga menjadi satu, yang menghasilkan pengurutan yang lebih efektif dibandingkan dengan pengaturan penambahan.

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

```
PS D:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11> go run "d:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11\guided1.go"
Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 3
Masukkan nomor rumah kerabat untuk daerah ke-1: 12 21 22
Hasil urutan rumah untuk daerah ke-1: 22 21 12
Masukkan jumlah rumah kerabat untuk daerah ke-2: 2
Masukkan nomor rumah kerabat untuk daerah ke-2: 21 22
Hasil urutan rumah untuk daerah ke-2: 22 21
PS D:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11>
```

Penjelasan

Program dimulai dengan meminta pengguna untuk memasukkan jumlah daerah (n), dengan batasan bahwa n harus lebih besar dari 0 dan tidak kurang dari 1000. Kemudian mereka diminta untuk memasukkan jumlah rumah kerabat (m) untuk setiap daerah, dengan batasan bahwa m harus lebih besar dari 0 dan tidak kurang dari 1.000.000.

Selain itu, program meminta pengguna untuk memasukkan nomor rumah anggota keluarga mereka yang tinggal di daerah tersebut. Algoritma pengurutan yang secara iteratif menemukan elemen terbesar dan menempatkannya di posisi yang tepat menggunakan pengurutan pilihan untuk mengurutkan angka-angka ini dalam array. Hasil untuk setiap area ditampilkan setelah mengurutkan array. Selain itu, program ini memastikan bahwa input pengguna memenuhi batasan, menghentikan eksekusi jika ada input yang tidak valid.

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

```
PS D:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11> go run "d:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11\guiged2.go"
Masukkan data (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6
PS D:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11>
```

Penjelasan

Program dimulai dengan meminta pengguna untuk memasukkan angka positif satu per satu; ketika mereka memasukkan angka negatif, proses berhenti. Data yang dimasukkan disimpan dalam slice.
Algoritma Insertion Sort digunakan oleh program untuk mengurutkan data dalam urutan menaik setelah mengumpulkannya. Algoritma ini menggunakan penggeseran elemen yang lebih besar untuk memindahkan elemen ke posisi yang tepat. Program menghitung selisih antar elemen dalam array yang berurutan setelah data diurutkan. Jika perbedaan tersebut sama, data dianggap memiliki jarak tetap. Jika perbedaan tidak konsisten, program akan mengatakan bahwa data tidak memiliki jarak tetap.
Selama data berupa bilangan bulat non-negatif, program ini dapat menangani berbagai ukuran data.

III. UNGUIDED

Unguided 1

Source code

```
package main

import (
	"fmt"
)

// Fungsi untuk mengurutkan array secara ascending menggunakan selection sort
func selectionSort(arr []int, ascending bool) {
	n := len(arr)
	for i := 0; i < n-1; i++ {
		selectedIdx := i
		for j := i + 1; j < n; j++ {
			if (ascending && arr[j] < arr[selectedIdx]) || (!ascending && arr[j] > arr[selectedIdx]) {
				selectedIdx = j
			}
		}
		arr[i], arr[selectedIdx] = arr[selectedIdx], 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("\nMasukkan jumlah rumah untuk daerah ke-%d: ", i+1)
		fmt.Scan(&m)

		if m <= 0 {
			fmt.Println("Jumlah rumah harus lebih besar dari 0.")
			return
		}

		// Membaca array angka sebagai input
		fmt.Printf("Masukkan nomor rumah untuk daerah ke-%d:\n", i+1)
		houses := make([]int, m)
		for j := 0; j < m; j++ {
			fmt.Scan(&houses[j])
		}

		// Pisahkan bilangan ganjil dan genap
		ganjil := make([]int, 0)
		genap := make([]int, 0)
		for _, num := range houses {
			if num%2 == 0 {
				genap = append(genap, num)
			} else {
				ganjil = append(ganjil, num)
			}
		}

		// Urutkan ganjil (ascending) dan genap (descending)
		selectionSort(ganjil, true)  // Urutkan ascending
		selectionSort(genap, false) // Urutkan descending

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

```
                for _, num := range ganjil {
                        fmt.Printf("%d ", num)
                }
                fmt.Print("Genap: ")
                for _, num := range genap {
                        fmt.Printf("%d ", num)
                }
                fmt.Println()
        }
}
```

Output :

```
PS D:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11> go run "d:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11\unguided1.go"
Masukkan jumlah daerah (n): 2

Masukkan jumlah rumah untuk daerah ke-1: 5
Masukkan nomor rumah untuk daerah ke-1:
1
2
3
4
5
Hasil urutan rumah untuk daerah ke-1: Ganjil: 1 3 5 Genap: 4 2

Masukkan jumlah rumah untuk daerah ke-2: 3
Masukkan nomor rumah untuk daerah ke-2:
5
7
8
Hasil urutan rumah untuk daerah ke-2: Ganjil: 5 7 Genap: 8
PS D:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11>
```

Penjelasan
Pengguna diminta oleh program untuk memasukkan jumlah daerah (n) dan rumah (m) untuk setiap daerah. Setelah itu, pengguna memasukkan nomor rumah yang terpisah menjadi dua kategori: bilangan ganjil dan bilangan genap. Nomor rumah ganjil diurutkan dalam urutan menaik (ascending) dan nomor rumah genap diurutkan dalam urutan menurun (descending). Fungsi Sorting Selection telah digeneralisasi untuk mendukung kedua arah pengurutan. Sehingga lebih mudah dibaca, hasil pengurutan ditampilkan dengan label yang menunjukkan kategori ganjil dan genap untuk setiap area. Program ini tetap dapat diandalkan dengan memastikan input valid dengan batasan tertentu.

Unguided 2

Source code

```
package main

import (

        "fmt"

)

// Fungsi untuk mengurutkan array menggunakan insertion sort

func insertionSort(arr []int) {

        for i := 1; i < len(arr); i++ {

                key := arr[i]

                j := i - 1
```

```
                // Pindahkan elemen yang lebih besar dari key ke satu posisi di depan
                for j >= 0 && arr[j] > key {
                        arr[j+1] = arr[j]
                        j--
                }
                arr[j+1] = key
        }
}
// Fungsi untuk menghitung median dari array yang sudah terurut
func getMedian(arr []int) int {
        if len(arr) == 0 {
                return 0 // Jika array kosong, median adalah 0
        }
        if len(arr)%2 == 1 {
                // Jika jumlah elemen ganjil, median adalah elemen tengah
                return arr[len(arr)/2]
        }
        // Jika jumlah elemen genap, median adalah rata-rata dua elemen tengah
        return (arr[len(arr)/2-1] + arr[len(arr)/2]) / 2
}
func main() {
        var input int
        data := make([]int, 0) // Inisialisasi array kosong
        fmt.Println("Masukkan bilangan bulat (akhiri dengan -5313):")
        for {
                fmt.Scan(&input)
                if input == -5313 {
                        // Marker untuk mengakhiri program
                        break
                }
                switch {
                case input == 0:
```

```
                        // Jika menemukan angka 0, urutkan array dan cetak median
                        insertionSort(data)
                        fmt.Println("Median saat ini:", getMedian(data))
                case input > 0:
                        // Hanya tambahkan bilangan bulat positif ke array
                        data = append(data, input)
                }
        }
}
```

Output

```
PS D:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11> go run "d:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11\unguided2.go"
Masukkan bilangan bulat (akhiri dengan -5313):
23 7 0 11 5 29 2 3 13 19 17 0
Median saat ini: 15
Median saat ini: 12
```

Penjelasan

Pengguna diminta oleh program untuk memasukkan jumlah daerah (n) dan rumah (m) untuk setiap daerah. Setelah itu, pengguna memasukkan nomor rumah yang terpisah menjadi dua kategori: bilangan ganjil dan bilangan genap. Nomor rumah ganjil diurutkan dalam urutan menaik (ascending) dan nomor rumah genap diurutkan dalam urutan menurun (descending). Fungsi Sorting Selection telah digeneralisasi untuk mendukung kedua arah pengurutan. Sehingga lebih mudah dibaca, hasil pengurutan ditampilkan dengan label yang menunjukkan kategori ganjil dan genap untuk setiap area. Program ini tetap dapat diandalkan dengan memastikan input valid dengan batasan tertentu.

Unguided 3

Source code

```
package main

import (
        "fmt"
)

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

// Fungsi untuk menambahkan data buku ke pustaka
func DaftarkanBuku(pustaka *[]Buku, n int) {
        for i := 0; i < n; i++ {
                var buku Buku
                fmt.Printf("Masukkan data untuk buku ke-%d (id, judul, penulis, penerbit, eksemplar, tahun,
rating):\n", i+1)
                fmt.Scan(&buku.id, &buku.judul, &buku.penulis, &buku.penerbit, &buku.eksemplar, &buku.tahun,
&buku.rating)
                *pustaka = append(*pustaka, buku)
        }
```

```
}

// Fungsi untuk mencetak buku dengan rating tertinggi
func CetakFavorit(pustaka []Buku) {
	if len(pustaka) == 0 {
		fmt.Println("Pustaka kosong.")
		return
	}
	terfavorit := pustaka[0]
	for _, buku := range pustaka {
		if buku.rating > terfavorit.rating {
			terfavorit = buku
		}
	}
	fmt.Println("Buku dengan rating tertinggi:")
	fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Eksemplar: %d, Tahun: %d, Rating: %d\n",
		terfavorit.id, terfavorit.judul, terfavorit.penulis, terfavorit.penerbit, terfavorit.eksemplar,
terfavorit.tahun, terfavorit.rating)
}

// Fungsi untuk mengurutkan array buku berdasarkan rating secara descending
func UrutkanBuku(pustaka *[]Buku) {
	for i := 1; i < len(*pustaka); i++ {
		key := (*pustaka)[i]
		j := i - 1
		for j >= 0 && (*pustaka)[j].rating < key.rating {
			(*pustaka)[j+1] = (*pustaka)[j]
			j--
		}
		(*pustaka)[j+1] = key
	}
}

// Fungsi untuk mencetak lima buku dengan rating tertinggi
func Cetak5Terbaik(pustaka []Buku) {
	fmt.Println("Lima buku dengan rating tertinggi:")
	for i := 0; i < 5 && i < len(pustaka); i++ {
		buku := pustaka[i]
		fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Eksemplar: %d, Tahun: %d, Rating:
%d\n",
			buku.id, buku.judul, buku.penulis, buku.penerbit, buku.eksemplar, buku.tahun,
buku.rating)
	}
}

// Fungsi untuk mencari buku dengan rating tertentu
func CariBuku(pustaka []Buku, rating int) {
	ditemukan := false
	for _, buku := range pustaka {
		if buku.rating == rating {
			ditemukan = true
			fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Eksemplar: %d, Tahun: %d,
Rating: %d\n",
				buku.id, buku.judul, buku.penulis, buku.penerbit, buku.eksemplar, buku.tahun,
buku.rating)
		}
	}
	if !ditemukan {
		fmt.Println("Tidak ada buku dengan rating tersebut.")
	}
}

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

	if n <= 0 || n > 7919 {
		fmt.Println("Jumlah buku harus antara 1 hingga 7919.")
```

```
                return
        }

        var pustaka []Buku

        // Input data buku
        DaftarkanBuku(&pustaka, n)

        // Cetak buku dengan rating tertinggi
        CetakFavorit(pustaka)

        // Urutkan buku berdasarkan rating
        UrutkanBuku(&pustaka)

        // Cetak lima buku dengan rating tertinggi
        Cetak5Terbaik(pustaka)

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

Output

```
PS D:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11> go run "d:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11\unguided3.go"
Masukkan jumlah buku di perpustakaan: 3
Masukkan data untuk buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 aku kita dia 230 2020 99
Masukkan data untuk buku ke-2 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 aku kita dia 300 2022 85
Masukkan data untuk buku ke-3 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
3 aku kita dia 450 2024 80
Buku dengan rating tertinggi:
ID: 1, Judul: aku, Penulis: kita, Penerbit: dia, Eksemplar: 230, Tahun: 2020, Rating: 99
Lima buku dengan rating tertinggi:
ID: 1, Judul: aku, Penulis: kita, Penerbit: dia, Eksemplar: 230, Tahun: 2020, Rating: 99
ID: 2, Judul: aku, Penulis: kita, Penerbit: dia, Eksemplar: 300, Tahun: 2022, Rating: 85
ID: 3, Judul: aku, Penulis: kita, Penerbit: dia, Eksemplar: 450, Tahun: 2024, Rating: 80
Masukkan rating buku yang ingin dicari: 99
ID: 1, Judul: aku, Penulis: kita, Penerbit: dia, Eksemplar: 230, Tahun: 2020, Rating: 99
PS D:\Alpro 2\Kartika Pringgo Hutomo_2311102196_Modul 11>
```

Penjelasan

Pengguna dapat memasukkan judul, ID, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating buku dengan kode, program yang dimaksudkan untuk mengelola data buku di perpustakaan. Program memiliki beberapa fitur utama setelah data dimasukkan:

Mencetak Buku Favorit: Menampilkan buku dengan rating tertinggi. Mengurutkan Buku: Mengurutkan daftar buku berdasarkan rating secara menurun (descending).

Menampilkan Lima Buku Terbaik: Setelah pengurutan, menampilkan lima buku dengan rating tertinggi.

Mencari Buku Berdasarkan Rating: Ini memungkinkan pengguna untuk mencari buku dengan skor tertentu.

Program ini menggunakan struktur data struct untuk merepresentasikan buku, memanfaatkan fungsi untuk modularitas, dan menggunakan logika sorting berbasis insertion sort untuk pengurutan data. Program juga melakukan validasi input pada jumlah buku yang diizinkan, memastikan bahwa jumlahnya berada dalam rentang 1 hingga 7919. Dengan pendekatan ini, kode menyediakan alat yang fleksibel untuk mengelola dan menganalisis data buku dalam sebuah perpustakaan.