LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII & XIII

PENGURUTAN DATA

Disusun Oleh :

FATTAH RIZQY ADHIPRATAMA / 2311102019

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

1. Selection Sort

Selection Sort adalah algoritma pengurutan yang bekerja dengan cara membagi array menjadi dua bagian: bagian terurut dan bagian tidak terurut. Pada setiap iterasi, elemen terkecil (untuk pengurutan menaik) dari bagian tidak terurut dipilih, lalu dipindahkan ke posisi akhir dari bagian terurut.

Langkah-langkah Selection Sort :

- Mulai dari indeks pertama, anggap elemen pertama sebagai elemen terkecil.
- Bandingkan elemen ini dengan elemen lainnya dalam bagian tidak terurut untuk menemukan elemen terkecil.
- Tukar elemen terkecil dengan elemen di indeks awal bagian tidak terurut.
- Pindahkan batas antara bagian terurut dan tidak terurut ke depan, lalu ulangi langkah ini sampai seluruh elemen terurut.

Kelebihan :

- Implementasinya sederhana.
- Tidak membutuhkan memori tambahan (in-place sorting).

Kekurangan:

- Tidak efisien untuk data yang besar karena kompleksitas waktunya.

2. Insertion Sort

Insertion Sort adalah algoritma pengurutan yang bekerja dengan cara membangun array terurut satu elemen pada satu waktu. Elemen dari bagian tidak terurut diambil dan disisipkan ke dalam posisi yang sesuai dalam bagian terurut.

Langkah-langkah Insertion Sort :

- Mulai dari indeks kedua (anggap elemen pertama sudah terurut).
- Ambil elemen di indeks saat ini, lalu bandingkan dengan elemen-elemen di bagian terurut.
- Geser elemen-elemen yang lebih besar ke kanan untuk memberikan ruang bagi elemen yang sedang diproses.
- Masukkan elemen ke posisi yang sesuai.
- Ulangi langkah ini untuk setiap elemen hingga seluruh array terurut.

Kelebihan :

- Efisien untuk data yang kecil atau data yang hampir terurut.
- Sederhana untuk diimplementasikan. Kompleksitas waktu terbaiknya jika data sudah terurut.

Kekurangan :

- Kompleksitas waktu terburuknya.
- Tidak cocok untuk data berukuran besar.

II. GUIDED

1.

Source Code

```
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 main() {
    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

```
PS D:\Data Semester 3\Praktikum Alpro 2\Modul 12> go run "d:\Data Semester 3\Praktikum Alpro 2\Modul 12\unguided1.go"
Masukkan jumlah daerah kerabat (n): 3

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 5 2 1 7 9 13
Masukkan 5 nomor rumah kerabat: Nomor rumah terurut untuk daerah 1: 1 2 7 9 13

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 6 189 15 27 39 75 133
Masukkan 6 nomor rumah kerabat: Nomor rumah terurut untuk daerah 2: 15 27 39 75 133 189

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 3 4 9 1
Masukkan 3 nomor rumah kerabat: Nomor rumah terurut untuk daerah 3: 1 4 9
PS D:\Data Semester 3\Praktikum Alpro 2\Modul 12>
```

Deskripsi Program

Program ini menggunakan Bahasa pemrograman Go. Program ini merupakan implementasi algoritma Selection Sort untuk mengurutkan nomor rumah dari beberapa daerah kerabat.

2.

Source Code

```
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
    }
}

// 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 main() {
    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

```
PS D:\Data Semester 3\Praktikum Alpro 2\Modul 12> go run "d:\Data Semester 3\Praktikum Alpro 2\Modul 12\unguided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
7 23 11 0 5 19 2 29 3 13 17 0 -5
Array setelah diurutkan:
0 0 2 3 5 7 11 13 17 19 23 29
Data berjarak tidak tetap
PS D:\Data Semester 3\Praktikum Alpro 2\Modul 12>
```

Deskripsi Program

Program ini menggunakan Bahasa pemrograman Go. Program ini menggunakan Insertion Sort untuk mengurutkan data integer yang dimasukkan oleh pengguna, dan kemudian memeriksa apakah selisih antar elemen dalam array yang telah diurutkan bersifat konstan.

III. UNGUIDED

Soal Studi Case 1

Belakangan diketahui ternyata Hercules itu tidak berani menyeberang jalan, maka selalu diusahakan agar hanya menyeberang jalan sesedikit mungkin, hanya diujung jalan. Karena nomor rumah sisi kiri jalan selalu ganjil dan sisi kanan jalan selalu genap, maka buatlah program **kerabat dekat** yang akan menampilkan nomor rumah mulai dari nomor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

Format **Masukan** masih persis sama seperti sebelumnya.

Keluaran terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap, di masing-masing daerah.

No	Masukan	Keluaran
1	3 5 2 1 7 9 13 6 189 15 27 39 75 133 3 4 9 1	1 13 12 8 2 15 27 39 75 133 189 8 4 2

Keterangan: Terdapat 3 daerah dalam contoh masukan. Baris kedua berisi campuran bilangan ganjil dan genap. Baris berikutnya hanya berisi bilangan ganjil, dan baris terakhir hanya berisi bilangan genap.

Petunjuk:

- Waktu pembacaan data, bilangan ganjil dan genap dipisahkan ke dalam dua array yang berbeda, untuk kemudian masing-masing diurutkan tersendiri.
- Atau, tetap disimpan dalam satu array, diurutkan secara keseluruhan. Tetapi pada waktu pencetakan, mulai dengan mencetak semua nilai ganjil lebih dulu, kemudian setelah selesai cetaklah semua nilai genapnya.

Source Code

```go
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 main() {
    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

```
PS D:\Data Semester 3\Praktikum Alpro 2\Modul 12> go run "d:\Data Semester 3\Praktikum Alpro 2\Modul 12\unguided1.go"
Masukkan jumlah daerah kerabat (n): 3

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 5 2 1 7 9 13
Masukkan 5 nomor rumah kerabat: Nomor rumah terurut untuk daerah 1: 1 2 7 9 13

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 6 189 15 27 39 75 133
Masukkan 6 nomor rumah kerabat: Nomor rumah terurut untuk daerah 2: 15 27 39 75 133 189

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 3 4 9 1
Masukkan 3 nomor rumah kerabat: Nomor rumah terurut untuk daerah 3: 1 4 9
PS D:\Data Semester 3\Praktikum Alpro 2\Modul 12>
```

Deskripsi Program

Program ini menggunakan Bahasa pemrograman Go. Program ini merupakan implementasi algoritma Selection Sort untuk mengurutkan nomor rumah dari beberapa daerah kerabat.

Soal Studi Case 2

Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi ternama. Dalam kompetisi tersebut, setiap tim berlomba untuk menyelesaikan sebanyak mungkin problem yang diberikan. Dari 13 problem yang diberikan, ada satu problem yang menarik. Problem tersebut mudah dipahami, hampir semua tim mencoba untuk menyelesaikannya, tetapi hanya 3 tim yang berhasil. Apa sih problemnya?

"Median adalah nilai tengah dari suatu koleksi data yang sudah terurut. Jika jumlah data genap, maka nilai median adalah rerata dari kedua nilai tengahnya. Pada problem ini, semua data merupakan bilangan bulat positif, dan karenanya rerata nilai tengah dibulatkan ke bawah."

Buatlah program **median** yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah 0.

Masukan berbentuk rangkaian bilangan bulat. Masukan tidak akan berisi lebih dari 1000000 data, tidak termasuk bilangan 0. Data 0 merupakan tanda bahwa median harus dicetak, tidak termasuk data yang dicari mediannya. Data masukan diakhiri dengan bilangan bulat -5313.

Keluaran adalah median yang diminta, satu data per baris.

No	Masukan	Keluaran
1	7 23 11 0 5 19 2 29 3 13 17 0 -5313	11 12

Source Code

```go
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
    "strings"
```

```
)

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] {
                minIdx = j
            }
        }
        arr[i], arr[minIdx] = arr[minIdx], arr[i]
    }
}

func calculateMedian(arr []int) float64 {
    n := len(arr)
    if n%2 == 0 {
        return float64(arr[n/2-1]+arr[n/2]) / 2.0
    }
    return float64(arr[n/2])
}

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Println("Masukkan angka yang dipisahkan dengan spasi
(akhiri dengan -5313):")
    scanner.Scan()
    line := scanner.Text()
    parts := strings.Split(line, " ")
    var data []int

    for _, part := range parts {
        num, err := strconv.Atoi(part)
        if err != nil {
            fmt.Println("Input tidak valid, harap masukkan
angka bulat saja.")
            return
        }

        if num == -5313 {
            break
        } else if num == 0 {
            selectionSort(data)
            median := calculateMedian(data)
```

```
            fmt.Printf("%.0f\n", median)
        } else {
            data = append(data, num)
        }
    }
}
```

Screenshoot Output

```
PS D:\Data Semester 3\Praktikum Alpro 2\Modul 12> go run "d:\Data Semester 3\Praktikum Alpro 2\Modul 12\unguided2.go"
Masukkan angka yang dipisahkan dengan spasi (akhiri dengan -5313):
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
PS D:\Data Semester 3\Praktikum Alpro 2\Modul 12>
```

Deskripsi Program

Program ini merupakan implementasi dalam bahasa Go untuk membaca, mengolah, dan menghitung nilai median dari sekumpulan angka. Program membaca masukan angka dari pengguna melalui terminal dalam format tertentu, memproses data menggunakan metode selection sort, dan menghitung median.

Soal Studi Case 3

Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Misalnya terdefinisi struct dan array seperti berikut ini:

```
const nMax : integer = 7919
type Buku = <
    id, judul, penulis, penerbit : string
    eksemplar, tahun, rating : integer >

type DaftarBuku = array [ 1..nMax] of Buku
Pustaka : DaftarBuku
nPustaka: integer
```

Masukan terdiri dari beberapa baris. Baris pertama adalah bilangan bulat N yang menyatakan banyaknya data buku yang ada di dalam perpustakaan. N baris berikutnya, masing-masingnya adalah data buku sesuai dengan atribut atau field pada struct. Baris terakhir adalah bilangan bulat yang menyatakan rating buku yang akan dicari.

Keluaran terdiri dari beberapa baris. Baris pertama adalah data buku terfavorit, baris kedua adalah lima judul buku dengan rating tertinggi, selanjutnya baris terakhir adalah data buku yang dicari sesuai rating yang diberikan pada masukan baris terakhir.

Lengkapi subprogram-subprogram dibawah ini, sesuai dengan I.S. dan F.S yang diberikan.

```
procedure DaftarkanBuku(in/out pustaka : DaftarBuku, n : integer)
{I.S. sejumlah n data buku telah siap para piranti masukan
 F.S. n berisi sebuah nilai, dan pustaka berisi sejumlah n data buku}

procedure CetakTerfavorit(in pustaka : DaftarBuku, in n : integer)
{I.S. array pustaka berisi n buah data buku dan belum terurut
 F.S. Tampilan data buku (judul, penulis, penerbit, tahun)
terfavorit, yaitu memiliki rating tertinggi}

procedure UrutBuku( in/out pustaka : DaftarBuku, n : integer )
{I.S. Array pustaka berisi n data buku
 F.S. Array pustaka terurut menurun/mengecil terhadap rating.
Catatan: Gunakan metoda Insertion sort}

procedure Cetak5Terbaru( in pustaka : DaftarBuku, n integer )
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
 F.S. Laporan 5 judul buku dengan rating tertinggi
 Catatan: Isi pustaka mungkin saja kurang dari 5}

procedure CariBuku(in pustaka : DaftarBuku, n : integer, r : integer )
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
 F.S. Laporan salah satu buku (judul, penulis, penerbit, tahun,
eksemplar, rating) dengan rating yang diberikan. Jika tidak ada buku
dengan rating yang ditanyakan, cukup tuliskan "Tidak ada buku dengan
rating seperti itu". Catatan: Gunakan pencarian biner/belah dua.}
```

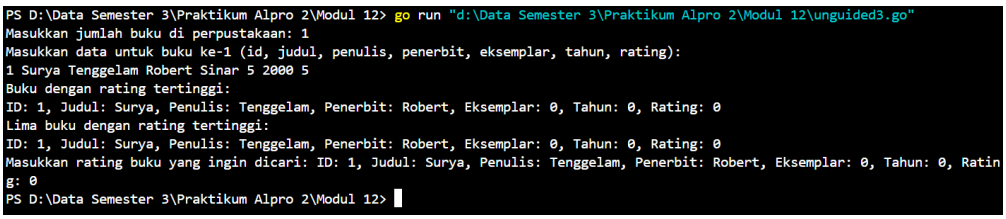
Source Code

```
package main
import (
    "fmt"
)

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

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)
    }
}

func CetakFavorit(pustaka []Buku, n int) {
    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)
}

func UrutkanBuku(pustaka *[]Buku, n int) {
    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
    }
}

func Cetak5Terbaik(pustaka []Buku, n int) {
    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)
    }
}

func CariBuku(pustaka []Buku, n int, r int) {
    ditemukan := false
    for _, buku := range pustaka {
        if buku.rating == r {
            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.")
    }
}
```

```go
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

    DaftarkanBuku(&pustaka, n)
    CetakFavorit(pustaka, n)
    UrutkanBuku(&pustaka, n)

    Cetak5Terbaik(pustaka, n)
    var rating int
    fmt.Print("Masukkan rating buku yang ingin dicari: ")
    fmt.Scan(&rating)
    CariBuku(pustaka, n, rating)
}
```

Screenshoot Program



Deskripsi Program

Program ini dibuat untuk mengelola data buku dalam sebuah perpustakaan menggunakan bahasa pemrograman Go. Program mampu menerima masukan data buku dari pengguna, menyimpan data dalam bentuk array dinamis, melakukan pencarian, pengurutan, dan menampilkan informasi buku berdasarkan kriteria tertentu.