

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII & XIII
PENGURUTAN DATA**



Disusun Oleh :

Maulisa Elvita Sari / 2311102259

IF-11-05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data (sorting) adalah salah satu operasi fundamental dalam pengolahan data yang bertujuan untuk Menyusun elemen-elemen dalam suatu struktur data ke dalam urutan tertentu, baik secara menaik (ascending) maupun menurun (descending). Dalam Bahasa pemrograman Go, pengurutan data didukung oleh Pustaka standar yang menyediakan fungsi dan mekanisme untuk menangani berbagai jenis kebutuhan pengurutan.

Algoritma pengurutan dapat dibedakan menjadi beberapa jenis berdasarkan cara kerjanya. Beberapa algoritma yang umum digunakan dalam Golang antara lain:

- **Insertion Sort:** Algoritma ini bekerja dengan membangun urutan secara bertahap. Elemen-elemen dari array diambil satu per satu dan disisipkan ke posisi yang tepat dalam sub-array yang sudah terurut.
- Selection sort adalah salah satu algoritma pengurutan yang sederhana dan mudah dipahami. Algoritma ini bekerja dengan cara membagi array menjadi dua bagian: bagian terurut dan bagian tidak terurut.

II. GUIDED

Soal Studi Case

Hercules, preman terkenal seantero ibukota, memiliki kerabat di banyak daerah. Tentunya Hercules sangat suka mengunjungi semua kerabatnya itu.

Diberikan masukan nomor rumah dari semua kerabatnya di suatu daerah, buatlah program rumahkerabat yang akan menyusun nomor-nomor rumah kerabatnya secara terurut membesar menggunakan algoritma selection sort.

Masukan dimulai dengan sebuah integer n ($0 < n < 1000$), banyaknya daerah kerabat Hercules tinggal. Isi n baris berikutnya selalu dimulai dengan sebuah integer m ($0 < m < 1000000$) yang menyatakan banyaknya rumah kerabat di daerah tersebut, diikuti dengan rangkaian bilangan bulat positif, nomor rumah para kerabat.

Keluaran terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar 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 2 7 9 13 15 27 39 75 133 189 1 4 9

Keterangan: Terdapat 3 daerah dalam contoh input, dan di masing-masing daerah mempunyai 5, 6, dan 3 kerabat.

Sourcecode

```
package main

import (
    "fmt"
)

func selectionSort (arr []int, n int){
    for i := 0; i < n-1; i++ {
        idxMin:= i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[idxMin]{
                idxMin = j
            }
        }
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}
```

```

}

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

    for daerah := 1; daerah <= n; daerah++ {
        var m int
        fmt.Printf ("\nMasukkan jumlah nomor rumah kerabat untuk daerah%d: ", daerah)
        fmt.Scan(&m)

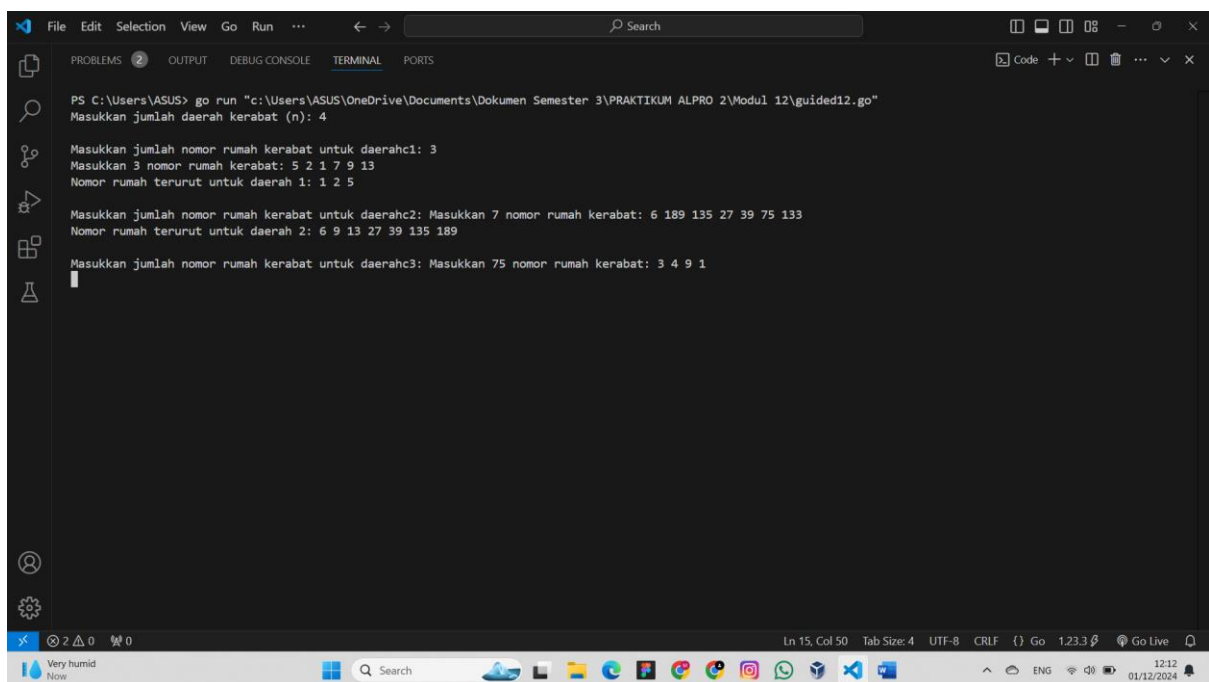
        arr := make ([]int, m)
        fmt.Printf ("Masukkan %d nomor rumah kerabat: ", m)
        for i := 0; i < m; i++ {
            fmt.Scan(&arr[i])
        }

        selectionSort (arr, m)

        fmt.Printf ("Nomor rumah terurut untuk daerah %d: ",
daerah)
        for _, num := range arr {
            fmt.Printf ("%d ", num)
        }
        fmt.Println()
    }
}

```

Screenshoot Output



```

PS C:\Users\ASUS> go run "c:\Users\ASUS\OneDrive\Documents\Dokumen Semester 3\PRAKTIKUM ALPRO 2\Modul 12\guided12.go"
Masukkan jumlah daerah kerabat (n): 4

Masukkan jumlah nomor rumah kerabat untuk daerah1: 3
Masukkan 3 nomor rumah kerabat: 5 2 1 7 9 13
Nomor rumah terurut untuk daerah 1: 1 2 5

Masukkan jumlah nomor rumah kerabat untuk daerah2: Masukkan 7 nomor rumah kerabat: 6 189 135 27 39 75 133
Nomor rumah terurut untuk daerah 2: 6 9 13 27 39 135 189

Masukkan jumlah nomor rumah kerabat untuk daerah3: Masukkan 75 nomor rumah kerabat: 3 4 9 1

```

Deskripsi Program

(deskripsikan program apa yang dibuat, memakai algoritma, dan cara kerja program sampai ke output yang dihasilkan dengan bahasa sendiri)

Program tersebut merupakan program dalam bahasa GO dimana program di atas meminta pengguna untuk mengurutkan nomor rumah kerabat berdasarkan urutan menaik. Program dirancang untuk menerima beberapa kumpulan nomor rumah dari beberapa daerah, lalu mengurutkannya satu per satu menggunakan algoritma selection sort. Setelah diurutkan, nomor rumah yang telah diurutkan ditampilkan untuk setiap daerah. Pertama, pengguna memulai dari elemen pertama untuk mencari elemen terkecil dalam array dan menempatkannya di posisi awal. Lalu, mengulang proses ini untuk elemen berikutnya dengan menyisahkan elemen-elemen yang sudah terurut, proses diulang hingga seluruh array terurut.

Soal Studi Case

Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metoda *insertion sort*), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.

Masukan terdiri dari sekumpulan bilangan bulat yang diakhiri oleh bilangan negatif. Hanya bilangan non negatif saja yang disimpan ke dalam array.

Keluaran terdiri dari dua baris. Baris pertama adalah isi dari array setelah dilakukan pengurutan, sedangkan baris kedua adalah status jarak setiap bilangan yang ada di dalam array. "Data berjarak x" atau "data berjarak tidak tetap".

Contoh masukan dan keluaran

No	Masukan	Keluaran
1	31 13 25 43 1 7 19 37 -5	1 7 13 19 25 31 37 43 Data berjarak 6
2	4 40 14 8 26 1 38 2 32 -31	1 2 4 8 14 26 32 38 40 Data berjarak tidak tetap

Sourcecode guided 2

```
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)

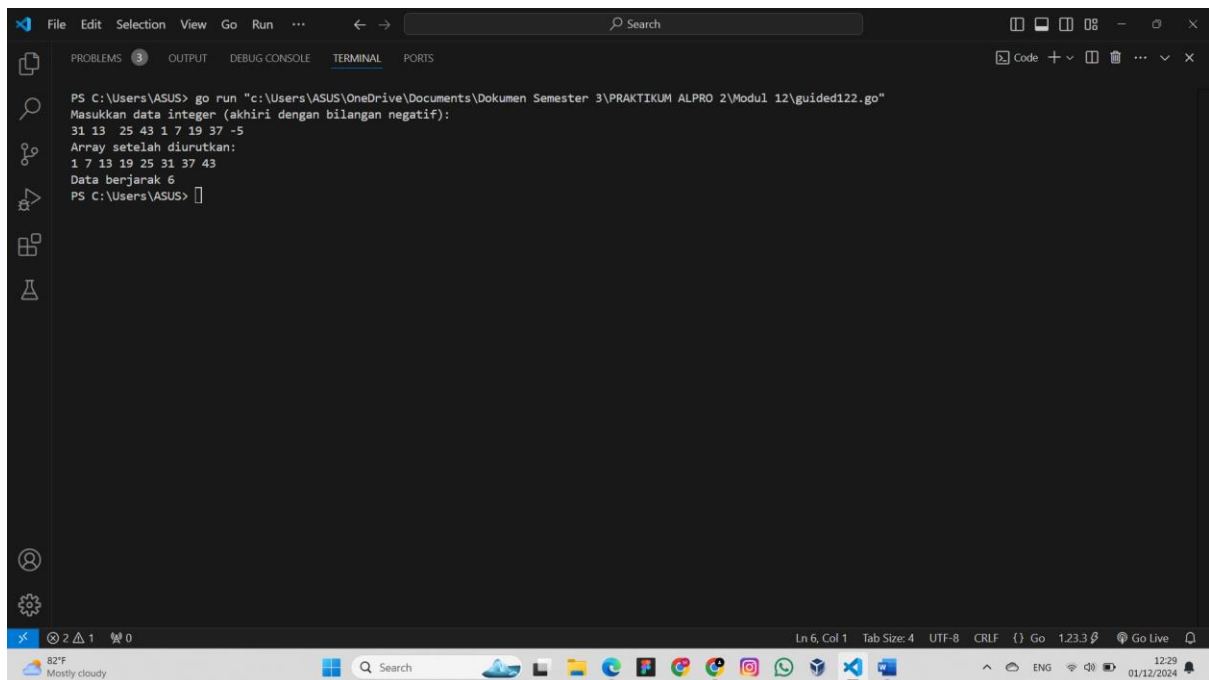
    // Periksa apakah selisih elemen tetap
    isConstant, difference := isConstantDifference(arr, n)

    // 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 C:\Users\ASUS> go run "c:\Users\ASUS\OneDrive\Documents\Dokumen Semester 3\PRAKTIKUM ALPRO 2\Modul 12\guided122.go"
Masukkan data integer (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Array setelah diurutkan:
1 7 13 19 25 31 37 43
Data berjarak 6
PS C:\Users\ASUS>
```

Deskripsi Program

(deskripsikan program apa yang dibuat, memakai algoritma, dan cara kerja program sampai ke output yang dihasilkan dengan bahasa sendiri)

Program tersebut merupakan program dalam bahasa GO dimana program di atas meminta pengguna untuk menerima masukan berupa bilangan bulat (integer) hingga ditemukan bilangan negatif sebagai tanda akhir input. Setelah itu, mengurutkan bilangan-bilangan tersebut dalam urutan menaik menggunakan algoritma insertion sort, pengguna memeriksa apakah elemen-elemen dalam array memiliki selisih yang tetap antara satu elemen dengan elemen berikutnya, maka program akan menampilkan array yang telah diurutkan dan hasil pemeriksaan mengenai selisih elemen.

UNGUIDED

1. Soal Studi Case

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.

Sourcecode

```
package main

import (
    "fmt"
)
```

```

func selectionSort(arr []int, descending bool) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        idx := i
        for j := i + 1; j < n; j++ {
            if descending {
                if arr[j] > arr[idx] {
                    idx = j
                }
            } else {
                if arr[j] < arr[idx] {
                    idx = j
                }
            }
        }
        arr[i], arr[idx] = arr[idx], arr[i]
    }
}

func separateOddEven(arr []int) ([]int, []int) {
    var odd, even []int
    for _, num := range arr {
        if num%2 == 0 {
            even = append(even, num)
        } else {
            odd = append(odd, num)
        }
    }
    return odd, even
}

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

    for i := 1; i <= t; i++ {
        var n int
        fmt.Printf("\nMasukkan jumlah nomor rumah untuk daerah %d: ", i)
        fmt.Scan(&n)

        arr := make([]int, n)
        fmt.Printf("Masukkan %d nomor rumah: ", n)
        for j := 0; j < n; j++ {
            fmt.Scan(&arr[j])
        }

        odd, even := separateOddEven(arr)

        selectionSort(odd, true)

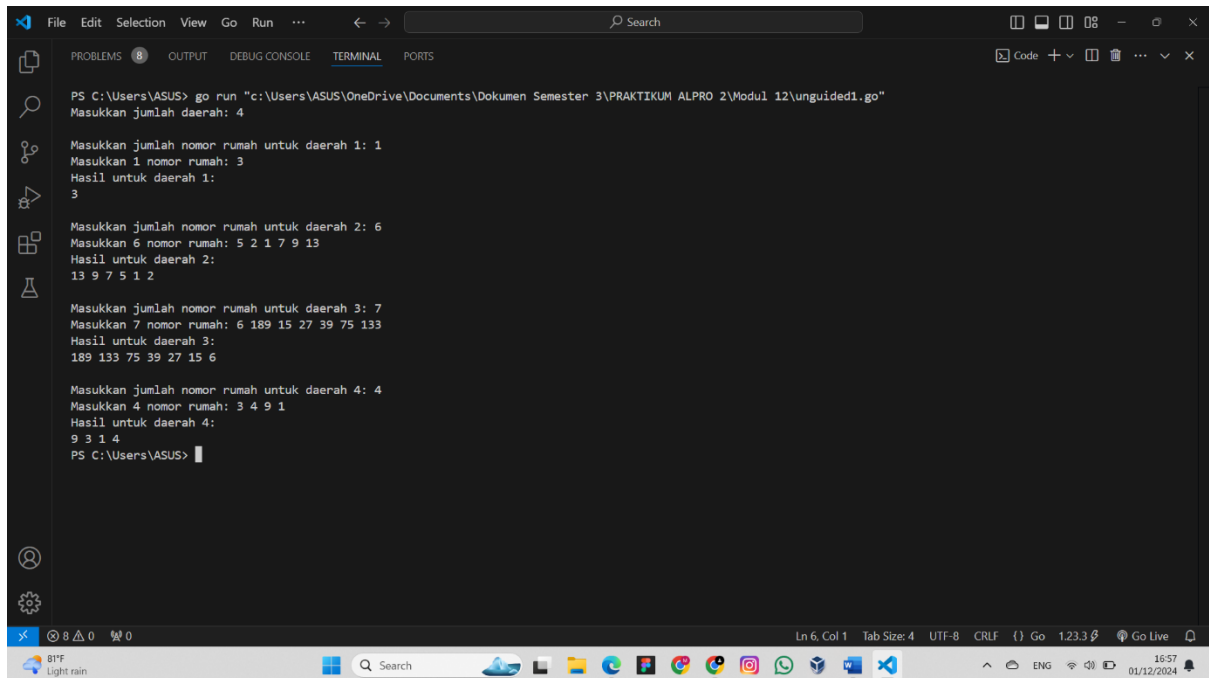
        selectionSort(even, false)

        fmt.Printf("Hasil untuk daerah %d:\n", i)
        for _, num := range odd {
            fmt.Printf("%d ", num)
        }
        for _, num := range even {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}

```

```
}  
}
```

Screenshoot Output



The screenshot shows a terminal window with the following output:

```
PS C:\Users\ASUS> go run "c:\Users\ASUS\OneDrive\Documents\Dokumen Semester 3\PRAKTIKUM ALPRO 2\Modul 12\unguided1.go"  
Masukkan jumlah daerah: 4  
  
Masukkan jumlah nomor rumah untuk daerah 1: 1  
Masukkan 1 nomor rumah: 3  
Hasil untuk daerah 1:  
3  
  
Masukkan jumlah nomor rumah untuk daerah 2: 6  
Masukkan 6 nomor rumah: 5 2 1 7 9 13  
Hasil untuk daerah 2:  
13 9 7 5 1 2  
  
Masukkan jumlah nomor rumah untuk daerah 3: 7  
Masukkan 7 nomor rumah: 6 189 15 27 39 75 133  
Hasil untuk daerah 3:  
189 133 75 39 27 15 6  
  
Masukkan jumlah nomor rumah untuk daerah 4: 4  
Masukkan 4 nomor rumah: 3 4 9 1  
Hasil untuk daerah 4:  
9 3 1 4  
PS C:\Users\ASUS>
```

Deskripsi Program

(deskripsikan program apa yang dibuat, memakai algoritma, dan cara kerja program sampai ke output yang dihasilkan dengan bahasa sendiri)

Program di atas merupakan program menggunakan bahasa Go yang dimana digunakan untuk memproses daftar nomor rumah di beberapa daerah, dimana nomor rumah dipisahkan menjadi ganjil dan genap, nomor ganjil diurutkan secara menurun (descending) dan nomor genap diurutkan secara menaik (ascending), maka hasil akhirnya akan ditampilkan dengan nomor ganjil yang diikuti oleh nomor genap pada setiap daerah.

2. Soal Studi Case

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."

Telkom University

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

Keterangan:

Sampai bilangan 0 yang pertama, data terbaca adalah 7 23 11, setelah tersusun: 7 11 23, maka median saat itu adalah 11.

Sampai bilangan 0 yang kedua, data adalah 7 23 11 5 19 2 29 3 13 17, setelah tersusun diperoleh: 2 3 5 7 11 13 17 19 23 29. Karena ada 10 data, genap, maka median adalah $(11+13)/2=12$.

Petunjuk:

Untuk setiap data bukan 0 (dan bukan marker -5313) simpan ke dalam array, Dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metode insertion sort dan ambil mediannya.

Sourcecode

```
package main

import (
    "fmt"
    "math"
)

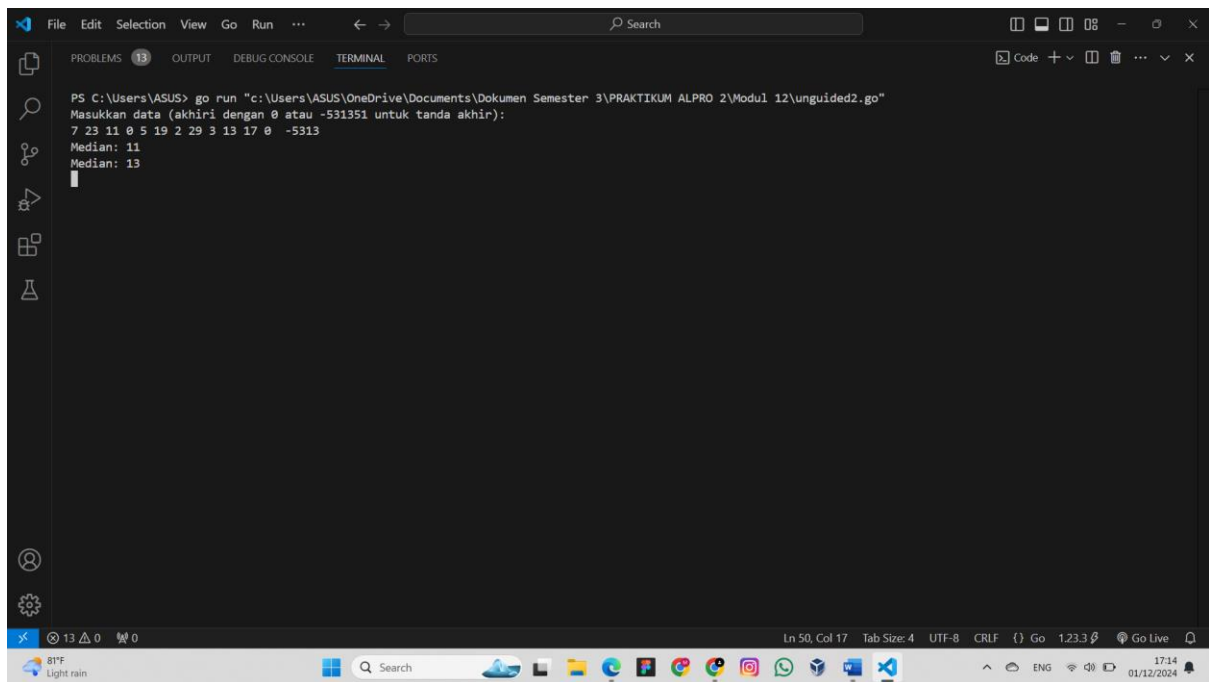
func selectionSort(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        idxMin := i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}

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

func main() {
    var num int
    var data []int

    fmt.Println("Masukkan data (akhiri dengan 0 atau -531351 untuk tanda akhir):")
    for {
        fmt.Scan(&num)
        if num == 0 || num == -531351 {
            if len(data) > 0 {
                selectionSort(data)
                median := findMedian(data)
                fmt.Println("Median:", median)
            }
            if num == 0 {
                data = []int{}
            } else {
                break
            }
        } else {
            data = append(data, num)
        }
    }
}
```

Screenshoot Output

A screenshot of a Visual Studio Code terminal window. The terminal shows the execution of a Go program. The prompt is 'PS C:\Users\ASUS> go run "c:\Users\ASUS\OneDrive\Documents\Dokumen Semester 3\PRAKTIKUM ALPRO 2\Modul 12\unguided2.go"'. The program prompts the user to 'Masukkan data (akhiri dengan 0 atau -531351 untuk tanda akhir):'. The user enters '7 23 11 0 5 19 2 29 3 13 17 0 -5313'. The program then outputs 'Median: 11' and 'Median: 13'. The terminal window has a dark theme and shows the file explorer on the left with a file named 'unguided2.go'. The status bar at the bottom indicates 'Ln 50, Col 17', 'Tab Size: 4', 'UTF-8', 'CRLF', and 'Go Live'.

Deskripsi Program

(deskripsikan program apa yang dibuat, memakai algoritma, dan cara kerja program sampai ke output yang dihasilkan dengan bahasa sendiri)

Program di atas merupakan program menggunakan bahasa Go yang dimana digunakan untuk membaca sejumlah data integer dari input pengguna, menghitung nilai median dari data tersebut, dan mencetak hasilnya. Program menggunakan selection sort untuk mengurutkan data, sehingga median dapat dihitung dengan tepat. Metode pengurutan yang memilih elemen terkecil dari daftar tidak terurut dan menukarnya dengan elemen pertama dari daftar, selanjutnya dilakukan iterasi hingga seluruh elemen terurut. Setelah median adalah nilai tengah dari data yang sudah diurutkan, maka jumlah elemen ganjil pada median adalah elemen tengah dan jumlah elemen genap, median adalah rata-rata dua elemen tengah dibulatkan ke bawah menggunakan fungsi 'math.Floor'.

3. Soal Studi Case

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.}
```

Sourcecode

```
package main

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

type Buku struct {
    ID          int
    Judul       string
    Penulis     string
    Penerbit    string
    Eksemplar   int
    Tahun       int
    Rating      int
}

func DaftarkanBuku(pustaka *[]Buku, n int) {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Println("Masukkan data buku:")
    for i := 0; i < n; i++ {
        var buku Buku
        fmt.Printf("Buku ke-%d:\n", i+1)

        fmt.Print("ID: ")
        scanner.Scan()
        buku.ID, _ = strconv.Atoi(scanner.Text())

        fmt.Print("Judul: ")
        scanner.Scan()
        buku.Judul = scanner.Text()

        fmt.Print("Penulis: ")
        scanner.Scan()
        buku.Penulis = scanner.Text()

        fmt.Print("Penerbit: ")
        scanner.Scan()
        buku.Penerbit = scanner.Text()

        fmt.Print("Eksemplar: ")
        scanner.Scan()
        buku.Eksemplar, _ = strconv.Atoi(scanner.Text())

        fmt.Print("Tahun: ")
        scanner.Scan()
        buku.Tahun, _ = strconv.Atoi(scanner.Text())

        fmt.Print("Rating: ")
        scanner.Scan()
        buku.Rating, _ = strconv.Atoi(scanner.Text())

        *pustaka = append(*pustaka, buku)
    }
}

func UrutkanBuku(pustaka *[]Buku) {
```



```

data := *pustaka
n := len(data)
for i := 1; i < n; i++ {
    key := data[i]
    j := i - 1

    // Urutkan berdasarkan rating (descending)
    for j >= 0 && data[j].Rating < key.Rating {
        data[j+1] = data[j]
        j--
    }
    data[j+1] = key
}
*pustaka = data
}

func CetakTerfavorit(pustaka []Buku) {
    if len(pustaka) == 0 {
        fmt.Println("Tidak ada buku dalam daftar.")
        return
    }

    // Buku dengan rating tertinggi
    fmt.Println("Buku dengan rating tertinggi:")
    fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Rating: %d\n",
        pustaka[0].ID, pustaka[0].Judul, pustaka[0].Penulis,
        pustaka[0].Penerbit, pustaka[0].Rating)
}

func CetakTopLima(pustaka []Buku) {
    fmt.Println("Lima buku dengan rating tertinggi:")
    top := 5
    if len(pustaka) < 5 {
        top = len(pustaka)
    }
    for i := 0; i < top; i++ {
        fmt.Printf("%d. Judul: %s, Rating: %d\n", i+1, pustaka[i].Judul,
            pustaka[i].Rating)
    }
}

func CariBuku(pustaka []Buku, targetRating int) {
    var hasil []Buku
    for _, buku := range pustaka {
        if buku.Rating == targetRating {
            hasil = append(hasil, buku)
        }
    }

    fmt.Printf("\nBuku dengan rating %d:\n", targetRating)
    if len(hasil) == 0 {
        fmt.Println("Tidak ada buku dengan rating tersebut.")
    } else {
        for _, buku := range hasil {
            fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Rating: %d\n",
                buku.Judul, buku.Penulis, buku.Penerbit, buku.Rating)
        }
    }
}

```

```

func main() {
    var pustaka []Buku
    var n, targetRating int

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

    if n <= 0 {
        fmt.Println("Jumlah buku harus lebih dari 0.")
        return
    }

    DaftarkanBuku(&pustaka, n)

    UrutkanBuku(&pustaka)

    CetakTerfavorit(pustaka)

    CetakTopLima(pustaka)

    fmt.Print("\nMasukkan rating buku yang ingin dicari: ")
    fmt.Scan(&targetRating)
    CariBuku(pustaka, targetRating)
}

```

Screenshoot Output

```

PS C:\Users\VASUS> go run "c:\Users\VASUS\OneDrive\Documents\Semester 3\PRAKTIKUM ALPRO 2\Modul 12\unguided3.go"
Masukkan jumlah buku: 3
Masukkan data buku:
Buku ke-1:
ID: Judul: gravitasi
Penulis: albert
Penerbit: gramedia
Eksemplar: 5
Tahun: 2023
Rating: 8
Buku ke-2:
ID: 111
Judul: matriks
Penulis: alkoarizmi
Penerbit: gramedia
Eksemplar: 4
Tahun: 2020
Rating: 9
Buku ke-3:
ID: 112
Judul: kalkulus
Penulis: einstein
Penerbit: gramedia
Eksemplar: 5
Tahun: 2017
Rating: 9
Buku dengan rating tertinggi:
ID: 111, Judul: matriks, Penulis: alkoarizmi, Penerbit: gramedia, Rating: 9
Lima buku dengan rating tertinggi:
1. Judul: matriks, Rating: 9
2. Judul: kalkulus, Rating: 9
3. Judul: gravitasi, Rating: 8
Masukkan rating buku yang ingin dicari: 9
Buku dengan rating 9:
Judul: matriks, Penulis: alkoarizmi, Penerbit: gramedia, Rating: 9
Judul: kalkulus, Penulis: einstein, Penerbit: gramedia, Rating: 9
PS C:\Users\VASUS>

```

Deskripsi Program

(deskripsikan program apa yang dibuat, memakai algoritma, dan cara kerja program sampai ke output yang dihasilkan dengan bahasa sendiri)

Program di atas merupakan program menggunakan bahasa Go yang dimana digunakan untuk mendaftarkan sejumlah buku beserta detailnya (ID, judul, penulis, penerbit, eksemplar, tahun,

dan rating), mengurutkan buku berdasarkan rating secara menurun (descending) menggunakan algoritma insertion sort, menampilkan buku dengan rating tertinggi, menampilkan lima buku terbaik berdasarkan rating, dan mencari buku berdasarkan rating tertentu dan menampilkan informasi buku-buku tersebut.

Kesimpulan:

Pengurutan data merupakan aspek fundamental dalam pemrograman dan sangat penting dalam aplikasi yang memerlukan pengolahan data efisien. Dengan berbagai algoritma yang tersedia dan dukungan dari pustaka standar Golang, pengguna dapat memilih metode yang paling sesuai dengan kebutuhan aplikasi mereka. Golang menawarkan kombinasi kinerja tinggi dan kemudahan penggunaan, membuatnya menjadi pilihan menarik untuk tugas-tugas pengolahan data.

DAFTAR PUSTAKA

<https://ejurnal.ulbi.ac.id/index.php/informatika/article/download/34/16>