

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII
PENGURUTAN DATA**



Disusun Oleh :

Daffa Aryaputra / 2311102272

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data (sorting) adalah proses mengatur elemen dalam sebuah struktur data seperti array atau slice berdasarkan urutan tertentu, baik secara menaik (ascending) maupun menurun (descending). Pengurutan ini penting dalam berbagai aplikasi untuk mempermudah pencarian, penyajian data, atau analisis. Dalam konteks algoritma, pengurutan dilakukan dengan membandingkan elemen-elemen data menggunakan berbagai metode, seperti Bubble Sort, Quick Sort, atau Merge Sort. Pemilihan algoritma pengurutan bergantung pada kebutuhan akan efisiensi dan kompleksitas algoritma terhadap jumlah data yang diolah.

Dalam bahasa Golang, pengurutan data dapat dilakukan menggunakan paket bawaan seperti `sort`. Paket ini menyediakan fungsi-fungsi untuk mengurutkan slice dengan tipe data bawaan seperti `int`, `float64`, atau `string`. Selain itu, Golang juga mendukung pengurutan kustom dengan menggunakan `interface sort.Interface`, memungkinkan pengguna untuk mendefinisikan logika pengurutan mereka sendiri. Pendekatan ini menjadikan Golang fleksibel dalam menangani berbagai skenario pengurutan data, baik untuk tipe data sederhana maupun kompleks.

II. GUIDED

1.

Sourcecode

```
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)

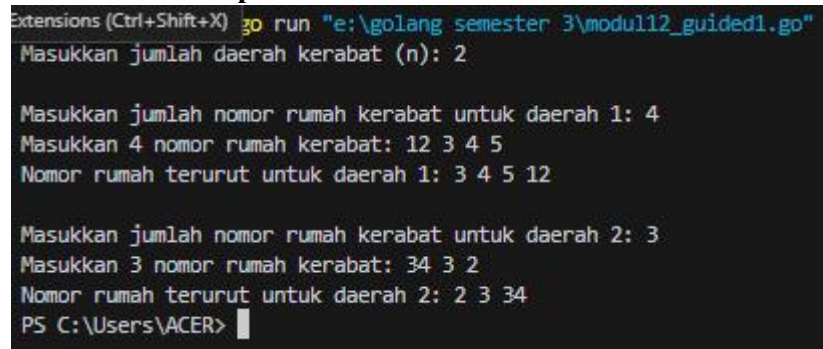
// Tampilkan hasil
fmt.Printf("Nomor rumah terurut untuk daerah %d: ",
daerah)

for _, num := range arr {
    fmt.Printf("%d ", num)
}

fmt.Println()
}
}

```

Screenshoot Output



```

Extensions (Ctrl+Shift+X) go run "e:\golang semester 3\modul12_guided1.go"
Masukkan jumlah daerah kerabat (n): 2

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 4
Masukkan 4 nomor rumah kerabat: 12 3 4 5
Nomor rumah terurut untuk daerah 1: 3 4 5 12

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 3
Masukkan 3 nomor rumah kerabat: 34 3 2
Nomor rumah terurut untuk daerah 2: 2 3 34
PS C:\Users\ACER>

```

Deskripsi Program

Program ini dirancang untuk mengurutkan nomor rumah kerabat di beberapa daerah menggunakan algoritma Selection Sort. Pengguna diminta memasukkan jumlah daerah terlebih dahulu, kemudian untuk setiap daerah, memasukkan jumlah nomor rumah yang ingin diurutkan. Setelah data dimasukkan, program akan menggunakan algoritma Selection Sort untuk menyusun nomor rumah dari nilai terkecil ke terbesar. Proses ini dilakukan untuk setiap daerah secara terpisah, dan hasilnya ditampilkan berupa daftar nomor rumah yang sudah terurut untuk setiap daerah.

Cara kerja program dimulai dengan membaca input jumlah daerah dan nomor rumah yang dimasukkan pengguna. Setelah itu, algoritma Selection Sort akan mencari elemen terkecil dari array nomor rumah, menukarnya dengan elemen di posisi saat ini, dan melanjutkan ke elemen berikutnya sampai semua elemen dalam array terurut. Setelah pengurutan selesai, program mencetak hasilnya ke layar untuk setiap daerah. Output program adalah daftar nomor rumah terurut untuk masing-masing daerah yang diproses secara bertahap sesuai input pengguna.

2.

Sourcecode

```
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)
    }

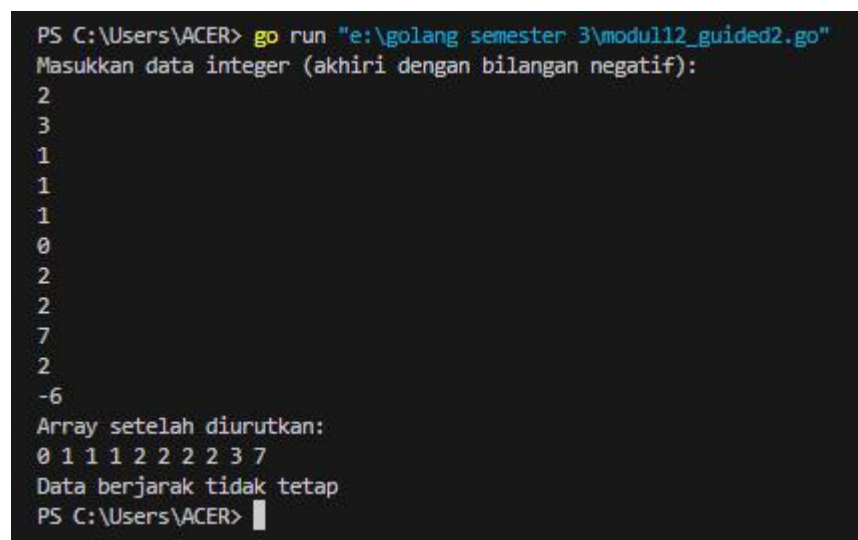
    fmt.Println()

    // Tampilkan status jarak

    if isConstant {
        fmt.Printf("Data berjarak %d\n", difference)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}
}

```

Screenshoot Output



```

PS C:\Users\ACER> go run "e:\golang semester 3\modul12_guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
2
3
1
1
1
0
2
2
7
2
-6
Array setelah diurutkan:
0 1 1 1 2 2 2 2 3 7
Data berjarak tidak tetap
PS C:\Users\ACER>

```

Deskripsi Program

Program ini dirancang untuk mengurutkan array dari angka yang dimasukkan oleh pengguna menggunakan algoritma Insertion Sort, kemudian memeriksa apakah selisih antara elemen-elemen dalam array tersebut tetap (konstan). Pengguna diminta untuk memasukkan sejumlah angka yang diakhiri dengan angka negatif, yang menandakan akhir dari input. Setelah data terkumpul, program akan mengurutkan angka-angka tersebut dan memeriksa apakah selisih antar elemen dalam array tetap konstan setelah diurutkan.

Proses pertama yang dilakukan adalah membaca input pengguna hingga ditemukan angka negatif. Angka-angka yang valid akan disimpan dalam array. Kemudian, algoritma Insertion Sort digunakan untuk mengurutkan array tersebut, dimana elemen yang lebih besar digeser ke kanan untuk memberi ruang bagi elemen yang lebih kecil, yang disisipkan pada posisi yang tepat. Setelah pengurutan, program akan memeriksa apakah selisih antara elemen-elemen dalam array tetap sama di seluruh array. Jika ya, program akan menampilkan nilai selisih tersebut, atau jika tidak, akan menyatakan bahwa selisih antar elemen tidak tetap. Output program mencakup array yang telah terurut dan informasi mengenai konsistensi selisih antar elemen.

III. UNGUIDED

1.

Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

func pisahkanGanjilGenap(arr []int) ([]int, []int) {
    var ganjil, genap []int
```

```
for _, num := range arr {
    if num%2 == 1 {
        ganjil = append(ganjil, num)
    } else {
        genap = append(genap, num)
    }
}

return ganjil, genap
}

func urutkanNomorRumah(ganjil []int, genap []int) []int {
    sort.Ints(ganjil)
    sort.Sort(sort.Reverse(sort.IntSlice(genap)))
    return append(ganjil, genap...)
}

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])
}

ganjil, genap := pisahkanGanjilGenap(arr)

hasil := urutkanNomorRumah(ganjil, genap)

fmt.Printf("Nomor rumah terurut untuk dikunjungi di
daerah %d: ", daerah)

for _, num := range hasil {
    fmt.Printf("%d ", num)
}

fmt.Println()
}
}
```

Screenshoot Output

```
PS C:\Users\ACER> go run "e:\golang semester 3\modul12_unguided1.go"
Masukkan jumlah daerah kerabat (n): 3

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 6
Masukkan 6 nomor rumah kerabat: 5 9 10 3 23 8
Nomor rumah terurut untuk dikunjungi di daerah 1: 3 5 9 23 10 8

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 7
Masukkan 7 nomor rumah kerabat: 9 98 8 2 3 4 5
Nomor rumah terurut untuk dikunjungi di daerah 2: 3 5 9 98 8 4 2

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 4
Masukkan 4 nomor rumah kerabat: 3 6 8 7
Nomor rumah terurut untuk dikunjungi di daerah 3: 3 7 8 6
PS C:\Users\ACER> |
```

Deskripsi Program

Program ini adalah aplikasi berbasis terminal yang dirancang untuk membantu mengatur urutan kunjungan ke rumah-rumah kerabat dalam beberapa daerah. Program ini menerima input berupa jumlah daerah yang ingin dikunjungi, jumlah nomor rumah di setiap daerah, serta daftar nomor rumah tersebut. Setiap nomor rumah kemudian dipisahkan menjadi dua kategori: ganjil dan genap. Nomor rumah ganjil diurutkan secara menaik (ascending), sedangkan nomor rumah genap diurutkan secara menurun (descending). Hasil pengurutan kedua kategori ini kemudian digabungkan menjadi satu daftar yang akan digunakan sebagai urutan kunjungan.

Algoritma program dimulai dengan membaca data dari pengguna, yaitu jumlah daerah dan nomor-nomor rumah di setiap daerah. Setelah itu, nomor rumah dipisahkan menjadi ganjil dan genap menggunakan fungsi `pisahkanGanjilGenap`. Kedua daftar tersebut diurutkan sesuai aturan menggunakan fungsi `urutkanNomorRumah`. Akhirnya, daftar ganjil dan genap yang sudah diurutkan digabungkan dan ditampilkan sebagai output. Dengan pendekatan ini, program memberikan urutan kunjungan rumah yang terorganisir sesuai aturan, yaitu nomor ganjil lebih dulu secara menaik dan nomor genap menyusul secara menurun, untuk setiap daerah yang dimasukkan. Output berupa daftar nomor rumah yang terurut untuk setiap daerah.

2.

Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

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

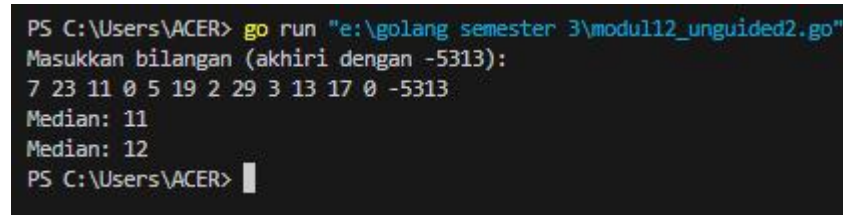
    fmt.Println("Masukkan bilangan (akhiri dengan -5313):")
    for {
        fmt.Scan(&input)

        if input == -5313 {
            break
        }

        if input == 0 {
            if len(data) > 0 {
                sort.Ints(data)
                median := calculateMedian(data)
                fmt.Printf("Median: %d\n", median)
            }
        } else {
            data = append(data, input)
        }
    }
}

func calculateMedian(data []int) int {
    n := len(data)
    if n%2 == 1 {
        return data[n/2]
    }
    return (data[(n/2)-1] + data[n/2]) / 2
}
```

Screenshoot Output



```
PS C:\Users\ACER> go run "e:\golang semester 3\modul12_unguided2.go"
Masukkan bilangan (akhiri dengan -5313):
7 23 11 0 5 19 2 29 3 13 17 0 -5313
Median: 11
Median: 12
PS C:\Users\ACER> |
```

Deskripsi Program

Program ini dibuat untuk menghitung nilai median dari sekumpulan bilangan yang dimasukkan oleh pengguna. Pengguna dapat memasukkan bilangan satu per satu, dengan dua aturan utama: masukan angka 0 untuk meminta program menghitung median dari data yang sudah dimasukkan sejauh ini, dan masukan angka -5313 untuk mengakhiri program. Median adalah nilai tengah dalam sekumpulan bilangan yang diurutkan; jika jumlah data ganjil, median adalah elemen di tengah, sedangkan jika jumlah data genap, median dihitung sebagai rata-rata dari dua elemen tengah.

Cara kerja program dimulai dengan membaca bilangan dari input pengguna secara bertahap dan menyimpannya ke dalam array. Ketika pengguna memasukkan angka 0, program akan mengurutkan array menggunakan fungsi bawaan `sort.Ints`, kemudian menghitung median berdasarkan panjang array. Jika jumlah elemen ganjil, median adalah elemen tengah; jika genap, median adalah rata-rata dari dua elemen tengah. Hasil perhitungan median ditampilkan di layar, dan program kembali menerima input. Proses ini berulang hingga pengguna memasukkan angka -5313 untuk keluar dari program. Program ini dirancang untuk menghitung median secara dinamis sesuai permintaan pengguna, tanpa perlu menghentikan input data.

3.

Sourcecode

```
package main

import "fmt"

const NMAX int = 7919

type Buku struct {
```

```

        id, judul, penulis, penerbit string
        eksemplar, tahun, rating    int
    }

type DaftarBuku [NMAX]Buku

func DaftarkanBuku(pustaka *DaftarBuku, N *int) {
    fmt.Print("Berapa banyaknya Buku: ")
    fmt.Scanln(N)

    for i := 0; i < *N; i++ {
        fmt.Printf("Data buku ke-%d:\n", i+1)
        fmt.Print("ID, Judul, Penulis, Penerbit, Eksemplar, Tahun,
Rating (pisahkan dengan spasi): ")
        fmt.Scanln(&pustaka[i].id, &pustaka[i].judul,
&pustaka[i].penulis, &pustaka[i].penerbit, &pustaka[i].eksemplar,
&pustaka[i].tahun, &pustaka[i].rating)
    }
}

func CetakTerfavorit(pustaka DaftarBuku, n int) {
    bukuFav := pustaka[0]
    for i := 1; i < n; i++ {
        if bukuFav.rating < pustaka[i].rating {
            bukuFav = pustaka[i]
        }
    }
    fmt.Printf("Buku terfavorit adalah %s oleh %s, penerbit %s,
tahun %d dengan rating %d\n",
        bukuFav.judul, bukuFav.penulis, bukuFav.penerbit,
bukuFav.tahun, bukuFav.rating)
}

func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        temp := pustaka[i]
        j := i
        for j > 0 && pustaka[j-1].rating < temp.rating {
            pustaka[j] = pustaka[j-1]
            j--
        }
        pustaka[j] = temp
    }
}

```

```

}

func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    fmt.Println("Top 5 Buku Terbaru:")
    limit := n
    if n > 5 {
        limit = 5
    }
    for i := 0; i < limit; i++ {
        fmt.Printf("%d. %s (Rating: %d)\n", i+1, pustaka[i].judul,
pustaka[i].rating)
    }
}

func CariBuku(pustaka DaftarBuku, n int, r int) {
    low, high := 0, n-1
    found := false
    var idx int

    for low <= high {
        mid := (low + high) / 2
        if pustaka[mid].rating == r {
            found = true
            idx = mid
            break
        } else if pustaka[mid].rating > r {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }

    if found {
        fmt.Println("Buku ditemukan:")
        fmt.Printf("%s oleh %s, penerbit %s, tahun %d,
eksemplar %d, rating %d\n",
            pustaka[idx].judul, pustaka[idx].penulis,
pustaka[idx].penerbit, pustaka[idx].tahun, pustaka[idx].eksemplar,
pustaka[idx].rating)
    } else {
        fmt.Println("Tidak ada buku dengan rating tersebut.")
    }
}

```



```

func main() {
    var pustaka DaftarBuku
    var nPustaka int
    var rating int

    DaftarkanBuku(&pustaka, &nPustaka)
    CetakTerfavorit(pustaka, nPustaka)
    UrutBuku(&pustaka, nPustaka)
    Cetak5Terbaru(pustaka, nPustaka)
    fmt.Print("Masukkan rating buku yang dicari: ")
    fmt.Scan(&rating)
    CariBuku(pustaka, nPustaka, rating)
}

```

Screenshoot Output

```

PS C:\Users\ACER> go run "e:\golang semester 3\modul12_unguided3.go"
Berapa banyaknya Buku: 2
Data buku ke-1:
ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating (pisahkan dengan spasi): 01 pemograman daffa gramedia 30 2024 5
Data buku ke-2:
ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating (pisahkan dengan spasi): 02 bahasa kevin gramedia 40 2023 4
Buku terfavorit adalah pemograman oleh daffa, penerbit gramedia, tahun 2024 dengan rating 5
Top 5 Buku Terbaru:
1. pemograman (Rating: 5)
2. bahasa (Rating: 4)
Masukkan rating buku yang dicari: 5
Buku ditemukan:
pemograman oleh daffa, penerbit gramedia, tahun 2024, eksemplar 30, rating 5
PS C:\Users\ACER>

```

Deskripsi Program

Program ini adalah aplikasi manajemen perpustakaan yang berfungsi untuk mendata koleksi buku, menemukan buku terfavorit, mengurutkan buku berdasarkan rating, mencetak daftar 5 buku dengan rating tertinggi, dan mencari buku dengan rating tertentu. Data setiap buku terdiri dari ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Program ini menggunakan struktur data berupa array statis DaftarBuku untuk menyimpan data buku hingga jumlah maksimum tertentu. Pengguna dapat memasukkan data buku melalui fungsi DaftarkanBuku, kemudian program akan memproses data tersebut untuk berbagai kebutuhan analisis.

Algoritma yang digunakan mencakup pencarian maksimum untuk menemukan buku terfavorit (CetakTerfavorit), algoritma insertion sort untuk mengurutkan buku berdasarkan rating dari tinggi ke rendah (UrutBuku), dan algoritma pencarian biner (CariBuku) untuk mencari buku dengan rating tertentu pada daftar yang sudah terurut. Setelah

pengurutan selesai, program menampilkan daftar 5 buku dengan rating tertinggi melalui fungsi Cetak5Terbaru. Output program berupa informasi buku terfavorit, daftar buku terbaru, dan hasil pencarian buku berdasarkan rating. Dengan algoritma ini, program efisien dalam mengolah dan menyajikan data buku secara terstruktur kepada pengguna.