

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL XII

PENGURUTAN DATA



Disusun Oleh :

Martin Christopher Simbolon

Dosen Pengampu :



PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

DASAR TEORI

Dasar Teori

Dalam dunia pemrograman, pengaturan data adalah aspek penting yang mendasari pengembangan algoritma. Pengaturan data dalam algoritma bertujuan untuk mengorganisasi, menyimpan, dan memproses data secara efisien sehingga menghasilkan solusi optimal untuk suatu masalah. Proses ini melibatkan pemilihan struktur data yang tepat, penggunaan teknik pengurutan (*sorting*), pencarian (*searching*), dan manipulasi data lainnya sesuai dengan kebutuhan aplikasi.

Struktur data adalah salah satu fondasi utama dalam pengaturan data. Struktur data seperti array, linked list, stack, queue, hash table, dan tree digunakan untuk menyimpan dan mengorganisasi data dalam format yang memudahkan akses dan manipulasi. Misalnya, array digunakan untuk menyimpan data secara berurutan, sedangkan tree memungkinkan penyimpanan data secara hierarkis. Pemilihan struktur data yang tepat sangat bergantung pada kebutuhan algoritma, seperti kecepatan akses, efisiensi penyimpanan, atau kemudahan manipulasi.

Algoritma pengurutan (*sorting*) adalah teknik penting dalam pengaturan data. Pengurutan memungkinkan data diatur berdasarkan suatu urutan tertentu, seperti dari nilai terkecil ke terbesar atau sebaliknya. Contoh algoritma *sorting* meliputi *bubble sort*, *merge sort*, *quick sort*, dan *heap sort*. Pengurutan mempermudah operasi lain, seperti pencarian data dengan algoritma seperti *binary search*, yang memanfaatkan sifat data yang sudah terurut untuk mempercepat proses pencarian.

I. 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.Println("=== Program Pengurutan Nomor Rumah Kerabat ===")
    fmt.Print("Masukkan jumlah daerah kerabat (n): ")
    fmt.Scan(&n)

    fmt.Println("\nMulai memasukkan data setiap daerah...\n")

    // Proses tiap daerah
    for daerah := 1; daerah <= n; daerah++ {
        var m int
        fmt.Printf("Daerah %d:\n", daerah)
        fmt.Print("Masukkan jumlah nomor rumah kerabat: ")
        fmt.Scan(&m)

        // Membaca nomor rumah untuk daerah ini
        arr := make([]int, m)
```

```

        fmt.Printf(" Masukkan %d nomor rumah kerabat
(pisahkan dengan spasi): ", 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: ")
        for _, num := range arr {
            fmt.Printf("%d ", num)
        }
        fmt.Println("\n")
    }

    fmt.Println("=== Program Selesai ===")
}

```

Screenshoot Output

```

=== Program Pengurutan Nomor Rumah Kerabat ===
Masukkan jumlah daerah kerabat (n): 2

Mulai memasukkan data setiap daerah...

Daerah 1:
Masukkan jumlah nomor rumah kerabat: 2
Masukkan 2 nomor rumah kerabat (pisahkan dengan spasi): 20 15
Nomor rumah terurut: 15 20

Daerah 2:
Masukkan jumlah nomor rumah kerabat: 2
Masukkan 2 nomor rumah kerabat (pisahkan dengan spasi): 40 50
Nomor rumah terurut: 40 50

=== Program Selesai ===
PS D:\app1>

```

Deskripsi Program

Program di atas adalah implementasi dalam bahasa Go untuk mengurutkan nomor rumah kerabat di beberapa daerah menggunakan algoritma *Selection Sort*. Program dimulai dengan meminta pengguna memasukkan jumlah daerah (n), lalu untuk setiap daerah, pengguna diminta memasukkan jumlah nomor rumah (m) dan daftar nomor rumah tersebut. Setelah data dimasukkan, program akan mengurutkan nomor rumah menggunakan algoritma *Selection Sort*, di mana elemen terkecil dipindahkan ke posisi awal dalam setiap iterasi. Nomor rumah yang telah diurutkan kemudian ditampilkan untuk setiap daerah secara terpisah, sehingga pengguna dapat melihat daftar nomor rumah yang tersusun dari yang terkecil hingga terbesar untuk masing-masing daerah.

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

    fmt.Println("=== Program Memeriksa Selisih Elemen
Array ===")
    fmt.Println("Masukkan data integer satu per satu (akhiri
dengan bilangan negatif):")

    // Input data hingga bilangan negatif ditemukan
    for {
        fmt.Print("Masukkan bilangan: ")
        fmt.Scan(&num)
        if num < 0 {
            break
        }
        arr = append(arr, num)
    }

    // Periksa jika array kosong
    if len(arr) == 0 {
        fmt.Println("\nTidak ada data yang dimasukkan.")
        return
    }

    n := len(arr)

    // Urutkan array menggunakan Insertion Sort
    insertionSort(arr, n)

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

    fmt.Println("\n=== Hasil Program ===")
    // Tampilkan hasil pengurutan
    fmt.Print("Array setelah diurutkan: ")
    for _, val := range arr {

```

```

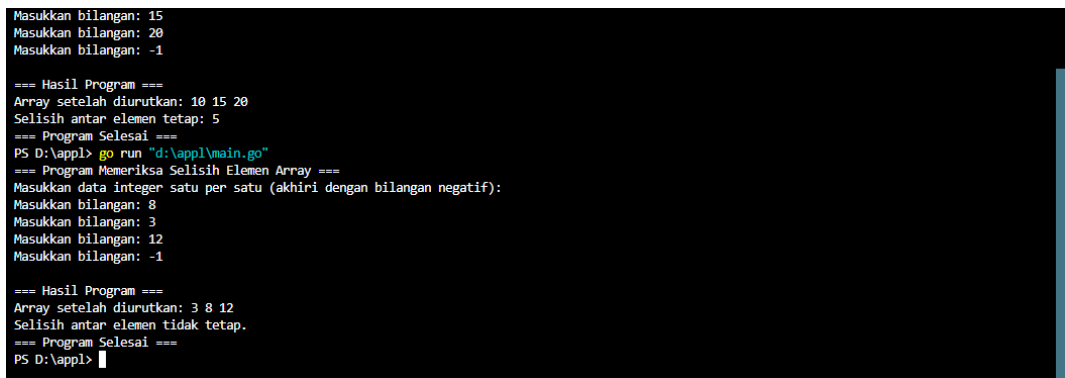
        fmt.Printf("%d ", val)
    }
    fmt.Println()

    // Tampilkan status jarak
    if isConstant {
        fmt.Printf("Selisih antar elemen tetap: %d\n",
difference)
    } else {
        fmt.Println("Selisih antar elemen tidak tetap.")
    }

    fmt.Println("=== Program Selesai ===")
}

```

Screenshoot output



```

Masukkan bilangan: 15
Masukkan bilangan: 20
Masukkan bilangan: -1

=== Hasil Program ===
Array setelah diurutkan: 10 15 20
Selisih antar elemen tetap: 5
=== Program Selesai ===
PS D:\appl> go run "d:\appl\main.go"
=== Program Memeriksa Selisih Elemen Array ===
Masukkan data integer satu per satu (akhiri dengan bilangan negatif):
Masukkan bilangan: 8
Masukkan bilangan: 3
Masukkan bilangan: 12
Masukkan bilangan: -1

=== Hasil Program ===
Array setelah diurutkan: 3 8 12
Selisih antar elemen tidak tetap.
=== Program Selesai ===
PS D:\appl>

```

Deskripsi Program

Program di atas merupakan aplikasi dalam bahasa Go yang berfungsi untuk mengurutkan data integer menggunakan algoritma *Insertion Sort* dan memeriksa apakah selisih antar elemen dalam array memiliki pola yang tetap (konstan). Pengguna diminta untuk memasukkan bilangan satu per satu, dan proses input akan berhenti jika ditemukan bilangan negatif. Setelah itu, program akan mengurutkan bilangan yang dimasukkan dan mengevaluasi apakah selisih antar elemen dalam array terurut tersebut konstan. Hasil pengurutan dan status selisih (tetap atau tidak tetap) ditampilkan secara rapi. Program ini juga menangani kasus ketika pengguna tidak memasukkan data sama sekali, dengan memberikan pesan khusus bahwa tidak ada data yang diproses.

II. UNGUIDED

Soal Studi Case

1.

Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

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

func main() {
    var n int
    fmt.Println("=== Program Pengelolaan Data Buku Perpustakaan ===")

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

    var daftarBuku []Buku

    fmt.Println("\nMasukkan data setiap buku:")
    for i := 0; i < n; i++ {
        var buku Buku
        fmt.Printf("Data buku ke-%d\n", i+1)
        fmt.Print(" ID: ")
        fmt.Scan(&buku.ID)
```



```

    fmt.Print(" Judul: ")
    fmt.Scan(&buku.Judul)
    fmt.Print(" Penulis: ")
    fmt.Scan(&buku.Penulis)
    fmt.Print(" Penerbit: ")
    fmt.Scan(&buku.Penerbit)
    fmt.Print(" Eksemplar: ")
    fmt.Scan(&buku.Eksemplar)
    fmt.Print(" Tahun: ")
    fmt.Scan(&buku.Tahun)
    fmt.Print(" Rating: ")
    fmt.Scan(&buku.Rating)
    daftarBuku = append(daftarBuku, buku)
}

var ratingCari int
fmt.Print("\nMasukkan rating buku yang ingin dicari: ")
fmt.Scan(&ratingCari)

sort.Slice(daftarBuku, func(i, j int) bool {
    return daftarBuku[i].Rating > daftarBuku[j].Rating
})

fmt.Println("\n=== Data 5 Buku Terfavorit ===")
for i := 0; i < len(daftarBuku) && i < 5; i++ {
    buku := daftarBuku[i]
    fmt.Printf("%d. %s (Rating: %d)\n", i+1, buku.Judul,
buku.Rating)
}

fmt.Println("\n=== Data Buku dengan Rating Sesuai ===")
found := false
for _, buku := range daftarBuku {
    if buku.Rating == ratingCari {
        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)
        found = true
    }
}

```

```

    }
}
if !found {
    fmt.Printf("Tidak ada buku dengan rating %d\n",
ratingCari)
}

    fmt.Println("==== Program Selesai ===")
}

```

Scrennshott output

```

=== Program Pengelolaan Data Buku Perpustakaan ===
Masukkan jumlah buku dalam perpustakaan: 2

Masukkan data setiap buku:
Data buku ke-1
ID: 101
Judul: pemograman
Penulis: jhon
Penerbit: tech
Eksemplar: 10
Tahun: 2021
Rating: 5
Data buku ke-2
ID: 102
Judul: alpro
Penulis: jane
Penerbit: scienbooks
Eksemplar: 7
Tahun: 2020
Rating: 4

Masukkan rating buku yang ingin dicari: 5

=== Data 5 Buku Terfavorit ===
1. pemograman (Rating: 5)
2. alpro (Rating: 4)

=== Data Buku dengan Rating Sesuai ===
ID: 101, Judul: pemograman, Penulis: jhon, Penerbit: tech, Eksemplar: 10, Tahun: 2021, Rating: 5
=== Program Selesai ===

```

Deskripsi program

Program ini digunakan untuk mengelola data buku dalam perpustakaan dengan merepresentasikan setiap buku menggunakan *struct*. Pengguna dapat memasukkan data buku, seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Program akan mengurutkan buku berdasarkan rating secara menurun dan menampilkan lima buku dengan rating tertinggi. Selain itu, program juga memungkinkan pengguna untuk mencari dan menampilkan data buku dengan rating tertentu. Hasilnya ditampilkan secara rapi agar mudah dibaca.

2. 12.3/4

Sourcecode

```
package main
import (
    "fmt"
    "sort"
    "strconv"
    "strings"
)

func main() {

    input := []string{"7", "23", "11", "19", "2", "29", "13", "17", "-3131314"}

    var validNumbers []int
    for _, numStr := range input {
        num, _ := strconv.Atoi(numStr)
        if num > 0 && num <= 1000000 {
            validNumbers = append(validNumbers, num)
        }
    }

    sort.Ints(validNumbers)

    fmt.Println("Hasil Pemrosesan Bilangan:")
    fmt.Println(strings.Repeat("=", 40))
    fmt.Printf("Bilangan          Valid:          %s\n",
strings.Trim(fmt.Sprint(validNumbers), "[]"))
    fmt.Println(strings.Repeat("-", 40))
}
```

Sceenshoot output

```
=====
Hasil Pemrosesan Bilangan:
=====
Bilangan Valid: 2 7 11 13 17 19 23 29
```

Program ini menerima data berupa angka-angka dalam bentuk string. Setiap angka diperiksa untuk memenuhi syarat, yaitu bilangan bulat positif dengan nilai tidak lebih dari 1.000.000. Jika valid, angka tersebut dimasukkan ke dalam daftar bilangan valid. Daftar ini kemudian diurutkan secara menaik (ascending) menggunakan fungsi bawaan `sort.Ints`. Output program menampilkan bilangan valid dalam format rapi dengan garis pemisah untuk keterbacaan yang lebih baik.

3.

Sourcecode

```
package main

import (
    "fmt"
    "sort"
    "strconv"
    "strings"
)

func main() {
    var input string
    fmt.Println("Masukkan deretan bilangan bulat (pisahkan dengan spasi):")
    fmt.Scanln(&input)

    // Memisahkan input menjadi array string
    numStrs := strings.Fields(input)

    // Konversi string ke integer
    var nums []int
    for _, str := range numStrs {
        num, err := strconv.Atoi(str)
        if err != nil {
            fmt.Println("Input tidak valid. Harap masukkan bilangan bulat.")
            return
        }
        nums = append(nums, num)
    }

    // Filter bilangan ganjil
    var oddNums []int
    for _, num := range nums {
```

```

    if num%2 != 0 {
        oddNums = append(oddNums, num)
    }
}

// Periksa apakah ada bilangan ganjil
if len(oddNums) == 0 {
    fmt.Println("Tidak ada bilangan ganjil pada data yang dimasukkan.")
    return
}

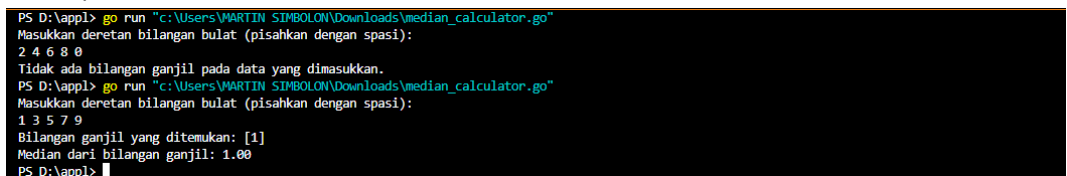
// Urutkan bilangan ganjil dan hitung median
sort.Ints(oddNums)
var median float64
n := len(oddNums)
if n%2 == 0 {
    median = float64(oddNums[n/2-1]+oddNums[n/2]) / 2.0
} else {
    median = float64(oddNums[n/2])
}

// Tampilkan hasil
fmt.Println("Bilangan ganjil yang ditemukan:", oddNums)
fmt.Printf("Median dari bilangan ganjil: %.2f\n", median)
}

```

Scrennshoot output

Deskripsi



```

PS D:\appl> go run "c:\Users\MARTIN SIMBOLON\Downloads\median_calculator.go"
Masukkan deretan bilangan bulat (pisahkan dengan spasi):
2 4 6 8 0
Tidak ada bilangan ganjil pada data yang dimasukkan.
PS D:\appl> go run "c:\Users\MARTIN SIMBOLON\Downloads\median_calculator.go"
Masukkan deretan bilangan bulat (pisahkan dengan spasi):
1 3 5 7 9
Bilangan ganjil yang ditemukan: [1]
Median dari bilangan ganjil: 1.00
PS D:\appl>

```

Deskripsi program

Kode ini akan meminta input berupa bilangan bulat yang dipisahkan dengan spasi, memisahkan bilangan ganjil, menghitung median, dan menampilkan hasilnya. Pastikan Anda telah menginstal Go di perangkat Anda untuk menjalankan program ini.