

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII & XIII
PENGURUTAN DATA**



Disusun Oleh :

SYAHRUL ROMADHONI / 2311102261

S1 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

Definisi Pengurutan Data dalam Golang

Pengurutan Data (Sorting) adalah proses menyusun elemen-elemen dalam suatu koleksi data, seperti array atau slice, ke dalam urutan tertentu berdasarkan kriteria tertentu, seperti urutan menaik (ascending) atau menurun (descending). Pengurutan data bertujuan untuk mempermudah proses pencarian, analisis, dan manipulasi data, serta meningkatkan efisiensi dalam berbagai algoritma dan aplikasi.

Dalam konteks **Golang**, pengurutan data dapat dilakukan dengan menggunakan algoritma pengurutan seperti **Insertion Sort**, **QuickSort**, atau dengan memanfaatkan fungsi built-in seperti `sort.Ints()` untuk tipe data tertentu. Pengurutan membantu dalam memeriksa pola dalam data, seperti selisih tetap antar elemen, dan memastikan bahwa data dapat diproses secara efisien sesuai dengan tujuan aplikasi yang diinginkan.

II. GUIDED

Soal Studi Case

1.

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)

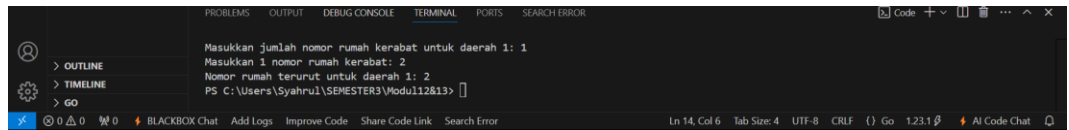
    for daerah := 1; daerah <= n; daerah++ {
        var m int
        fmt.Printf("\nMasukkan jumlah nomor rumah kerabat untuk daerah %d: ", daerah)
        fmt.Scan(&m)

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

        // Sort the house numbers
        selectionsort(arr, m)

        // Print sorted house numbers for the current daerah
        fmt.Printf("Nomor rumah terurut untuk daerah %d: ", daerah)
        for _, num := range arr {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}
```

SCREENSHOOT OUTPUT



DESKRIPSI PROGRAM

Program di atas merupakan implementasi algoritma **Selection Sort** untuk mengurutkan data nomor rumah dalam beberapa daerah. Program meminta pengguna untuk memasukkan jumlah daerah kerabat, lalu untuk setiap daerah, pengguna memasukkan jumlah nomor rumah dan daftar nomor rumah. Data nomor rumah dalam setiap daerah diurutkan menggunakan algoritma **Selection Sort**, yang bekerja dengan mencari elemen terkecil dari array dan menempatkannya di posisi awal secara iteratif. Setelah data diurutkan, program menampilkan nomor rumah yang telah terurut untuk setiap daerah. Program ini sederhana, efisien untuk jumlah data kecil, dan dirancang untuk memproses input per daerah secara terpisah.

2.

SOURCECODE

```
package main

import (
    "fmt"
)

func insertionSort(arr []int, n int) {
    for i := 1; i < n; i++ {
        key := arr[i]
        j := i - 1

        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

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("Masukkan data integer (akhiri dengan bilangan negatif):")
    for {
        fmt.Scan(&num)
        if num < 0 {
            break
        }
        arr = append(arr, num)
    }

    n := len(arr)

    insertionSort(arr, n)

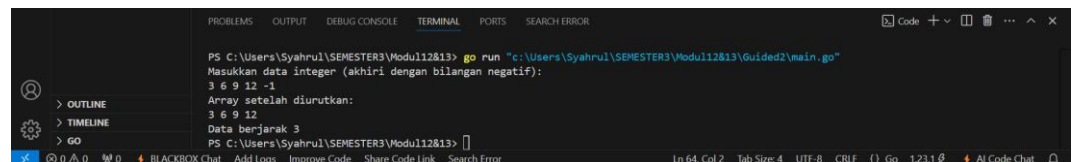
    isConstant, difference := isConstantDifference(arr, n)

    fmt.Println("Array setelah diurutkan:")
    for _, val := range arr {
        fmt.Printf("%d ", val)
    }
    fmt.Println()

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

```

SCREENSHOOT OUTPUT



```

PS C:\Users\Syahrul\SEMESTER3\Modul12&13> go run "c:\Users\Syahrul\SEMESTER3\Modul12&13\Guided2\main.go"
Masukkan data integer (akhiri dengan bilangan negatif):
3 6 9 12 -1
Array setelah diurutkan:
3 6 9 12
Data berjarak 3
PS C:\Users\Syahrul\SEMESTER3\Modul12&13>

```

DESKRIPSI PROGRAM

Program di atas adalah aplikasi untuk membaca sekumpulan bilangan integer dari pengguna, mengurutkannya menggunakan metode Insertion Sort, dan memeriksa apakah bilangan-bilangan dalam array memiliki jarak (selisih) yang tetap antara elemen-elemen berturutannya.

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

Sourcecode

```
package main

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

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Println("Masukkan jumlah baris data:")

    scanner.Scan()
    n, _ := strconv.Atoi(scanner.Text())

    fmt.Println("Masukkan data masing-masing baris:")
    results := make([]string, n)

    for i := 0; i < n; i++ {
        scanner.Scan()
        line := scanner.Text()
```

```

    numbers := parseNumbers(line)

    odd, even := splitOddEven(numbers)

    sort.Sort(sort.Reverse(sort.IntSlice(odd)))
    sort.Ints(even)

    results[i] = formatOutput(odd, even)
}

fmt.Println("\nHasil keluaran:")
for _, result := range results {
    fmt.Println(result)
}
}

func parseNumbers(input string) []int {
    parts := strings.Fields(input)
    numbers := make([]int, len(parts))
    for i, part := range parts {
        numbers[i], _ = strconv.Atoi(part)
    }
    return numbers
}

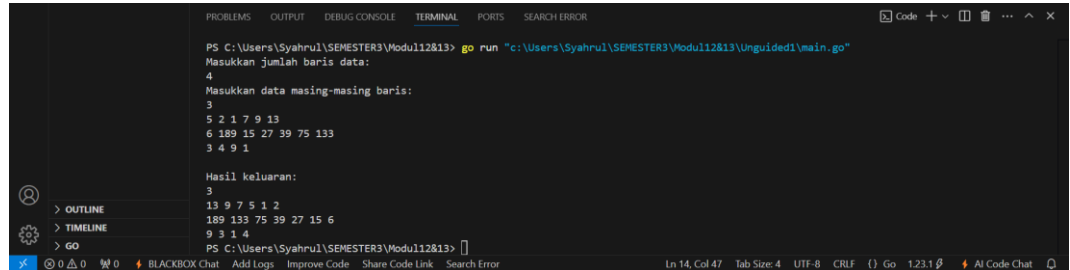
func splitOddEven(numbers []int) (odd []int, even []int) {
    for _, num := range numbers {
        if num%2 == 0 {
            even = append(even, num)
        } else {
            odd = append(odd, num)
        }
    }
    return
}

func formatOutput(odd []int, even []int) string {
    result := make([]string, 0, len(odd)+len(even))
    for _, num := range odd {
        result = append(result, strconv.Itoa(num))
    }
    for _, num := range even {
        result = append(result, strconv.Itoa(num))
    }
    return strings.Join(result, " ")
}

```



Screenshoot Output



```
PS C:\Users\Syahrul\SEMESTER3\Modul112813> go run "c:\Users\Syahrul\SEMESTER3\Unguided1\main.go"
Masukkan jumlah baris data:
4
Masukkan data masing-masing baris:
3
5 2 1 7 9 13
6 189 15 27 39 75 133
3 4 9 1

Hasil keluaran:
3
13 9 7 5 1 2
189 133 75 39 27 15 6
9 3 1 4
PS C:\Users\Syahrul\SEMESTER3\Modul112813>
```

Deskripsi Program

Program untuk mengurutkan data nomor rumah berdasarkan kriteria tertentu. Setiap baris masukan diproses dengan memisahkan nomor rumah menjadi dua kelompok: nomor ganjil dan nomor genap. Nomor ganjil diurutkan secara menurun (descending), sedangkan nomor genap diurutkan secara menaik (ascending). Program membaca jumlah baris dan data nomor rumah dari pengguna, kemudian mengolah setiap baris secara independen. Hasil pengurutan untuk nomor ganjil dan genap digabungkan menjadi satu string yang disusun sesuai urutan tersebut. Selanjutnya, hasil keluaran ditampilkan ke layar untuk setiap baris masukan secara berurutan. Program ini menggunakan fungsi-fungsi untuk parsing angka, pemisahan ganjil-genap, pengurutan, dan format keluaran agar mudah dipahami dan terstruktur.

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

Sourcecode

```
package main

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

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Println("Masukkan data (akhiri setiap baris dengan -5313):")

    for {
        scanner.Scan()
        line := scanner.Text()
        if strings.TrimSpace(line) == "" {
            break
        }
    }
}
```

```

    numbers := parseNumbers(line)

    if len(numbers) == 1 && numbers[0] == 0 {
        break
    }

    numbers = filterData(numbers, 0, -5313)

    sort.Ints(numbers)

    median := calculateMedian(numbers)

    fmt.Println(median)
}

func parseNumbers(line string) []int {
    parts := strings.Fields(line)
    numbers := make([]int, len(parts))
    for i, part := range parts {
        numbers[i], _ = strconv.Atoi(part)
    }
    return numbers
}

func filterData(numbers []int, excludes ...int) []int {
    filtered := []int{}
    for _, num := range numbers {
        exclude := false
        for _, ex := range excludes {
            if num == ex {
                exclude = true
                break
            }
        }
        if !exclude {
            filtered = append(filtered, num)
        }
    }
    return filtered
}

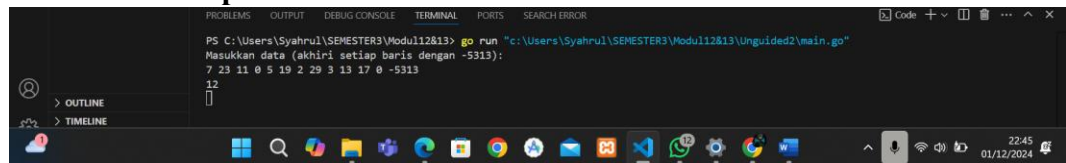
func calculateMedian(numbers []int) int {
    n := len(numbers)
    if n == 0 {
        return 0
    }

    if n%2 == 1 {
        return numbers[n/2]
    }
}

```

```
mid1, mid2 := numbers[n/2-1], numbers[n/2]
return (mid1 + mid2) / 2
}
```

Scenshoot Output



Deskripsi Program

Program di atas dirancang untuk menghitung nilai median dari sekumpulan data bilangan bulat yang dimasukkan pengguna. Setiap baris input diproses secara terpisah hingga mencapai tanda akhir data berupa bilangan -5313. Program pertama-tama membaca input, menghapus nilai -5313, dan mengurutkan bilangan dalam urutan menaik. Jika jumlah bilangan dalam baris ganjil, median diambil sebagai elemen tengah. Jika jumlahnya genap, median dihitung sebagai rata-rata dua elemen tengah dan dibulatkan ke bawah. Hasil median untuk setiap baris ditampilkan langsung setelah proses selesai. Program ini dirancang untuk menangani beberapa baris input dengan struktur sederhana dan efisien.

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.

n 90 | Modul Praktikum Algoritma dan Pemrograman 2

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 (
    "fmt"
    "sort"
    "strings"
```

```

)

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

type DaftarBuku struct {
    Pustaka []Buku
}

func (d *DaftarBuku) DaftarkanBuku(buku Buku) {
    d.Pustaka = append(d.Pustaka, buku)
}

func (d *DaftarBuku) DataTerfavorit() Buku {
    if len(d.Pustaka) == 0 {
        return Buku{}
    }

    terfavorit := d.Pustaka[0]
    for _, buku := range d.Pustaka {
        if buku.Rating > terfavorit.Rating {
            terfavorit = buku
        }
    }
    return terfavorit
}

func (d *DaftarBuku) UrutkanBuku() {
    sort.Slice(d.Pustaka, func(i, j int) bool {
        return d.Pustaka[i].Tahun < d.Pustaka[j].Tahun
    })
}

func (d *DaftarBuku) CariBuku(rating int) []Buku {
    var hasil []Buku
    for _, buku := range d.Pustaka {
        if buku.Rating == rating {
            hasil = append(hasil, buku)
        }
    }
    return hasil
}

func main() {
    var pustaka DaftarBuku

```

```

    fmt.Println("Masukkan jumlah buku:")
    var n int
    fmt.Scanln(&n)

    for i := 0; i < n; i++ {
        var id, tahun, rating int
        var judul, penulis, penerbit string

        fmt.Printf("Masukkan data buku ke-%d (ID, Judul, Penulis, Penerbit, Tahun,
Rating):\n", i+1)
        fmt.Scanln(&id)
        fmt.Scanln(&judul)
        fmt.Scanln(&penulis)
        fmt.Scanln(&penerbit)
        fmt.Scanln(&tahun)
        fmt.Scanln(&rating)

        pustaka.DaftarkanBuku(Buku{
            ID: id,
            Judul: strings.TrimSpace(judul),
            Penulis: strings.TrimSpace(penulis),
            Penerbit: strings.TrimSpace(penerbit),
            Tahun: tahun,
            Rating: rating,
        })
    }

    fmt.Println("\nBuku dengan rating tertinggi:")
    terfavorit := pustaka.DataTerfavorit()
    fmt.Printf("%d %s %s %s %d %d\n", terfavorit.ID, terfavorit.Judul,
    terfavorit.Penulis, terfavorit.Penerbit, terfavorit.Tahun, terfavorit.Rating)

    fmt.Println("\nBuku setelah diurutkan berdasarkan tahun terbit:")
    pustaka.UrutkanBuku()
    for _, buku := range pustaka.Pustaka {
        fmt.Printf("%d %s %s %s %d %d\n", buku.ID, buku.Judul, buku.Penulis,
    buku.Penerbit, buku.Tahun, buku.Rating)
    }

    fmt.Println("\nMasukkan rating untuk mencari buku:")
    var cariRating int
    fmt.Scanln(&cariRating)
    cariHasil := pustaka.CariBuku(cariRating)
    if len(cariHasil) > 0 {
        fmt.Println("Buku dengan rating yang dicari:")
        for _, buku := range cariHasil {
            fmt.Printf("%d %s %s %s %d %d\n", buku.ID, buku.Judul, buku.Penulis,
    buku.Penerbit, buku.Tahun, buku.Rating)
        }
    } else {

```

```

    }
    }
    fmt.Println("Tidak ada buku dengan rating tersebut.")
}
}

```

Screenshoot

```

PS C:\Users\Syahrul\SEMESTER3\Modul12&13> go run "c:\Users\Syahrul\SEMESTER3\Modul12&13\Unguided3\main.go"
Masukkan jumlah buku:
3
Masukkan data buku ke-1 (ID, Judul, Penulis, Penerbit, Tahun, Rating):
1 Algoritma_Dasar John_Doe Penerbit_A 2020 5
Masukkan data buku ke-2 (ID, Judul, Penulis, Penerbit, Tahun, Rating):
2 Struktur_Data Jane_Doe Penerbit_B 2018 4
3 Pemrograman_Lanjut Alice Penerbit_C 2021 5

Buku dengan rating tertinggi:
2 Struktur_Data Jane_Doe Penerbit_B 2018 4

Buku setelah diurutkan berdasarkan tahun terbit:
2 Struktur_Data Jane_Doe Penerbit_B 2018 4
1 Algoritma_Dasar John_Doe Penerbit_A 2020 5
3 Pemrograman_Lanjut Alice Penerbit_C 2021 5

Masukkan rating untuk mencari buku:
5
Tidak ada buku dengan rating tersebut.
PS C:\Users\Syahrul\SEMESTER3\Modul12&13>

```

Deskripsi Program

Program di atas adalah aplikasi pengelolaan data buku untuk sebuah perpustakaan menggunakan struktur data struct di Golang. Program ini dapat melakukan beberapa fungsi utama: mendaftarkan buku, menampilkan buku dengan rating tertinggi, mengurutkan buku berdasarkan tahun terbit, dan mencari buku berdasarkan rating tertentu. Data buku mencakup ID, judul, penulis, penerbit, tahun, dan rating, yang disimpan dalam array dinamis. Program membaca input pengguna untuk menambahkan buku, kemudian memproses data sesuai kebutuhan, seperti menampilkan buku favorit atau mencari buku dengan kriteria tertentu. Dengan pengurutan otomatis dan pencarian spesifik, program ini mempermudah pengelolaan serta penelusuran koleksi buku secara efisien.

KESIMPULAN

Kesimpulan dari materi **Pengurutan Data pada Golang** adalah bahwa Golang menyediakan berbagai metode dan algoritma untuk mengurutkan data dengan cara yang efisien dan sederhana. Salah satu algoritma pengurutan yang sering digunakan adalah **Insertion Sort**, yang bekerja dengan menyisipkan elemen yang belum terurut ke posisi yang sesuai dalam bagian array yang sudah terurut.

Dalam Golang, pengurutan dapat dilakukan dengan menggunakan fungsi built-in seperti `sort.Ints()` untuk array atau slice bertipe integer, yang memanfaatkan algoritma **Quicksort** atau **Heapsort** di belakang layar. Penggunaan algoritma pengurutan secara manual, seperti **Insertion Sort**, berguna untuk memahami bagaimana algoritma bekerja dan memberikan fleksibilitas dalam memilih metode pengurutan yang sesuai dengan kebutuhan aplikasi.

Selain itu, Golang memungkinkan untuk memeriksa pola-pola tertentu dalam data setelah pengurutan, seperti **jarak tetap** antara elemen-elemen berturutannya. Ini berguna dalam berbagai aplikasi, seperti dalam analisis data atau aplikasi yang memerlukan pemeriksaan kesamaan atau pola aritmatika dalam data.

Secara keseluruhan, pengurutan data di Golang menawarkan efisiensi dan kemudahan, baik dengan menggunakan fungsi built-in maupun mengimplementasikan algoritma pengurutan secara manual untuk kebutuhan khusus.

DAFTAR PUSTAKA

<https://journal.lppmunindra.ac.id/index.php/STRING/article/view/17972>