

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 12 & 13  
PENGURUTAN DATA**



**Disusun Oleh :**

**HANIF REYHAN ZHAFRAN ADRYTONA / 2311102266**

**11 IF 06**

**Dosen Pengampu :**

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

Pengurutan data (sorting) adalah proses menyusun elemen dalam suatu koleksi data, seperti array atau slice, berdasarkan urutan tertentu, baik secara menaik (ascending) maupun menurun (descending). Dalam bahasa pemrograman Go, pengurutan data dapat dilakukan dengan mudah menggunakan paket standar `sort`. Paket ini menyediakan fungsi bawaan seperti `sort.Ints` untuk mengurutkan slice integer, `sort.Strings` untuk slice string, dan `sort.Float64s` untuk slice float secara otomatis. Selain itu, Go juga mendukung pengurutan kostum menggunakan tipe `sort.Interface`, di mana pengguna dapat mengimplementasikan tiga metode: `Len()`, `Less(i, j int)`, dan `Swap(i, j int)`, untuk mengurutkan struktur data yang kompleks. Dalam praktiknya, algoritma bawaan `sort` di Go menggunakan metode yang sangat efisien dengan kompleksitas waktu rata-rata  **$O(n \log n)$** , memungkinkan pengurutan dilakukan secara optimal. Dengan dukungan ini, Go menawarkan fleksibilitas yang besar untuk berbagai kebutuhan pengurutan, baik pada data sederhana maupun kompleks.

## II. GUIDED

Guided 1.0

Sourcecode

```
// 2311102266_Hanif Reyhan Zhafran Arytona

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++ {
            // Cari elemen terkecil
            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\nuntuk 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

```

PROBLEMS 9 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 12 & 13\GUIDED.1.go"
Masukkan jumlah daerah kerabat (n): 3

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 3
Masukkan 3 nomor rumah kerabat: 10 11 12
Nomor rumah terurut untuk daerah 1: 10 11 12

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 5
Masukkan 5 nomor rumah kerabat: 13 9 10 8 11
Nomor rumah terurut untuk daerah 1: 10 11 12

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 5
Masukkan jumlah nomor rumah kerabat untuk daerah 2: 5
Masukkan 5 nomor rumah kerabat: 13 9 10 8 11
Nomor rumah terurut untuk daerah 2: 8 9 10 11 13

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 2
Masukkan 2 nomor rumah kerabat: 6 2
Nomor rumah terurut untuk daerah 3: 2 6

```

### Deskripsi Program

Program di atas adalah program untuk mengurutkan nomor rumah kerabat pada beberapa daerah menggunakan algoritma **Selection Sort**. Pengguna pertama-tama diminta untuk memasukkan jumlah daerah ( $n$ ), kemudian untuk setiap daerah, dimasukkan jumlah nomor rumah ( $m$ ) dan daftar nomor rumah yang ingin diurutkan. Program menggunakan fungsi `selectionSort` untuk mengurutkan nomor rumah dari terkecil ke terbesar. Algoritma ini bekerja dengan mencari elemen terkecil dari sisa array dan menukarnya ke posisi yang benar secara iteratif. Setelah pengurutan selesai, program menampilkan daftar nomor rumah yang telah diurutkan untuk setiap daerah. Program ini dirancang agar mudah digunakan dan cocok untuk mengolah data nomor rumah dalam jumlah kecil hingga menengah, dengan logika yang efisien dan tampilan hasil yang jelas.

## Guided 2.0

### Sourcecode

```
// 2311102266_Hanif Reyhan Zhafran Arytona

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)
if n == 0 {
    fmt.Println("Tidak ada data yang dimasukkan.")
    return
}

insertionSort(arr, n)

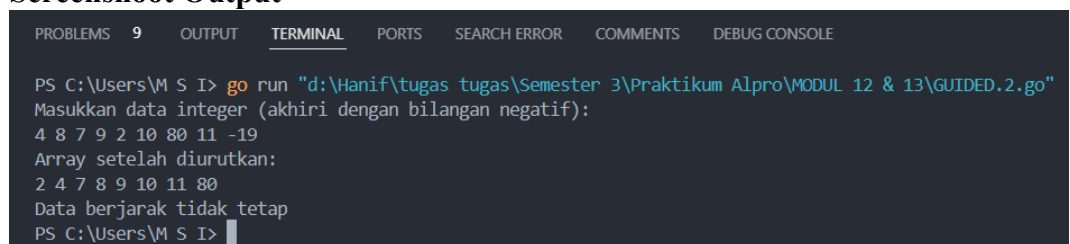
isConstant, difference := isConstantDifference(arr, n)

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

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

```

### Screenshoot Output



```

PROBLEMS 9 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 12 & 13\GUIDED.2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
4 8 7 9 2 10 80 11 -19
Array setelah diurutkan:
2 4 7 8 9 10 11 80
Data berjarak tidak tetap
PS C:\Users\M S I>

```

### Deskripsi Program

Program ini bertujuan untuk mengurutkan data yang dimasukkan oleh pengguna dan memeriksa apakah selisih antara elemen-elemen array yang terurut tetap konstan. Pengguna diminta untuk memasukkan serangkaian angka hingga mereka memasukkan angka negatif, yang menandakan akhir input. Program menggunakan algoritma **Insertion Sort** untuk mengurutkan data, di mana setiap elemen dibandingkan dengan elemen sebelumnya dan disisipkan pada posisi yang

tepat. Setelah array terurut, program memeriksa apakah selisih antara elemen-elemen array tetap konstan menggunakan fungsi `isConstantDifference`. Jika selisihnya konstan, program menampilkan nilai selisih tersebut, namun jika tidak, program memberikan informasi bahwa data berjarak tidak tetap. Program ini juga menangani kasus ketika tidak ada data yang dimasukkan dengan menampilkan pesan yang sesuai. Dengan demikian, program ini tidak hanya mengurutkan data, tetapi juga memberikan informasi tambahan mengenai pola dalam data yang dimasukkan.

### III. UNGUIDED

#### Unguided 1 Sourcecode

```
// 2311102266_Hanif Reyhan Zhafran Arytona
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 rumah 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

```

PROBLEMS 9 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 12 & 13\UNGUIDED.1.go"
Masukkan jumlah rumah kerabat (n): 3

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 5
Masukkan 5 nomor rumah kerabat: 10 4 9 8 2
Nomor rumah terurut untuk daerah 1: 2 4 8 9 10

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 3
Masukkan 3 nomor rumah kerabat: 3 20 11

● Masukkan jumlah nomor rumah kerabat untuk daerah 2: 3
  Masukkan 3 nomor rumah kerabat: 3 20 11
  Masukkan 3 nomor rumah kerabat: 3 20 11
  Nomor rumah terurut untuk daerah 2: 3 11 20
○
Masukkan jumlah nomor rumah kerabat untuk daerah 3: 6
Masukkan 6 nomor rumah kerabat: 22 10 21 39 19 49
Nomor rumah terurut untuk daerah 3: 10 19 21 22 39 49
PS C:\Users\M S I>

```

## Deskripsi Program

Program di atas bertujuan untuk mengurutkan nomor rumah kerabat pada beberapa daerah menggunakan algoritma **Selection Sort**. Pertama, program meminta pengguna untuk memasukkan jumlah daerah yang akan diproses. Setiap daerah kemudian diproses secara terpisah, di mana untuk setiap daerah, pengguna diminta memasukkan jumlah nomor rumah yang ada di daerah tersebut dan



nomor-nomor rumah itu sendiri. Setelah nomor rumah dimasukkan, program akan mengurutkan nomor-nomor tersebut menggunakan algoritma Selection Sort.

Algoritma **Selection Sort** bekerja dengan cara mencari elemen terkecil dalam array dan menukarnya dengan elemen yang ada pada posisi yang tepat. Proses ini dimulai dari elemen pertama dan berlanjut ke elemen-elemen berikutnya, masing-masing menemukan elemen terkecil dari sisa array dan menukarnya dengan elemen pada posisi tersebut. Proses ini diulang hingga seluruh elemen dalam array terurut. Dalam program ini, Selection Sort diterapkan pada setiap daftar nomor rumah yang dimasukkan untuk tiap daerah.

Setelah nomor rumah untuk tiap daerah diurutkan, program kemudian menampilkan nomor rumah yang telah terurut untuk setiap daerah secara terpisah. Output akan menunjukkan daftar nomor rumah yang telah diurutkan dari yang terkecil hingga yang terbesar. Program ini memudahkan pengguna untuk mengelompokkan dan mengurutkan data berdasarkan wilayah, dengan pendekatan yang sederhana dan mudah dipahami menggunakan algoritma Selection Sort.

## Unguided 2

### Sourcecode

```
// 2311102266_Hanif Reyhan Zhafran Arytona

package main

import (
    "fmt"
    "sort"
)

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() {
    var data []int
    var num int
```

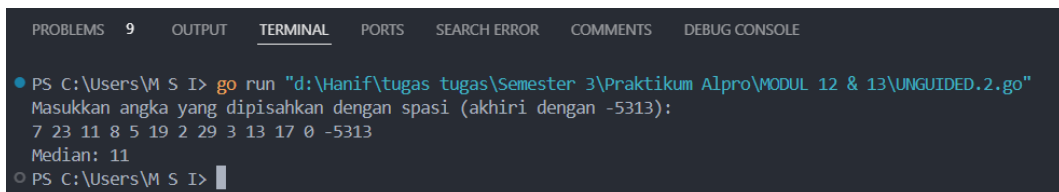
```

    fmt.Println("Masukkan angka yang dipisahkan dengan spasi (akhiri
dengan -5313):")

    for {
        fmt.Scan(&num)
        if num == -5313 {
            break
        }
        if num == 0 {
            sort.Ints(data)
            median := calculateMedian(data)
            fmt.Printf("Median: %.0f\n", median)
        } else {
            data = append(data, num)
        }
    }
}

```

## Screenshoot Output



```

PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 12 & 13\UNGUIDED.2.go"
Masukkan angka yang dipisahkan dengan spasi (akhiri dengan -5313):
7 23 11 8 5 19 2 29 3 13 17 0 -5313
Median: 11
PS C:\Users\M S I>

```

## Deskripsi Program

Program di atas meminta pengguna untuk memasukkan serangkaian angka satu per satu hingga menemukan angka -5313, yang menandakan akhir input. Setiap angka yang dimasukkan selain 0 akan disimpan dalam array `data`. Ketika pengguna memasukkan angka 0, program akan mengurutkan array menggunakan fungsi `sort.Ints` dari pustaka standar Go dan menghitung median dari array yang sudah diurutkan. Median dihitung dengan memeriksa apakah jumlah elemen array ganjil atau genap. Jika jumlahnya genap, median adalah rata-rata dua elemen tengah; jika ganjil, median adalah elemen tengah. Program kemudian mencetak nilai median tersebut. Proses berlanjut hingga pengguna memasukkan -5313, yang akan menghentikan input dan mengakhiri program. Program ini memudahkan pengguna untuk menghitung median data secara dinamis dengan cara yang sederhana dan efisien.

### UNGUIDED 3

#### Sourcecode

```
// 2311102266_Hanif Reyhan Zhafran Arytona

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

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

    var pustaka []Buku
    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)
    }

    terfavorit := pustaka[0]
    for _, buku := range pustaka {
```

```

        if buku.rating > terfavorit.rating {
            terfavorit = buku
        }
    }
    fmt.Println("\nBuku 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)

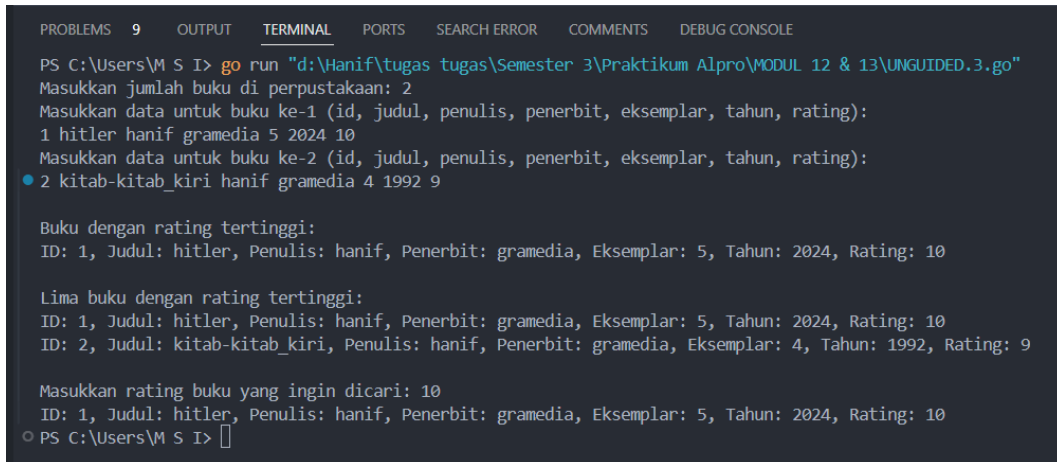
    sort.SliceStable(pustaka, func(i, j int) bool {
        return pustaka[i].rating > pustaka[j].rating
    })

    fmt.Println("\nLima 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)
    }

    var rating int
    fmt.Print("\nMasukkan rating buku yang ingin dicari: ")
    fmt.Scan(&rating)
    ditemukan := false
    for _, buku := range pustaka {
        if buku.rating == rating {
            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.")
    }
}

```

## Screenshoot Output



```
PROBLEMS 9 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE
PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 12 & 13\UNGUIDED.3.go"
Masukkan jumlah buku di perpustakaan: 2
Masukkan data untuk buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 hitler hanif gramedia 5 2024 10
Masukkan data untuk buku ke-2 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 kitab-kitab_kiri hanif gramedia 4 1992 9

Buku dengan rating tertinggi:
ID: 1, Judul: hitler, Penulis: hanif, Penerbit: gramedia, Eksemplar: 5, Tahun: 2024, Rating: 10

Lima buku dengan rating tertinggi:
ID: 1, Judul: hitler, Penulis: hanif, Penerbit: gramedia, Eksemplar: 5, Tahun: 2024, Rating: 10
ID: 2, Judul: kitab-kitab_kiri, Penulis: hanif, Penerbit: gramedia, Eksemplar: 4, Tahun: 1992, Rating: 9

Masukkan rating buku yang ingin dicari: 10
ID: 1, Judul: hitler, Penulis: hanif, Penerbit: gramedia, Eksemplar: 5, Tahun: 2024, Rating: 10
PS C:\Users\M S I>
```

## Deskripsi Program

Program di atas mengelola data buku di perpustakaan dengan cara sederhana namun efisien. Pengguna diminta untuk memasukkan jumlah buku yang akan didaftarkan, dan kemudian input data buku dilakukan satu per satu, termasuk informasi seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Setelah semua data dimasukkan, program pertama-tama mencari dan menampilkan buku dengan rating tertinggi. Kemudian, data buku diurutkan berdasarkan rating menggunakan `sort.SliceStable`, dan lima buku dengan rating tertinggi ditampilkan. Program juga memungkinkan pengguna untuk mencari buku berdasarkan rating tertentu, dan jika ditemukan, buku tersebut akan ditampilkan. Jika tidak ada buku dengan rating tersebut, program memberikan informasi yang sesuai. Program ini menyederhanakan pengelolaan data buku dengan pengurutan dan pencarian berdasarkan rating.