

LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL 12 & 13
PENGURUTAN DATA



Oleh:

NAMA : HAIKAL SATRIATAMA

NIM : 2311102066

KELAS : IF 11 02

S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024

I. DASAR TEORI

Algoritma Selection Sort

Algoritma Selection Sort berfokus pada pencarian nilai ekstrim (terkecil atau terbesar) dalam suatu kumpulan data dan menempatkannya pada posisi yang sesuai. Dalam konteks ini, data akan diurutkan dalam urutan menaik (ascending), di mana data dengan nilai terkecil berada di bagian kiri, dan data dengan nilai terbesar berada di bagian kanan.

Langkah-langkah algoritma Selection Sort adalah sebagai berikut:

1. Temukan nilai terkecil dalam rentang data yang tersisa.
2. Tukar nilai terkecil tersebut dengan nilai yang berada pada posisi paling kiri dalam rentang data yang tersisa.
3. Ulangi langkah ini sampai hanya tersisa satu elemen yang belum terurut.

Selection Sort melibatkan dua proses utama, yaitu:

- Pencarian indeks dari nilai ekstrim (terkecil atau terbesar).
- Proses pertukaran nilai atau swap antara dua elemen.

	Notasi Algoritma	Notasi dalam bahasa Go
1	$i \leftarrow 1$	$i = 1$
2	while $i \leq n-1$ do	for $i \leq n-1$ {
3	$idx_min \leftarrow i - 1$	$idx_min = i - 1$
4	$j \leftarrow i$	$j = i$
5	while $j < n$ do	for $j < n$ {
6	if $a[idx_min] > a[j]$ then	if $a[idx_min] > a[j]$ {
7	$idx_min \leftarrow j$	$idx_min = j$
8	endif	}
9	$j \leftarrow j + 1$	$j = j + 1$
10	endwhile	}
11	$t \leftarrow a[idx_min]$	$t = a[idx_min]$
12	$a[idx_min] \leftarrow a[i-1]$	$a[idx_min] = a[i-1]$
13	$a[i-1] \leftarrow t$	$a[i-1] = t$
14	$i \leftarrow i + 1$	$i = i + 1$
15	endwhile	}

Algoritma Insertion Sort

Insertion Sort bekerja dengan cara menyisipkan setiap elemen yang belum terurut ke dalam posisi yang sesuai dalam bagian data yang sudah terurut. Dalam algoritma ini, tidak dilakukan pencarian nilai ekstrim seperti pada Selection Sort.

Sebaliknya, setiap elemen yang belum terurut akan dicari posisinya melalui pencarian sekuensial.

Langkah-langkah algoritma Insertion Sort adalah sebagai berikut:

1. Untuk setiap data yang belum terurut, cari posisi yang tepat dalam data yang sudah terurut dengan cara membandingkan dan menggeser elemen-elemen yang lebih besar ke kanan, sehingga ada ruang kosong untuk menyisipkan data yang belum terurut.
2. Ulangi langkah tersebut untuk setiap elemen yang belum terurut.

Insertion Sort melibatkan dua proses utama:

- Pencarian posisi yang tepat untuk elemen yang belum terurut secara sekuensial.
- Penyisipan elemen tersebut ke dalam posisi yang sesuai dalam bagian data yang sudah terurut.

II. GUIDED

Guided 1

Sourcecode :

```
package main

import "fmt"

// Fungsi untuk mengurutkan array menggunakan selection sort
func selectionSort(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        maxIdx := i
```

```

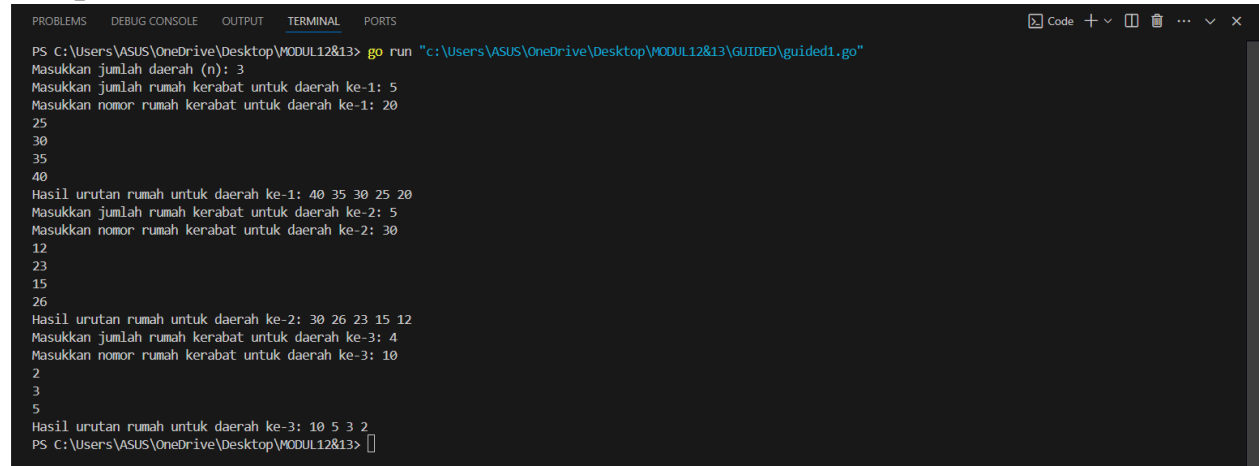
        for j := i + 1; j < n; j++ {
            if arr[j] > arr[maxIdx] { // Cari elemen terbesar
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx], arr[i] // Tukar elemen
    }
}

func main() {
    var n int
    fmt.Print("Masukkan jumlah daerah (n): ")
    fmt.Scan(&n)
    if n <= 0 || n >= 1000 {
        fmt.Println("n harus lebih besar dari 0 dan kurang dari 1000.")
        return
    }
    for i := 0; i < n; i++ {
        var m int
        fmt.Printf("Masukkan jumlah rumah kerabat untuk daerah ke-%d: ", i+1)
        fmt.Scan(&m)
        if m <= 0 || m >= 1000000 {
            fmt.Println("m harus lebih besar dari 0 dan kurang dari 1000000.")
            return
        }
        // Masukkan nomor rumah
        houses := make([]int, m)
        fmt.Printf("Masukkan nomor rumah kerabat untuk daerah ke-%d: ", i+1)
        for j := 0; j < m; j++ {
            fmt.Scan(&houses[j])
        }
        // Urutkan dengan selection sort
        selectionSort(houses)
        // Cetak hasil
        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ", i+1)
        for _, house := range houses {
            fmt.Printf("%d ", house)
        }
    }
}

```

```
        fmt.Println()
    }
}
```

Output :



```
PS C:\Users\ASUS\OneDrive\Desktop\MODUL12&13> go run "c:\Users\ASUS\OneDrive\Desktop\MODUL12&13\GUIDED\guided1.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5
Masukkan nomor rumah kerabat untuk daerah ke-1: 20
25
30
35
40
Hasil urutan rumah untuk daerah ke-1: 40 35 30 25 20
Masukkan jumlah rumah kerabat untuk daerah ke-2: 5
Masukkan nomor rumah kerabat untuk daerah ke-2: 30
12
23
15
26
Hasil urutan rumah untuk daerah ke-2: 30 26 23 15 12
Masukkan jumlah rumah kerabat untuk daerah ke-3: 4
Masukkan nomor rumah kerabat untuk daerah ke-3: 10
2
3
5
Hasil urutan rumah untuk daerah ke-3: 10 5 3 2
PS C:\Users\ASUS\OneDrive\Desktop\MODUL12&13>
```

Penjelasan :

Program ini mengurutkan nomor rumah kerabat di beberapa daerah secara menurun (descending) menggunakan algoritma selection sort. Input berupa jumlah daerah, jumlah rumah per daerah, dan nomor rumah. Hasilnya adalah daftar nomor rumah yang telah diurutkan untuk setiap daerah.

Guided 2

Sourcecode :

```
package main

import (
    "fmt"
    "math"
)

func insertionSort(arr []int) {
```

```

n := len(arr)
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 isDataConsistentlySpaced(arr []int) (bool, int) {
    if len(arr) < 2 {
        return true, 0 // Array dengan kurang dari 2 elemen dianggap
        berjarak

    }
    diff := int(math.Abs(float64(arr[1] - arr[0])))
    for i := 1; i < len(arr)-1; i++ {
        currentDiff := int(math.Abs(float64(arr[i+1] - arr[i])))
        if currentDiff != diff {
            return false, 0 // Jika ada selisih yang berbeda, tidak berjarak
        }
    }
    return true, diff
}

func main() {
    var data []int
    var input int
    fmt.Println("Masukkan data (akhiri dengan bilangan negatif):")
    for {
        fmt.Scan(&input)
        if input < 0 {
            break
        }
        data = append(data, input)
    }
    insertionSort(data)
    isConsistent, diff := isDataConsistentlySpaced(data)
    fmt.Println("Hasil pengurutan:", data)
}

```

```

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

```

Output :

```

PROBLEMS 2 DEBUG CONSOLE OUTPUT TERMINAL PORTS
PS C:\Users\ASUS\OneDrive\Desktop\MODUL12&13> go run "c:\Users\ASUS\OneDrive\Desktop\MODUL12&13\GUIDED\guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6
PS C:\Users\ASUS\OneDrive\Desktop\MODUL12&13>

```

Penjelasan :

Program ini mengurutkan data angka yang dimasukkan pengguna, lalu memeriksa apakah jarak antar elemen dalam data yang diurutkan **konsisten** (tetap). Jika konsisten, program menampilkan jaraknya; jika tidak, program menyatakan jarak tidak tetap.

III. UNGUIDED

Unguided 1

Sourcecode :

```

package main

import (
    "fmt"
    "sort"
)

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

```

```

if n <= 0 || n >= 1000 {
    fmt.Println("n harus lebih besar dari 0 dan kurang dari 1000.")
    return
}

for i := 0; i < n; i++ {
    var m int
    fmt.Printf("Masukkan jumlah rumah kerabat untuk daerah ke-%d: ", i+1)
    fmt.Scan(&m)

    if m <= 0 || m >= 1000000 {
        fmt.Println("m harus lebih besar dari 0 dan kurang dari 1000000.")
        return
    }

    houses := make([]int, m)
    fmt.Printf("Masukkan nomor rumah kerabat untuk daerah ke-%d: ", i+1)
    for j := 0; j < m; j++ {
        fmt.Scan(&houses[j])
    }

    var odd, even []int
    for _, house := range houses {
        if house%2 == 0 {
            even = append(even, house)
        } else {
            odd = append(odd, house)
        }
    }

    sort.Ints(odd)
    sort.Ints(even)

    fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ", i+1)
    for _, house := range odd {
        fmt.Printf("%d ", house)
    }
}

```

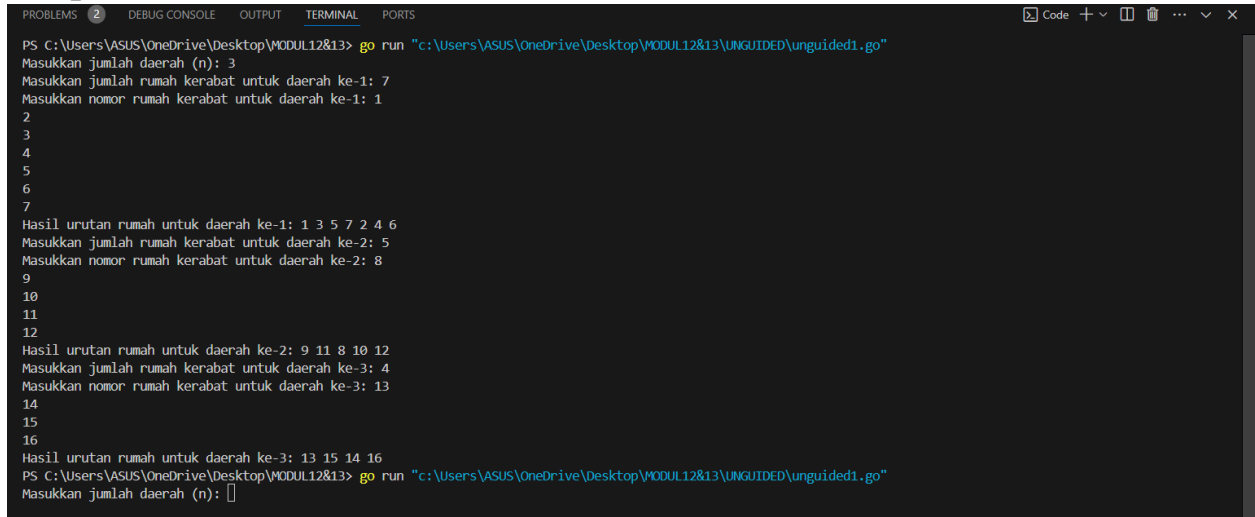


```

    }
    for _, house := range even {
        fmt.Printf("%d ", house)
    }
    fmt.Println()
}
}

```

Output :



```

PROBLEMS 2 DEBUG CONSOLE OUTPUT TERMINAL PORTS
PS C:\Users\ASUS\OneDrive\Desktop\MODUL12&13> go run "c:\Users\ASUS\OneDrive\Desktop\MODUL12&13\UNGUIDED\unguided1.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 7
Masukkan nomor rumah kerabat untuk daerah ke-1: 1
2
3
4
5
6
7
Hasil urutan rumah untuk daerah ke-1: 1 3 5 7 2 4 6
Masukkan jumlah rumah kerabat untuk daerah ke-2: 5
Masukkan nomor rumah kerabat untuk daerah ke-2: 8
9
10
11
12
Hasil urutan rumah untuk daerah ke-2: 9 11 8 10 12
Masukkan jumlah rumah kerabat untuk daerah ke-3: 4
Masukkan nomor rumah kerabat untuk daerah ke-3: 13
14
15
16
Hasil urutan rumah untuk daerah ke-3: 13 15 14 16
PS C:\Users\ASUS\OneDrive\Desktop\MODUL12&13> go run "c:\Users\ASUS\OneDrive\Desktop\MODUL12&13\UNGUIDED\unguided1.go"
Masukkan jumlah daerah (n): 

```

Penjelasan :

Program Go di atas bertujuan untuk menerima input nomor rumah dari beberapa daerah, memisahkannya menjadi nomor ganjil dan genap, lalu mengurutkan masing-masing secara **ascending (menaik)** sebelum menampilkan hasilnya.

Unguided 2

Sourcecode :

```

package main

import (
    "fmt"
    "sort"
)

func hitungMedian(data []int) int {
    n := len(data)

```

```

    if n == 0 {
        return 0
    }
    if n%2 != 0 {
        return data[n/2]
    }
    return (data[n/2-1] + data[n/2]) / 2
}

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

    for {
        var num int
        fmt.Scan(&num)

        if num < 0 {
            break
        }

        if num == 0 {

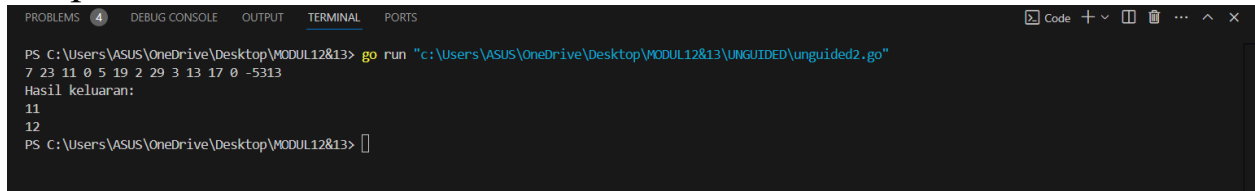
            tempData := make([]int, len(data))
            copy(tempData, data)
            sort.Ints(tempData)
            median := hitungMedian(tempData)
            masukan = append(masukan, median)
        } else {

            data = append(data, num)
        }
    }

    fmt.Println("Hasil keluaran:")
    for _, median := range masukan {
        fmt.Println(median)
    }
}

```

Output :



```
PROBLEMS 4 DEBUG CONSOLE OUTPUT TERMINAL PORTS
PS C:\Users\ASUS\OneDrive\Desktop\MODUL12&13> go run "c:\Users\ASUS\OneDrive\Desktop\MODUL12&13\UNGUIDED\unguided2.go"
7 23 11 0 5 19 2 29 3 13 17 0 -5313
Hasil keluaran:
11
12
PS C:\Users\ASUS\OneDrive\Desktop\MODUL12&13>
```

Penjelasan :

Program ini menghitung **median** dari kumpulan angka yang dimasukkan secara bertahap oleh pengguna. Median dihitung setiap kali angka **0** dimasukkan, dan hasilnya disimpan untuk ditampilkan setelah semua input selesai.

Kesimpulan

- Median adalah nilai tengah dari data yang sudah diurutkan.
- Program menghitung dan menyimpan median setiap kali angka **0** dimasukkan.
- Angka negatif menandakan akhir proses, dan program mencetak semua median yang dihitung.

Unguided 3

Sourcecode :

```
package main

import (
    "fmt"
    "sort"
)

const nMax = 7919

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

```

    Rating    int
}

type DaftarBuku []Buku

func tampilkanBuku(buku Buku) {
    fmt.Printf("ID: %s, Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d,
        Rating: %d\n",
        buku.ID, buku.Judul, buku.Penulis, buku.Penerbit, buku.Tahun,
        buku.Rating)
}

func cetakTerfavorit(pustaka DaftarBuku) {
    if len(pustaka) == 0 {
        fmt.Println("Tidak ada buku di pustaka.")
        return
    }
    terfavorit := pustaka[0]
    for _, buku := range pustaka {
        if buku.Rating > terfavorit.Rating {
            terfavorit = buku
        }
    }
    fmt.Println("Buku Terfavorit:")
    tampilkanBuku(terfavorit)
}

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

func cetak5Terbaru(pustaka DaftarBuku) {
    if len(pustaka) == 0 {
        fmt.Println("Tidak ada buku di pustaka.")
        return
    }
    fmt.Println("5 Buku dengan Rating Tertinggi:")
    for i := 0; i < 5 && i < len(pustaka); i++ {

```

```

        tampilkanBuku(pustaka[i])
    }
}

func cariBuku(pustaka DaftarBuku, rating int) {
    urutkanBuku(pustaka)
    left, right := 0, len(pustaka)-1
    for left <= right {
        mid := (left + right) / 2
        if pustaka[mid].Rating == rating {
            fmt.Println("Buku ditemukan dengan rating tersebut:")
            tampilkanBuku(pustaka[mid])
            return
        } else if pustaka[mid].Rating < rating {
            right = mid - 1
        } else {
            left = mid + 1
        }
    }
    fmt.Println("Tidak ada buku dengan rating seperti itu.")
}

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

    if n <= 0 || n > nMax {
        fmt.Printf("Jumlah buku harus lebih dari 0 dan kurang dari %d.\n",
            nMax)
        return
    }

    pustaka := make(DaftarBuku, n)
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan data untuk buku ke-%d (ID, Judul, Penulis,
            Penerbit, Eksemplar, Tahun, Rating):\n", i+1)
        fmt.Scan(&pustaka[i].ID, &pustaka[i].Judul, &pustaka[i].Penulis,
            &pustaka[i].Penerbit, &pustaka[i].Eksemplar, &pustaka[i].Tahun,
            &pustaka[i].Rating)
    }
}

```

```

    }

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

    cetakTerfavorit(pustaka)
    urutkanBuku(pustaka)
    cetak5Terbaru(pustaka)
    cariBuku(pustaka, ratingCari)
}

```

Output :

```

PROBLEMS 6 DEBUG CONSOLE OUTPUT TERMINAL PORTS
Masukkan jumlah buku: 2
Masukkan data untuk buku ke-1 (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating):
1 Langit Haikal Gramedia 6 2023 7
Masukkan data untuk buku ke-2 (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating):
2 Bulan Haikal Gramedia 7 2023 8
Masukkan rating buku yang ingin dicari: 2
Buku Terfavorit:
ID: 2, Judul: Bulan, Penulis: Haikal, Penerbit: Gramedia, Tahun: 2023, Rating: 8
5 Buku dengan Rating Tertinggi:
ID: 2, Judul: Bulan, Penulis: Haikal, Penerbit: Gramedia, Tahun: 2023, Rating: 8
ID: 1, Judul: Langit, Penulis: Haikal, Penerbit: Gramedia, Tahun: 2023, Rating: 7
Tidak ada buku dengan rating seperti itu.
PS C:\Users\ASUS\OneDrive\Desktop\MODUL12&13>

```

Penjelasan :

Program di atas bertujuan untuk mengelola data koleksi buku dalam sebuah pustaka. Aplikasi ini dapat menampilkan buku terbaik, mengurutkan koleksi berdasarkan rating, dan mencari buku berdasarkan rating yang diinginkan.

Program ini memberikan solusi untuk pengelolaan data buku yang memungkinkan pengguna:

- Memasukkan data buku.
- Menemukan buku terbaik.
- Mengurutkan dan mencetak buku berdasarkan rating.
- Mencari buku secara efisien menggunakan binary search.