

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL XII
PENGURUTAN DATA**



Oleh:

MUHAMAD IHSAN

2311102077

IF - 11 – 02

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024**

I. DASAR TEORI

1. Ide Algoritma Selection Sort

Pengurutan secara seleksi Ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian meletakkan pada posisi yang seharusnya. Pada penjelasan berikut ini data akan diurut membesar (ascending), dan data dengan Indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

- 1) Cari nilai terkecil di dalam rentang data tersisa
- 2) Pindahkan/tukar tempat dengan data yang berada pada posisi paling kiri pada rentang data tersisa tersebut.
- 3) Ulangi proses ini sampai tersisa hanya satu data saja.

Algoritma ini dikenal juga dengan nama Selection Sort, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian indeks nilai ekstrim dan proses pertukaran dua nilai atau swap.

	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	}

2. Algoritma selection sort

Adapun algoritma selection sort pada untuk mengurutkan array bertipe data bilangan bulat secara membesar atau ascending adalah sebagai berikut ini.

```
..    ..
5    type arrInt [4321]int
..    ..
15   func selectionSort1(T *arrInt, n int){
16   /* I.S. terdefinisi array T yang berisi n bilangan bulat
17      F.S. array T terurut secara ascending atau membesar dengan SELECTION SORT */
18      var t, i, j, idx_min int
19
```

```

20     i = 1
21     for i <= n-1 {
22         idx_min = i - 1
23         j = i
24         for j < n {
25             if T[idx_min] > T[j] {
26                 idx_min = j
27             }
28             j = j + 1
29         }
30         t = T[idx_min]
31         T[idx_min] = T[i-1]
32         T[i-1] = t
33         i = i + 1
34     }
35 }

```

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada saat proses perbandingan nilai ekstrim, kemudian tipe data dari variabel t sama dengan struct dari arraynya.

```

..     ...
5     type mahasiswa struct {
..         nama, nim, kelas, jurusan string
..         ipk float64
..     }
..     type arrMhs [2023]mahasiswa
..     ...
15 func selectionSort2(T * arrMhs, n int){
16     /* I.S. terdefinisi array T yang berisi n data mahasiswa
17        F.S. array T terurut secara ascending atau membesar berdasarkan ipk dengan
18        menggunakan algoritma SELECTION SORT */
19     var i, j, idx_min int
20     var t mahasiswa
21     i = 1
22     for i <= n-1 {
23         idx_min = i - 1
24         j = i
25         for j < n {
26             if T[idx_min].ipk > T[j].ipk {
27                 idx_min = j
28             }
29             j = j + 1
30         }
31         t = T[idx_min]
32         T[idx_min] = T[i-1]
33         T[i-1] = t
34         i = i + 1
35     }
36 }

```

II. GUIDED

Guided 1 selection sort

```
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)
```

```

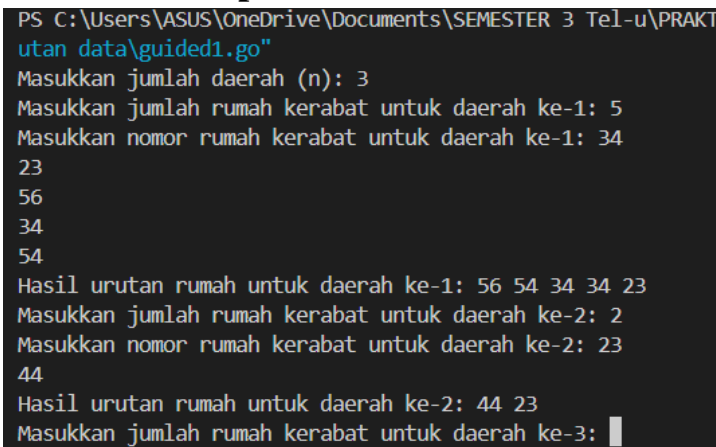
        fmt.Printf("Masukkan nomor rumah kerabat untuk daerah ke-%d: ", i+1)
        for j := 0; j < m; j++ {
            fmt.Scan(&houses[j])
        }

        // 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()
    }
}

```

Screenshot output



```

PS C:\Users\ASUS\OneDrive\Documents\SEMESTER 3 Tel-u\PRAKTIKUM 1> go run data\guided1.go
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5
Masukkan nomor rumah kerabat untuk daerah ke-1: 34
23
56
34
54
Hasil urutan rumah untuk daerah ke-1: 56 54 34 34 23
Masukkan jumlah rumah kerabat untuk daerah ke-2: 2
Masukkan nomor rumah kerabat untuk daerah ke-2: 23
44
Hasil urutan rumah untuk daerah ke-2: 44 23
Masukkan jumlah rumah kerabat untuk daerah ke-3: 

```

Deskripsi program

Program ini mengurutkan nomor rumah kerabat di beberapa daerah dari terbesar ke terkecil menggunakan algoritma *selection sort*. Pengguna memasukkan jumlah daerah (n) dan jumlah rumah di setiap daerah (m), kemudian memasukkan nomor rumah untuk masing-masing daerah. Setelah diurutkan, program mencetak nomor rumah yang telah terurut untuk setiap daerah. Jika input n atau m tidak valid, program menampilkan pesan kesalahan dan berhenti.

Guided 2 insertion sort

```
package main

import (
    "fmt"
    "math"
)

// Fungsi insertion sort untuk mengurutkan array
func insertionSort(arr []int) {
    n := len(arr)
    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 data berjarak tetap
func isDataConsistentlySpaced(arr []int) (bool, int) {
    if len(arr) < 2 {
        return true, 0 // Array dengan kurang dari 2 elemen dianggap
        berjarak tetap
    }

    // Hitung selisih awal
    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
            tetap
        }
    }

    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)
    }

    // Urutkan data menggunakan insertion sort
    insertionSort(data)

    // Periksa apakah data berjarak tetap
    isConsistent, diff := isDataConsistentlySpaced(data)

    // Cetak hasil
    fmt.Println("Hasil pengurutan:", data)
    if isConsistent {
        fmt.Printf("Data berjarak %d\n", diff)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}

```

Screenshot output

```

utan data\guided1insertionsort.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\Documents\SEMESTER 3 Tel-u\PRAKTIKUM ALPRO2\Golang>

```

Deskripsi program

Program ini menerima serangkaian bilangan dari pengguna, mengurutkannya dengan algoritma *insertion sort*, lalu memeriksa apakah jarak antar elemen array terurut tersebut konstan. Jika jaraknya tetap, program mencetak jaraknya; jika tidak, program menyatakan jarak tidak tetap. Input berakhir saat pengguna memasukkan bilangan negatif.

III. UNGUIDED

Soal Latihan selection sort 2 dan 3

- 2) 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

Source code

```
package main

import "fmt"

func selectionSort(arr []int, ascending bool) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        idx := i
        for j := i + 1; j < n; j++ {
            if (ascending && arr[j] < arr[idx]) || (!ascending
&& arr[j] > arr[idx]) {
                idx = j
            }
        }
        arr[i], arr[idx] = arr[idx], arr[i]
    }
}

func main() {
    var n int
    fmt.Scan(&n)

    for i := 0; i < n; i++ {
```



```

var m int
fmt.Scan(&m)
houses := make([]int, m)
odd := []int{}
even := []int{}

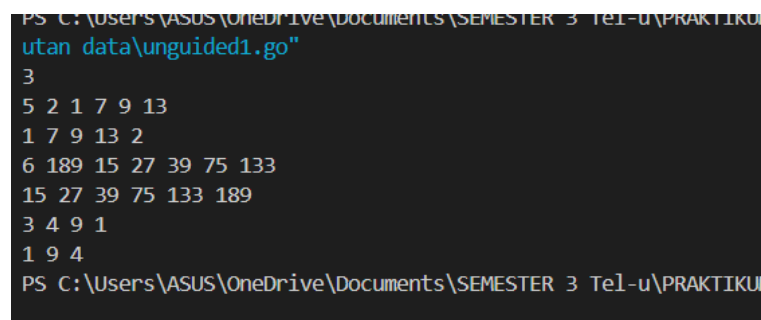
for j := 0; j < m; j++ {
    fmt.Scan(&houses[j])
    if houses[j]%2 == 0 {
        even = append(even, houses[j])
    } else {
        odd = append(odd, houses[j])
    }
}

selectionSort(odd, true)
selectionSort(even, false)

for i := 0; i < len(odd); i++ {
    fmt.Print(odd[i], " ")
}
for i := 0; i < len(even); i++ {
    fmt.Print(even[i], " ")
}
fmt.Println()
}

```

Screenshot output



```

PS C:\Users\ASUS\OneDrive\Documents\SEMESTER 3 Tel-u\PRAKTIKUM\Koding\Go\untutan data\unguided1.go"
3
5 2 1 7 9 13
1 7 9 13 2
6 189 15 27 39 75 133
15 27 39 75 133 189
3 4 9 1
1 9 4
PS C:\Users\ASUS\OneDrive\Documents\SEMESTER 3 Tel-u\PRAKTIKUM\Koding\Go\untutan data\unguided1.go"

```

Deskripsi

Program ini memisahkan angka menjadi ganjil dan genap, lalu mengurutkan ganjil secara naik dan genap secara turun. Hasilnya digabungkan dengan ganjil terlebih dahulu, diikuti genap. Contoh: untuk input 5, 2, 3, 8, 1, outputnya adalah 1, 3, 5, 8, 2.

- 3) 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

Source code

```
package main

import "fmt"

func selectionSort(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        idx := i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[idx] {
                idx = j
            }
        }
        arr[i], arr[idx] = arr[idx], arr[i]
    }
}

func printMedian(arr []int) {
    selectionSort(arr)
    n := len(arr)
    if n%2 == 1 {
        fmt.Println(arr[n/2])
    }
}
```

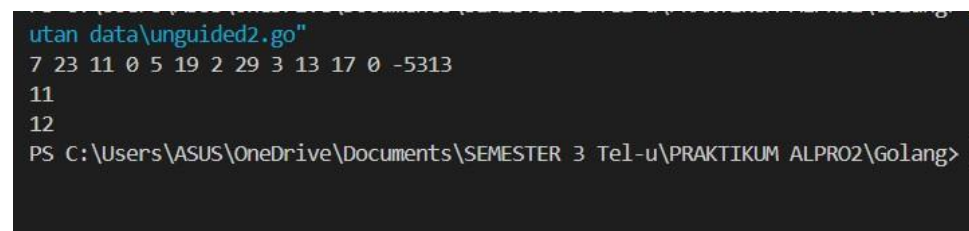
```

    } else {
        mid := n / 2
        median := (arr[mid-1] + arr[mid]) / 2
        fmt.Println(median)
    }
}

func main() {
    var data []int
    var input int
    for {
        fmt.Scan(&input)
        if input == -5313 {
            break
        }
        if input == 0 {
            printMedian(data)
        } else {
            data = append(data, input)
        }
    }
}

```

Screenshot output



The screenshot shows a terminal window with the following text:

```

utan data\unguided2.go"
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
PS C:\Users\ASUS\OneDrive\Documents\SEMESTER 3 Tel-u\PRAKTIKUM ALPRO2\Golang>

```

Deskripsi program

Program ini membaca angka dari pengguna dan mencetak median setiap kali input 0 diberikan. Angka diurutkan menggunakan *selection sort*, lalu median dihitung: jika jumlah angka ganjil, median adalah elemen tengah; jika genap, median adalah rata-rata dua elemen tengah. Program berakhir saat input -5313 dimasukkan. Contoh: untuk input 5, 3, 8, 0, median yang dicetak adalah 5.

Soal Latihan insertion sort

- 2) 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.

SOURCE CODE

```
package main

import "fmt"

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

type DaftarBuku struct {
    Pustaka []Buku
}

func DaftarkanBuku(n int) DaftarBuku {
    var pustaka DaftarBuku
    for i := 0; i < n; i++ {
        var buku Buku
```

```

        fmt.Scan(&buku.ID, &buku.Judul, &buku.Penulis, &buku.Penerbit,
        &buku.Eksemplar, &buku.Tahun, &buku.Rating)
        pustaka.Pustaka = append(pustaka.Pustaka, buku)
    }
    return pustaka
}

func CetakTerfavorit(pustaka DaftarBuku) {
    if len(pustaka.Pustaka) == 0 {
        fmt.Println("Tidak ada buku di perpustakaan.")
        return
    }
    terfavorit := pustaka.Pustaka[0]
    for i := 1; i < len(pustaka.Pustaka); i++ {
        if pustaka.Pustaka[i].Rating > terfavorit.Rating {
            terfavorit = pustaka.Pustaka[i]
        }
    }
    fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d, Rating:
    %d\n",
        terfavorit.Judul, terfavorit.Penulis, terfavorit.Penerbit,
    terfavorit.Tahun, terfavorit.Rating)
}

func UrutBuku(pustaka *DaftarBuku) {
    n := len(pustaka.Pustaka)
    for i := 0; i < n-1; i++ {
        for j := 0; j < n-i-1; j++ {
            if pustaka.Pustaka[j].Rating < pustaka.Pustaka[j+1].Rating {
                pustaka.Pustaka[j], pustaka.Pustaka[j+1] =
                pustaka.Pustaka[j+1], pustaka.Pustaka[j]
            }
        }
    }
}

func Cetak5Terbaru(pustaka DaftarBuku) {
    if len(pustaka.Pustaka) == 0 {
        fmt.Println("Tidak ada buku di perpustakaan.")
        return
    }
    limit := 5
    if len(pustaka.Pustaka) < 5 {
        limit = len(pustaka.Pustaka)
    }
    for i := 0; i < limit; i++ {

```

```

        buku := pustaka.Pustaka[i]
        fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d, Rating:
%d\n",
                buku.Judul, buku.Penulis, buku.Penerbit, buku.Tahun,
buku.Rating)
    }
}

func CariBuku(pustaka DaftarBuku, rating int) {
    UrutBuku(&pustaka)
    low, high := 0, len(pustaka.Pustaka)-1
    found := false
    for low <= high {
        mid := (low + high) / 2
        if pustaka.Pustaka[mid].Rating == rating {
            found = true
            buku := pustaka.Pustaka[mid]
            fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d,
Eksemplar: %d, Rating: %d\n",
                    buku.Judul, buku.Penulis, buku.Penerbit, buku.Tahun,
buku.Eksemplar, buku.Rating)
            break
        } else if pustaka.Pustaka[mid].Rating < rating {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
    if !found {
        fmt.Println("Tidak ada buku dengan rating tersebut.")
    }
}

func main() {
    var n int
    fmt.Scan(&n)
    pustaka := DaftarkanBuku(n)
    CetakTerfavorit(pustaka)
    UrutBuku(&pustaka)
    Cetak5Terbaru(pustaka)
    var rating int
    fmt.Scan(&rating)
    CariBuku(pustaka, rating)
}

```

Screenshot output

```
3
01 akukamu nyong gpu 10 2024 10
02 gatau kamu nasmedia 9 2023 9
03 blast make megapres 8 2022 8
Judul: akukamu, Penulis: nyong, Penerbit: gpu, Tahun: 2024, Rating: 10
Judul: akukamu, Penulis: nyong, Penerbit: gpu, Tahun: 2024, Rating: 10
Judul: gatau, Penulis: kamu, Penerbit: nasmedia, Tahun: 2023, Rating: 9
Judul: blast, Penulis: make, Penerbit: megapres, Tahun: 2022, Rating: 8
8
Judul: blast, Penulis: make, Penerbit: megapres, Tahun: 2022, Eksemplar: 8, Rating: 8
PS C:\Users\ASUS\OneDrive\Documents\SEMESTER 3 Tel-u\PRAKTIKUM ALPRO2\Golang>
```

Deskripsi program

Program ini mengelola koleksi buku perpustakaan dengan fitur pendaftaran buku, pencetakan buku terbaik, pengurutan berdasarkan rating, pencetakan lima buku terbaru, dan pencarian buku berdasarkan rating. Pengguna memasukkan data buku, lalu program mencetak buku dengan rating tertinggi, mengurutkan koleksi, menampilkan lima buku teratas, dan mencari buku berdasarkan rating. Jika buku ditemukan, detailnya dicetak; jika tidak, pesan "tidak ditemukan" ditampilkan.