

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL X

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Disusun Oleh :

Reza Alvonzo / 2311102026

IF-11-06

Dosen Pengampu :

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pencarian nilai ekstrem (maksimum atau minimum) dalam suatu kumpulan data adalah salah satu operasi dasar dalam pemrograman. Di Golang, konsep ini dapat diimplementasikan menggunakan struktur data seperti **array**, **slice**, atau bahkan dalam **struct** yang memiliki atribut numerik.

1. Konsep Nilai Ekstrem

Nilai ekstrem merujuk pada:

Nilai Minimum: Nilai terkecil dalam kumpulan data.

Nilai Maksimum: Nilai terbesar dalam kumpulan data.

Pencarian nilai ekstrem melibatkan iterasi melalui kumpulan data dan membandingkan setiap elemen dengan nilai yang sudah ditemukan sebelumnya.

2. Kompleksitas Waktu

Pencarian nilai ekstrem melibatkan iterasi penuh melalui data:

Best Case, Worst Case, dan Average Case: Semua sama dengan $O(n)$, di mana n adalah jumlah elemen dalam kumpulan data.

Kesimpulan

Pencarian nilai ekstrem di Golang mengandalkan konsep iterasi sederhana dan perbandingan elemen. Dengan memanfaatkan tipe data seperti array, slice, dan struct, kita dapat mengimplementasikan pencarian yang fleksibel sesuai kebutuhan aplikasi.

II. GUIDED

1. Guided 1

Sourcecode

```
package main

import "fmt"

// Mendeklarasikan tipe data array arrInt dengan panjang
2023
type arrInt [2023]int

// Fungsi untuk mencari indeks elemen terkecil dalam array
func terkecil(tabInt arrInt, n int) int {
    var idx int = 0 // idx menyimpan indeks elemen
    terkecil
    var j int = 1
    for j < n {
        if tabInt[idx] > tabInt[j] {
            idx = j // Simpan indeks j jika elemen di
            indeks j lebih kecil
        }
        j = j + 1
    }
    return idx
}

// Fungsi main untuk menguji fungsi terkecil
func main() {
    var n int
    var tab arrInt

    // Meminta input jumlah elemen array
    fmt.Print("Masukkan jumlah elemen (maks 2023): ")
    fmt.Scan(&n)

    // Validasi input jumlah elemen
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah elemen harus antara 1 dan
2023.")
        return
    }

    // Memasukkan elemen-elemen array
    fmt.Println("Masukkan elemen-elemen array:")
    for i := 0; i < n; i++ {
        fmt.Print("Elemen ke-", i+1, ": ")
        fmt.Scan(&tab[i])
    }
}
```

```
// Memanggil fungsi terkecil untuk menemukan indeks  
elemen terkecil  
idxMin := terkecil(tab, n)  
  
// Menampilkan nilai dan indeks terkecil  
fmt.Println("Nilai terkecil dalam array adalah:",  
tab[idxMin], "pada indeks:", idxMin)  
}
```

Screenshoot Output

```
Masukkan jumlah elemen (maks 2023): 2  
Masukkan elemen-elemen array:  
Elemen ke-1: 7  
Elemen ke-2: 12  
Nilai terkecil dalam array adalah: 7 pada indeks: 0
```

Deskripsi Program

Program Go ini dibuat untuk mencari indeks dari elemen terkecil dalam sebuah array. Mendefinisikan tipe array `arrInt` dengan panjang tetap 2023. Fungsi `terkecil()` untuk mencari indeks elemen minimal. Fungsi `main()` untuk input dan pengujian.

2. Guided 2

Sourcecode

```
package main

import "fmt"

// Definisi struct mahasiswa dengan atribut nama, nim,
kelas, jurusan, dan ipk
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk                                float64
}

// Definisi tipe data array mahasiswa dengan kapasitas
maksimal 2023
type arrMhs [2023]mahasiswa

// Fungsi untuk mencari IPK tertinggi dalam array
mahasiswa
func ipk(T arrMhs, n int) float64 {
    var tertinggi float64 = T[0].ipk
    var j int = 1
    for j < n {
        if tertinggi < T[j].ipk {
            tertinggi = T[j].ipk
        }
        j = j + 1
    }
    return tertinggi
}

// Fungsi main untuk mengisi data mahasiswa dan mencari
IPK tertinggi
func main() {
    var n int
    var dataMhs arrMhs

    // Meminta input jumlah mahasiswa
    fmt.Print("Masukkan jumlah mahasiswa (maks 2023): ")
    fmt.Scan(&n)

    // Validasi jumlah mahasiswa yang dimasukkan
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah mahasiswa harus antara 1 dan
2023.")
        return
    }

    // Mengisi data mahasiswa
    for i := 0; i < n; i++ {
```

```

        fmt.Printf("\nMasukkan data mahasiswa ke-%d\n",
i+1)
        fmt.Print("Nama: ")
        fmt.Scan(&dataMhs[i].nama)
        fmt.Print("NIM: ")
        fmt.Scan(&dataMhs[i].nim)
        fmt.Print("Kelas: ")
        fmt.Scan(&dataMhs[i].kelas)
        fmt.Print("Jurusan: ")
        fmt.Scan(&dataMhs[i].jurusan)
        fmt.Print("IPK: ")
        fmt.Scan(&dataMhs[i].ipk)
    }

    // Mencari dan menampilkan IPK tertinggi
    tertinggi := ipk(dataMhs, n)
    fmt.Printf("\nIPK tertinggi dari %d mahasiswa adalah:
%.2f\n", n, tertinggi)

}

```

Screenshoot Output

```

Masukkan jumlah mahasiswa (maks 2023): 2

Masukkan data mahasiswa ke-1
Nama: Reza
NIM: 2311102026
Kelas: 06
Jurusan: IF
IPK: 3.2

Masukkan data mahasiswa ke-2
Nama: Oland
NIM: 2311102010
Kelas: 06
Jurusan: IF
IPK: 3.4

IPK tertinggi dari 2 mahasiswa adalah: 3.40

```

Deskripsi Program

program Go ini yang mengelola data mahasiswa dan mencari IPK tertinggi. Program meminta input jumlah mahasiswa (n), Melakukan validasi input ($1 \leq n \leq 2023$), Mengisi data n mahasiswa (nama, NIM, kelas, jurusan, IPK) , Mencari IPK tertinggi menggunakan fungsi ipk(), Menampilkan hasil IPK tertinggi

.

III. UNGUIDED

Unguided 1

Sourcecode

```
package main
import "fmt"

func main() {
    var N int
    var berat [1000]float64

    fmt.Print("Masukkan jumlah anak kelinci ")
    fmt.Scan(&N)
    fmt.Print("Masukkan berat anak kelinci ")
    for i := 0; i < N; i++ {
        fmt.Scan(&berat[i])
    }

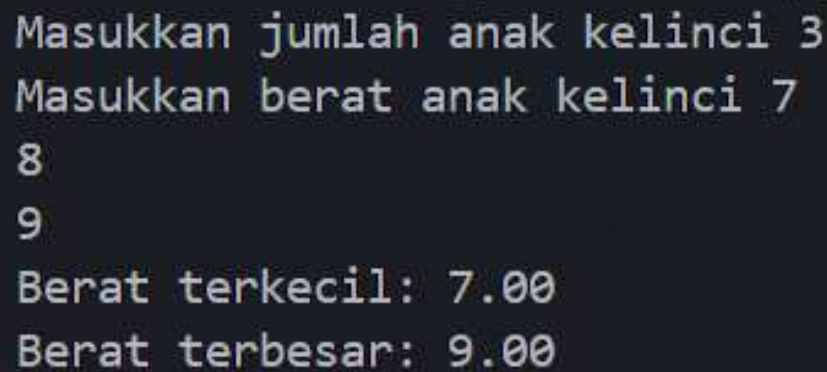
    min := berat[0]
    max := berat[0]

    for i := 0; i < N; i++ {
        if berat[i] < min {
            min = berat[i]
        }
        if berat[i] > max {
            max = berat[i]
        }
    }

    fmt.Printf("Berat terkecil: %.2f\n", min)
    fmt.Printf("Berat terbesar: %.2f\n", max)
}

//Reza Alvonzo IF 11 06 2311102026//
```


Screenshoot Output



```
Masukkan jumlah anak kelinci 3
Masukkan berat anak kelinci 7
8
9
Berat terkecil: 7.00
Berat terbesar: 9.00
```

Deskripsi Program

Program ini adalah aplikasi sederhana untuk mencatat dan menganalisis berat anak kelinci. Program meminta pengguna memasukkan jumlah anak kelinci yang akan dicatat beratnya, Kemudian pengguna diminta memasukkan berat untuk setiap anak kelinci satu per satu, Program akan mencari berat terkecil dan terbesar dari semua data yang dimasukkan, Terakhir, program menampilkan hasil berat terkecil dan terbesar dengan 2 angka decimal.

Unguided 2

Sourcecode

```
package main
import "fmt"

func main() {
    var x, y int
    fmt.Print("Masukkan jumlah ikan dan kapasitas wadah: ")
    fmt.Scan(&x, &y)

    berat := make([]float64, x)
    fmt.Println("Masukkan berat tiap ikan: ")
    for i := 0; i < x; i++ {
        fmt.Scan(&berat[i])
    }

    jumlahWadah := (x + y - 1) / y
    totalBeratWadah := make([]float64, jumlahWadah)
```

```

        for i := 0; i < x; i++ {
            indeksWadah := i / y
            totalBeratWadah[indeksWadah] += berat[i]
        }

        fmt.Print("Berat total wadah: ")
        for _, total := range totalBeratWadah {
            fmt.Printf("%.2f ", total)
        }
        fmt.Println()

        fmt.Print("Rata-rata berat tiap wadah: ")
        for _, total := range totalBeratWadah {
            rataRata := total / float64(y)
            fmt.Printf("%.2f ", rataRata)
        }
        fmt.Println()
    }

    //Reza Alvonzo IF 11 06 2311102026//

```

Screenshoot Output

```

Masukkan jumlah ikan dan kapasitas wadah: 2 20
Masukkan berat tiap ikan:
5
6
Berat total wadah: 11.00
Rata-rata berat tiap wadah: 0.55

```

Deskripsi Program

Program ini adalah aplikasi untuk mengelola distribusi ikan ke dalam wadah dan menghitung statistiknya. Membagi sejumlah ikan ke dalam beberapa wadah dengan kapasitas tertentu, Menghitung total berat ikan di setiap wadah, Menghitung rata-rata berat ikan per wadah.

Unguided 3

Sourcecode

```
package main
import (
    "fmt"
)
//2311102064 Aji Tri Prasetyo
type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita, x int, bMin, bMax
*float64){
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]
    for i := 1; i < x; i++ {
        if arrBerat[i] > *bMax {
            *bMax = arrBerat[i]
        }
        if arrBerat[i] < *bMin {
            *bMin = arrBerat[i]
        }
    }
}

func rerata(arrBerat arrBalita, x int) float64{
    sum:= 0.0
    for i := 0; i < x; i++ {
        sum += arrBerat[i]
    }
    return sum/ (float64(x))
}

func main(){
    var x int
    var arrBalita[100] float64
    var bMin,bMax float64

    fmt.Print("Masukkan banyak data berat balita: ")
    fmt.Scan(&x)

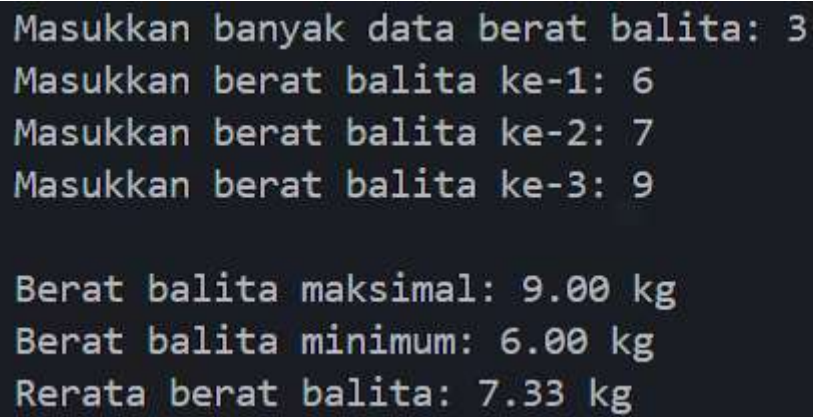
    for i := 0; i < x; i++ {
        fmt.Printf("Masukkan berat balita ke-%d: ",i+1)
        fmt.Scan(&arrBalita[i])
    }
    fmt.Println()

    hitungMinMax(arrBalita,x,&bMin,&bMax)
    avg := rerata(arrBalita,x)

    fmt.Printf("Berat balita maksimal: %.2f kg\n", bMax)
    fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)
```

```
    fmt.Printf("Rerata berat balita: %.2f kg", avg )  
}  
  
//Reza Alvonzo IF 11 06 2311102026//
```

Screenshoot Output



```
Masukkan banyak data berat balita: 3  
Masukkan berat balita ke-1: 6  
Masukkan berat balita ke-2: 7  
Masukkan berat balita ke-3: 9  
  
Berat balita maksimal: 9.00 kg  
Berat balita minimum: 6.00 kg  
Rerata berat balita: 7.33 kg
```

Deskripsi Program

Program ini adalah aplikasi untuk menganalisis data berat badan balita. Program ini menggunakan pointer dan fungsi terpisah untuk menghitung statistik berat balita.