LAPORAN PRAKTIKUM ALGORITMA DAN PEMROGRAMAN 2

MODUL 11

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Disusun Oleh :

Fahri Ramadhan / 2311102024

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 adalah proses untuk menemukan elemen terbesar (maksimum) dan terkecil (minimum) dalam suatu himpunan data. Nilai ekstrem ini penting karena dapat memberikan informasi penting tentang batasan, pola, atau anomali dalam dataset. Nilai maksimum merepresentasikan elemen dengan nilai tertinggi, sedangkan nilai minimum menunjukkan elemen dengan nilai terendah. Proses pencarian biasanya dilakukan dengan membandingkan setiap elemen dalam data secara langsung, yang memiliki kompleksitas waktu O(n)O(n)O(n).

Selain metode sederhana ini, pendekatan lain seperti penggunaan struktur data heap atau algoritma divide and conquer dapat digunakan untuk meningkatkan efisiensi, terutama dalam dataset besar. Nilai ekstrem memiliki banyak aplikasi, seperti mendeteksi anomali dalam analisis keuangan, membantu pengambilan keputusan dalam optimasi, atau memantau fenomena lingkungan seperti suhu ekstrem. Oleh karena itu, pencarian nilai ekstrem merupakan langkah penting dalam analisis data untuk mengidentifikasi informasi kunci dari suatu dataset.

II. GUIDED

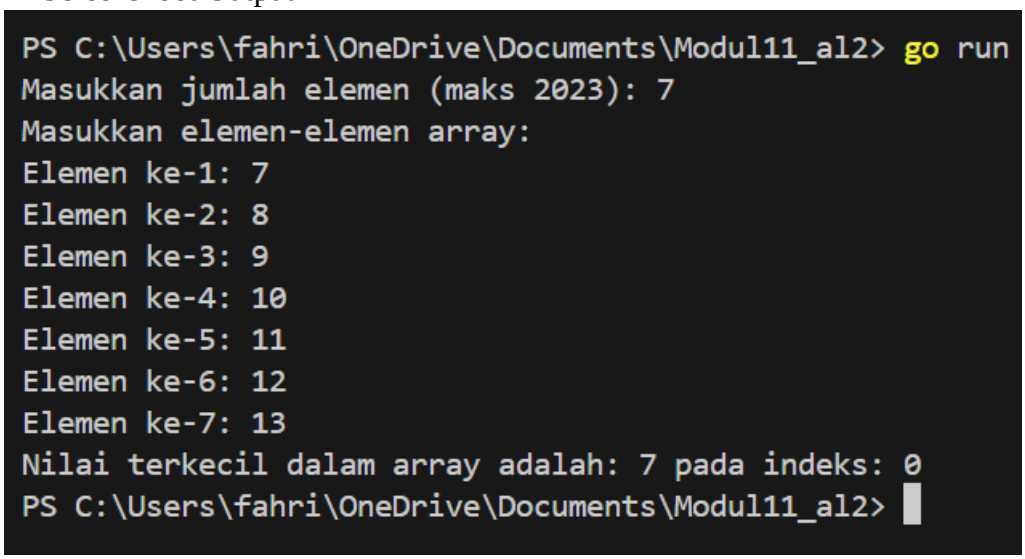
1. Guided 1

Sourcecode

```
//2311102024_FahriRamadhan
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



Deskripsi Program

Variabel idx diinisialisasi dengan 0, yaitu indeks elemen pertama. Variabel j dimulai dari 1, karena elemen pertama (idx = 0) dianggap sebagai elemen terkecil awal. Dalam for loop, setiap elemen array diperiksa: Jika elemen tabInt[j] lebih kecil dari elemen tabInt[idx], maka idx diubah menjadi j. Fungsi mengembalikan idx yang berisi indeks elemen terkecil.

2. Guided 2

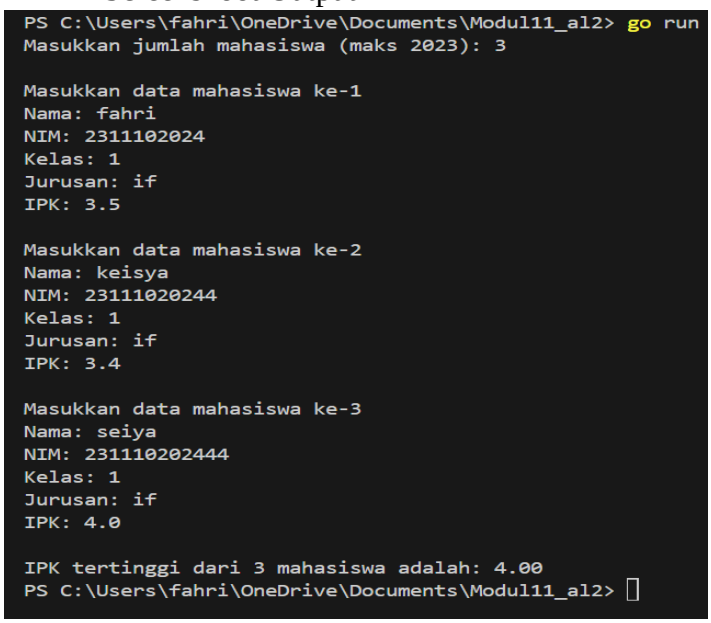
Sourcecode

```
//2311102024_FahriRamadhan

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 mahasiswaa
func ipk(T arrMhs, n int) float64 {
   var tertinggi float64 = T[0].ipk
```

```
    for j := 1; j < n; j++ {
        if tertinggi < T[j].ipk {
            tertinggi = T[j].ipk
        }
    }
    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



Deskripsi Program

Validasi IPK: Program saat ini tidak memeriksa apakah nilai ipk berada dalam rentang valid (0.0 - 4.0). Validasi ini dapat ditambahkan untuk menghindari input salah. Error Handling: Program tidak menangani input non-numerik untuk n atau ipk, sehingga berisiko gagal jika ada kesalahan input.

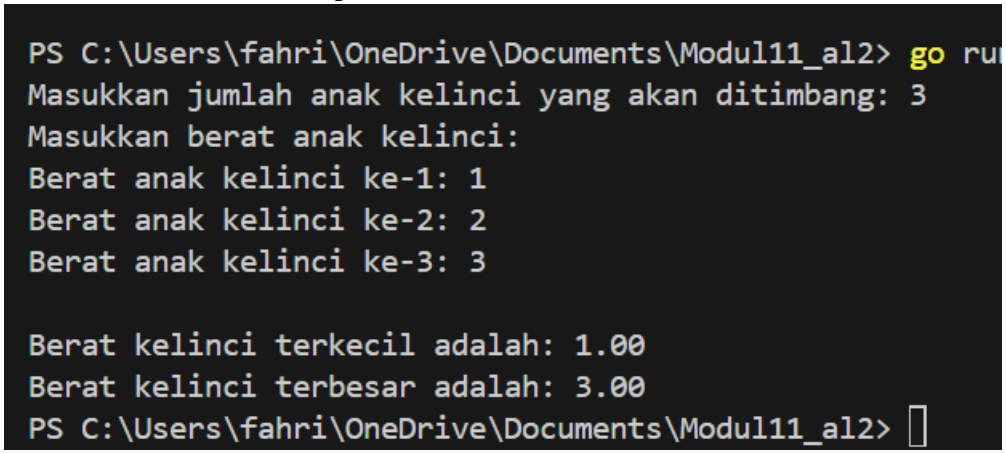
III. UNGUIDED

Unguided 1 Sourcecode

```
//2311102024_FahriRamadhan

 package main

import (
    "fmt"
)

func cariBerat(berat []float64, n int)
(float64, float64) {
    terkecil := berat[0]
    terbesar := berat[0]

    for i := 1; i < n; i++ {
         if berat[i] < terkecil {
              terkecil = berat[i]
         }
         if berat[i] > terbesar {
              terbesar = berat[i]
         }
    }

    return terkecil, terbesar
}

func main() {
    var n int

    fmt.Print("Masukkan  jumlah  anak
kelinci yang akan ditimbang: ")
    fmt.Scan(&n)

    if n < 1 || n > 1000 {
         fmt.Println("Jumlah     anak
kelinci harus antara 1 dan 1000.")
```

```
            return
        }

        berat := make([]float64, n)

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

        terkecil,        terbesar        :=
cariBerat(berat, n)

        fmt.Printf("\nBerat        kelinci
terkecil adalah: %.2f\n", terkecil)
        fmt.Printf("Berat kelinci terbesar
adalah: %.2f\n", terbesar)
}
```

Screenshoot Output



Deskripsi Program

Array berat: Sebuah array dengan kapasitas maksimum 1000 elemen yang digunakan untuk menyimpan berat masing-masing anak kelinci. Tipe datanya adalah float64, yang memungkinkan penyimpanan nilai desimal. Program meminta pengguna untuk memasukkan: Jumlah anak kelinci (N). Berat masing-masing anak kelinci yang disimpan dalam array berat.

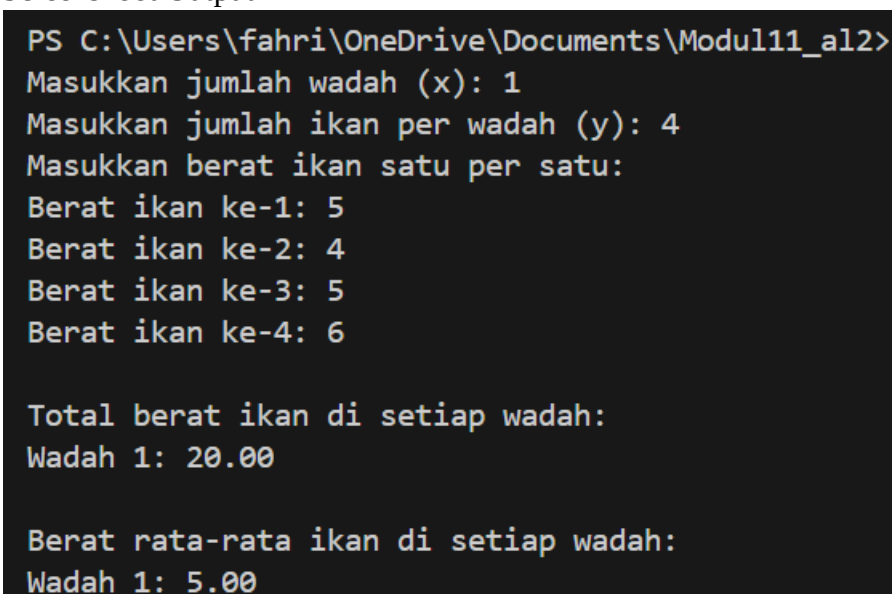
Unguided 2

Sourcecode

```
//2311102024_FahriRamadhan

package main

import (
	"fmt"
)

func main() {
	var x, y int

	fmt.Print("Masukkan jumlah wadah (x): ")
	fmt.Scan(&x)
	fmt.Print("Masukkan jumlah ikan per wadah
(y): ")
	fmt.Scan(&y)

	if x < 1 || x > 1000 || y < 1 {
		fmt.Println("Nilai x harus antara 1-
1000 dan y harus lebih dari 0.")
		return
	}

	ikan := make([]float64, x*y)
	fmt.Println("Masukkan berat ikan satu per
satu:")
	for i := 0; i < x*y; i++ {
		fmt.Printf("Berat ikan ke-%d: ", i+1)
		fmt.Scan(&ikan[i])
	}
```

```
	totalBerat := make([]float64, x)
	rataRata := make([]float64, x)

	for i := 0; i < x; i++ {
		sum := 0.0
		for j := 0; j < y; j++ {
			sum += ikan[i*y+j]
		}
		totalBerat[i] = sum
		rataRata[i] = sum / float64(y)
	}

	fmt.Println("\nTotal berat ikan di setiap
wadah:")
	for i := 0; i < x; i++ {
		fmt.Printf("Wadah %d: %.2f\n", i+1,
totalBerat[i])
	}

	fmt.Println("\nBerat rata-rata ikan di
setiap wadah:")
	for i := 0; i < x; i++ {
		fmt.Printf("Wadah %d: %.2f\n", i+1,
rataRata[i])
	}
}
```

Screenshoot Output



Deskripsi Program

Program menerima jumlah ikan (x) dan kapasitas wadah (y). Berat setiap ikan dimasukkan ke dalam slice berat. Program menghitung jumlah wadah yang diperlukan. Berat setiap ikan dialokasikan ke wadah yang sesuai, lalu total berat tiap wadah dihitung. Program mencetak: Berat total setiap wadah, Rata-rata berat ikan per wadah.

Unguided 3

Sourcecode

```
//2311102024_FahriRamadhan

package main

import (
	"fmt"
)

type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita, n int, bMin
*float64, bMax *float64) {
	*bMin = arrBerat[0]
	*bMax = arrBerat[0]

	for i := 1; i < n; i++ {
		if arrBerat[i] < *bMin {
			*bMin = arrBerat[i]
		}
		if arrBerat[i] > *bMax {
			*bMax = arrBerat[i]
		}
	}
}

func rata(arrBerat arrBalita, n int) float64 {
	var total float64 = 0.0

	for i := 0; i < n; i++ {
		total += arrBerat[i]
	}

	return total / float64(n)
}

func main() {
	var n int
	var berat arrBalita
	var bMin, bMax float64

	fmt.Print("Masukkan banyak data berat
balita: ")
```

```
	fmt.Scan(&n)

	if n < 1 || n > 100 {
		fmt.Println("Jumlah balita harus
antara 1 dan 100.")
		return
	}

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

	hitungMinMax(berat, n, &bMin, &bMax)
	rerata := rata(berat, n)

	fmt.Printf("\nBerat balita minimum: %.2f
kg\n", bMin)
	fmt.Printf("Berat balita maksimum: %.2f
kg\n", bMax)
	fmt.Printf("Rata-rata berat balita: %.2f
kg\n", rerata)
}
```

Screenshoot Output

```
PS C:\Users\fahri\OneDrive\Documents\Modul11_al2> go ru
Masukkan banyak data berat balita: 3
Masukkan berat balita ke-1: 24
Masukkan berat balita ke-2: 22
Masukkan berat balita ke-3: 25

Berat balita minimum: 22.00 kg
Berat balita maksimum: 25.00 kg
Rata-rata berat balita: 23.67 kg
PS C:\Users\fahri\OneDrive\Documents\Modul11_al2>
```

Deskripsi Program

Fungsi rerata Parameter: arrBerat: Array berat balita. x: Jumlah elemen yang digunakan dalam array. Proses: Menghitung jumlah seluruh elemen dalam array (sum). Membagi sum dengan jumlah elemen (x) untuk mendapatkan rata-rata. Hasil: Mengembalikan rata-rata berat balita dalam bentuk float64.