LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL X

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Disusun Oleh :

Rasyid Nafsyarie / 2311102011

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 dalam himpunan data adalah proses untuk menemukan nilai maksimum (nilai tertinggi) dan minimum (nilai terendah) dari sekumpulan elemen data. Proses ini sering digunakan dalam berbagai aplikasi seperti analisis statistik, optimasi, dan pemrosesan data.

1. Konsep Dasar Nilai Ekstrim Nilai Minimum: Elemen dengan nilai terkecil dalam suatu himpunan data. Nilai Maksimum: Elemen dengan nilai terbesar dalam suatu himpunan data.

2. Metode Pencarian Nilai Ekstrim Pencarian nilai ekstrem dapat dilakukan dengan algoritma sederhana: Inisialisasi nilai minimum dan maksimum dengan elemen pertama dalam himpunan. Iterasi melalui setiap elemen dalam himpunan: Jika elemen lebih kecil dari nilai minimum saat ini, perbarui nilai minimum., Jika elemen lebih besar dari nilai maksimum saat ini, perbarui nilai maksimum. Kompleksitas waktu algoritma ini adalah O(n), di mana nnn adalah jumlah elemen dalam himpunan.

3. Implementasi dalam Golang Golang adalah bahasa pemrograman statically typed yang efisien untuk pengolahan data. Untuk mencari nilai ekstrem, kita memanfaatkan tipe data slice karena mendukung penyimpanan elemen dengan ukuran dinamis.

II. GUIDED

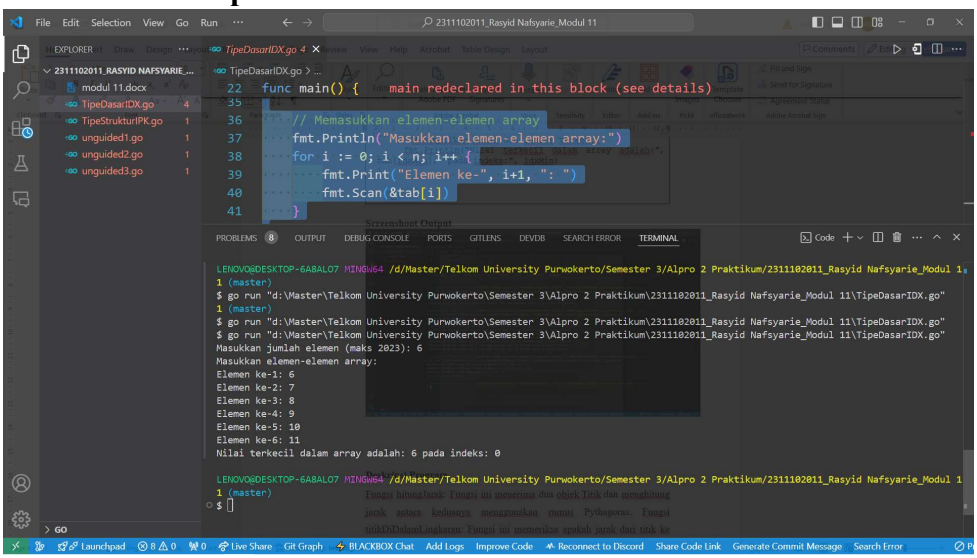
1. Guided 1

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



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.

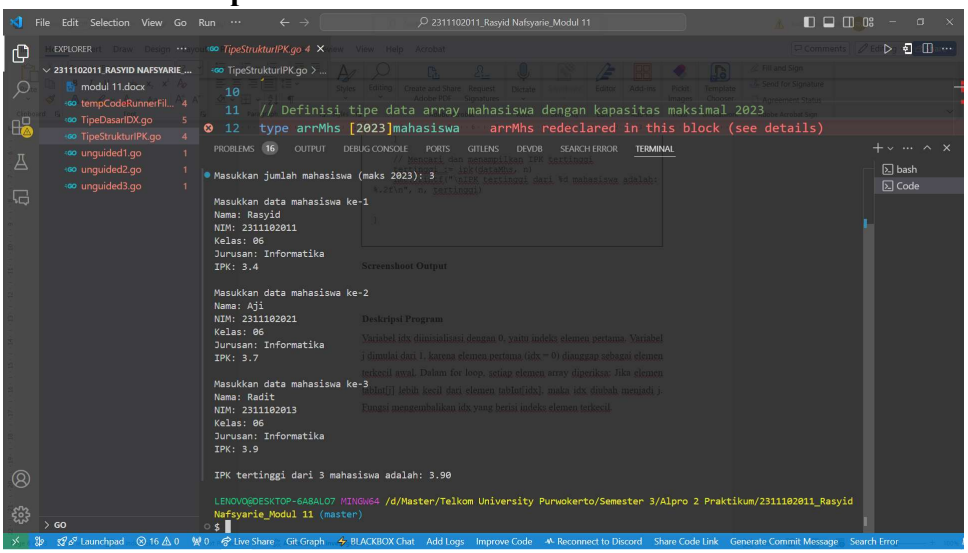
2. Guided 2

Sourcecode

```
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



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.

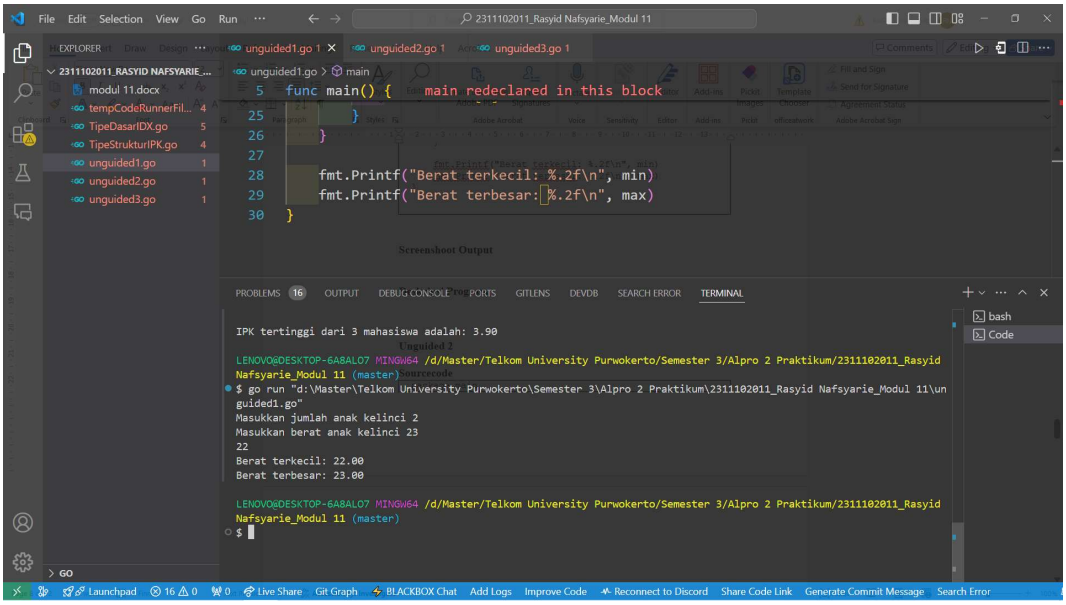
III. UNGUIDED

Unguided 1

Sourcecode

```
package main
//Rasyid Nafsyarie 2311102011 IF 11 06
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)
}
```

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.

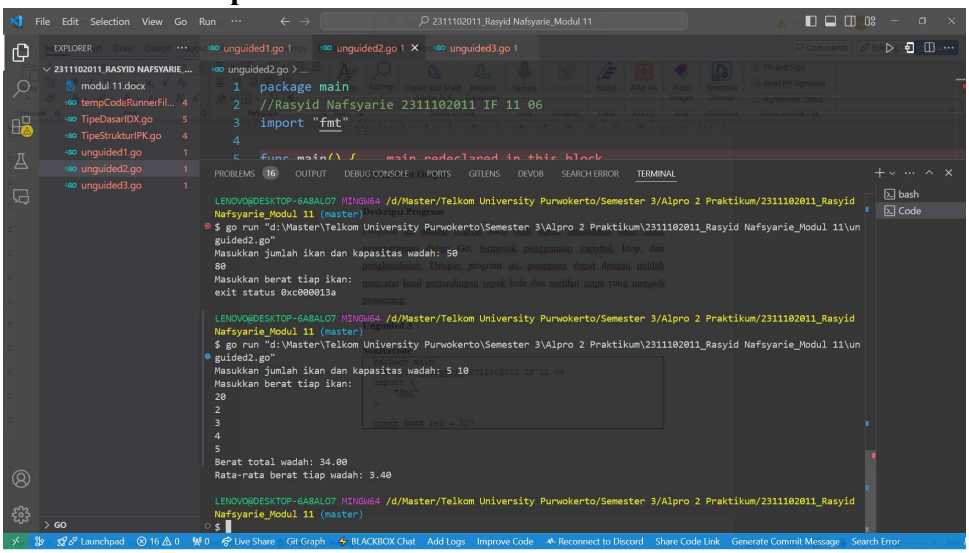
Unguided 2

Sourcecode

```
package main
//Rasyid Nafsyarie 2311102011 IF 11 06
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()

}
```

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.

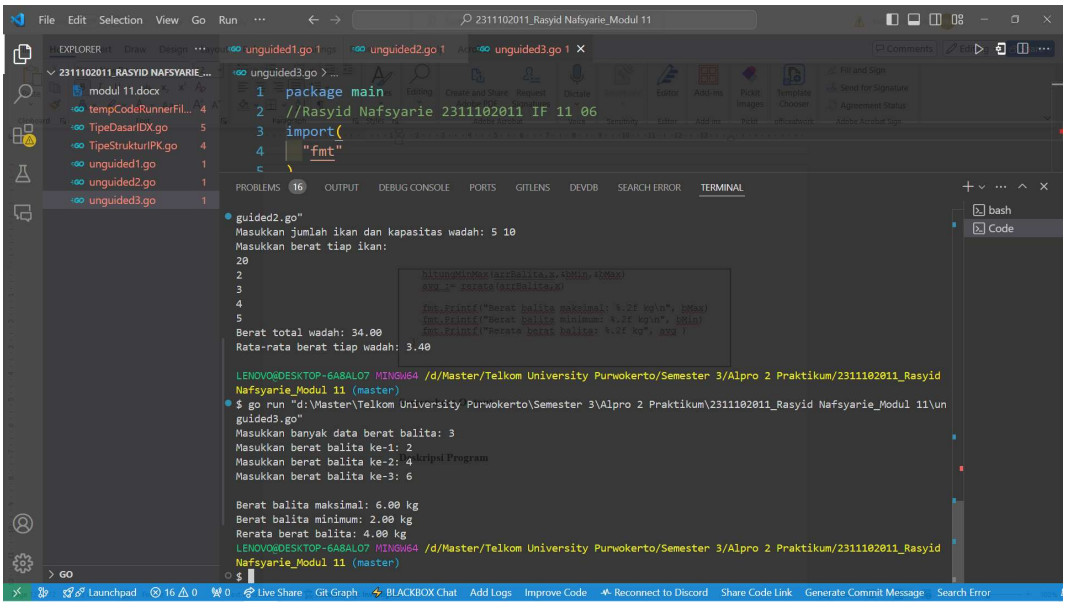
Unguided 3

Sourcecode

```
package main
//Rasyid Nafsyarie 2311102011 IF 11 06
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 )
}
```

Screenshoot Output



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.