

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XI
PENCARIAN NILAI EKSTRIM
PADA HIMPUNAN DATA**



**Disusun Oleh :
PETRA PRIADI S.P GINTING (2311102273)
IF-06**

**Dosen Pengampu :
ABEDNEGO DWI SEPTIADI**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2024

I. DASAR TEORI

Pengertian Pencarian nilai ekstrim adalah proses menemukan nilai tertinggi (maksimum) atau terendah (minimum) dalam sekumpulan data. Ini adalah konsep dasar dalam statistik dan analisis data yang sering digunakan dalam berbagai bidang seperti meteorologi, keuangan, sains, dan teknik.

Aplikasi

1. **Meteorologi:** Untuk memantau cuaca ekstrem, seperti suhu tertinggi dan terendah dalam sehari, minggu, atau bulan.
2. **Keuangan:** Untuk menganalisis harga saham atau nilai tukar mata uang, mencari titik tertinggi atau terendah dalam periode waktu tertentu.
3. **Ilmu Data:** Dalam data mining dan machine learning, untuk memahami distribusi data dan mendeteksi anomali atau outlier.

Proses Pencarian Nilai Ekstrim

1. **Inisialisasi:** Mulailah dengan menetapkan nilai pertama dari data sebagai nilai ekstrim awal (baik maksimum atau minimum).
2. **Perbandingan:** Bandingkan setiap nilai dalam himpunan data dengan nilai ekstrim saat ini. Jika nilai baru lebih besar (untuk pencarian maksimum) atau lebih kecil (untuk pencarian minimum) daripada nilai ekstrim saat ini, maka perbarui nilai ekstrim tersebut.
3. **Iterasi:** Lakukan perbandingan ini untuk setiap elemen dalam himpunan data hingga seluruh data telah dievaluasi.
4. **Pengembalian Hasil:** Setelah semua elemen dalam himpunan data telah dievaluasi, hasil nilai ekstrim (maksimum atau minimum) dikembalikan sebagai output.

II. 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)

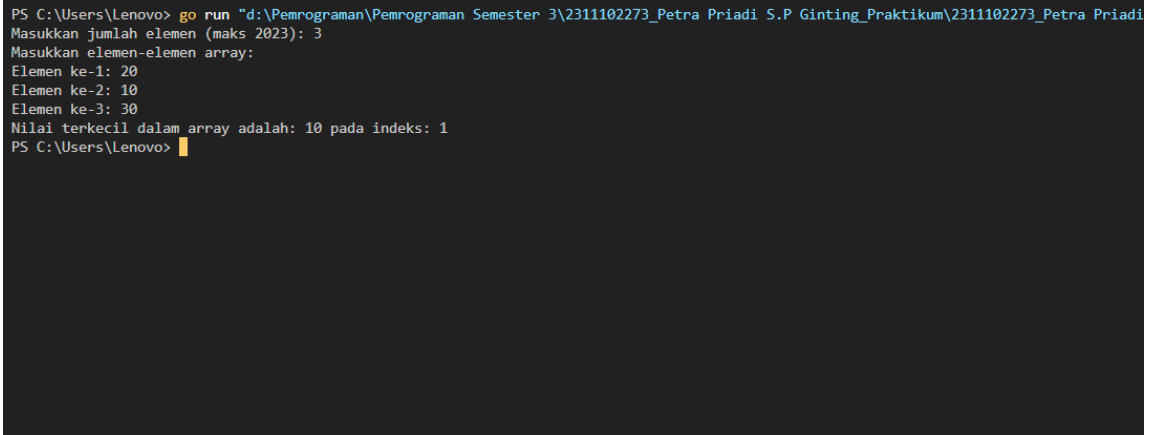
    // Validasi input jumlah elemen
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah elemen harus antara 1 dan 2023.")
        return
    }

    // 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



```
PS C:\Users\Lenovo> go run "d:\Pemrograman\Pemrograman Semester 3\2311102273_Petra Priadi S.P Ginting_Praktikum\2311102273_Petra Priadi
Masukkan jumlah elemen (maks 2023): 3
Masukkan elemen-elemen array:
Elemen ke-1: 20
Elemen ke-2: 10
Elemen ke-3: 30
Nilai terkecil dalam array adalah: 10 pada indeks: 1
PS C:\Users\Lenovo>
```

Deskripsi Program

Program dimulai dengan mendeklarasikan tipe data array `arrInt` untuk menampung integer hingga 2023 elemen. Fungsi `terkecil` digunakan untuk mencari indeks elemen terkecil dalam array, dengan iterasi dari elemen pertama ke elemen ke-*n* dan membandingkan setiap elemen untuk memperbarui indeks jika ditemukan elemen yang lebih kecil. Dalam fungsi `main`, pengguna diminta memasukkan jumlah elemen array (*n*), yang harus berada dalam rentang 1 hingga 2023. Setelah itu, pengguna memasukkan nilai elemen array satu per satu. Program kemudian memanggil fungsi `terkecil` untuk menemukan indeks elemen terkecil dan menampilkan nilai serta indeksnya kepada pengguna.

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

```
PS C:\Users\Lenovo> go run "d:\Pemrograman\Pemrograman Semester 3\2311102273_Petra Priadi S.P Ginting_Prak
Masukkan jumlah mahasiswa (maks 2023): 3

Masukkan data mahasiswa ke-1
Nama: Petra
NIM: 2311102273
Kelas: IF06
Jurusan: Teknik_Informatika
IPK: 3.8

Masukkan data mahasiswa ke-2
Nama: Bagus
NIM: 2311119374
Kelas: IF09
Jurusan: Teknik_Sipil
IPK: 3.5

Masukkan data mahasiswa ke-3
Nama: Budi
NIM: 231119277
Kelas: IF05
Jurusan: Teknik_Lingkungan
IPK: 3.5

IPK tertinggi dari 3 mahasiswa adalah: 3.80
PS C:\Users\Lenovo>
```

Deskripsi Program

Struct mahasiswa didefinisikan dengan atribut: nama, nim, kelas, jurusan, dan ipk. Array arrMhs bertipe struct mahasiswa memiliki kapasitas maksimum 2023 elemen. Fungsi ipk digunakan untuk menghitung nilai IPK tertinggi dalam array dengan cara membandingkan setiap elemen mulai dari elemen pertama hingga ke-n. Dalam fungsi main, program meminta input jumlah mahasiswa (n), yang harus berada dalam rentang 1 hingga 2023. Kemudian, pengguna diminta untuk mengisi data setiap mahasiswa, termasuk nama, NIM, kelas, jurusan, dan IPK. Setelah data mahasiswa diinput, fungsi ipk dipanggil untuk menentukan nilai IPK tertinggi. Hasilnya ditampilkan dalam format desimal dengan dua angka di belakang koma.

III. UNGUIDED

1. Soal Studi Case

Sebuah program digunakan untuk mendata berat anak kelinci yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat anak kelinci yang akan dijual.

Masukan terdiri dari sekumpulan bilangan, yang mana bilangan pertama adalah bilangan bulat N yang menyatakan banyaknya anak kelinci yang akan ditimbang beratnya. Selanjutnya N bilangan riil berikutnya adalah berat dari anak kelinci yang akan dijual.

Keluaran terdiri dari dua buah bilangan riil yang menyatakan berat kelinci terkecil dan terbesar.

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var berat [1000]float64

    var p int
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&p)

    if p <= 0 || p > 1000 {
        fmt.Println("Jumlah anak kelinci harus antara 1 dan 1000.")
        return
    }

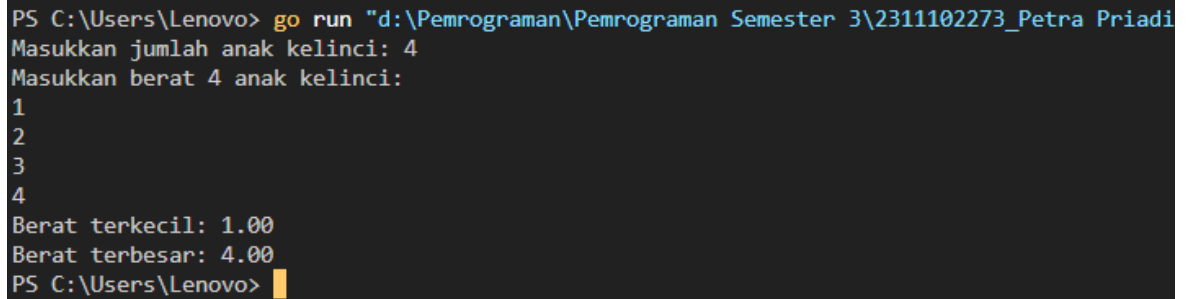
    fmt.Printf("Masukkan berat %d anak kelinci:\n", p)
    for i := 0; i < p; i++ {
        fmt.Scan(&berat[i])
    }

    minberat := berat[0]
    maxberat := berat[0]

    for i := 1; i < p; i++ {
        if berat[i] < minberat {
            minberat = berat[i]
        }
        if berat[i] > maxberat {
            maxberat = berat[i]
        }
    }
}
```

```
    fmt.Printf("Berat terkecil: %.2f\n", minberat)
    fmt.Printf("Berat terbesar: %.2f\n", maxberat)
}
```

Screenshoot Output



```
PS C:\Users\Lenovo> go run "d:\Pemrograman\Pemrograman Semester 3\2311102273_Petra Priadi
Masukkan jumlah anak kelinci: 4
Masukkan berat 4 anak kelinci:
1
2
3
4
Berat terkecil: 1.00
Berat terbesar: 4.00
PS C:\Users\Lenovo>
```

Deskripsi Program

Program menggunakan array berat berukuran maksimum 1000 elemen untuk menyimpan berat setiap anak kelinci dalam tipe data float64. Dalam fungsi main, pengguna diminta memasukkan jumlah anak kelinci (p), yang divalidasi agar berada di antara 1 hingga 1000. Jika jumlah tidak valid, program akan berhenti dengan pesan kesalahan. Selanjutnya, program meminta input berat masing-masing anak kelinci, yang disimpan dalam array berat. Variabel minberat dan maxberat diinisialisasi dengan berat anak kelinci pertama. Melalui iterasi dari elemen kedua hingga elemen ke-p, program membandingkan setiap berat dengan minberat dan maxberat untuk memperbarui nilai berat terkecil dan terbesar. Akhirnya, program mencetak berat terkecil dan terbesar dengan format dua desimal.

2. Studi Case

Sebuah program digunakan untuk menentukan tarif ikan yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat ikan yang akan dijual.

Masukan terdiri dari dua baris, yang mana baris pertama terdiri dari dua bilangan bulat x dan y. Bilangan x menyatakan banyaknya ikan yang akan dijual, sedangkan y adalah banyaknya ikan yang akan dimasukkan ke dalam wadah. Baris kedua terdiri dari sejumlah x bilangan riil yang menyatakan banyaknya ikan yang akan dijual.

Keluaran terdiri dari dua baris. Baris pertama adalah kumpulan bilangan riil yang menyatakan total berat ikan di setiap wadah (jumlah wadah tergantung pada nilai x dan y, urutan ikan yang dimasukkan ke dalam wadah sesuai urutan pada masukan baris ke-2). Baris kedua adalah sebuah bilangan riil yang menyatakan berat rata-rata ikan di setiap wadah.

Sourcecode

```
package main

import (
    "fmt"
)

type dataIkan struct {
    berat      []float64
    jumlahIkan int
    kapasitasWadah int
}

type wadahAkhir struct {
    beratWadah      []float64
    beratRataWadah []float64
}

func getInput() (dataIkan, error) {
    var data dataIkan
    data.berat = make([]float64, 1000)

    fmt.Println("Masukkan jumlah ikan (x) dan kapasitas wadah (y): ")
    fmt.Scan(&data.jumlahIkan, &data.kapasitasWadah)
```

```

        if err := validateInput(data.jumlahIkan, data.kapasitasWadah); err != nil {
            return data, err
        }

        fmt.Printf("Masukkan berat %d ikan:\n", data.jumlahIkan)
        for i := 0; i < data.jumlahIkan; i++ {
            fmt.Scan(&data.berat[i])
        }

        return data, nil
    }

func validateInput(jumlahIkan, kapasitasWadah int) error {
    if jumlahIkan <= 0 || jumlahIkan > 1000 {
        return fmt.Errorf("Jumlah ikan (x) harus antara 1 dan 1000")
    }
    if kapasitasWadah <= 0 {
        return fmt.Errorf("Kapasitas wadah (y) harus lebih besar dari 0")
    }
    return nil
}

func calculateDistribution(data dataIkan) wadahAkhir {
    jumlahWadah := (data.jumlahIkan + data.kapasitasWadah - 1) /
data.kapasitasWadah
    var results wadahAkhir
    results.beratWadah = make([]float64, jumlahWadah)
    results.beratRataWadah = make([]float64, jumlahWadah)

    ikanPerWadah := make([]int, jumlahWadah)

    for i := 0; i < data.jumlahIkan; i++ {
        indexWadah := i / data.kapasitasWadah
        results.beratWadah[indexWadah] += data.berat[i]
        ikanPerWadah[indexWadah]++
    }

    for i := 0; i < jumlahWadah; i++ {
        if ikanPerWadah[i] > 0 {

```

```

        results.beratRataWadah[i] = results.beratWadah[i] /
float64(ikanPerWadah[i])
    }
}

return results
}

func displayResults(results wadahAkhir) {
    fmt.Println("\nTotal berat ikan di setiap wadah:")
    for i, berat := range results.beratWadah {
        fmt.Printf("Wadah %d: %.2f\n", i+1, berat)
    }

    fmt.Println("\nRata-rata berat ikan di setiap wadah:")
    for i, rata := range results.beratRataWadah {
        fmt.Printf("Wadah %d: %.2f\n", i+1, rata)
    }
}

func main() {
    data, err := getInput()
    if err != nil {
        fmt.Println(err)
        return
    }

    results := calculateDistribution(data)
    displayResults(results)
}

```

Screenshoot Output

```
PS C:\Users\Lenovo> go run "d:\Pemrograman\Pemrograman Semester 3\2311102273_Petra Priadi S.P Ginting_Praktikum\2
Masukkan jumlah ikan (x) dan kapasitas wadah (y): 4 2
Masukkan berat 4 ikan:
2
1.5
1.5
2.5

Total berat ikan di setiap wadah:
Wadah 1: 3.50
Wadah 2: 4.00

Rata-rata berat ikan di setiap wadah:
Wadah 1: 1.75
Wadah 2: 2.00
PS C:\Users\Lenovo>
```

Deskripsi Program

Struct `dataIkan` menyimpan data ikan, termasuk array berat ikan, jumlah ikan (`jumlahIkan`), dan kapasitas wadah (`kapasitasWadah`). Struct `wadahAkhir` menyimpan hasil distribusi, yaitu total berat ikan di setiap wadah (`beratWadah`) dan rata-rata berat ikan per wadah (`beratRataWadah`). Dalam fungsi `main`, data ikan diambil dari input pengguna melalui fungsi `getInput`, yang juga memvalidasi jumlah ikan (1-1000) dan kapasitas wadah (>0). Jika input valid, fungsi `calculateDistribution` menghitung distribusi ikan ke wadah, dengan menghitung total berat ikan per wadah dan rata-ratanya berdasarkan indeks ikan. Distribusi dilakukan menggunakan perhitungan indeks wadah ($i / \text{kapasitasWadah}$) sehingga ikan dikelompokkan secara berurutan ke dalam wadah. Hasil distribusi ditampilkan dengan fungsi `displayResults`, yang mencetak total berat dan rata-rata berat untuk setiap wadah.

3. Studi Case

Pos Pelayanan Terpadu (posyandu) sebagai tempat pelayanan kesehatan perlu mencatat data berat balita (dalam kg). Petugas akan memasukkan data tersebut ke dalam array. Dari data yang diperoleh akan dicari berat balita terkecil, terbesar, dan reratanya.

Buatlah program dengan spesifikasi subprogram sebagai berikut:

```
Masukan banyak data berat balita : 4
Masukan berat balita ke-1: 5.3
Masukan berat balita ke-2: 6.2
Masukan berat balita ke-3: 4.1
Masukan berat balita ke-4: 9.9
Berat balita minimum: 4.10 kg
Berat balita maksimum: 9.90 kg
Rerata berat balita: 6.38 kg
```

Sourcecode

```
package main

import (
    "fmt"
)

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] < *bMin {
            *bMin = arrBerat[i]
        }
        if arrBerat[i] > *bMax {
            *bMax = arrBerat[i]
        }
    }
}

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

```

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

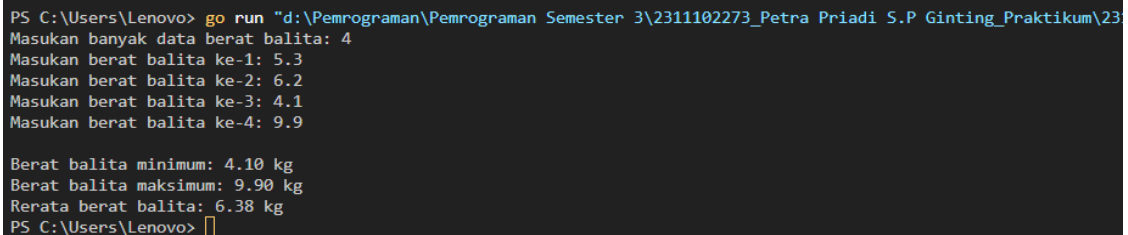
    fmt.Print("Masukan banyak data berat balita: ")
    fmt.Scan(&x)

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

    hitungMinMax(berat, x, &bMin, &bMax)
    rataRata := rerata(berat, x)
    fmt.Printf("\nBerat balita minimum: %.2f kg\n", bMin)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)
    fmt.Printf("Rerata berat balita: %.2f kg\n", rataRata)
}

```

Screenshoot Output



```

PS C:\Users\Lenovo> go run "d:\Pemrograman\Pemrograman Semester 3\2311102273_Petra Priadi S.P Ginting_Praktikum\23
Masukan banyak data berat balita: 4
Masukan berat balita ke-1: 5.3
Masukan berat balita ke-2: 6.2
Masukan berat balita ke-3: 4.1
Masukan berat balita ke-4: 9.9

Berat balita minimum: 4.10 kg
Berat balita maksimum: 9.90 kg
Rerata berat balita: 6.38 kg
PS C:\Users\Lenovo> 

```

Deskripsi Program

Program menggunakan array `arrBalita` berukuran maksimum 100 elemen untuk menyimpan berat balita, dengan tipe data `float64`. Fungsi `hitungMinMax` menghitung nilai berat minimum dan maksimum dengan menggunakan pointer untuk memperbarui nilai `bMin` dan `bMax` berdasarkan perbandingan setiap elemen dalam array. Fungsi `rerata` menghitung rata-rata berat balita dengan menjumlahkan semua elemen array dan membagi totalnya dengan jumlah data (`x`). Dalam fungsi `main`, pengguna diminta memasukkan jumlah data berat balita (`x`) dan berat masing-masing balita, yang disimpan dalam array `berat`. Program memanggil `hitungMinMax` untuk mendapatkan berat minimum dan maksimum, serta `rerata` untuk menghitung rata-rata berat. Akhirnya, hasil perhitungan ditampilkan dalam format dua desimal.