

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

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



Disusun Oleh :

Andika Indra Prastawa/ 2311102033

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 adalah suatu proses yang lazim dilakukan di dalam kehidupan sehari-hari. Contoh penggunaannya dalam kehidupan nyata sangat beragam, misalnya pencarian file di dalam directory komputer, pencarian suatu teks di dalam sebuah dokumen, pencarian buku pada rak buku, dan contoh lainnya. Pertama pada modul ini akan dipelajari salah satu algoritma pencarian nilai terkecil atau terbesar pada sekumpulan data, atau biasa disebut pencarian nilai ekstrim.

Pencarian nilai ekstrem dapat dilakukan dengan memanfaatkan loop untuk menjelajahi elemen array atau slice. Pendekatan yang umum dimulai dengan mengasumsikan elemen pertama sebagai nilai minimum atau maksimum, kemudian membandingkannya dengan elemen-elemen lain. Jika ditemukan elemen yang lebih kecil (atau lebih besar), elemen tersebut akan menjadi nilai ekstrem baru. Proses ini terus berlanjut hingga semua elemen telah diperiksa. Bahasa Go mendukung pengelolaan array dan slice secara efisien, yang membuat implementasi algoritma pencarian nilai ekstrem menjadi sederhana dan cepat.

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

```
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

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\LENOVO\Downloads\MODUL11_PRAKTIKUM ALPRO2> go run "c:\Users\LENOVO\Download
Masukkan jumlah elemen (maks 2023): 4
Masukkan elemen-elemen array:
Elemen ke-1: 2
Elemen ke-2: 4
Elemen ke-3: 6
Elemen ke-4: 8
Nilai terkecil dalam array adalah: 2 pada indeks: 0
PS C:\Users\LENOVO\Downloads\MODUL11_PRAKTIKUM ALPRO2>

```

Deskripsi Program

Program mencari elemen terkecil dalam sebuah array. Program dimulai dengan mendeklarasikan tipe data array bernama `arrInt` `yterkecil`, digunakan `main`, `pterkecil` dipanggil untuk mengidentifikasi elemen terkecil dan indeksinya, yang kemudian ditampilkan kepada pengguna. Program juga mencakup validasi input untuk memastikan jumlah elemen berada dalam rentang yang diperbolehkan.

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 indeks IPK tertinggi dalam array
    mahasiswa
    func indeksIPKTertinggi(T arrMhs, n int) int {
        var idx int = 0 // Inisialisasi indeks IPK tertinggi
        pada indeks pertama
        for j := 1; j < n; j++ {
            if T[idx].ipk < T[j].ipk {
                idx = j // Update indeks jika ditemukan IPK
                yang lebih tinggi
            }
        }
        return idx
    }

    // Fungsi main untuk mengisi data mahasiswa dan mencari
    IPK tertinggi beserta indeksnya
    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: ")

```

```

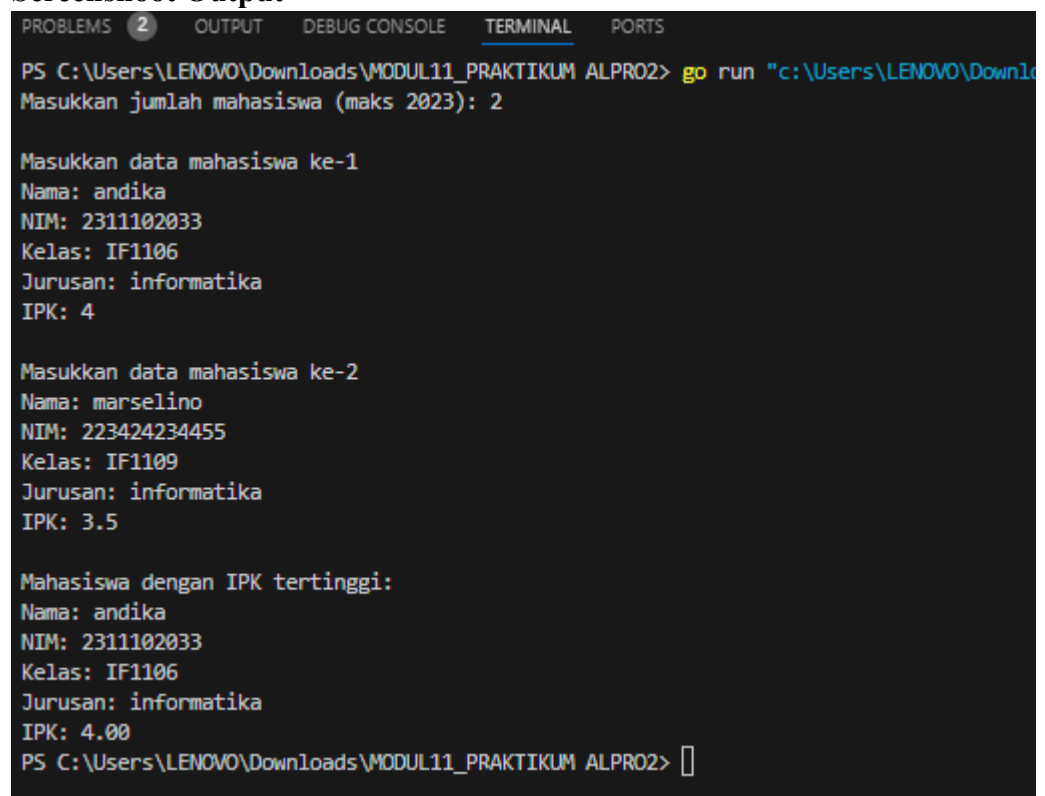
        fmt.Scan(&dataMhs[i].ipk)
    }

    // Mendapatkan indeks IPK tertinggi
    idxTertinggi := indeksIPKTertinggi(dataMhs, n)

    // Menampilkan data mahasiswa dengan IPK tertinggi
    fmt.Printf("\nMahasiswa dengan IPK tertinggi:\n")
    fmt.Printf("Nama: %s\n", dataMhs[idxTertinggi].nama)
    fmt.Printf("NIM: %s\n", dataMhs[idxTertinggi].nim)
    fmt.Printf("Kelas: %s\n",
dataMhs[idxTertinggi].kelas)
    fmt.Printf("Jurusan: %s\n",
dataMhs[idxTertinggi].jurusan)
    fmt.Printf("IPK: %.2f\n", dataMhs[idxTertinggi].ipk)
}

```

Screenshoot Output



```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\LENOVO\Downloads\MODUL11_PRAKTIKUM ALPRO02> go run "c:\Users\LENOVO\Downlo
Masukkan jumlah mahasiswa (maks 2023): 2

Masukkan data mahasiswa ke-1
Nama: andika
NIM: 2311102033
Kelas: IF1106
Jurusan: informatika
IPK: 4

Masukkan data mahasiswa ke-2
Nama: marselino
NIM: 223424234455
Kelas: IF1109
Jurusan: informatika
IPK: 3.5

Mahasiswa dengan IPK tertinggi:
Nama: andika
NIM: 2311102033
Kelas: IF1106
Jurusan: informatika
IPK: 4.00
PS C:\Users\LENOVO\Downloads\MODUL11_PRAKTIKUM ALPRO02> 

```

Deskripsi Program

Program mencari data mahasiswa dengan IPK tertinggi dari daftar yang diinputkan. Program menggunakan tipe data struct untuk merepresentasikan

atribut mahasiswa, yaitu nama, NIM, kelas, jurusan, dan IPK, serta mendefinisikan tipe data array `arrMhs` dengan kapasitas hingga 2023 mahasiswa. Fungsi `indeksIPKTertinggi` digunakan untuk menentukan indeks mahasiswa dengan IPK tertinggi. Dalam fungsi `main`, program meminta pengguna untuk memasukkan jumlah mahasiswa dan mengisi data masing-masing mahasiswa. Setelah data dimasukkan, program mencari dan menampilkan informasi mahasiswa dengan IPK tertinggi, termasuk nama, NIM, kelas, jurusan, dan IPK. Program juga memvalidasi jumlah input untuk memastikan sesuai dengan batasan yang ditentukan.

III. UNGUIDED

1. Unguided 1

Sourcecode

```
package main

import (
    "fmt"
)

func cariberatdarikecilkebesar_2311102033(berat
[]float64, n int) (float64, float64) {
    min := berat[0]
    max := berat[0]

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

    return min, max
}

func main() {
    const maxKapasitas = 1000
    var berat [maxKapasitas]float64
    var n int

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

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

    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])
    }

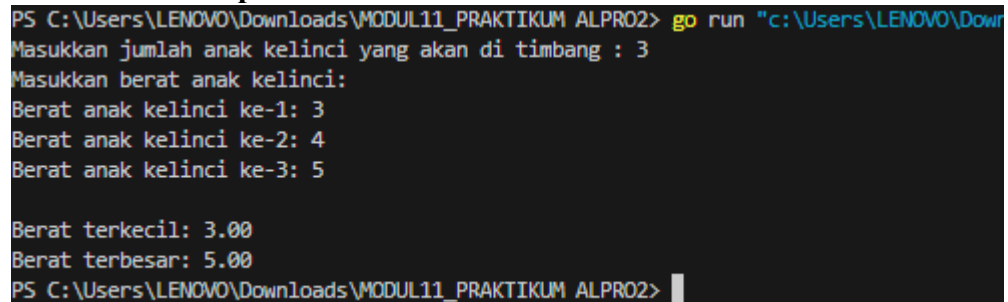
    terkecil, terbesar :=
cariberatdarikecilkebesar_2311102033(berat[:n], n)

    fmt.Printf("\nBerat terkecil: %.2f\n", terkecil)
```



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

Screenshoot Output



```
PS C:\Users\LENOVO\Downloads\MODUL11_PRAKTIKUM ALPRO2> go run "c:\Users\LENOVO\Downr
Masukkan jumlah anak kelinci yang akan di timbang : 3
Masukkan berat anak kelinci:
Berat anak kelinci ke-1: 3
Berat anak kelinci ke-2: 4
Berat anak kelinci ke-3: 5

Berat terkecil: 3.00
Berat terbesar: 5.00
PS C:\Users\LENOVO\Downloads\MODUL11_PRAKTIKUM ALPRO2> |
```

Deskripsi Program

Program mendata berat anak kelinci yang akan ditimbang dan menentukan berat terkecil serta terbesar dari data yang dimasukkan. Program ini menggunakan array dengan kapasitas maksimal 1000 untuk menyimpan data berat anak kelinci. Pengguna diminta untuk memasukkan jumlah anak kelinci (dengan validasi bahwa jumlah harus antara 1 hingga 1000) dan berat masing-masing anak kelinci. Program kemudian menggunakan fungsi untuk mencari nilai berat terkecil dan terbesar dengan membandingkan setiap elemen array. Hasil berupa berat terkecil dan terbesar ditampilkan dengan format dua desimal.

2. Unguided 2

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var x, y int
    fmt.Print("Masukkan jumlah wadah (x): ")
    fmt.Scanln(&x)
    fmt.Print("Masukkan jumlah ikan per wadah (y): ")
    fmt.Scanln(&y)
```

```

var berat_2311102033 []float64
totalberat := make([]float64, x)
totalFish := x * y

fmt.Println("Masukkan berat ikan satu per satu:")
for i := 0; i < totalFish; i++ {
    var weight float64
    fmt.Printf("Berat ikan ke-%d: ", i+1)
    fmt.Scanln(&weight)
    berat_2311102033 = append(berat_2311102033,
weight)

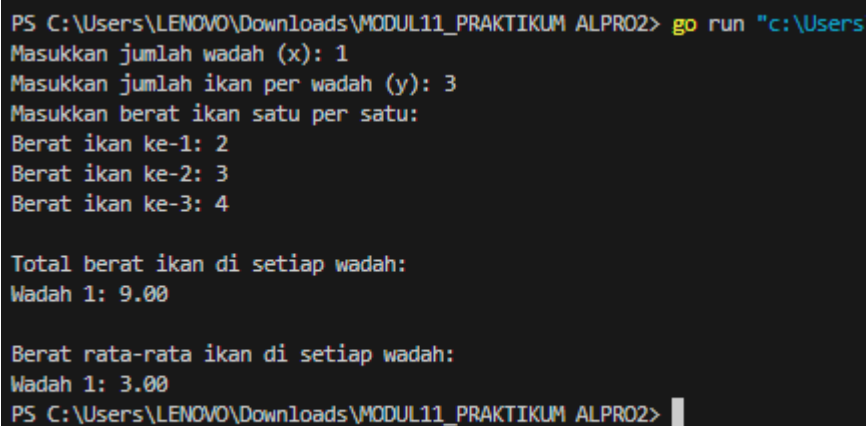
    currentWadah := i / y
    totalberat[currentWadah] += weight
}

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++ {
    average := totalberat[i] / float64(y)
    fmt.Printf("Wadah %d: %.2f\n", i+1, average)
}
}

```

Screenshoot Output



```

PS C:\Users\LENOVO\Downloads\MODUL11_PRAKTIKUM ALPRO2> go run "c:\Users
Masukkan jumlah wadah (x): 1
Masukkan jumlah ikan per wadah (y): 3
Masukkan berat ikan satu per satu:
Berat ikan ke-1: 2
Berat ikan ke-2: 3
Berat ikan ke-3: 4

Total berat ikan di setiap wadah:
Wadah 1: 9.00

Berat rata-rata ikan di setiap wadah:
Wadah 1: 3.00
PS C:\Users\LENOVO\Downloads\MODUL11_PRAKTIKUM ALPRO2>

```

Deskripsi Program

Program menghitung total berat dan berat rata-rata ikan di setiap wadah berdasarkan input dari pengguna. Pengguna diminta untuk memasukkan jumlah wadah (x), jumlah ikan per wadah (y), dan berat masing-masing ikan secara berurutan. Program menyimpan berat ikan dalam array dinamis dan menghitung total berat ikan di setiap wadah dengan membagi ikan secara berurutan ke dalam wadah berdasarkan kapasitas. Setelah semua data dimasukkan, program menampilkan total berat ikan di setiap wadah dan berat rata-rata ikan per wadah dengan format dua desimal, memberikan gambaran distribusi berat ikan di seluruh wadah.

3. Unguided 3 Sourcecode

```
package main

import (
    "fmt"
)

type arrBalita [100]float64

func hitungMinimalMaksimum_2311102033(arrBerat
[]float64, bMin, bMax *float64) {
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]
    for _, berat := range arrBerat {
        if berat < *bMin {
            *bMin = berat
        }
        if berat > *bMax {
            *bMax = berat
        }
    }
}

func hitungratarata_2311102033(arrBerat []float64)
float64 {
    var total float64
    for _, berat := range arrBerat {
        total += berat
    }
    return total / float64(len(arrBerat))
}
```

```

    }

    func main() {
        var n int
        fmt.Print("Masukkan banyak data berat balita: ")
        fmt.Scanln(&n)

        arrBerat := make([]float64, n)

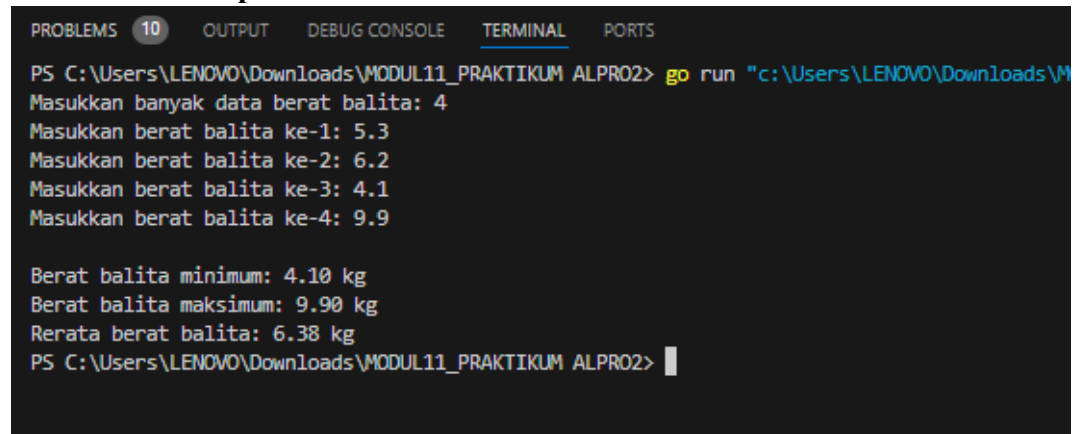
        for i := 0; i < n; i++ {
            fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
            fmt.Scanln(&arrBerat[i])
        }

        var bMin, bMax float64
        hitungMinimalMaksimum_2311102033(arrBerat, &bMin,
        &bMax)
        rerata := hitungratarata_2311102033(arrBerat)

        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", rerata)
    }

```

Screenshoot Output



```

PS C:\Users\LENOVO\Downloads\MODUL11_PRAKTIKUM ALPRO2> go run "c:\Users\LENOVO\Downloads\W
Masukkan banyak data berat balita: 4
Masukkan berat balita ke-1: 5.3
Masukkan berat balita ke-2: 6.2
Masukkan berat balita ke-3: 4.1
Masukkan 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\Downloads\MODUL11_PRAKTIKUM ALPRO2>

```

Deskripsi Program

Program ini digunakan untuk menghitung berat minimum, maksimum, dan rata-rata dari sekumpulan data berat balita yang dimasukkan oleh pengguna. Program meminta pengguna untuk menentukan jumlah data yang akan dimasukkan, kemudian membaca berat masing-masing balita. Fungsi

hitungMinimalMaksimum_2311102033 digunakan untuk menghitung berat balita terkecil dan terbesar dalam array, sedangkan fungsi hitungratarata_2311102033 digunakan untuk menghitung rata-rata dari seluruh data berat balita. Hasil perhitungan berupa berat minimum, maksimum, dan rata-rata ditampilkan dengan format dua angka desimal.