

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL XI

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Disusun Oleh :

ERWIN RIVALDO SILABAN/2311102248

S1-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 ekstrim adalah teknik dasar tetapi penting dalam pengolahan data, terutama dalam bidang statistik, analisis data, dan aplikasi berbasis data lainnya. Penggunaan algoritma yang tepat untuk pencarian nilai ekstrim dapat memberikan efisiensi dan akurasi yang lebih tinggi tergantung pada konteks data dan kebutuhan aplikasi.

Teori ini mencakup berbagai teknik yang dapat disesuaikan dengan jenis data dan kompleksitas masalah yang dihadapi, membuatnya menjadi konsep fundamental dalam pemrograman dan ilmu komputer.

II. GUIDED

1. Guided 1

Soal Studi Case

XXXXXXXXXXXXXXXXXX

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

```

29
30 // Validasi input jumlah elemen
31 if n < 1 || n > 2023 {
    PROBLEMS 29 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
    PS D:\tes\praktek golang> go run "d:\tes\praktek golang\.vscode\pencarian nilai ekstrim\guided1.go"
    Masukkan jumlah elemen (maks 2023): 5
    Masukkan elemen-elemen array:
    Elemen ke-1: 4
    Elemen ke-2: 7
    Elemen ke-3: 3
    Elemen ke-4: 2
    Elemen ke-5: 8
    Nilai terkecil dalam array adalah: 2 pada indeks: 3
    PS D:\tes\praktek golang>
  
```

Deskripsi Program

Fungsi utama, terkecil, mencari indeks elemen terkecil dalam array dengan iterasi dari awal hingga elemen terakhir, membandingkan setiap elemen

dengan elemen terkecil yang ditemukan sejauh ini. Jika elemen yang lebih kecil ditemukan, indeksnya diperbarui. Di dalam fungsi main, pengguna diminta memasukkan jumlah elemen yang ingin dievaluasi, dengan batas antara 1 hingga 2023. Setelah menerima input data elemen, program memanggil fungsi terkecil untuk menemukan elemen terkecil dan kemudian menampilkan nilai terkecil beserta indeksnya. Program ini memiliki kompleksitas $O(n)$, di mana pencarian elemen terkecil dilakukan dalam satu lintasan array.

2. Guided 2

Soal Studi Case

XXXXXXXXXXXXXXXXXX

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)

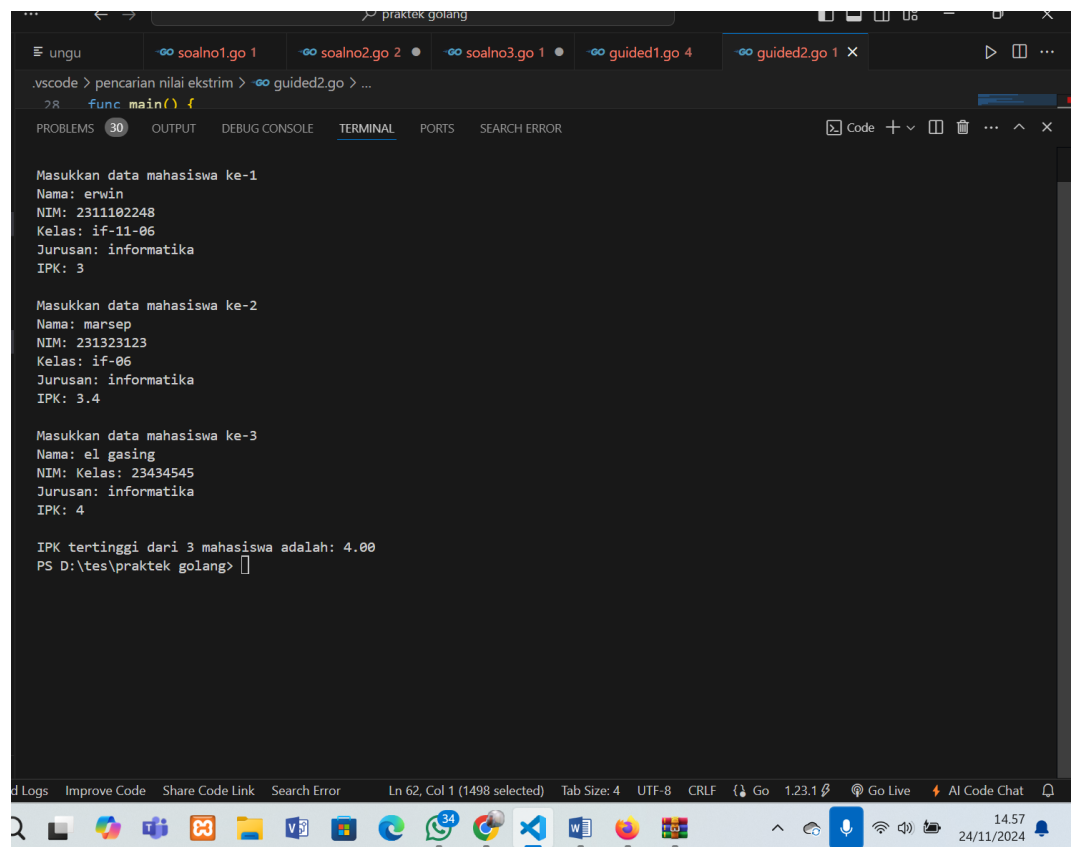
    // Validasi jumlah mahasiswa yang
dimasukkan
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah mahasiswa
harus antara 1 dan 2023.")
        return
    }

    // 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)
}

```

Screenshot Output



The screenshot shows a terminal window with the following output:

```
Masukkan data mahasiswa ke-1
Nama: erwin
NIM: 2311102248
Kelas: if-11-06
Jurusan: informatika
IPK: 3

Masukkan data mahasiswa ke-2
Nama: marsep
NIM: 231323123
Kelas: if-06
Jurusan: informatika
IPK: 3.4

Masukkan data mahasiswa ke-3
Nama: el gasing
NIM: 23434545
Jurusan: informatika
IPK: 4

IPK tertinggi dari 3 mahasiswa adalah: 4.00
PS D:\tes\praktek golang>
```

The terminal window is titled "praktek golang" and shows the execution of a Go program. The program prompts the user to enter student data three times. The first student is erwin (NIM: 2311102248, Kelas: if-11-06, Jurusan: informatika, IPK: 3). The second student is marsep (NIM: 231323123, Kelas: if-06, Jurusan: informatika, IPK: 3.4). The third student is el gasing (NIM: 23434545, Jurusan: informatika, IPK: 4). The program then outputs the highest IPK value, which is 4.00.

Deskripsi Program

Program ini memiliki kapasitas menyimpan data hingga 2023 mahasiswa menggunakan tipe data array. Program memiliki fungsi khusus 'ipk' yang bertugas mencari nilai IPK tertinggi dari seluruh data mahasiswa yang telah dimasukkan. Dalam fungsi main, program akan meminta pengguna memasukkan jumlah mahasiswa (dengan validasi antara 1-2023) dan

kemudian meminta input data lengkap untuk setiap mahasiswa termasuk nama, NIM, kelas, jurusan, dan IPK mereka. Setelah semua data dimasukkan, program akan memanggil fungsi ipk untuk menghitung dan menampilkan IPK tertinggi dari seluruh mahasiswa yang datanya telah diinput.

III. UNGUIDED

1. Soal Studi Case

XXXXXXXXXXXXXXXXXX

Sourcecode

```
package main

//Erwin Rivaldo Silaban
//23111022248
import (
    "fmt"
    "sort"
)

func main() {
    var beratKelinci = make([]float64, 0, 1000)

    var N int
    fmt.Print("Masukkan Jumlah Kelinci: ")
    fmt.Scan(&N)

    if N > 1000 {
        fmt.Println("Jumlah kelinci melebihi
kapasitas.")
        return
    }

    fmt.Println("Masukkan Berat Kelinci Yang Akan
Ditimbang:")
    for i := 0; i < N; i++ {
        var berat float64
        fmt.Printf("Berat Kelinci ke-%d (kg): ", i+1)
        fmt.Scan(&berat)
        beratKelinci = append(beratKelinci, berat)
    }
    sort.Float64s(beratKelinci)

    fmt.Println("\nBerat Kelinci dari yang terkecil
sampai terbesar:")
    for i := 0; i < N; i++ {
        fmt.Printf("%.1f kg\n", beratKelinci[i])
    }
}
```


Screenshoot Output

```
PROBLEMS 30 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\.vscode\pencarian nilai ekstrim\soalno1.go"
Masukkan Jumlah Kelinci: 4
Masukkan Berat Kelinci Yang Akan Ditimbang:
Berat Kelinci ke-1 (kg): 3
Berat Kelinci ke-2 (kg): 6
Berat Kelinci ke-3 (kg): 8
Berat Kelinci ke-4 (kg): 7

Berat Kelinci dari yang terkecil sampai terbesar:
3.0 kg
6.0 kg
7.0 kg
8.0 kg
PS D:\tes\praktek golang> 
```

Deskripsi Program

Program dimulai dengan meminta pengguna memasukkan jumlah kelinci yang akan ditimbang (N), dengan validasi agar tidak melebihi kapasitas maksimal. Selanjutnya, program akan meminta input berat untuk setiap kelinci satu per satu. Setelah semua data berat kelinci dimasukkan, program menggunakan fungsi `sort.Float64s` untuk mengurutkan berat kelinci dari yang terkecil hingga terbesar. Terakhir, program menampilkan hasil pengurutan berat kelinci dalam format satu baris per data dengan satuan kilogram.

2. Soal Studi Case

XXXXXXXXXXXXXXXXXXXX

Sourcecode

```
package main

//2311102248
//Erwin Rivaldo Silaban
import (
    "fmt"
)

func main() {
    var x, y int
```

```

    fmt.Print("Masukkan jumlah ikan yang
    akan dijual dan banyaknya ikan per wadah (x
    y): ")
    fmt.Scan(&x, &y)

    if x <= 0 || y <= 0 {
        fmt.Println("Jumlah ikan dan jumlah
        ikan per wadah harus lebih besar dari 0.")
        return
    }

    ikan := make([]float64, x)

    fmt.Printf("Masukkan berat %d ikan yang
    akan dijual:\n", x)
    for i := 0; i < x; i++ {
        fmt.Printf("Berat ikan ke-%d: ",
i+1)
        fmt.Scan(&ikan[i])
    }

    var totalBerat []float64
    var total float64
    for i := 0; i < x; i++ {
        total += ikan[i]
        if (i+1)%y == 0 || i == x-1 {
            totalBerat = append(totalBerat,
total)
            total = 0
        }
    }

    var rataRataBerat []float64
    for _, berat := range totalBerat {
        rataRata := berat / float64(y)
        rataRataBerat =
append(rataRataBerat, rataRata)
    }

    fmt.Println("\nTotal berat ikan di
    setiap wadah:")
    for _, berat := range totalBerat {
        fmt.Printf("%.2fkg\n ", berat)
    }

```

```

    }
    fmt.Println()

    fmt.Println("Rata-rata berat ikan di
    setiap wadah:")
    for _, rata := range rataRataBerat {
        fmt.Printf("%.2fkg\n ", rata)
    }
    fmt.Println()
}

```

Screenshoot Output

```

PS D:\tes\praktek golang> go run "d:\tes\praktek golang\.vscode\pencarian nilai ekstrim\soalno2.go"
Masukkan jumlah ikan yang akan dijual dan banyaknya ikan per wadah (x y): 7 3
Masukkan berat 7 ikan yang akan dijual:
Berat ikan ke-1: 2
Berat ikan ke-2: 4
Berat ikan ke-3: 5
Berat ikan ke-4: 3.3
Berat ikan ke-5: 2.3
Berat ikan ke-6: 4.5
Berat ikan ke-7: 2

Total berat ikan di setiap wadah:
11.00kg
10.10kg
2.00kg

Rata-rata berat ikan di setiap wadah:
3.67kg
3.37kg
0.67kg

PS D:\tes\praktek golang>

```

Deskripsi Program

program akan meminta pengguna memasukkan berat untuk setiap ikan secara berurutan. Setelah semua data berat ikan dimasukkan, program akan memproses data tersebut dengan menghitung total berat ikan di setiap wadah dan menyimpannya dalam array totalBerat, kemudian menghitung rata-rata berat ikan per wadah dengan membagi total berat dengan jumlah ikan di setiap wadah, hasilnya disimpan dalam array rataRataBerat.

Akhirnya, program menampilkan hasil perhitungan dalam dua bagian: pertama menampilkan total berat ikan di setiap wadah, dan kedua menampilkan rata-rata berat ikan per wadah, keduanya ditampilkan dengan format dua angka desimal dan satuan kilogram dalam baris yang terpisah.

3. Soal Studi Case

XXXXXXXXXXXXXXXXXX

Sourcecode

```
package main

//Erwin Rivaldo Silaban
//2311102248

import "fmt"

type arrBalita [100]float64

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

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

func rerata(arrBerat arrBalita, n int)
float64 {
    var rata float64 = 0
    bagi := 0
    for i := 0; i < n; i++ {
        rata += arrBerat[i]
        bagi++
    }
}
```

```
        return rata / float64(bagi)
    }

    func main() {
        var beratMin, beratMax float64
        var arrBerat arrBalita
        var n int

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

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

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

        hitungMinMax(arrBerat, &beratMin,
        &beratMax, n)
        rataRata := rerata(arrBerat, n)

        fmt.Printf("Berat balita minimum : %.2f
        Kg\n", beratMin)
        fmt.Printf("Berat balita maksimum :
        %.2f Kg\n", beratMax)
        fmt.Printf("Rerata berat balita : %.2f
        Kg\n", rataRata)
    }
```

Screenshoot Output

```
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\.vscode\pencarian nilai ekstrim\soalno3.go"
Masukkan banyak data berat balita : 7
Masukkan berat balita ke-1: 6
Masukkan berat balita ke-2: 7
Masukkan berat balita ke-3: 8
Masukkan berat balita ke-4: 9
Masukkan berat balita ke-5: 8
Masukkan berat balita ke-6: 7
Masukkan berat balita ke-7: 9
Berat balita minimum : 6.00 Kg
Berat balita maksimum : 9.00 Kg
Rerata berat balita : 7.71 Kg
PS D:\tes\praktek golang>
```

Deskripsi Program

Program dapat menampung hingga 100 data berat balita dan memiliki dua fungsi utama yaitu fungsi hitungMinMax untuk mencari berat terendah dan tertinggi, serta fungsi rerata untuk menghitung rata-rata berat seluruh balita. Program memulai dengan meminta jumlah balita yang akan didata (dengan batasan 1-100), kemudian meminta input berat untuk setiap balita. Setelah data lengkap, program akan menampilkan hasil analisis berupa berat minimum, maksimum, dan rata-rata dalam satuan kilogram dengan dua angka di belakang koma.