

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL XI

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Disusun Oleh :

Daffa Aryaputra / 2311102272

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pencarian nilai ekstrem pada himpunan data adalah proses untuk menemukan elemen dengan nilai minimum atau maksimum dalam sebuah kumpulan data. Nilai ekstrem sering digunakan dalam analisis data untuk mengidentifikasi elemen terkecil atau terbesar yang mungkin memiliki makna penting dalam konteks tertentu, seperti mencari skor tertinggi dalam ujian atau berat badan terendah dalam data kesehatan. Dalam implementasi komputasi, nilai ekstrem dicari dengan memindai elemen-elemen dalam struktur data seperti array atau slice dan melakukan perbandingan elemen secara iteratif. Proses ini memastikan bahwa elemen yang diinginkan ditemukan dengan efisien.

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.

Penerapan pencarian nilai ekstrem sangat berguna dalam berbagai aplikasi seperti statistik, pengolahan data kesehatan, atau bahkan analisis performa. Selain itu, fungsi pencarian ini dapat diperluas untuk mendukung berbagai jenis data dan kondisi, seperti pencarian nilai ekstrem pada subset data tertentu atau pencarian berdasarkan kriteria khusus. Bahasa Go, dengan performa dan sintaksis sederhananya, memungkinkan pengembang untuk mengimplementasikan algoritma pencarian ini secara efisien, baik dalam aplikasi skala kecil maupun skala besar.

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
    }

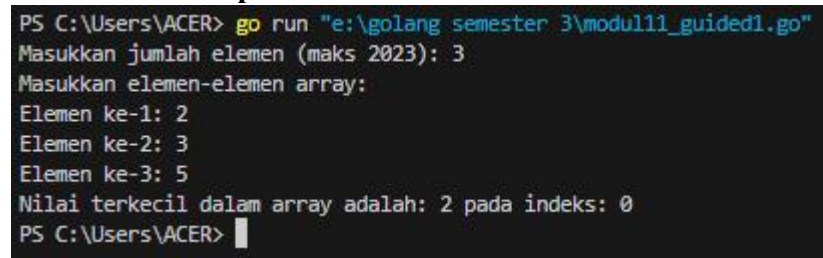
    // Memasukkan elemen-elemen array
    fmt.Println("Masukkan elemen-elemen array:")
    for i := 0; i < n; i++ {
```

```
        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)
}
```

Screenshot Output



```
PS C:\Users\ACER> go run "e:\golang semester 3\modul11_guided1.go"
Masukkan jumlah elemen (maks 2023): 3
Masukkan elemen-elemen array:
Elemen ke-1: 2
Elemen ke-2: 3
Elemen ke-3: 5
Nilai terkecil dalam array adalah: 2 pada indeks: 0
PS C:\Users\ACER>
```

Deskripsi Program

Program di atas untuk mencari elemen terkecil dalam sebuah array dengan panjang maksimum 2023 elemen. Program ini menggunakan sebuah tipe data array khusus (arrInt) yang memiliki panjang tetap 2023. Dengan memanfaatkan fungsi bernama terkecil, program mencari indeks elemen terkecil dari array yang diberikan oleh pengguna. Proses pencarian elemen terkecil dilakukan dengan membandingkan setiap elemen dalam array secara berulang (iteratif). Program ini juga mengandalkan validasi input untuk memastikan jumlah elemen array berada dalam rentang yang diperbolehkan (antara 1 dan 2023).

Cara kerja program dimulai dengan meminta pengguna memasukkan jumlah elemen array yang akan diisi, kemudian memverifikasi bahwa jumlahnya valid. Setelah itu, pengguna diminta memasukkan elemen-elemen array satu per satu. Setelah seluruh elemen dimasukkan, program memanggil fungsi terkecil, yang melakukan perbandingan berulang pada elemen array untuk menentukan elemen terkecil. Fungsi ini menggunakan variabel indeks (idx) untuk menyimpan posisi elemen terkecil selama iterasi. Setelah proses selesai, program menampilkan elemen terkecil beserta indeksnya kepada pengguna sebagai output. Misalnya, jika

pengguna memasukkan array [2,3,5], program akan menampilkan "Nilai terkecil dalam array adalah: 2 pada indeks: 0".

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\ACER> go run "e:\golang semester 3\modul11_guided2.go"
Masukkan jumlah mahasiswa (maks 2023): 3

Masukkan data mahasiswa ke-1
Nama: Daffa
NIM: 2311102272
Kelas: IF1106
Jurusan: Informatika
IPK: 4

Masukkan data mahasiswa ke-2
Nama: Ucup
NIM: 21143244334
Kelas: IF1106
Jurusan: informatika
IPK: 3

Masukkan data mahasiswa ke-3
Nama: jarwo
NIM: 36576897966
Kelas: rpl1106
Jurusan: rpl
IPK: 3.5

IPK tertinggi dari 3 mahasiswa adalah: 4.00
```

Deskripsi Program

Program di atas untuk mencari Indeks Prestasi Kumulatif (IPK) tertinggi dari sejumlah data mahasiswa yang dimasukkan oleh pengguna. Program ini menggunakan struktur data struct untuk merepresentasikan informasi mahasiswa, yang terdiri dari atribut nama, NIM, kelas, jurusan, dan IPK. Data mahasiswa disimpan dalam sebuah array dengan kapasitas maksimal 2023 elemen. Proses pencarian IPK tertinggi dilakukan melalui fungsi ipk, yang membandingkan setiap elemen IPK dari data mahasiswa secara iteratif untuk menentukan nilai tertinggi.

Cara kerja program dimulai dengan meminta pengguna memasukkan jumlah mahasiswa yang akan dimasukkan ke dalam program, kemudian memverifikasi bahwa jumlah tersebut berada dalam batasan yang diperbolehkan. Selanjutnya, pengguna mengisi data untuk masing-masing mahasiswa, meliputi nama, NIM, kelas, jurusan, dan IPK. Setelah data mahasiswa terisi, program memanggil fungsi ipk untuk mencari IPK tertinggi dengan cara membandingkan nilai IPK dari setiap mahasiswa di dalam array. Fungsi ini mengembalikan nilai IPK tertinggi, yang kemudian ditampilkan sebagai output. Misalnya, jika pengguna memasukkan data lima mahasiswa dengan IPK masing-masing 4, 3, dan 3.5, program akan menampilkan "IPK tertinggi dari 3 mahasiswa adalah: 4".

III. UNGUIDED

1.

Sourcecode

```
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

```
PS C:\Users\ACER> go run "e:\golang semester 3\modul11_unguided1.go"
Masukkan jumlah anak kelinci yang akan ditimbang: 3
Masukkan berat anak kelinci:
Berat anak kelinci ke-1: 2
Berat anak kelinci ke-2: 4
Berat anak kelinci ke-3: 3

Berat kelinci terkecil adalah: 2.00
Berat kelinci terbesar adalah: 4.00
PS C:\Users\ACER> |
```

Deskripsi Program

Program di atas dibuat untuk membantu pengguna mendata berat anak kelinci yang akan dijual. Program ini menerima input berupa jumlah anak kelinci yang akan ditimbang dan berat masing-masing anak kelinci. Berdasarkan data yang dimasukkan, program menghitung dan menampilkan berat kelinci terkecil dan terbesar. Program dirancang untuk menangani maksimal 1000 data berat kelinci.

Cara kerja program dimulai dengan meminta pengguna memasukkan jumlah anak kelinci, lalu memvalidasi bahwa jumlah tersebut berada dalam rentang 1 hingga 1000. Selanjutnya, program meminta berat masing-masing anak kelinci yang disimpan dalam sebuah array. Setelah itu, fungsi `cariBerat` digunakan untuk membandingkan berat kelinci satu per satu, guna menemukan nilai terkecil dan terbesar. Hasil akhir berupa berat terkecil dan terbesar ditampilkan ke layar dengan format desimal dua angka di belakang koma. Program ini memastikan input yang diterima valid dan memberikan hasil yang informatif kepada pengguna.

2.

Sourcecode

```
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

```
PS C:\Users\ACER> go run "e:\golang semester 3\modul11_unguided2.go"
Masukkan jumlah wadah (x): 2
Masukkan jumlah ikan per wadah (y): 3
Masukkan berat ikan satu per satu:
Berat ikan ke-1: 1.5
Berat ikan ke-2: 3
Berat ikan ke-3: 2.5
Berat ikan ke-4: 3.1
Berat ikan ke-5: 1
Berat ikan ke-6: 2

Total berat ikan di setiap wadah:
Wadah 1: 7.00
Wadah 2: 6.10

Berat rata-rata ikan di setiap wadah:
Wadah 1: 2.33
Wadah 2: 2.03
PS C:\Users\ACER> |
```

Deskripsi Program

Program di atas dirancang untuk membantu menghitung total berat dan berat rata-rata ikan di setiap wadah. Pengguna memasukkan jumlah wadah (x) dan jumlah ikan per wadah (y), diikuti dengan input berat masing-masing ikan. Program ini memanfaatkan array untuk menyimpan berat ikan dan menghitung total serta rata-rata berat ikan untuk setiap wadah secara terpisah. Output dari program berupa total berat dan rata-rata berat ikan di setiap wadah, yang ditampilkan secara terstruktur.

Cara kerja program dimulai dengan menerima input jumlah wadah dan jumlah ikan per wadah, yang kemudian divalidasi untuk memastikan nilainya sesuai (wadah maksimal 1000 dan jumlah ikan lebih dari 0). Setelah itu, program meminta input berat ikan satu per satu hingga seluruh data tersimpan dalam array. Program kemudian menghitung total berat ikan di setiap wadah dengan menjumlahkan berat ikan yang berada pada indeks-indeks terkait, dan menghitung rata-rata dengan membagi total berat setiap wadah dengan jumlah ikan per wadah. Hasil perhitungan tersebut ditampilkan ke layar dalam bentuk nilai desimal dengan dua angka di belakang koma untuk setiap wadah.

3.

Sourcecode

```
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
    }
}

```

```

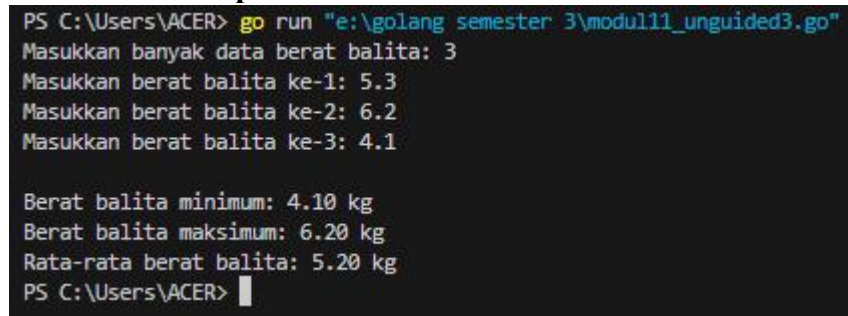
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
        fmt.Scan(&berat[i])
    }

    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\ACER> go run "e:\golang semester 3\modul11_unguided3.go"
Masukkan banyak data berat balita: 3
Masukkan berat balita ke-1: 5.3
Masukkan berat balita ke-2: 6.2
Masukkan berat balita ke-3: 4.1

Berat balita minimum: 4.10 kg
Berat balita maksimum: 6.20 kg
Rata-rata berat balita: 5.20 kg
PS C:\Users\ACER>

```

Deskripsi Program

Program di atas dibuat untuk membantu petugas Posyandu mencatat dan menganalisis data berat badan balita. Dengan memasukkan jumlah balita serta berat badan masing-masing, program ini dapat menghitung berat badan minimum, maksimum, dan rata-rata dari data yang dimasukkan. Program menggunakan tipe data array untuk menyimpan berat badan hingga maksimal 100 balita, dan hasil analisis ditampilkan dalam format desimal dua angka di belakang koma.

Cara kerja program dimulai dengan meminta input jumlah balita, yang kemudian divalidasi agar berada dalam rentang 1 hingga 100. Selanjutnya, pengguna memasukkan berat badan setiap balita yang disimpan dalam array. Program menggunakan subprogram `hitungMinMax` untuk mencari berat badan terkecil dan terbesar dengan membandingkan nilai setiap elemen array secara berulang. Subprogram `rata` menghitung rata-rata berat badan dengan menjumlahkan semua berat badan dan membaginya dengan jumlah balita. Hasil perhitungan berupa berat badan minimum, maksimum,

dan rata-rata ditampilkan ke layar untuk membantu pengguna melakukan analisis lebih lanjut.