LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN 2

MODUL 7

“PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA”

DISUSUN OLEH:

RAIHAN ADI ARBA

103112400071

S1 IF-12-01

DOSEN:

Dimas Fanny Hebrasianto Permadi, S.ST., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024/2025

DASAR TEORI

Algoritma pencarian nilai minimum dan maksimum adalah metode untuk menemukan nilai terkecil dan terbesar dalam suatu kumpulan data. Salah satu pendekatan yang umum digunakan adalah pencarian linear, di mana setiap elemen data diperiksa satu per satu secara berurutan. Proses ini dimulai dengan menginisialisasi nilai minimum dan maksimum menggunakan elemen pertama array atau slice. Selanjutnya, algoritma melakukan iterasi melalui sisa elemen, membandingkan setiap nilai dengan nilai minimum dan maksimum sementara. Jika ditemukan elemen yang lebih kecil dari nilai minimum saat ini, nilai minimum diperbarui. Demikian pula, jika ada elemen yang lebih besar dari nilai maksimum sementara, nilai maksimum akan disesuaikan. Setelah seluruh data diperiksa, nilai minimum dan maksimum yang diperoleh merupakan hasil akhir pencarian.

Algoritma ini memiliki keunggulan dalam kesederhanaan dan kemampuannya untuk diterapkan dalam berbagai konteks. Misalnya, dalam pengolahan data seperti pengukuran berat badan balita, pertumbuhan tinggi tanaman, fluktuasi harga buku, atau analisis nilai ujian siswa, pencarian nilai ekstrim sangat penting untuk mendukung proses analisis dan pengambilan keputusan. Dengan memeriksa setiap elemen secara sistematis, metode ini menjamin akurasi hasil meskipun data yang diolah berukuran besar atau tidak terurut.

A. GUIDED

1. Source code :

```go
package main

import "fmt"

func main() {
    var n int
    fmt.Print("Masukan jumlah tanaman: ")
    fmt.Scan(&n)

    var tinggiTanaman [500]float64
    for i := 0; i < n; i++ {
        fmt.Printf("Masukan tinggi tanaman ke-%d (cm): ", i+1)
        fmt.Scan(&tinggiTanaman[i])
    }

    min, max := tinggiTanaman[0], tinggiTanaman[0]
    for i := 1; i < n; i++ {
        if tinggiTanaman[i] < min {
            min = tinggiTanaman[i]
        }
        if tinggiTanaman[i] > max {
            max = tinggiTanaman[i]
        }
    }

    fmt.Printf("\nTinggi tanaman tertinggi: %.2f cm\n", max)
    fmt.Printf("Tinggi tanaman terpendek: %.2f cm\n", min)
}
```

Output :

```
raihan@Raihans-MacBook-Pro Laprak-Modul-7 % go run "/U
han/Documents/
GitHub/PRAKTIKUM-2/Laprak-Modul-7/103112400071_MODUL7/
0071_Guided1.g
o"
Masukan jumlah tanaman: 10
Masukan tinggi tanaman ke-1 (cm): 1
Masukan tinggi tanaman ke-2 (cm): 2
Masukan tinggi tanaman ke-3 (cm): 1
Masukan tinggi tanaman ke-4 (cm): 4
Masukan tinggi tanaman ke-5 (cm): 99
Masukan tinggi tanaman ke-6 (cm): 22
Masukan tinggi tanaman ke-7 (cm): 3
Masukan tinggi tanaman ke-8 (cm): 24
Masukan tinggi tanaman ke-9 (cm): 55
Masukan tinggi tanaman ke-10 (cm): 33

Tinggi tanaman tertinggi: 99.00 cm
Tinggi tanaman terpendek: 1.00 cm
raihan@Raihans-MacBook-Pro Laprak-Modul-7 %
```

Deskripsi :
Pengguna diminta untuk memasukkan jumlah tanaman, lalu memasukkan tinggi masing-masing tanaman satu per satu. Data tinggi tanaman disimpan dalam array, kemudian program membandingkan setiap nilai untuk menentukan mana yang tertinggi dan terendah. Hasil akhirnya ditampilkan dalam satuan sentimeter dengan format dua angka di belakang koma.

2. Source code :

```go
package main

import "fmt"

func main() {
    var x, y int
    fmt.Print("Masukan jumlah buku dan jumlah buku per rak: ")
    fmt.Scan(&x, &y)

    var hargaBuku [500]float64
    fmt.Println("\nMasukan harga setiap buku (dalam ribuan Rp): ")
    for i := 0; i < x; i++ {
        fmt.Scan(&hargaBuku[i])
    }

    var hargaRataRata []float64
    for i := 0; i < x; i += y {
```

```
        total := 0.0
        for j := i; j < i+y && j < x; j++ {
            total += hargaBuku[j]
        }
        hargaRataRata = append(hargaRataRata, total/float64(y))
    }

    min, max := hargaBuku[0], hargaBuku[0]
    for _, harga := range hargaBuku[:x] {
        if harga < min {
            min = harga
        }
        if harga > max {
            max = harga
        }
    }

    fmt.Printf("\nRata-rata harga per rak: ")
    for _, avg := range hargaRataRata {
        fmt.Printf("%.2f ", avg)
    }
    fmt.Printf("\nHarga termahal: %.2f Rp\n", max)
    fmt.Printf("Harga termurah: %.2f Rp\n", min)
}
```

Output :

```
raihan@Raihans-MacBook-Pro Laprak-Modul-7 % go run "/Users/rai
han/Documents/GitHub/PRAKTIKUM-2/Laprak-Modul-7/103112400071_M
ODUL7/103112400071_Guided2.go"
Masukan jumlah buku dan jumlah buku per rak: 1 2

Masukan harga setiap buku (dalam ribuan Rp):
90000

Rata-rata harga per rak: 45000.00
Harga termahal: 90000.00 Rp
Harga termurah: 90000.00 Rp
raihan@Raihans-MacBook-Pro Laprak-Modul-7 %
```

Deskripsi :

Program ini meminta pengguna untuk memasukkan jumlah buku dan jumlah buku per rak, kemudian meminta input harga setiap buku. Setelah data dimasukkan, program menghitung rata-rata harga buku per rak, serta menentukan harga termahal dan termurah dari seluruh buku yang dimasukkan.

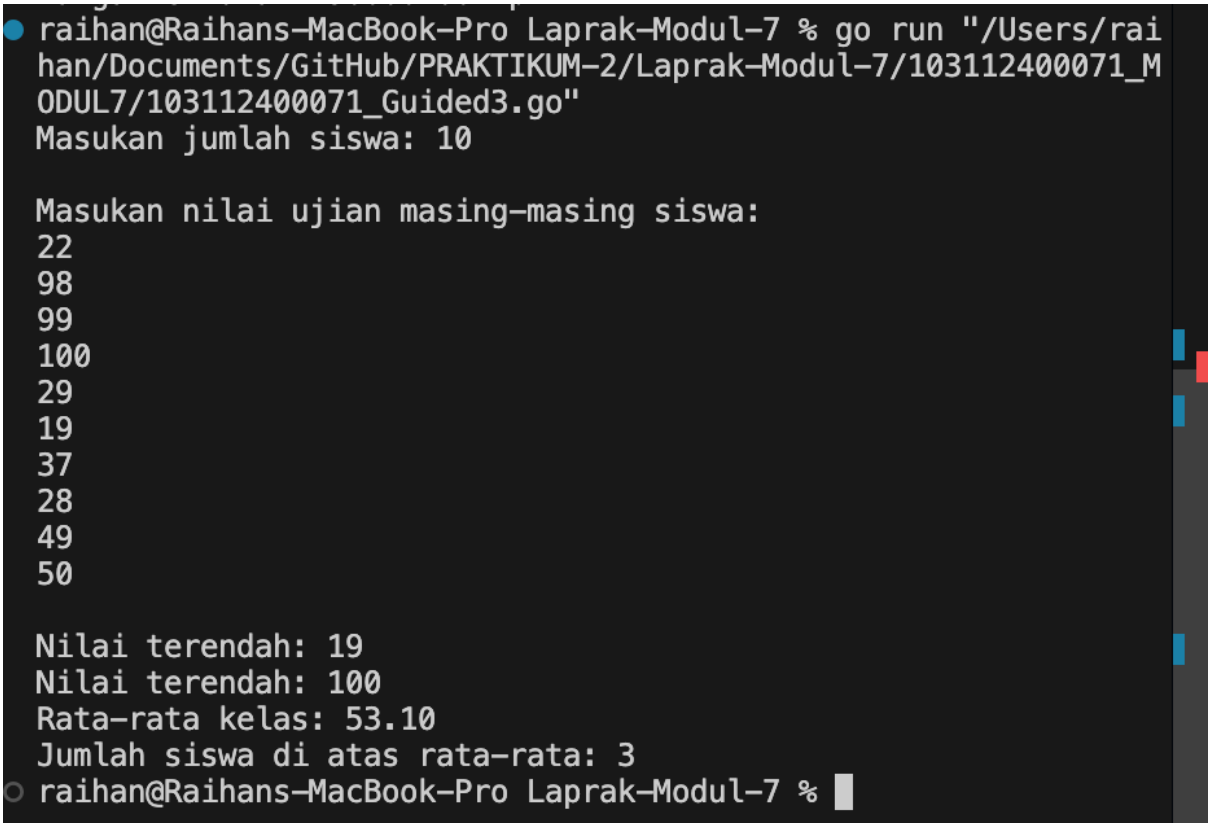
Ketika dijalankan, program pertama-tama meminta input jumlah buku dan kapasitas buku per rak. Misalnya, jika pengguna memasukkan 1 buku dengan 2 buku per rak, program tetap akan menghitung rata-rata dengan membagi harga buku yang ada dengan

kapasitas rak, meskipun jumlah buku kurang dari kapasitas rak. Setelah itu, program menampilkan rata-rata harga per rak, harga termahal, dan harga termurah. Contoh output menunjukkan bahwa jika hanya ada 1 buku dengan harga 90000, rata-ratanya dihitung sebagai 45000 (karena dibagi kapasitas rak 2), sedangkan harga termahal dan termurah sama karena hanya ada satu data buku.

3. Source code :

```go
package main

import "fmt"

func main() {
    var n int
    fmt.Print("Masukan jumlah siswa: ")
    fmt.Scan(&n)

    var nilaiSiswa [200]float64
    var totalNilai float64 = 0

    fmt.Println("\nMasukan nilai ujian masing-masing siswa:")
    for i := 0; i < n; i++ {
        fmt.Scan(&nilaiSiswa[i])
        totalNilai += nilaiSiswa[i]
    }

    rataRata := totalNilai / float64(n)

    min, max := nilaiSiswa[0], nilaiSiswa[0]
    var diAtasRataRata int = 0
    for _, nilai := range nilaiSiswa[:n] {
        if nilai < min {
            min = nilai
        }
        if nilai > max {
            max = nilai
        }
        if nilai > rataRata {
            diAtasRataRata++
        }
    }

    fmt.Printf("\nNilai terendah: %.0f\n", min)
    fmt.Printf("Nilai terendah: %.0f\n", max)
    fmt.Printf("Rata-rata kelas: %.2f\n", rataRata)
    fmt.Printf("Jumlah siswa di atas rata-rata: %d\n", diAtasRataRata)
}
```

Output:



Deskripsi :

Program ini meminta pengguna untuk memasukkan jumlah siswa, kemudian mengumpulkan nilai ujian masing-masing siswa. Setelah data dimasukkan, program menghitung rata-rata nilai kelas, menentukan nilai terendah dan tertinggi, serta menghitung berapa banyak siswa yang nilainya berada di atas rata-rata kelas.

Ketika dijalankan, program pertama-tama meminta input jumlah siswa. Misalnya, jika pengguna memasukkan 10 siswa, program akan meminta 10 nilai ujian. Setelah semua nilai dimasukkan, program menghitung rata-rata kelas, mencari nilai minimum dan maksimum, serta mengecek berapa siswa yang nilainya melebihi rata-rata. Contoh output menunjukkan bahwa dari 10 siswa, rata-rata kelas adalah 53.10, nilai terendah 19, nilai tertinggi 100, dan terdapat 3 siswa yang nilainya di atas rata-rata.

B. UNGUIDED

1. Latihan 1

Source Code:

```go
// Raihan Adi Arba
// 103112400071

package main
```

```go
import "fmt"

func inputBeratKelinci(n int, berat *[1000]float64) {
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan berat kelinci ke-%d (kg): ", i+1)
        fmt.Scan(&berat[i])
    }
}

func cariMinMax(n int, berat [1000]float64) (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() {
    var jumlahKelinci int
    var beratKelinci [1000]float64

    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&jumlahKelinci)

    inputBeratKelinci(jumlahKelinci, &beratKelinci)

    min, max := cariMinMax(jumlahKelinci, beratKelinci)

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

Output :

```
raihan@Raihans-MacBook-Pro Laprak-Modul-7 % go run "/Users/rai
han/Documents/GitHub/PRAKTIKUM-2/Laprak-Modul-7/103112400071_M
ODUL7/103112400071_Unguided1.go"
Masukkan jumlah anak kelinci: 4
Masukkan berat kelinci ke-1 (kg): 3
Masukkan berat kelinci ke-2 (kg): 24
Masukkan berat kelinci ke-3 (kg): 32
Masukkan berat kelinci ke-4 (kg): 3
Berat kelinci terkecil: 3.00
Berat kelinci terbesar: 32.00
```

Penjelasan Program:

Aplikasi ini memungkinkan pengguna untuk memasukkan data berat beberapa ekor kelinci sekaligus, kemudian secara otomatis mengidentifikasi kelinci dengan berat badan terkecil dan terbesar dalam kelompok tersebut. Program diawali dengan meminta input jumlah kelinci yang akan diukur, kemudian pengguna diminta memasukkan berat masing-masing kelinci dalam satuan kilogram. Data berat disimpan dalam sebuah array dengan kapasitas hingga 1000 ekor kelinci, membuatnya cocok untuk peternakan skala kecil hingga menengah.

Setelah semua data berat dimasukkan, program akan memproses data tersebut melalui dua fungsi utama: fungsi pertama bertugas mengumpulkan input berat dari pengguna, sedangkan fungsi kedua menganalisis data untuk menemukan nilai ekstrem (minimum dan maximum). Hasil analisis kemudian ditampilkan dalam format yang jelas, menunjukkan berat terkecil dan terbesar dengan presisi dua angka di belakang koma. Contoh eksekusi program menunjukkan efektivitasnya, dimana untuk input 4 kelinci dengan berat 3kg, 24kg, 32kg, dan 3kg, program berhasil mengidentifikasi rentang berat 3.00kg hingga 32.00kg dengan akurat.

2. Latihan 2

Source Code:

```go
// Raihan Adi Arba
// 103112400071

package main

import "fmt"

func main() {
    var jumlahIkan, jumlahPerWadah int

    // Input jumlah ikan dan jumlah ikan per wadah
    fmt.Print("Masukkan jumlah ikan dan jumlah per wadah: ")
    fmt.Scan(&jumlahIkan, &jumlahPerWadah)

    var ikan [1000]float64
```

```
    // Input berat tiap ikan
    for i := 0; i < jumlahIkan; i++ {
        fmt.Printf("Berat ikan ke-%d: ", i+1)
        fmt.Scan(&ikan[i])
    }

    var wadah [1000]float64
    jumlahWadah := 0
    totalSemua := 0.0

    // Proses pembagian ke wadah
    for i := 0; i < jumlahIkan; i += jumlahPerWadah {
        batas := i + jumlahPerWadah
        if batas > jumlahIkan {
            batas = jumlahIkan
        }

        total := 0.0
        for j := i; j < batas; j++ {
            total += ikan[j]
        }

        wadah[jumlahWadah] = total
        jumlahWadah++
        totalSemua += total
    }

    // Output berat per wadah
    fmt.Println("\nBerat per wadah:")
    for i := 0; i < jumlahWadah; i++ {
        fmt.Printf("Wadah %d: %.2f kg\n", i+1, wadah[i])
    }

    // Output rata-rata
    rataRata := totalSemua / float64(jumlahIkan)
    fmt.Printf("\nRata-rata berat ikan: %.2f kg\n", rataRata)
}
```

Output:

```
raihan@Raihans-MacBook-Pro Laprak-Modul-7 % go run "/Users/rai
han/Documents/GitHub/PRAKTIKUM-2/Laprak-Modul-7/103112400071_M
ODUL7/103112400071_Unguided2.go"
Masukkan jumlah ikan dan jumlah per wadah: 5
2
Berat ikan ke-1: 1
Berat ikan ke-2: 2
Berat ikan ke-3: 3
Berat ikan ke-4: 2
Berat ikan ke-5: 4

Berat per wadah:
Wadah 1: 3.00 kg
Wadah 2: 5.00 kg
Wadah 3: 4.00 kg

Rata-rata berat ikan: 2.40 kg
```

Deskripsi Program:
Program ini menggunakan metode iteratif untuk membagi ikan ke dalam wadah-wadah dengan kapasitas tertentu. Pertama, program meminta input jumlah ikan dan jumlah ikan per wadah. Kemudian, program membaca berat masing-masing ikan dan menyimpannya dalam sebuah array. Selanjutnya, program membagi ikan ke dalam wadah-wadah dengan menjumlahkan berat ikan sesuai kapasitas per wadah yang ditentukan. Setiap wadah berisi sekelompok ikan dengan total berat yang dihitung, dan hasilnya ditampilkan. Terakhir, program menghitung rata-rata berat semua ikan dengan membagi total berat seluruh ikan dengan jumlah ikan.

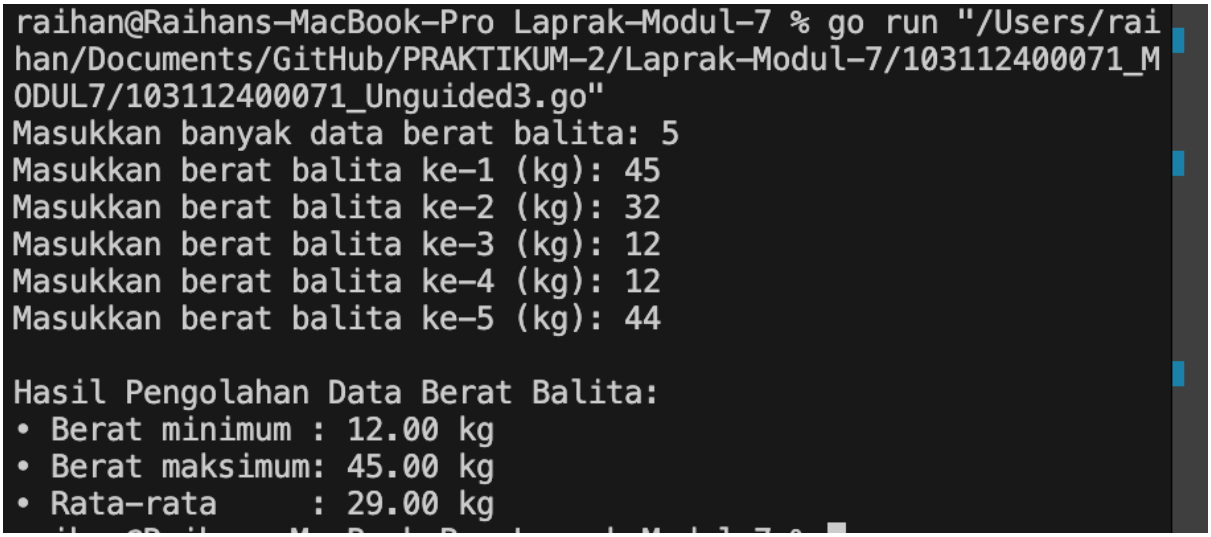
Algoritma yang digunakan bersifat sequential dan linear, karena program melakukan iterasi melalui array ikan secara berurutan dan membaginya ke wadah tanpa memerlukan teknik pengurutan atau pencarian yang kompleks. Pendekatan ini efisien untuk masalah pembagian sederhana seperti ini, dengan kompleksitas waktu O(n), di mana n adalah jumlah ikan.

3. Latihan 3
   Source Code:

```go
// Raihan Adi Arba
// 103112400071

package main

import (
    "fmt"
)

type arrBalita [100]float64

// Fungsi untuk menghitung berat minimum dan maksimum
```

```go
func hitungMinMax(arr arrBalita, jumlah int, bMin *float64, bMax *float64) {
    *bMin = arr[0]
    *bMax = arr[0]

    for i := 1; i < jumlah; i++ {
        if arr[i] < *bMin {
            *bMin = arr[i]
        }
        if arr[i] > *bMax {
            *bMax = arr[i]
        }
    }
}

// Fungsi untuk menghitung rata-rata berat balita
func hitungRerata(arr arrBalita, jumlah int) float64 {
    var total float64
    for i := 0; i < jumlah; i++ {
        total += arr[i]
    }
    return total / float64(jumlah)
}

func main() {
    var data arrBalita
    var n int

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

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

    var min, max float64
    hitungMinMax(data, n, &min, &max)
    rata := hitungRerata(data, n)

    fmt.Println("\nHasil Pengolahan Data Berat Balita:")
    fmt.Printf("• Berat minimum : %.2f kg\n", min)
    fmt.Printf("• Berat maksimum: %.2f kg\n", max)
    fmt.Printf("• Rata-rata     : %.2f kg\n", rata)
}
```

Output:



Deskripsi Program:

Program ini merupakan implementasi sederhana dalam bahasa Go untuk menganalisis data berat badan balita. Algoritma yang digunakan bersifat linear dan iteratif, dimana program melakukan tiga proses utama: menerima input data, menghitung statistik dasar (minimum, maksimum, dan rata-rata), serta menampilkan hasil. Proses pencarian nilai minimum dan maksimum menggunakan pendekatan perbandingan elemen secara berurutan (sequential search) dengan kompleksitas waktu O(n), sementara perhitungan rata-rata dilakukan dengan menjumlahkan semua nilai kemudian membaginya dengan jumlah data. Program ini cocok untuk pengolahan data skala kecil hingga menengah, seperti dalam pemantauan kesehatan balita di posyandu atau klinik, karena sifatnya yang sederhana namun efektif untuk mendapatkan gambaran statistik dasar dari sekumpulan data berat badan. Implementasi pointer pada parameter fungsi hitungMinMax menunjukkan penerapan konsep passing by reference untuk efisiensi memori.

DAFTAR PUSTAKA

Prayogo, N. A. (2021). ***Dasar Pemrograman Go. Ebook***