LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 10
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Oleh:

NAMA : HAKAN ISMAIL AFNAN

NIM : 103112400038

KELAS : 12-IF-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Algoritma Pencarian Nilai Minimum dan Maksimum

Pencarian nilai ekstrim adalah proses menemukan nilai terkecil dan terbesar dalam sebuah kumpulan data. Algoritma yang umum digunakan adalah pencarian linear, yang memeriksa setiap elemen data satu per satu. Nilai minimum dan maksimum diinisialisasi dari elemen pertama, kemudian diperbarui jika ditemukan nilai yang lebih kecil atau lebih besar. Dalam bahasa Go, data biasanya disimpan dalam array atau slice, dan pencarian dilakukan dengan membandingkan setiap elemen secara berurutan.

Prinsip Kerja Algoritma:

- Inisialisasi:
  Tetapkan nilai minimum dan maksimum awal dengan elemen pertama dari array.
- Iterasi Data:
  Lakukan perulangan dari elemen kedua hingga terakhir, bandingkan setiap elemen dengan nilai minimum dan maksimum yang sudah ada.
  - ~ Jika elemen lebih kecil dari nilai minimum, perbarui nilai minimum.
  - ~ Jika elemen lebih besar dari nilai maksimum, perbarui nilai maksimum.
- Hasil:
  Setelah seluruh data diperiksa, nilai minimum dan maksimum yang tersimpan merupakan nilai ekstrim data tersebut.

Keunggulan:

Metode ini banyak digunakan dalam berbagai aplikasi seperti pengolahan data berat badan balita, tinggi tanaman, harga buku, dan nilai ujian siswa, yang memerlukan informasi nilai ekstrim untuk analisis dan pengambilan keputusan.

II. GUIDED

Contoh 1

```go
//Nama  : Hakan Ismail Afnan
//NIM   : 103112400038
package main

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

    var tinggitanaman [500]float64
    for i := 0; i < n; i++ {
        fmt.Printf("masukkan 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("\ntinggi tanaman terpendek : %.2f cm\n", min)

}
```

Output:

```
PS C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODUL7> go run "c:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODUL7\Guided\1.go"
masukkan jumlah tanaman :5
masukkan tinggi tanaman ke-1 (cm):23
masukkan tinggi tanaman ke-2 (cm):25
masukkan tinggi tanaman ke-3 (cm):18
masukkan tinggi tanaman ke-4 (cm):31
masukkan tinggi tanaman ke-5 (cm):22

tinggi tanaman tertinggi : 31.00 cm

tinggi tanaman terpendek : 18.00 cm
PS C:\Users\User\Documents\HAKAN ISMAIL AFNAN\103112400038_MODUL7>
```

Penjelasan:

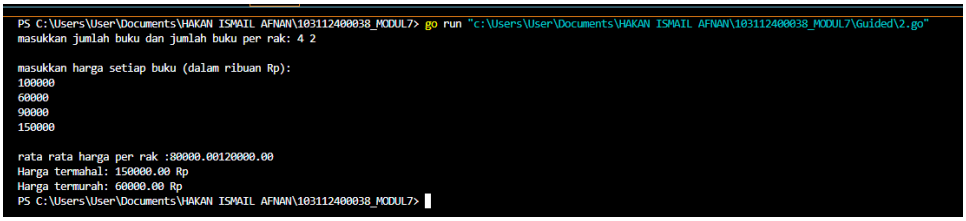
Program ini meminta pengguna memasukkan jumlah tanaman dan tinggi tiap tanaman secara berurutan, kemudian menyimpan data tersebut dalam array. Selanjutnya, program mencari tinggi tanaman minimal dan maksimal dengan membandingkan nilai-nilai dalam array.

Contoh Soal 2

```go
//Nama  : Hakan Ismail Afnan
//NIM   : 103112400038
package main

import (
    "fmt"
)

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

    var hargabuku [500]float64
    fmt.Println("\nmasukkan 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:



Penjelasan

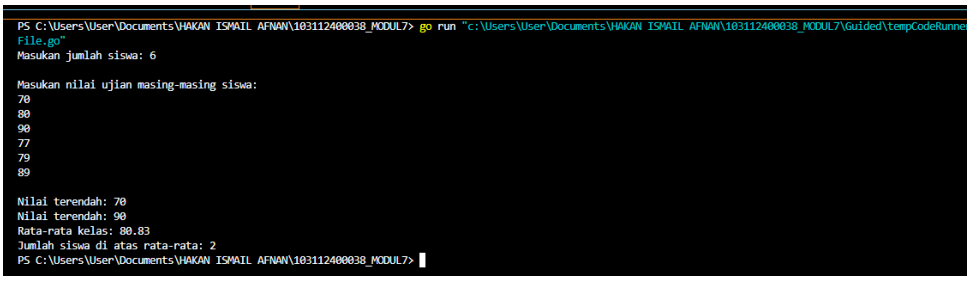
Program ini meminta pengguna memasukkan jumlah buku dan jumlah buku yang dapat dimasukkan ke setiap rak, lalu membaca harga masing-masing buku. Harga buku disimpan dalam array, kemudian program mengelompokkan buku berdasarkan kapasitas rak untuk menghitung rata-rata harga per rak. Selanjutnya, program mencari harga buku tertinggi dan terendah dengan membandingkan seluruh harga yang dimasukkan. Hasil rata-rata harga per rak dan harga ekstrim ditampilkan secara berurutan..

.

Contoh 3

```go
//Nama  : Hakan Ismail Afnan
//NIM   : 103112400038
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:



Penjelasan:

Program ini meminta pengguna memasukkan jumlah siswa dan nilai ujian mereka satu per satu. Nilai-nilai ini disimpan dalam array, dan program menghitung rata-rata nilai seluruh siswa. Selanjutnya, program menentukan nilai minimum dan maksimum serta menghitung jumlah siswa yang nilainya lebih tinggi dari rata-rata. Hasil akhir berupa nilai terendah, nilai tertinggi, rata-rata kelas, dan jumlah siswa di atas rata-rata ditampilkan ke layar.

III. UNGUIDED

Soal 1

```
//HAKAN ISMAIL AFNAN 103112400038
package main

import "fmt"

// Fungsi untuk menghitung faktorial
func faktorial(n int) int {
    hasil := 1
    for i := 1; i <= n; i++ {
        hasil *= i
    }
    return hasil
}

// Fungsi untuk menghitung permutasi
func permutasi(n, r int) int {
    if r > n {
        return 0
    }
    return faktorial(n) / faktorial(n-r)
}

// Fungsi untuk menghitung kombinasi
func kombinasi(n, r int) int {
    if r > n {
        return 0
    }
    return faktorial(n) / (faktorial(r) * faktorial(n-r))
}

func main() {
    var x, y, z, w int

    fmt.Print("Masukkan empat angka (x, y, z, w): ")
    fmt.Scan(&x, &y, &z, &w)

    if x >= z && y >= w {
        fmt.Println("Hasil Permutasi dan Kombinasi:")
        fmt.Printf("Permutasi(%d, %d): %d | Kombinasi(%d, %d):
%d\n", x, z, permutasi(x, z), x, z, kombinasi(x, z))
        fmt.Printf("Permutasi(%d, %d): %d | Kombinasi(%d, %d):
%d\n", y, w, permutasi(y, w), y, w, kombinasi(y, w))
```

```
    } else {
        fmt.Println("Tidak sesuai")
    }
}
```

Output:

```
PS C:\Users\User\Downloads\Hakan Ismail> go run "c:\Users\User\Downloads\Hakan Ismail\Unguided1.go"
Masukkan empat angka (x, y, z, w): 5 10 3 10
Hasil Permutasi dan Kombinasi:
Permutasi(5, 3): 60 | Kombinasi(5, 3): 10
Permutasi(10, 10): 3628800 | Kombinasi(10, 10): 1
PS C:\Users\User\Downloads\Hakan Ismail> go run "c:\Users\User\Downloads\Hakan Ismail\Unguided1.go"
Masukkan empat angka (x, y, z, w): 8 0 2 0
Hasil Permutasi dan Kombinasi:
Permutasi(8, 2): 56 | Kombinasi(8, 2): 28
Permutasi(0, 0): 1 | Kombinasi(0, 0): 1
PS C:\Users\User\Downloads\Hakan Ismail>
```

Penjelasan

Program ini menggunakan array dengan kapasitas tetap 1000 sesuai soal. Input berat disimpan di array, kemudian dilakukan pencarian nilai ekstrim dengan loop for biasa. Pendekatan ini cocok untuk data dengan ukuran tetap dan mudah diimplementasikan.

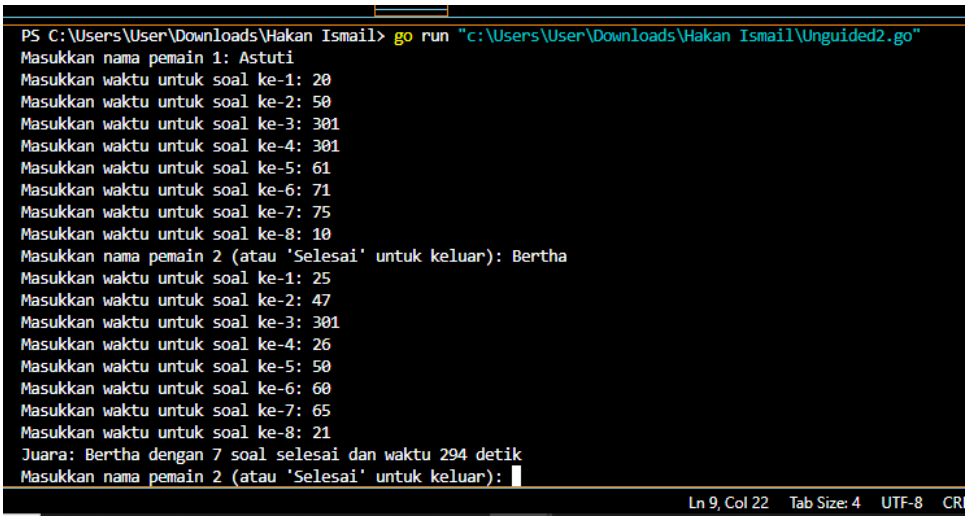
Soal 2

```
//HAKAN ISMAIL AFNAN 103112400038
package main

import "fmt"

// Fungsi untuk menghitung skor berdasarkan waktu yang
dihabiskan
func hitungSkor(waktu int, jumlahSoal *int, totalDurasi *int)
{
    if waktu <= 300 {
        *jumlahSoal++
        *totalDurasi += waktu
    }
}

func main() {
    var juara string
    var pemain1, pemain2 string
    var durasi1, durasi2 int
    var durasiTotal1, durasiTotal2 int
    var soalSelesai1, soalSelesai2 int
    var skorTertinggi, waktuTerbaik int
```

```
    // Input pemain pertama
    fmt.Print("Masukkan nama pemain 1: ")
    fmt.Scan(&pemain1)

    if pemain1 == "Selesai" {
        return
    }

    // Menghitung skor untuk pemain 1
    for i := 0; i < 8; i++ {
        fmt.Print("Masukkan waktu untuk soal ke-", i+1, ": ")
        fmt.Scan(&durasi1)
        hitungSkor(durasi1, &soalSelesai1, &durasiTotal1)
    }

    // Input pemain kedua
    for {
        fmt.Print("Masukkan nama pemain 2 (atau 'Selesai'
untuk keluar): ")
        fmt.Scan(&pemain2)

        if pemain2 == "Selesai" {
            break
        }

        // Menghitung skor untuk pemain 2
        for i := 0; i < 8; i++ {
            fmt.Print("Masukkan waktu untuk soal ke-", i+1, ":
")
            fmt.Scan(&durasi2)
            hitungSkor(durasi2, &soalSelesai2, &durasiTotal2)
        }

        // Menentukan pemenang berdasarkan skor
        switch {
        case soalSelesai1 > soalSelesai2:
            skorTertinggi, waktuTerbaik, juara = soalSelesai1,
durasiTotal1, pemain1
        case soalSelesai2 > soalSelesai1:
            skorTertinggi, waktuTerbaik, juara = soalSelesai2,
durasiTotal2, pemain2
        case soalSelesai1 == soalSelesai2:
            if durasiTotal1 < durasiTotal2 {
                skorTertinggi, waktuTerbaik, juara =
soalSelesai1, durasiTotal1, pemain1
```

```
            } else {
                skorTertinggi, waktuTerbaik, juara =
soalSelesai2, durasiTotal2, pemain2
            }
        }

        fmt.Printf("Juara: %s dengan %d soal selesai dan waktu
%d detik\n", juara, skorTertinggi, waktuTerbaik)
    }
}
```

Output:



Penjelasan

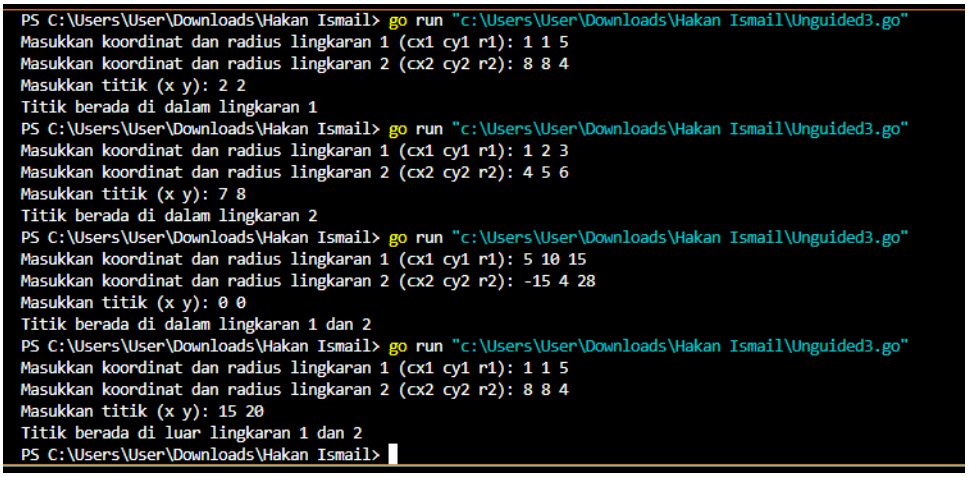
Program ini menerima input jumlah ikan (x) dan kapasitas wadah (y), kemudian membaca berat ikan sebanyak x data ke dalam array dengan kapasitas tetap 1000. Program mengelompokkan ikan ke dalam wadah sesuai kapasitas y, menghitung total berat ikan pada setiap wadah, dan menampilkan total berat per wadah secara berurutan. Selanjutnya, program menghitung dan menampilkan rata-rata berat ikan dari seluruh ikan yang telah dimasukkan.

Soal 3

```
/ //HAKAN ISMAIL AFNAN 103112400038

package main

import "fmt"

// Prosedur untuk mencetak deret Collatz
func cetakDeretCollatz(n int) {
    fmt.Print(n, " ") // Mencetak angka awal
```

```
    for n != 1 {
        if n%2 == 0 {
            n /= 2
        } else {
            n = 3*n + 1
        }
        fmt.Print(n, " ") // Mencetak setiap angka dalam deret
    }
}

func main() {
    var input int

    fmt.Print("Masukkan angka (kurang dari 1.000.000): ")
    fmt.Scan(&input)

    if input < 1000000 && input > 0 {
        cetakDeretCollatz(input)
    } else {
        fmt.Println("Error: Masukkan angka positif kurang dari
1.000.000")
    }
}
```

Output:



Penjelasan:

Program ini menggunakan tipe data beratBalita sebagai array dengan kapasitas 100 elemen bertipe float64. Fungsi temukanMinMax berfungsi menemukan berat balita minimum dan maksimum dengan membandingkan tiap elemen dalam array. Fungsi rataRataBerat menghitung rata-rata dengan menjumlahkan seluruh berat dan

membaginya dengan jumlah data. Program utama menerima input data berat balita, memproses data melalui fungsi-fungsi tersebut, dan menampilkan hasil ke layar.

IV. KESIMPULAN

Algoritma pencarian nilai minimum dan maksimum pada himpunan data dapat diimplementasikan dengan mudah menggunakan bahasa Go melalui iterasi sederhana pada array atau slice. Dengan memulai dari nilai awal elemen pertama, program secara berurutan membandingkan setiap data untuk menemukan nilai terkecil dan terbesar. Metode ini efektif untuk data yang tidak terurut dan memberikan hasil yang cepat dengan kompleksitas O(n). Kombinasi pencarian nilai ekstrim dengan perhitungan rata-rata juga memungkinkan analisis data yang lebih menyeluruh.

REFERENSI

Modul 10 Algoritma Pemrograman Pencarian Nilai Ekstrim pada Himpunan Data