LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XI

Pencarian Nilai Extreme pada Himpunan Data

Disusun Oleh :

Nadhif Atha Zaki / 2311102007

IF-11-05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Nilai ekstrem (extreme value) adalah nilai yang menunjukkan titik terendah (minimum) atau tertinggi (maksimum) dalam suatu himpunan data. Dalam analisis data atau statistika, nilai ekstrem penting untuk memahami rentang nilai dalam data, dan dapat digunakan untuk mendeteksi outlier atau nilai yang tidak biasa. Berikut adalah penjelasan dasar mengenai nilai ekstrem:

1. Nilai Minimum dan Maksimum

- Nilai Minimum (Minimum Value): Nilai terkecil dalam suatu himpunan data. Dalam konteks data numerik, nilai minimum adalah elemen yang lebih kecil dari atau sama dengan semua elemen lainnya.
- Nilai Maksimum (Maximum Value): Nilai terbesar dalam suatu himpunan data. Nilai ini lebih besar dari atau sama dengan semua elemen lainnya.

Contoh: Pada himpunan data {3, 5, 1, 9, 7}, nilai minimum adalah 1 dan nilai maksimum adalah 9.

2. Pencarian Nilai Ekstrem

Pencarian nilai ekstrem dapat dilakukan dengan metode pencarian langsung atau menggunakan algoritma tertentu. Proses ini sangat penting dalam berbagai aplikasi, seperti analisis statistik, optimasi, dan pembelajaran mesin.

- Algoritma Pencarian Ekstrem: Salah satu cara umum untuk mencari nilai ekstrem adalah dengan iterasi melalui seluruh elemen dalam data dan membandingkan setiap elemen dengan nilai ekstrem yang ditemukan sebelumnya. Misalnya, jika kita mencari nilai minimum, kita akan terus memperbarui nilai minimum saat menemukan elemen yang lebih kecil.

II. UNGUIDED

1.

Sebuah program digunakan untuk mendata berat anak kelinci yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat anak kelinci yang akan dijual.

Masukan terdiri dari sekumpulan bilangan, yang mana bilangan pertama adalah bilangan bulat N yang menyatakan banyaknya anak kelinci yang akan ditimbang beratnya. Selanjutnya N bilangan riil berikutnya adalah berat dari anak kelinci yang akan dijual.

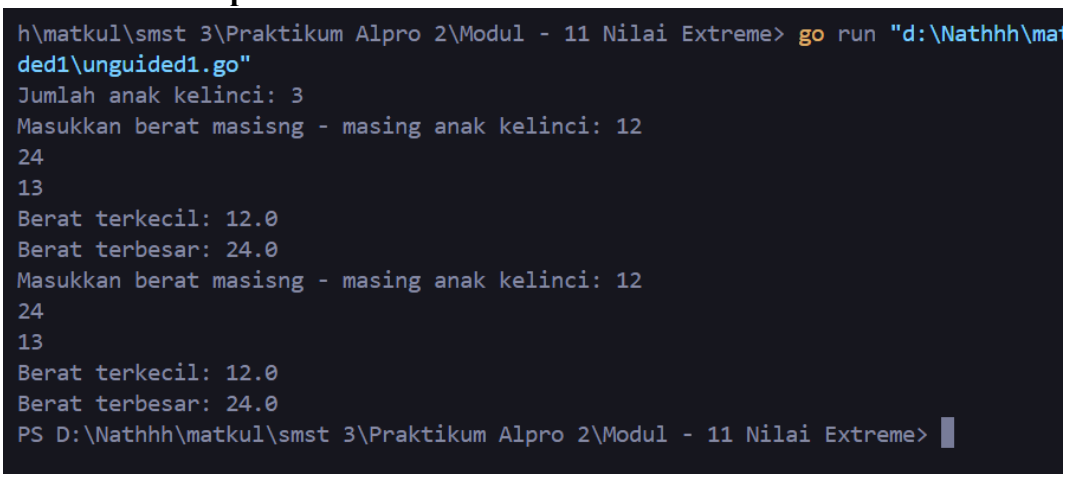
Keluaran terdiri dari dua buah bilangan riil yang menyatakan berat kelinci terkecil dan terbesar.

Sourcecode

```
package main

import "fmt"

func main() {
    var n int
    var berat []float64

    fmt.Print("Jumlah anak kelinci: ")
    fmt.Scan(&n)

    //input berat anak kelinci
    berat = make([]float64, n)
    fmt.Print("Masukkan berat masisng - masing anak kelinci:
")
    for i := 0; i < n; i++ {
        fmt.Scan(&berat[i])
    }

    //inisialisasi nilai extreme
    beratMinim := berat[0]
    beratMaks := berat[0]

    //pencarian nilai extreme
    for _, berat := range berat {
        if berat < beratMinim {
            beratMinim = berat
        }
        if berat > beratMaks {
            beratMaks = berat
```

```
        }
    }

    //output
    fmt.Printf("Berat terkecil: %.1f\n", beratMinim)
    fmt.Printf("Berat terbesar: %.1f\n", beratMaks)
}
```

Screenshoot Output



Deskripsi Program

Program di atas adalah aplikasi sederhana dalam bahasa Go untuk menentukan berat terkecil dan terbesar dari sejumlah anak kelinci. Program pertama-tama meminta pengguna untuk memasukkan jumlah anak kelinci dan berat masing-masing anak kelinci, yang kemudian disimpan dalam sebuah slice bertipe float64. Selanjutnya, program melakukan iterasi melalui slice untuk menemukan berat minimum dan maksimum dengan membandingkan setiap elemen dengan nilai awal yang diinisialisasi dari elemen pertama. Hasil akhirnya adalah berat terkecil dan terbesar yang ditampilkan dengan format satu angka di belakang desimal. Program ini mengilustrasikan cara menangani input array, iterasi, dan logika perbandingan secara efisien dalam Go.

2.

Sebuah program digunakan untuk menentukan tarif ikan yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat ikan yang akan dijual.

Masukan terdiri dari dua baris, yang mana baris pertama terdiri dari dua bilangan bulat x dan y. Bilangan x menyatakan banyaknya ikan yang akan dijual, sedangkan y adalah banyaknya ikan yang akan dimasukan ke dalam wadah. Baris kedua terdiri dari sejumlah x bilangan riil yang menyatakan banyaknya ikan yang akan dijual.

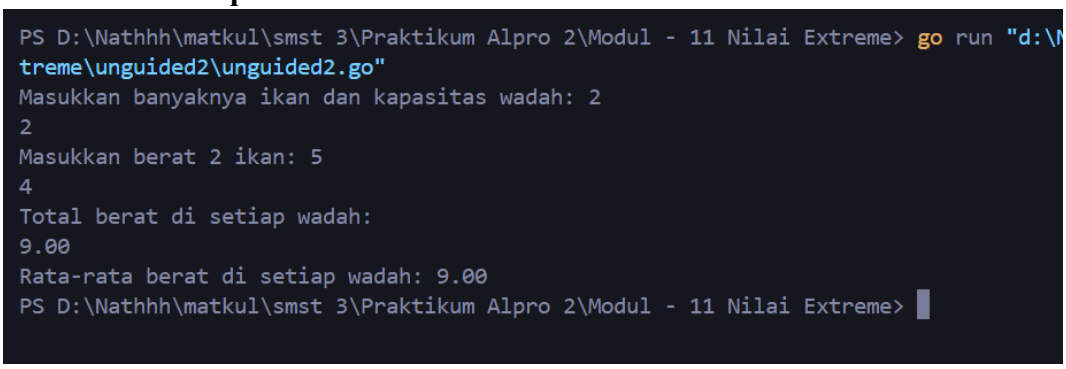
Keluaran terdiri dari dua baris. Baris pertama adalah kumpulan bilangan riil yang menyatakan total berat ikan di setiap wadah (jumlah wadah tergantung pada nilai x dan y, urutan ikan yang dimasukan ke dalam wadah sesuai urutan pada masukan baris ke-2). Baris kedua adalah sebuah bilangan riil yang menyatakan berat rata-rata ikan di setiap wadah.

Sourcecode

```
package main

import "fmt"

func main() {
    // Deklarasi variabel
    var x, y int
    var ikan [1000]float64

    // Input jumlah ikan dan kapasitas wadah
    fmt.Print("Masukkan banyaknya ikan dan kapasitas wadah:
")
    fmt.Scan(&x, &y)

    // Input berat ikan
    fmt.Printf("Masukkan berat %d ikan: ", x)
    for i := 0; i < x; i++ {
        fmt.Scan(&ikan[i])
    }

    // Proses penghitungan
    var totalBerat []float64
    for i := 0; i < x; i += y {
        jumlah := 0.0
        for j := i; j < i+y && j < x; j++ {
            jumlah += ikan[j]
        }
```

```
        totalBerat = append(totalBerat, jumlah)
    }

    // Hitung rata-rata berat per wadah
    jumlahTotal := 0.0
    for _, berat := range totalBerat {
        jumlahTotal += berat
    }
    rataRata := jumlahTotal / float64(len(totalBerat))

    // Output
    fmt.Println("Total berat di setiap wadah:")
    for _, berat := range totalBerat {
        fmt.Printf("%.2f ", berat)
    }
    fmt.Println()

    fmt.Printf("Rata-rata berat di setiap wadah: %.2f\n",
rataRata)
}
```

Screenshoot Output



Deskripsi Program

Program ini menghitung total berat ikan di setiap wadah dan rata-rata berat ikan per wadah. Pengguna memasukkan jumlah ikan x, kapasitas wadah y, dan berat masing-masing ikan. Program membagi ikan ke dalam wadah sesuai kapasitas, lalu menghitung total berat per wadah menggunakan loop. Hasil total berat semua wadah dijumlahkan untuk menentukan rata-rata berat per wadah. Program mencetak total berat per wadah dan rata-rata tersebut dalam format angka desimal.

3.

Pos Pelayanan Terpadu (posyandu) sebagai tempat pelayanan kesehatan perlu mencatat data berat balita (dalam kg). Petugas akan memasukkan data tersebut ke dalam array. Dari data yang diperoleh akan dicari berat balita terkecil, terbesar, dan reratanya.

Buatlah program dengan spesifikasi subprogram sebagai berikut:

```
type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita; bMin, bMax *float64) {
/* I.S. Terdefinisi array dinamis arrBerat
```



```
  Proses: Menghitung berat minimum dan maksimum dalam array
  F.S. Menampilkan berat minimum dan maksimum balita */
    ...
}

function rerata (arrBerat arrBalita) real {
/* menghitung dan mengembalikan rerata berat balita dalam array */
    ...
}
```

Perhatikan sesi interaksi pada contoh berikut ini (**teks bergaris bawah adalah input/read**)

```
Masukan banyak data berat balita : 4
Masukan berat balita ke-1: 5.3
Masukan berat balita ke-2: 6.2
Masukan berat balita ke-3: 4.1
Masukan berat balita ke-4: 9.9
Berat balita minimum: 4.10 kg
Berat balita maksimum: 9.90 kg
Rerata berat balita: 6.38 kg
```

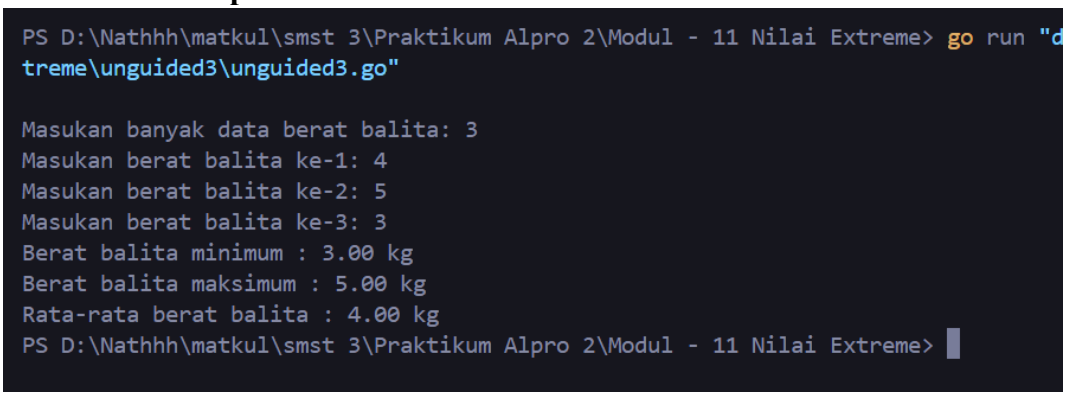
Sourcecode

```go
package main

import (
    "fmt"
    "strings"
)

// Definisi tipe data untuk array berat balita
type arrBalita [100]float64
```

```go
// Fungsi untuk menghitung berat minimum dan maksimum
func hitungMinMax(arrBerat arrBalita, n int, bMin, 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]
        }
    }
}

// Fungsi untuk menghitung rata-rata berat balita
func rerata(arrBerat arrBalita, n int) float64 {
    total := 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

    // Inputkan jumlah data balita
    fmt.Print("\nMasukan banyak data berat balita: ")
    fmt.Scanln(&n)

    // Input data berat balita satu per satu
    for i := 0; i < n; i++ {
        fmt.Printf("Masukan berat balita ke-%d: ", i+1)
        fmt.Scanln(&berat[i])
    }

    // Memanggil fungsi untuk menghitung minimum dan
maksimum
    hitungMinMax(berat, n, &bMin, &bMax)
```

```
    // Menghitung rata-rata berat balita
    rata := rerata(berat, n)

    // Menampilkan hasil
    fmt.Printf("Berat balita minimum : %.2f kg\n", bMin)
    fmt.Printf("Berat balita maksimum : %.2f kg\n", bMax)
    fmt.Printf("Rata-rata berat balita : %.2f kg\n", rata)

    // Menutup program dengan ucapan terima kasih
    fmt.Println("\nTerima kasih telah menggunakan program
ini!")
    fmt.Println(strings.Repeat("=", 40))
}
```

Screenshoot Output



Deskripsi Program

Program ini digunakan untuk mengolah data berat balita, termasuk menghitung berat minimum, maksimum, dan rata-rata dari sejumlah balita. Pertama, pengguna diminta untuk memasukkan jumlah data balita, kemudian memasukkan berat masing-masing balita yang disimpan dalam array arrBalita. Fungsi hitungMinMax digunakan untuk mencari berat balita dengan nilai minimum dan maksimum, sedangkan fungsi rerata menghitung rata-rata berat balita. Hasil pengolahan, berupa berat minimum, maksimum, dan rata-rata, kemudian ditampilkan dengan format dua angka desimal.