LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 7

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Oleh:

Muhammad Faris Rachmadi

103112400079

IF12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

7.1 Ide pencarian Nilai Max/Min

Pencarian adalah suatu proses yang lazim dilakukan di dalam kehidupan sehari-hari. Contoh penggunaannya dalam kehidupan nyata sangat beragam, misalnya pencarian file di dalam directory komputer, pencarian suatu teks di dalam sebuah dokumen, pencarian buku pada rak buku, dan contoh lainnya. Pertama pada modul ini akan dipelajari salah satu algoritma pencarian nilai terkecil atau terbesar pada sekumpulan data, atau biasa disebut pencarian nilai ekstrim.

Ide algoritma sederhana sekali. Karena data harus diproses secara sekuensial, maka nilai atau indeks ke nilai maksimum dari data yang telah diproses disimpan untuk dibandingkan dengan data berikutnya. Nilai yang berhasil disimpan sampai algoritma tersebut berakhir adalah nilai maksimum yang dicari. Adapun algoritmanya secara umum adalah sebagai berikut:

1) Jadikan data pertama sebagai nilai ekstrim

2) Lakukan validasi nilai ekstrim dari data kedua hingga data terakhir.

• Apabila nilai ekstrim tidak valid, maka update nilai ekstrims tersebut dengan data yang dicek.

3) Apabila semua data telah dicek, maka nilai ekstrim yang dimiliki adalah valid.

7.2 Pencarian Nilai Ekstrim pada Array Bertipe Data Dasar

Untuk array bertipe data dasar, pencarian nilai ekstrim dapat dilakukan dengan menggunakan algoritma pencarian linier. Berikut adalah langkah-langkah dasar dalam algoritma pencarian nilai ekstrim:

- **Inisialisasi**: Tentukan nilai awal untuk elemen maksimum dan minimum, yang biasanya diambil dari elemen pertama array.
- **Iterasi**: Lakukan iterasi untuk membandingkan setiap elemen array dengan nilai maksimum dan minimum yang sudah ada. Jika ditemukan elemen yang lebih besar dari nilai maksimum, perbarui nilai maksimum.

Begitu pula, jika ditemukan elemen yang lebih kecil dari nilai minimum, perbarui nilai minimum.

- **Hasil**: Setelah seluruh elemen array dibandingkan, hasilnya adalah nilai maksimum dan minimum dalam array tersebut.

7.3 Pencarian Nilai Ekstrim pada Array Bertipe Data Terstruktur

Pencarian nilai ekstrim pada array bertipe data terstruktur umumnya melibatkan pemilihan atribut yang akan dijadikan kriteria untuk pencarian. Langkah-langkahnya adalah sebagai berikut:

- **Inisialisasi:** Tentukan nilai ekstrim pertama berdasarkan atribut yang dipilih (misalnya, usia atau nilai). Nilai ini bisa diambil dari elemen pertama array.

- **Iterasi:** Lakukan iterasi terhadap setiap elemen dalam array dan bandingkan atribut yang dipilih untuk memperbarui nilai ekstrim jika ditemukan nilai yang lebih besar atau lebih kecil.

- **Hasil:** Setelah semua elemen diperiksa, hasilnya adalah elemen dengan nilai ekstrim berdasarkan atribut yang dipilih.

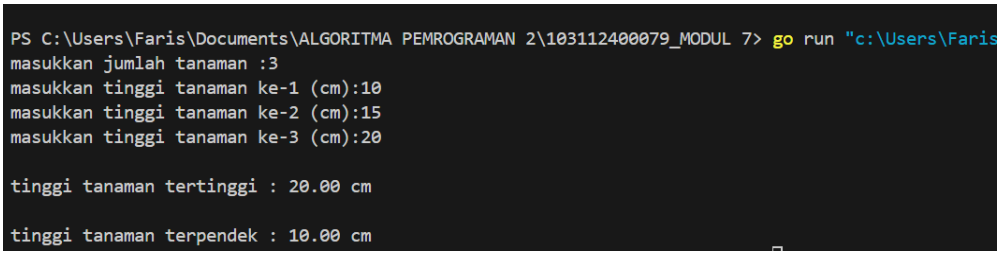
II. GUIDED

Guided 1

Code:

```
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:



Deskripsi:

Kode Go ini meminta pengguna untuk memasukkan jumlah tanaman yang ingin diukur tingginya, kemudian meminta input tinggi masing-masing tanaman dalam satuan cm. Setelah itu, kode akan menghitung dan menampilkan tinggi tanaman tertinggi dan terpendek dari data yang dimasukkan. Program ini menggunakan array untuk menyimpan tinggi tanaman, dan dengan iterasi, nilai tertinggi (maksimum) dan terendah (minimum) dihitung menggunakan kondisi if. Hasil akhirnya akan menampilkan tinggi tanaman tertinggi dan terpendek dengan format dua angka desimal.

Guided 2

Code:

```
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 harag 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:

```
PS C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODUL 7> go run "c:\Use
masukkan jumlah buku dan jumlah buku per rak: 6 3

masukkan harga setiap buku (dalam ribuan Rp):
100000
120000
110000
80000
75000
60000

rata rata harag per rak :110000.0071666.67
Harga termahal: 120000.00 Rp
Harga termurah: 60000.00 Rp
```

Deskripsi:

Kode Go ini meminta pengguna untuk memasukkan jumlah buku dan jumlah buku per rak. Selanjutnya, pengguna diminta untuk memasukkan harga setiap buku dalam satuan ribuan Rupiah. Program ini kemudian menghitung rata-rata harga per rak berdasarkan jumlah buku per rak yang dimasukkan. Untuk itu, program mengelompokkan harga-harga buku ke dalam rak sesuai jumlah yang ditentukan, menghitung total harga per rak, dan membaginya dengan jumlah buku per rak untuk mendapatkan rata-rata harga setiap rak. Selain itu, program juga mencari harga buku termahal dan termurah di antara semua buku yang dimasukkan. Hasil akhirnya menampilkan rata-rata harga per rak, harga termahal, dan harga termurah dalam format dua angka desimal.

Guided 3

Code:

```
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:

```
PS C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODUL 7> go run "c:\Users\Faris\
Masukan jumlah siswa: 5

Masukan nilai ujian masing-masing siswa:
80
85
95
75
100

Nilai terendah: 75
Nilai terendah: 100
Rata-rata kelas: 87.00
Jumlah siswa di atas rata-rata: 2
```

Deskripsi:

Kode Go ini meminta pengguna untuk memasukkan jumlah siswa dan nilai ujian masing-masing siswa. Kemudian, program menghitung total nilai dari semua siswa dan menghitung rata-rata nilai kelas. Program ini juga mencari nilai terendah dan tertinggi, serta menghitung jumlah siswa yang memiliki nilai di atas rata-rata kelas. Setelah itu, hasilnya akan ditampilkan, yaitu nilai terendah, nilai tertinggi, rata-rata kelas, dan jumlah siswa yang nilainya lebih tinggi dari rata-rata. Hasil tersebut ditampilkan dengan format dua angka desimal untuk rata-rata, dan angka bulat untuk nilai terendah, nilai tertinggi, dan jumlah siswa di atas rata-rata.

III. UNGUIDED

Unguided 1

Code:

```
package main

//Muhammad Faris Rachmadi
//103112400079

import (
    "fmt"
)

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

    var beratkelinci [1000]float64
    for i := 0; i < n; i++ {
        fmt.Printf("masukkan berat kelinci ke-%d (kg):", i+1)
        fmt.Scan(&beratkelinci[i])
    }
    min, max := beratkelinci[0], beratkelinci[0]
    for i := 1; i < n; i++ {
        if beratkelinci[i] > min {
            min = beratkelinci[i]
        }
        if beratkelinci[i] < max {
            max = beratkelinci[i]
        }
    }
    fmt.Printf("\nberat kelinci terkecil : %.2f kg\n", max)
    fmt.Printf("\nberat kelinci terbesar : %.2f kg\n", min)

}
```

Output:

```
PS C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODUL 7> go run "c:\Users\Fa
masukkan jumlah kelinci :3
masukkan berat kelinci ke-1 (kg):2.3
masukkan berat kelinci ke-2 (kg):1.5
masukkan berat kelinci ke-3 (kg):1.8

berat kelinci terkecil : 1.50 kg

berat kelinci terbesar : 2.30 kg
```

Deskripsi:

Kode Go ini meminta pengguna untuk memasukkan jumlah kelinci yang ingin diukur beratnya, dan kemudian memasukkan berat masing-masing kelinci dalam satuan kilogram. Program ini menggunakan array untuk menyimpan berat kelinci dan kemudian mencari berat kelinci terkecil dan terbesar. Dengan menggunakan kondisi if, program membandingkan setiap berat kelinci untuk mencari nilai maksimum (terberat) dan minimum (terringan). Hasil akhirnya akan menampilkan berat kelinci terbesar dan terkecil dalam format dua angka desimal. Namun, terdapat kesalahan dalam perbandingan antara min dan max yang seharusnya dibalik (nilai terbesar harus disimpan di max dan nilai terkecil di min), sehingga hasilnya perlu dibetulkan.

Unguided 2

Code:

```
package main

//Muhammad Faris Rachmadi
//103112400079

import "fmt"

func main() {
   var x, y int
   fmt.Print("Masukkan jumlah ikan yang akan dijual (x) dan jumlah ikan per
wadah (y): ")
   fmt.Scan(&x, &y)

   var beratIkan [1000]float64

   fmt.Println("Masukkan berat masing-masing ikan:")
   for i := 0; i < x; i++ {
      fmt.Printf("Ikan ke-%d: ", i+1)
      fmt.Scan(&beratIkan[i])
   }

   fmt.Println("\nTotal berat ikan per wadah:")
   for i := 0; i < x; i += y {
      var total float64
      for j := i; j < i+y && j < x; j++ {
         total += beratIkan[j]
      }
      fmt.Printf("Wadah %d: %.2f kg\n", i/y+1, total)
   }

   fmt.Println("\nRata-rata berat ikan per wadah:")
   for i := 0; i < x; i += y {
      var total float64
      var jumlah int
      for j := i; j < i+y && j < x; j++ {
         total += beratIkan[j]
         jumlah++
      }
      rata := total / float64(jumlah)
      fmt.Printf("Wadah %d: %.2f kg\n", i/y+1, rata)
   }
}
```

Output:

```
PS C:\Users\Faris\Documents\ALGORITMA PEMROGRAMAN 2\103112400079_MODUL 7> go run "c:\Users\Faris\
Masukkan jumlah ikan yang akan dijual (x) dan jumlah ikan per wadah (y): 6 3
Masukkan berat masing-masing ikan:
Ikan ke-1: 1.2
Ikan ke-2: 1.4
Ikan ke-3: 1.3
Ikan ke-4: 1
Ikan ke-5: 2.1
Ikan ke-6: 1.8

Total berat ikan per wadah:
Wadah 1: 3.90 kg
Wadah 2: 4.90 kg

Rata-rata berat ikan per wadah:
Wadah 1: 1.30 kg
Wadah 2: 1.63 kg
```

Deskripsi:

Kode Go ini meminta pengguna untuk memasukkan jumlah ikan yang akan dijual (x) dan jumlah ikan per wadah (y). Kemudian, pengguna diminta untuk memasukkan berat masing-masing ikan dalam satuan kilogram. Program ini pertama-tama menghitung total berat ikan dalam setiap wadah berdasarkan jumlah ikan per wadah yang dimasukkan. Setiap wadah akan dijumlahkan beratnya dan hasilnya ditampilkan dalam format dua angka desimal. Setelah itu, program menghitung rata-rata berat ikan per wadah dengan cara menjumlahkan berat ikan dalam wadah tersebut dan membaginya dengan jumlah ikan di dalam wadah. Hasil rata-rata berat ikan per wadah juga ditampilkan dalam format dua angka desimal.

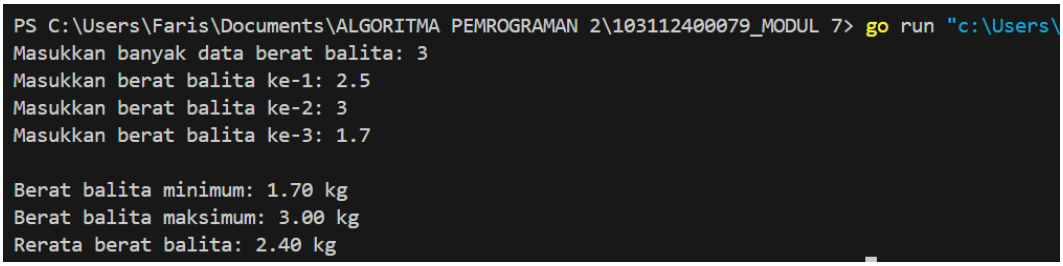
Unguided 3

Code:

```
package main

//Muhammad Faris Rachmadi
//103112400079
import "fmt"

type arrBalita [100]float64

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]
        }
    }
}

func rerata(arrBerat arrBalita, n int) float64 {
    var total float64
    for i := 0; i < n; i++ {
        total += arrBerat[i]
    }
    return total / float64(n)
}

func main() {
    var n int
    fmt.Print("Masukkan banyak data berat balita: ")
    fmt.Scan(&n)

    var beratBalita arrBalita

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

```
    var min, max float64
    hitungMinMax(beratBalita, n, &min, &max)
    rata := rerata(beratBalita, n)

    fmt.Printf("\nBerat balita minimum: %.2f kg\n", min)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", max)
    fmt.Printf("Rerata berat balita: %.2f kg\n", rata)
}
```

Output:



Deskripsi:

Kode Go ini menggunakan struktur array untuk menyimpan data berat balita dan memprosesnya untuk mencari berat balita minimum, maksimum, dan rata-rata. Fungsi hitungMinMax menerima parameter array arrBalita, jumlah data n, serta dua pointer untuk menyimpan nilai minimum (bMin) dan maksimum (bMax). Fungsi ini mengiterasi array untuk mencari nilai terkecil dan terbesar. Fungsi rerata digunakan untuk menghitung rata-rata berat balita dengan menjumlahkan seluruh berat dan membaginya dengan jumlah data n.

Dalam fungsi main, pengguna diminta untuk memasukkan banyak data berat balita yang akan diproses, kemudian berat setiap balita dimasukkan ke dalam array beratBalita. Setelah itu, fungsi hitungMinMax dipanggil untuk menentukan berat balita terendah dan tertinggi, sementara fungsi rerata digunakan untuk menghitung rata-rata berat balita. Hasil akhir yang ditampilkan mencakup berat balita minimum, maksimum, dan rata-rata dengan format dua angka desimal.

IV. KESIMPULAN

Kesimpulan dari laporan praktikum ini adalah bahwa pencarian nilai ekstrim, baik itu nilai maksimum maupun minimum, pada himpunan data dapat dilakukan dengan menggunakan algoritma pencarian linier. Melalui berbagai contoh aplikasi, seperti pencarian berat tanaman, harga buku, nilai ujian siswa, berat kelinci, dan ikan, dapat dilihat bahwa algoritma pencarian ini efektif untuk menemukan nilai ekstrim dalam kumpulan data yang besar. Selain itu, algoritma ini juga dapat diperluas untuk menghitung nilai rata-rata dalam kelompok data, seperti pada contoh perhitungan rata-rata berat ikan per wadah. Dengan pemahaman ini, diharapkan peserta praktikum dapat mengimplementasikan teknik pencarian nilai ekstrim ini pada berbagai kasus lainnya dengan lebih efisien.

V. REFRENSI

MODUL 10 PRAKTIKUM ALGORITMA PEMROGRAMAN

2 – PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA