LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL X

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Oleh :

NAMA : Felix Pedrosa Valentino

NIM : 103112400056

KELAS : IF – 12 – 01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Ide Pencarian Nilai Maksimum dan Minimum

Pencarian merupakan proses yang sering kita lakukan dalam kehidupan sehari-hari. Contoh penerapannya bervariasi, mulai dari mencari file di dalam direktori komputer, menemukan teks dalam sebuah dokumen, hingga mencari buku di rak. Dalam modul ini, kita akan mempelajari salah satu algoritma untuk mencari nilai terkecil atau terbesar dalam sekumpulan data, yang biasa dikenal dengan istilah pencarian nilai ekstrim.

Ide dasar dari algoritma ini sangat sederhana. Karena data perlu diproses secara sekuensial, kita akan menyimpan nilai atau indeks dari nilai maksimum yang telah diproses untuk dibandingkan dengan data yang berikutnya. Nilai yang berhasil disimpan hingga akhir proses adalah nilai maksimum yang kita cari.

Secara umum, langkah-langkah algoritma ini adalah sebagai berikut:

1. Anggap data pertama sebagai nilai ekstrim.

2. Validasi nilai ekstrim dari data kedua hingga data terakhir.

- Jika nilai ekstrim tidak valid, lakukan pembaruan nilai ekstrim tersebut dengan data yang sedang diperiksa.

3. Setelah semua data diperiksa, nilai ekstrim yang tersimpan adalah valid.

Dengan mengikuti langkah-langkah ini, kita dapat dengan mudah menemukan nilai maksimum atau minimum dari sekumpulan data.

Pencarian Nilai Ekstrim pada Array Bertipe Data Dasar

Misalkan kita memiliki sebuah array integer dengan kapasitas 2023, yang diisi dengan N bilangan bulat. Selanjutnya, kita akan melakukan pencarian untuk menemukan nilai terkecil dalam array tersebut. Selanjutnya, pada penjelasan di awal Bab 3, telah diuraikan bahwa salah satu aspek terpenting dalam pencarian adalah posisi atau indeks dari nilai yang dicari dalam suatu kumpulan data atau array.

Pencarian Nilai Ekstrem pada Array Bertipe Data Terstruktur

Dalam situasi yang lebih kompleks, pencarian nilai ekstrem dapat dilakukan dengan berbagai cara. Contohnya, kita bisa mencari mahasiswa dengan nilai tertinggi, lagu dengan durasi terpanjang, atau pembalap dengan waktu catatan balap tercepat. Sebagai ilustrasi, bayangkan kita memiliki array yang menyimpan data mahasiswa, dan kemudian kita menggunakan fungsi IPK untuk menemukan mahasiswa dengan IPK tertinggi.

II. GUIDED

1. Guided 1

Source Code :

```
// Felix Pedrosa V

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("Tinggi tanaman terpendek: %.2f cm\n", min)
}
```

Output :

Penjelasan Program :
Program diatas ditulis menggunakan bahasa Go dan bertujuan untuk mengukur tinggi tanaman. Pengguna akan diminta untuk memasukkan jumlah tanaman yang ingin diukur, kemudian program akan meminta input tinggi masing-masing tanaman dalam satuan sentimeter. Setelah semua data tinggi tanaman dimasukkan, program akan menghitung dan menampilkan tinggi tanaman tertinggi dan terendah. Dengan menggunakan array untuk menyimpan data tinggi tanaman, program ini mampu menangani hingga 500 tanaman dan memberikan hasil yang akurat dalam format dua desimal.

2. Guided 2

Source Code :

```
// Felix Pedrosa V

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 :

```
PS D:\KULIAH\SEMESTER 2\Algoritma & Pemrograman 2\CODING - GOLANG - Alpro 2> go run "d:\KULIAH\SEMESTER 2\Algoritma & Pemrograman 2\CODING -
 GOLANG - Alpro 2\alpro2_week9_modul10\103112400056_Guided2\103112400056_Guided2.go"
Masukkan jumlah buku dan jumlah buku per rak: 6 2

Masukkan harga setiap buku (dalam ribuan Rp):
20
30
70
80
56
34

Rata-rata harga per rak: 25.00 75.00 45.00
Harga termahal: 80.00 Rp
Harga termurah: 20.00 Rp
```

Penjelasan Program :

Program diatas ditulis menggunakan bahasa Go dan bertujuan untuk menghitung rata-rata harga buku yang disimpan di rak. Pengguna akan diminta untuk memasukkan jumlah total buku serta jumlah buku yang bisa ditampung per rak. Setelah itu, program akan meminta pengguna untuk memasukkan harga setiap buku dalam satuan ribuan Rupiah. Setelah semua harga buku terkumpul, program akan menghitung rata-rata

harga untuk masing-masing rak berdasarkan jumlah buku yang telah ditentukan. Selain itu, aplikasi ini juga mampu menentukan harga buku termahal dan termurah dari daftar yang telah diinput. Hasil perhitungan rata-rata harga per rak, bersama dengan harga termahal dan termurah, akan ditampilkan dengan format dua angka desimal, sehingga informasi yang disajikan menjadi jelas dan terstruktur untuk pengguna.

3. Guided 3

Source Code :

```
// Felix Pedrosa V

package main

import "fmt"

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

    var nilaiSiswa [200]float64
    var totalNilai float64 = 0

    fmt.Println("\nMasukkan 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 tertinggi: %.0f\n", max)
    fmt.Printf("Rata-rata kelas: %.2f\n", rataRata)
    fmt.Printf("Jumlah siswa di atas rata-rata: %d\n", diAtasRataRata)
}
// Felix Pedrosa V

package main

import "fmt"

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

    var nilaiSiswa [200]float64
    var totalNilai float64 = 0

    fmt.Println("\nMasukkan 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 tertinggi: %.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 :

Program diatas ditulis menggunakan bahasa Go dan bertujuan untuk menghitung dan menganalisis nilai ujian siswa. Pengguna akan diminta untuk memasukkan jumlah siswa terlebih dahulu. Selanjutnya, program akan mengajak pengguna untuk memasukkan nilai ujian masing-masing siswa. Setelah semua nilai terkumpul, program akan menghitung total nilai, rata-rata kelas, serta menentukan nilai terendah dan tertinggi. Selain itu, program juga akan menghitung jumlah siswa yang memiliki nilai di atas rata-rata. Hasil analisis, yang mencakup nilai terendah, tertinggi, rata-rata kelas, dan jumlah siswa di atas rata-rata, akan ditampilkan dalam format yang jelas, sehingga memberikan informasi yang bermanfaat bagi pengguna.

III. UNGUIDED

1. UnGuided 1

Source Code :

```
// Felix Pedrosa V

package main

import (
   "fmt"
   "math"
)

func main() {
   var N int
   fmt.Print("Masukkan jumlah anak kelinci: ")
   fmt.Scan(&N)

   if N <= 0 || N > 1000 {
      fmt.Println("Jumlah anak kelinci harus antara 1 dan 1000.")
      return
   }

   var beratKelinci [1000]float64
   fmt.Println("Masukkan berat anak kelinci:")
   for i := 0; i < N; i++ {
      fmt.Scan(&beratKelinci[i])
   }

   terkecil := math.MaxFloat64
   terbesar := -math.MaxFloat64

   for i := 0; i < N; i++ {
      if beratKelinci[i] < terkecil {
         terkecil = beratKelinci[i]
      }
      if beratKelinci[i] > terbesar {
         terbesar = beratKelinci[i]
```

```
            }
        }

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

Output :

```
PS D:\KULIAH\SEMESTER 2\Algoritma & Pemrograman 2\CODING - GOLANG - Alpro 2> go run "d:\KULIAH\SEMESTER 2\Algoritma & Pemrograman 2\CODING -
 GOLANG - Alpro 2\alpro2_week9_modul10\103112400056_Unguided1\103112400056_Unguided1.go"
Masukkan jumlah anak kelinci: 5
Masukkan berat anak kelinci:
11
10
12
14
21
Berat kelinci terkecil: 10.00
Berat kelinci terbesar: 21.00
```

Penjelasan Program :

Program diatas ditulis menggunakan bahasa Go dan bertujuan untuk menghitung dan menampilkan berat terkecil serta terbesar dari anak kelinci yang akan dijual di pasar. Pengguna diminta untuk memasukkan jumlah anak kelinci (N), yang harus berada dalam rentang 1 hingga 1000. Selanjutnya, program akan meminta pengguna untuk menginput berat dari masing-masing anak kelinci. Dengan memanfaatkan array berkapasitas 1000, program akan menyimpan data berat tersebut, lalu menghitung nilai berat terkecil dan terbesar melalui perbandingan. Akhirnya, hasil perhitungan akan disajikan kepada pengguna dalam format yang jelas, menampilkan berat terkecil dan terbesar dengan dua angka desimal.

2. UnGuided 2

Source Code :

```
// Felix Pedrosa V

package main

import "fmt"

func main() {
    const maxIkan = 1000
    var beratIkan [maxIkan]float64
    var x, y int
```

```
    // Input jumlah ikan dan jumlah ikan per wadah
    fmt.Print("Masukkan jumlah ikan yang akan dijual (x): ")
    fmt.Scan(&x)
    fmt.Print("Masukkan jumlah ikan per wadah (y): ")
    fmt.Scan(&y)

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

    // Proses pengelompokan ke dalam wadah
    var wadah [][]float64
    for i := 0; i < x; i += y {
        akhir := i + y
        if akhir > x {
            akhir = x
        }
        wadah = append(wadah, beratIkan[i:akhir])
    }

    // Hitung dan tampilkan total berat setiap wadah
    var totalSemuaWadah float64
    fmt.Println("\nTotal berat ikan di setiap wadah:")
    for i, w := range wadah {
        var total float64
        for _, berat := range w {
            total += berat
        }
        fmt.Printf("Wadah %d: %.2f kg\n", i+1, total)
        totalSemuaWadah += total
    }

    // Hitung dan tampilkan rata-rata berat per wadah
    rataRata := totalSemuaWadah / float64(len(wadah))
    fmt.Printf("\nBerat rata-rata ikan di setiap wadah: %.2f kg\n",
rataRata)
```

```
}
```

Output :

```
PS D:\KULIAH\SEMESTER 2\Algoritma & Pemrograman 2\CODING - GOLANG - Alpro 2> go run "d:\KULIAH\SEMESTER 2\Algoritma & Pemrograman 2\CODING -
 GOLANG - Alpro 2\alpro2_week9_modul10\103112400056_Unguided2\103112400056_Unguided2.go"
Masukkan jumlah ikan yang akan dijual (x): 4
Masukkan jumlah ikan per wadah (y): 4
Masukkan berat ikan satu per satu:
Berat ikan ke-1: 2
Berat ikan ke-2: 4
Berat ikan ke-3: 7
Berat ikan ke-4: 8

Total berat ikan di setiap wadah:
Wadah 1: 21.00 kg

Berat rata-rata ikan di setiap wadah: 21.00 kg
```

Penjelasan Program :

Program diatas ditulis menggunakan bahasa Go dan bertujuan untuk menghitung dan menampilkan total berat ikan yang akan dijual berdasarkan input dari pengguna. Dengan memanfaatkan array yang dapat menampung hingga 1000 data berat ikan, pengguna diminta untuk memasukkan total jumlah ikan (x) serta jumlah ikan per wadah (y). Setelah itu, pengguna akan diminta untuk memasukkan berat setiap ikan secara individual. Program ini akan mengelompokkan ikan ke dalam wadah sesuai dengan jumlah yang telah ditentukan dan menghitung total berat di setiap wadah. Di akhir proses, program juga akan menghitung rata-rata berat ikan per wadah dan menyajikannya kepada pengguna.

3. UnGuided 3

Source Code :

```
// Felix Pedrosa V

package main

import "fmt"

// Tipe data untuk menyimpan berat balita
type arrBalita [100]float64

// Fungsi untuk menghitung nilai minimum dan maksimum
func hitungMinMax(arr arrBalita, bMin, bMax *float64) {
    *bMin = arr[0]
    *bMax = arr[0]
```

```
    for _, berat := range arr {
        if berat != 0 { // Hanya hitung yang bukan nol
            if berat < *bMin {
                *bMin = berat
            }
            if berat > *bMax {
                *bMax = berat
            }
        }
    }
}

// Fungsi untuk menghitung dan mengembalikan rerata
func rerata(arr arrBalita) float64 {
    var total float64
    var count int

    for _, berat := range arr {
        if berat != 0 { // Hanya hitung yang bukan nol
            total += berat
            count++
        }
    }

    if count == 0 {
        return 0 // Menghindari pembagian dengan nol
    }
    return total / float64(count)
}

func main() {
    var data arrBalita
    var jumlahData int

    // Input jumlah data berat balita
    fmt.Print("Masukkan banyak data berat balita: ")
    fmt.Scan(&jumlahData)

    // Input berat balita
    for i := 0; i < jumlahData; i++ {
```

```
        fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
        fmt.Scan(&data[i])
    }

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

    fmt.Printf("Berat 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 :

Penjelasan Program :

Program diatas ditulis menggunakan bahasa Go dan bertujuan untuk menghitung berat balita yang disimpan dalam sebuah array. Aplikasi ini memiliki tiga fungsi utama: fungsi `hitungMinMax` untuk menentukan berat balita yang minimum dan maksimum, serta fungsi `rerata` untuk menghitung rata-rata berat balita.

Pengguna perlu menyiapkan sebuah array dengan kapasitas hingga 100 data bertipe `float64` yang akan digunakan untuk menyimpan berat balita. Setelah itu, pengguna akan diminta untuk memasukkan jumlah data yang relevan, serta berat setiap balita secara berurutan. Begitu data telah diinput, program akan memproses informasi tersebut dan menghasilkan output yang mencakup berat balita terkecil, tertinggi, dan rata-rata. Selain itu, program juga dilengkapi dengan pengecekan agar berat nol tidak dihitung dalam perhitungan.

IV. KESIMPULAN

Kesimpulan dari modul ini menunjukkan bahwa pencarian nilai ekstrim, baik maksimum maupun minimum, adalah langkah penting dalam pengolahan data yang relevan di berbagai aspek kehidupan sehari-hari. Dengan menggunakan algoritma yang sederhana, pengguna dapat dengan mudah menentukan nilai tertinggi dan terendah dalam suatu kumpulan data, baik dalam bentuk angka biasa maupun data terstruktur seperti informasi mahasiswa. Modul ini juga menjelaskan cara implementasi algoritma pencarian menggunakan bahasa pemrograman Go, termasuk pemanfaatan array untuk menyimpan data serta metode efisien untuk menghitung nilai ekstrim. Dengan pemahaman yang mendalam tentang konsep ini, pengguna akan mampu mengembangkan program yang lebih kompleks dan berguna, seperti analisis data akademik atau pengelolaan informasi lainnya, sehingga meningkatkan keterampilan pemrograman dan analisis data secara keseluruhan.

V. REFERENSI

Modul 10 - Praktikum Alpro 2