LAPORAN

PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL7

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Oleh:

NAMA: MOHAMMAD REYHAN ARETHA FATIN

NIM: 103112400078

KELAS: IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

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.

2. Pencarian Nilai Ekstrem di Array Bertipe Data Dasar

Pencarian nilai ekstrim pada array bertipe data dasar adalah proses untuk menemukan nilai maksimum atau minimum dalam sebuah kumpulan data. Algoritma ini dimulai dengan menganggap elemen pertama sebagai nilai ekstrim sementara, lalu membandingkan setiap elemen berikutnya dengan nilai tersebut. Jika ditemukan nilai yang lebih besar (untuk maksimum) atau lebih kecil (untuk minimum), nilai ekstrim akan diperbarui. Proses ini dilakukan secara sekuensial hingga seluruh elemen array diperiksa, dan nilai ekstrim yang ditemukan adalah hasil akhir dari pencarian. Dengan kompleksitas waktu O(n), algoritma ini efisien dan sering digunakan dalam berbagai aplikasi seperti analisis data dan pengolahan informasi numerik.

3. Pencarian Nilai Ekstrem di Array Bertipe Data Terstruktur

Pencarian nilai ekstrim pada array bertipe data terstruktur melibatkan pencarian nilai tertinggi atau terendah dari atribut tertentu dalam objek atau record. Misalnya, untuk menemukan mahasiswa dengan IPK tertinggi, kita membandingkan nilai IPK dari setiap elemen dalam array. Proses dimulai dengan memilih elemen pertama sebagai nilai ekstrim sementara dan memperbarui nilai ekstrim saat ditemukan data dengan atribut yang lebih besar atau lebih kecil. Setelah seluruh elemen diperiksa, nilai ekstrim yang ditemukan adalah hasil akhir pencarian.

GUIDED 1

SOURCE CODE:

```
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 teerpendek: %.2f cm/n", min)
}
```

OUTPUT:

```
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL7> go run "d:\ALGORITMA
masukan jumlah tanaman: 4
masukan tinggi tanaman ke -1(cm) : 2
masukan tinggi tanaman ke -2(cm) : 3
masukan tinggi tanaman ke -3(cm) : 4
masukan tinggi tanaman ke -4(cm) : 5

Tinggi Tanaman Tertinggi: 5.00 cm/nTinggi  tanaman teerpendek: 2.00 cm/n
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL7>
```

DEKSRIPSI:

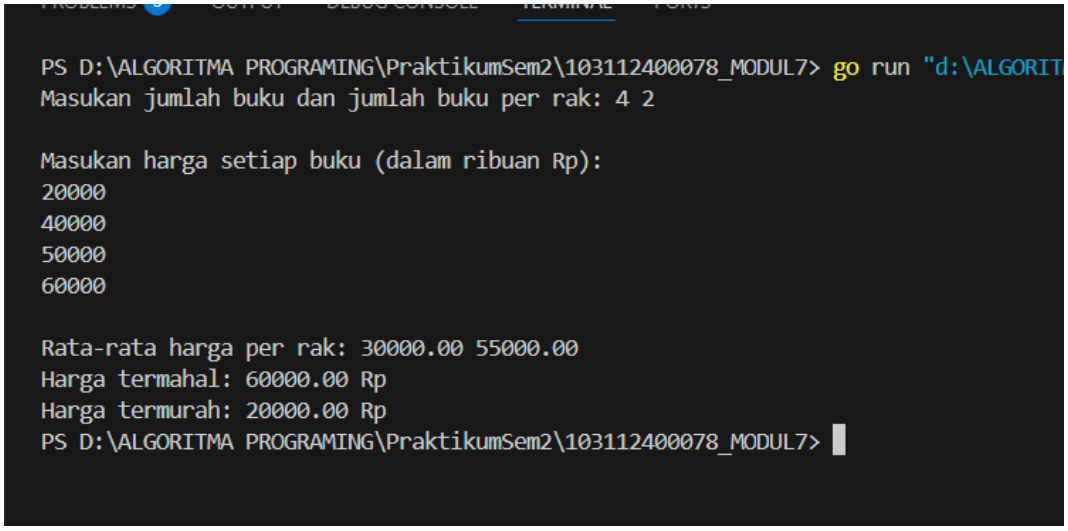
Program ini meminta pengguna untuk memasukkan jumlah tanaman yang ingin diukur, kemudian meminta input tinggi tanaman satu per satu dalam satuan cm. Setelah itu, program menghitung tinggi tanaman tertinggi dan terpendek dari daftar yang telah dimasukkan dengan cara membandingkan setiap nilai tinggi tanaman yang dimasukkan. Hasilnya, program akan menampilkan tinggi tanaman tertinggi dan terpendek dalam format dua angka desimal. Program ini menggunakan array dengan kapasitas maksimum 500 untuk menyimpan data tinggi tanaman dan mengoptimalkan proses pencarian nilai minimum dan maksimum menggunakan perulangan.

GUIDED 2

SOURCE CODE:

```
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:



DEKSRIPSI:

Program ini meminta pengguna untuk memasukkan jumlah buku dan jumlah buku yang dapat ditempatkan per rak, kemudian meminta input harga setiap buku dalam satuan ribuan rupiah. Program kemudian menghitung rata-rata harga buku per rak berdasarkan jumlah buku yang dimasukkan per rak, dengan membagi total harga buku dalam satu rak dengan jumlah buku per rak. Selain itu, program juga mencari harga buku termahal dan termurah dari seluruh daftar harga yang dimasukkan. Hasilnya, program akan menampilkan rata-rata harga per rak, harga termahal, dan harga termurah dengan format dua angka desimal.

GUIDED 3

SOURCE 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 D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL7> go run "d:\ALGORITMA
Masukan jumlah siswa: 3

Masukan nilai ujian masing-masing siswa:
90
88
72

Nilai terendah: 72
Nilai terendah: 90
Rata-rata kelas: 83.33
Jumlah siswa di atas rata-rata: 2
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL7>
```

DEKSRIPSI:

Program 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 juga mencari nilai terendah dan tertinggi di antara semua nilai siswa. Selain itu, program menghitung berapa banyak siswa yang memiliki nilai di atas rata-rata kelas. Hasil yang ditampilkan meliputi nilai terendah, nilai tertinggi, rata-rata kelas, dan jumlah siswa yang nilai ujian mereka berada di atas rata-rata kelas. Semua hasil ditampilkan dengan format yang sesuai.

II. UNGUIDED

UNGUIDED 1

SOURCE CODE

```
package main
//103112400078 Mohammad Reyhan Aretha Fatin
import "fmt"

func main() {
    var jumlahKelinci int
    var beratKelinci [1000]float64
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&jumlahKelinci)

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

    min := beratKelinci[0]
    max := beratKelinci[0]

    for i := 1; i < jumlahKelinci; i++ {
        if beratKelinci[i] < min {
            min = beratKelinci[i]
        }
        if beratKelinci[i] > max {
            max = beratKelinci[i]
        }
    }

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

OUTPUT

```
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL7> go run "d:\ALGORITMA
Masukkan jumlah anak kelinci: 3
Masukkan berat kelinci ke-1 (kg): 1
Masukkan berat kelinci ke-2 (kg): 2
Masukkan berat kelinci ke-3 (kg): 3
Berat kelinci terkecil: 1.00 kg
Berat kelinci terbesar: 3.00 kg
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL7>
```

DEKSRIPSI

Program ini meminta pengguna untuk memasukkan jumlah anak kelinci dan berat masing-masing kelinci dalam satuan kilogram. Program kemudian menyimpan berat kelinci-kelinci tersebut dalam sebuah array dan menghitung berat kelinci terkecil dan terbesar. Setelah melakukan perbandingan antara setiap berat kelinci, program akan menampilkan hasil berat kelinci terkecil dan terbesar dengan format dua angka desimal. Program ini efektif untuk memantau rentang berat anak kelinci yang dimasukkan.

UNGUIDED 2

SOURCE CODE

```
package main
//103112400078 Mohammad Reyhan Aretha Fatin
import "fmt"

func main() {
   var totalIkan, ikanPerWadah int
   var beratIkan [1000]float64

   fmt.Print("Masukkan jumlah ikan (x) dan jumlah ikan per wadah (y): ")
   fmt.Scan(&totalIkan, &ikanPerWadah)

   fmt.Print("Masukkan berat setiap ikan: ")
   for i := 0; i < totalIkan; i++ {
      fmt.Scan(&beratIkan[i])
   }

   jumlahWadah := (totalIkan + ikanPerWadah - 1) / ikanPerWadah
   totalBeratPerWadah := make([]float64, jumlahWadah)
   jumlahIkanPerWadah := make([]int, jumlahWadah)

   for i := 0; i < totalIkan; i++ {
      wadahIdx := i / ikanPerWadah
      totalBeratPerWadah[wadahIdx] += beratIkan[i]
      jumlahIkanPerWadah[wadahIdx]++
   }

   for i := 0; i < jumlahWadah; i++ {
      fmt.Printf("| Wadah ke-%d: %.2f kg |", i+1, totalBeratPerWadah[i])
      if jumlahIkanPerWadah[i] > 0 {
                              fmt.Printf(" Berat rata-rata: %.2f kg |",
totalBeratPerWadah[i]/float64(jumlahIkanPerWadah[i]))
      }
      fmt.Println()
   }
}
```

OUTPUT

```
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL7> go run "d:\ALGORI
Masukkan jumlah ikan (x) dan jumlah ikan per wadah (y): 4 4
Masukkan berat setiap ikan: 3
4
6
8
| Wadah ke-1: 21.00 kg | Berat rata-rata: 5.25 kg |
PS D:\ALGORITMA PROGRAMING\PraktikumSem2\103112400078_MODUL7>
```

DEKSRIPSI

Program ini meminta pengguna untuk memasukkan jumlah ikan dan jumlah ikan yang akan ditempatkan dalam setiap wadah. Kemudian, pengguna diminta untuk memasukkan berat setiap ikan. Program menghitung berapa banyak wadah yang diperlukan berdasarkan jumlah ikan dan ikan per wadah, serta menghitung total berat ikan dalam setiap wadah. Program juga menghitung berat rata-rata ikan dalam setiap wadah dan menampilkan hasilnya. Setiap wadah akan ditampilkan dengan total berat ikan dan, jika ada lebih dari satu ikan di wadah tersebut, berat rata-ratanya juga ditampilkan dengan format dua angka desimal.

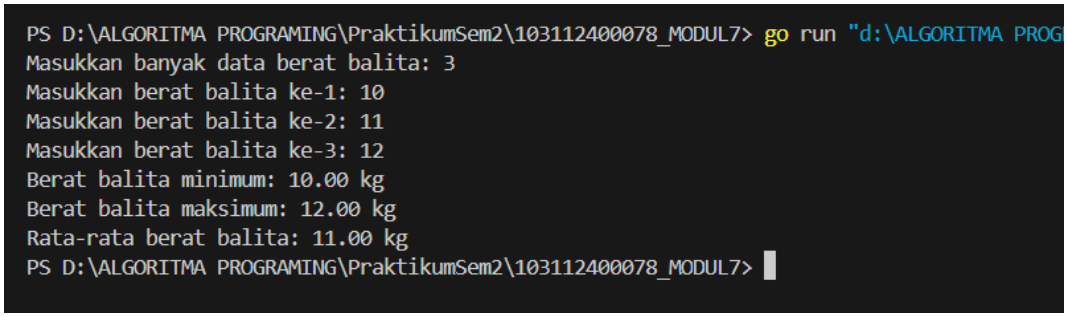
UNGUIDED 3

SOURCE CODE

```
package main
//103112400078 Mohammad Reyhan Aretha Fatin
import "fmt"

func hitungMinMaxRata(arr []float64, n int) (min, max, rata float64) {
    min, max = arr[0], arr[0]
    var total, count float64

    for i := 0; i < n; i++ {
        if arr[i] > 0 {
            if arr[i] < min {
                min = arr[i]
            }
            if arr[i] > max {
                max = arr[i]
            }
            total += arr[i]
            count++
        }
    }
    if count > 0 {
        rata = total / count
    }
    return
}

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

    if n <= 0 || n > 100 {
        fmt.Println("Jumlah data tidak valid.")
        return
    }

    var dataBalita [100]float64
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
        fmt.Scan(&dataBalita[i])
    }
```

```
    min, max, rata := hitungMinMaxRata(dataBalita[:n], n)
    fmt.Printf("Berat balita minimum: %.2f kg\n", min)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", max)
    fmt.Printf("Rata-rata berat balita: %.2f kg\n", rata)
}
```

OUTPUT



DEKSRIPSI

Program ini meminta pengguna untuk memasukkan jumlah data berat balita, kemudian meminta input berat balita satu per satu. Program menyimpan data berat balita dalam array dan menghitung berat balita minimum, maksimum, serta rata-rata berat balita yang valid (berat lebih besar dari 0). Fungsi hitungMinMaxRata digunakan untuk melakukan perhitungan tersebut. Setelah semua data dimasukkan, program akan menampilkan berat balita minimum, maksimum, dan rata-rata dengan format dua angka desimal. Program juga melakukan validasi jumlah data agar berada dalam rentang yang sesuai (1 hingga 100 data).

III. KESIMPULAN

Kesimpulan dari laporan ini adalah bahwa program-program yang telah dibahas berhasil mengimplementasikan algoritma pencarian nilai ekstrim (tertinggi dan terendah) dalam berbagai konteks penggunaan data, seperti pada tanaman, buku, siswa, kelinci, ikan, dan balita. Setiap program menggunakan konsep dasar pencarian nilai maksimum dan minimum dengan memanfaatkan array untuk menyimpan data dan perulangan untuk memproses setiap elemen. Selain itu, beberapa program juga mengimplementasikan perhitungan nilai rata-rata dan melakukan validasi terhadap data yang dimasukkan oleh pengguna. Algoritma yang digunakan memiliki efisiensi waktu O(n) dan diterapkan pada berbagai jenis data bertipe dasar maupun terstruktur, memberikan hasil yang valid dan sesuai dengan konteksnya.

REFERENSI

MODUL 7 PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA ALGORITMA PEMOGRAMAN 2