LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL X

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Disusun Oleh :

Nok Nadia

2311102298

IF-11-08

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM PURWOKERTO

2024

I. DASAR TEORI

1. Pengenalan Algoritma Pencarian Nilai Ekstrim

Pencarian nilai ekstrim adalah proses untuk menemukan nilai terbesar (maksimum) atau terkecil (minimum) dari sekumpulan data. Proses ini sangat umum diterapkan dalam berbagai situasi, baik di bidang ilmu komputer maupun dalam kehidupan sehari-hari, seperti pencarian file di komputer atau menentukan nilai tertinggi dalam ujian.

Pada dasarnya, algoritma pencarian nilai ekstrim bekerja dengan memeriksa elemen data secara sekuensial (linear). Nilai ekstrim sementara disimpan, lalu diperbarui jika ditemukan nilai yang lebih ekstrim (lebih besar untuk maksimum atau lebih kecil untuk minimum) di elemen berikutnya.

2. Langkah Algoritma Pencarian Nilai Ekstrim

Algoritma pencarian nilai ekstrim dimulai dengan menginisialisasi elemen pertama sebagai nilai ekstrim sementara. Selanjutnya, dilakukan iterasi untuk membandingkan nilai ekstrim sementara dengan elemen-elemen berikutnya dalam data. Jika elemen saat ini memenuhi kriteria sebagai nilai ekstrim, maka nilai ekstrim sementara diperbarui dengan elemen tersebut. Setelah semua elemen selesai diperiksa, nilai ekstrim yang tersimpan pada akhir proses adalah hasil yang valid.

3. Implementasi pada Array Bertipe Dasar

Pada array yang berisi elemen bertipe data dasar (seperti integer atau float), algoritma pencarian nilai ekstrim dilakukan dengan membandingkan nilai elemen array satu per satu. Implementasi ini sederhana dan efisien untuk dataset kecil hingga menengah.

```
max ← array[0]
for i ← 1 to n - 1 do
     if array[i] > max then
         max ← array[i]
return max
```

4. Implementasi pada Array Bertipe Terstruktur

Pada kasus lebih kompleks, seperti pencarian nilai ekstrim pada array bertipe terstruktur (misalnya, data mahasiswa), algoritma dapat dimodifikasi untuk mengembalikan indeks dari elemen ekstrim, sehingga identitas data yang bersangkutan juga dapat diperoleh.

```
idx ← 0
for i ← 1 to n - 1 do
    if T[i].ipk > T[idx].ipk then
        idx ← i
return idx
```

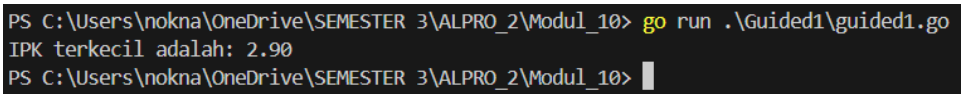
II. GUIDED

Guided 1

```go
//2311102298
//Nok Nadia
package main

import "fmt"

type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk                       float64
}

type arrMhs [2023]mahasiswa

func IPK_1(T arrMhs, n int) float64 {
    var terkecil float64 = T[0].ipk
    var j int = 1

    for j < n {
        if terkecil > T[j].ipk {
            terkecil = T[j].ipk
        }
        j = j + 1
    }
    return terkecil
}

func main() {
    var mahasiswaArr arrMhs
    mahasiswaArr[0] = mahasiswa{"Alice", "2211102123", "IF-1", "Teknik Informatika", 3.2}
    mahasiswaArr[1] = mahasiswa{"Bob", "2211102124", "IF-2", "Teknik Informatika", 3.8}
    mahasiswaArr[2] = mahasiswa{"Charlie", "22111021235", "IF-1", "Teknik Informatika", 2.9}
    mahasiswaArr[3] = mahasiswa{"Diana", "22111022189", "IF-3", "Teknik Informatika", 3.0}
    n := 4

    ipkTerkecil := IPK_1(mahasiswaArr, n)

    fmt.Printf("IPK terkecil adalah: %.2f\n", ipkTerkecil)
```

```
}
```

Screenshots Output



Deskripsi:

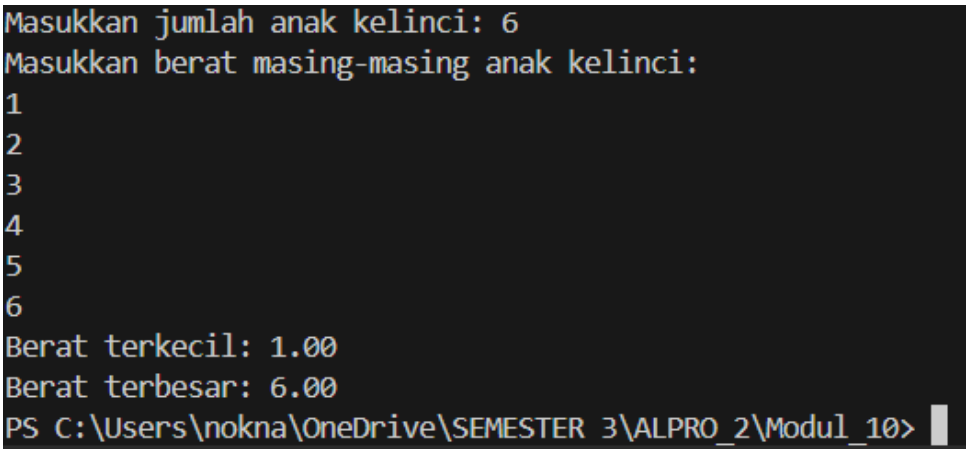
Program di atas digunakan untuk mencari IPK terkecil dari sekumpulan data mahasiswa yang disimpan dalam array bertipe terstruktur. Struktur `mahasiswa` menyimpan informasi seperti nama, NIM, kelas, jurusan, dan IPK. Fungsi `IPK_1` menerima array mahasiswa (`arrMhs`) dan jumlah data (`n`), kemudian melakukan iterasi untuk membandingkan nilai IPK setiap mahasiswa, memperbarui nilai terkecil jika ditemukan IPK yang lebih kecil dari nilai sebelumnya. Pada fungsi `main`, array diisi dengan beberapa data mahasiswa, dan fungsi `IPK_1` dipanggil untuk mencari IPK terkecil, yang kemudian ditampilkan sebagai output dengan format desimal dua angka.

III. UNGUIDED

Unguided 1

```go
//2311102298
//Nok Nadia
package main

import "fmt"

func main() {
    var beratKelinci [1000]float64

    var n int
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&n)

    fmt.Println("Masukkan berat masing-masing anak
kelinci:")
    for i := 0; i < n; i++ {
        fmt.Scan(&beratKelinci[i])
    }

    minBerat := beratKelinci[0]
    maxBerat := beratKelinci[0]

    for i := 1; i < n; i++ {
        if beratKelinci[i] < minBerat {
            minBerat = beratKelinci[i]
        }
        if beratKelinci[i] > maxBerat {
            maxBerat = beratKelinci[i]
        }
    }

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

Screenshots Output



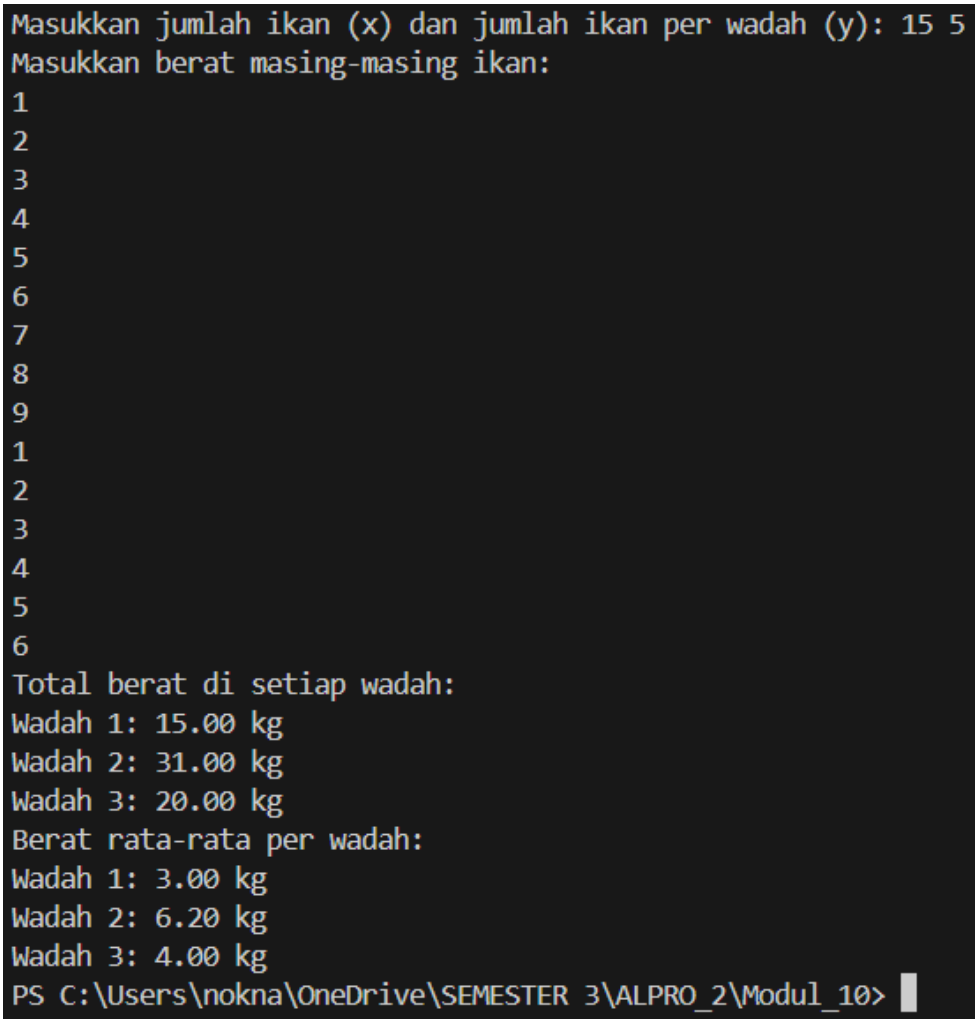
Deskripsi:

Program dalam bahasa Go ini digunakan untuk menentukan berat terkecil dan terbesar dari sekumpulan data berat anak kelinci yang akan dijual di pasar. Program memulai dengan meminta input jumlah anak kelinci (n) dan berat masing-masing kelinci yang disimpan dalam array berkapasitas maksimum 1000. Nilai terkecil dan terbesar diinisialisasi dengan berat kelinci pertama, kemudian dilakukan iterasi melalui array untuk membandingkan setiap berat dengan nilai terkecil dan terbesar yang telah ditemukan sebelumnya. Jika ditemukan berat yang lebih kecil atau lebih besar, nilai tersebut diperbarui. Akhirnya, program menampilkan berat terkecil dan terbesar dengan format dua angka desimal.

Unguided 2

```go
//2311102298
//Nok Nadia
package main

import "fmt"

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

    beratIkan := make([]float64, x)
    fmt.Println("Masukkan berat masing-masing ikan:")
    for i := 0; i < x; i++ {
        fmt.Scan(&beratIkan[i])
    }

    jumlahWadah := (x + y - 1) / y
    totalBerat := make([]float64, jumlahWadah)
    for i := 0; i < x; i++ {
        wadahIdx := i / y
        totalBerat[wadahIdx] += beratIkan[i]
    }

    fmt.Println("Total berat di setiap wadah:")
    for i := 0; i < jumlahWadah; i++ {
        fmt.Printf("Wadah %d: %.2f kg\n", i+1,
totalBerat[i])
    }

    fmt.Println("Berat rata-rata per wadah:")
    for i := 0; i < jumlahWadah; i++ {
        jumlahIkanDiWadah := y
        if i == jumlahWadah-1 && x%y != 0 {
            jumlahIkanDiWadah = x % y
        }
        rataRata := totalBerat[i] /
float64(jumlahIkanDiWadah)
        fmt.Printf("Wadah %d: %.2f kg\n", i+1, rataRata)
    }
}
```

Screenshots Output



Deskripsi:

Program di atas digunakan untuk menghitung total berat dan berat rata-rata ikan yang dimasukkan ke dalam sejumlah wadah berdasarkan jumlah ikan dan kapasitas wadah. Program menerima input jumlah total ikan (x) dan kapasitas wadah (y), lalu meminta berat masing-masing ikan. Berat ikan disimpan dalam array, dan program menghitung jumlah wadah yang diperlukan dengan membagi total ikan dengan kapasitas wadah (dibulatkan ke atas jika tidak habis dibagi). Selanjutnya, program menghitung total berat ikan di setiap wadah dan menampilkan hasilnya. Selain itu, program juga

menghitung berat rata-rata per wadah dengan membagi total berat ikan dalam wadah dengan jumlah ikan yang ada di dalamnya. Hasil akhirnya berupa daftar total berat dan berat rata-rata untuk setiap wadah.

Unguided 3

```go
//2311102298
//Nok Nadia
package main

import (
    "fmt"
)

func hitungMinMax(arrBerat []float64, bMin, bMax *float64) {
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]
    for _, berat := range arrBerat {
        if berat < *bMin {
            *bMin = berat
        }
        if berat > *bMax {
            *bMax = berat
        }
    }
}

func rerata(arrBerat []float64) float64 {
    total := 0.0
    for _, berat := range arrBerat {
        total += berat
    }
    return total / float64(len(arrBerat))
}

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

    arrBerat := make([]float64, n)
    for i := 0; i < n; i++ {
        fmt.Printf("Masukan berat balita ke-%d: ", i+1)
        fmt.Scan(&arrBerat[i])
    }

    var bMin, bMax float64
    hitungMinMax(arrBerat, &bMin, &bMax)
```

```
    rataRata := rerata(arrBerat)

    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",
rataRata)
}
```

Screenshots Output

```
Masukan banyak data berat balita: 5
Masukan berat balita ke-1: 12
Masukan berat balita ke-2: 10.5
Masukan berat balita ke-3: 7
Masukan berat balita ke-4: 11
Masukan berat balita ke-5: 9
Berat balita minimum: 7.00 kg
Berat balita maksimum: 12.00 kg
rata-rata berat balita: 9.90 kg
PS C:\Users\nokna\OneDrive\SEMESTER 3\ALPRO 2\Modul 10>
```

Deskripsi:

Program ini adalah implementasi dalam bahasa Go untuk mengolah data berat balita. Program menerima input berupa jumlah data berat balita dan nilai berat masing-masing balita, lalu menghitung berat minimum, maksimum, dan rata-rata. Fungsi `hitungMinMax` digunakan untuk menentukan nilai minimum dan maksimum dalam array, dengan memanfaatkan pointer untuk mengembalikan hasil. Fungsi `rerata` menghitung rata-rata berat dengan menjumlahkan seluruh elemen dalam array dan membaginya dengan jumlah elemen. Hasil perhitungan ditampilkan dalam format desimal dengan dua angka di belakang koma. Program ini dirancang untuk input dinamis sesuai dengan jumlah data yang dimasukkan pengguna.

IV. DAFTAR PUSTAKA

1) Asisten praktikum, Akmelia Zahara dan Kyla Azzahra Kinan “Modul X PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA” Learning Management System, 2024