LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN2

MODUL 10

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Oleh:

Destia Ananda Putra

2311102176

IF-11-02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

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 ekstrem

Berikut ini adalah notasi dasar pseudocode dan bahasa Go, misalnya untuk pencarian nilai terbesar atau maksimum:

	Notasi Algoritma	Notasi dalam bahasa Go
1	`max ← 1`	`max = 0`
2	`i ← 2`	`i = 1`
3	`while i <= n do`	`for i <`
4	`    if a[i]`	
5	`        max ← i`	`        max = i`
6	`    endif`	`    }`
7	`    i ← i + 1`	`    i = i + 1`
8	`endwhile`	`}`

2. Pencarian Nilai Ekstrim Pada Array Bertipe Data Dasar

Misalnya terdefinisi sebuah array of integer dengan kapasitas 2023, dan array terisi sejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut. Perhatikan potongan program dalam bahasa Go berikut ini!

```
type arrInt [2023]int
...

func terkecil_1(tabInt arrInt, n int) int {
/* mengembalikan nilai terkecil yang terdapat di dalam tabInt yang berisi n
bilangan bulat */
    var min int = tabInt[0]        // min berisi data pertama
    var j int = 1                  // pencarian dimulai dari data berikutnya
    for j < n {
        if min > tabInt[j] {       // pengecekan apakah nilai minimum valid
            min = tabInt[j]        // update nilai minimum dengan yang valid
        }
        j = j + 1
    }
    return min                     // returnkan nilai minimumnya
}
```

Program penggunaan indeks array pada bahasa Go dimulai dari nol atau "0" Posisi atau indeks dari nilai yang dicari dalam kumpulan data atau array. Oleh karena itu, modifikasi pada program di atas dapat dilihat pada potongan program berikut ini!

Pada program ini menunjukkan bahwa untuk mencari indeks nilai terkecil, perulangan dimulai dari indeks pertama array (0) hingga elemen terakhir (n-1). Hal ini memungkinkan program untuk secara akurat menentukan **indeks** yang sesuai dengan nilai terkecil pada array tersebut.

3. Pencarian Nilai Ekstrim pada Array Bertipe Data Terstruktur

Pada kasus yang lebih kompleks pencarian ekstrim dapat juga dilakukan, misalnya mencari data mahasiswa dengan nilai terbesar, mencari lagu dengan durasi terlama, mencari pembalap yang memiliki catatan waktu balap tercepat, dan sebagainya. Sebagai contoh misalnya terdapat array yang digunakan untuk menyimpan data mahasiswa, kemudian terdapat fungsi IPK yang digunakan untuk mencari data mahasiswa dengan IPK tertinggi**.**

```
...
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk float64
}
type arrMhs [2023]mahasiswa
...

func IPK_1(T arrMhs, n int) float64 {
/* mengembalikan ipk terkecil yang dimiliki mahasiswa pada array T yang berisi
n mahasiswa */
    var tertinggi float64 = T[0].ipk
    var j int = 1
    for j < n {
        if tertinggi < T[j].ipk {
            tertinggi = T[j].ipk
        }
        j = j + 1
    }
    return tertinggi
}
```

Apabila diperhatikan potongan program di atas, maka kita akan memperoleh ipk tertinggi, tetapi kita tidak memperoleh identitas mahasiswa dengan ipk tertinggi tersebut. Maka seperti penjelasan yang sudah diberikan sebelumnya, maka pencarian yang dilakukan bisa mengembalikan indeks mahasiswa dengan ipk tertinggi tersebut. Berikut ini adalah modifikasinya!

```
...
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk float64
}
type arrMhs [2023]mahasiswa
...

func IPK_2(T arrMhs, n int) int {
/* mengembalikan indeks mahasiswa yang memiliki ipk tertinggi pada array T yang
berisi n mahasiswa */
    var idx int = 0
    var j int = 1
    for j < n {
        if T[idx].ipk < T[j].ipk {
            idx = j
        }
        j = j + 1
    }
    return idx
}
```

II. GUIDED

Guided 1

```
package main

import "fmt"

func main() {
    var N int
    var berat [1000]float64

    fmt.Print("masukkan jumlah anak kelinci: ")
    fmt.Scan(&N)

    for i := 0; i < N; i++ {
    fmt.Print("masukkan berat anak kelinci: ")
    fmt.Scan(&berat[i])
    }

    min := berat[0]
    max := berat[0]

    for i := 1; i < N; i++ {
    if berat[i] < min {
            min = berat[i]
    }
    if berat[i] > max {
            max = berat[i]
    }
    }
    fmt.Printf("Berat terkecil: %.2f\n", min)
    fmt.Printf("Berat terbesar: %.2f", max)
}
```

Output

```
PS C:\Users\Lenovo> go run "c:\Users\Lenov
masukkan jumlah anak kelinci: 3
masukkan berat anak kelinci: 1
masukkan berat anak kelinci: 2
masukkan berat anak kelinci: 3
Berat terkecil: 1.00
Berat terbesar: 3.00
PS C:\Users\Lenovo>
```

Penjelasan

Program dirancang untuk menghitung berat terkecil dan berat terbesar dari sejumlah anak kelinci berdasarkan data yang dimasukkan oleh pengguna. Dalam output yang diberikan, pengguna memasukkan jumlah anak kelinci sebanyak 3, menunjukkan bahwa program berhasil menjalankan dengan baik, yaitu menentukan nilai minimum dan maksimum dari data berat yang dimasukkan. Program ini bermanfaat untuk memproses dan menganalisis data berat secara sederhana dan cepat.

Guided II

```
package main

import "fmt"

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

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

	jumlahWadah := (x + y - 1) / y
	totalBeratWadah := make([]float64, jumlahWadah)
	jumlahIkanDiWadah := make([]int, jumlahWadah)

	for i := 0; i < x; i++ {
		indeksWadah := i / y
		totalBeratWadah[indeksWadah] += berat[i]
		jumlahIkanDiWadah[indeksWadah]++
	}

	// Output total berat tiap wadah
	fmt.Println("Total berat tiap wadah:")
```

```
        for _, total := range totalBeratWadah {
                fmt.Printf("%.2f ", total)
        }
        fmt.Println()

        // Output rata-rata berat tiap wadah
        fmt.Println("Rata-rata berat tiap wadah:")
        for i := 0; i < jumlahWadah; i++ {
                if jumlahIkanDiWadah[i] > 0 {
                        rataRata := totalBeratWadah[i] / float64(jumlahIkanDiWadah[i])
                        fmt.Printf("%.2f ", rataRata)
                } else {
                        fmt.Printf("0.00 ") // jika wadah kosong
                }
        }
        fmt.Println()
}
```

OUTPUT

```
PS C:\Users\Lenovo> go run "c:\Users\Lenovo\Documents\Fi
Masukkan jumlah ikan dan kapasitas wadah: 6 4
Masukkan berat tiap ikan:
1
2
3
5
6
7
Total berat tiap wadah:
11.00 13.00
Rata-rata berat tiap wadah:
2.75 6.50
PS C:\Users\Lenovo>
```

Penjelasan

Program menghitung jumlah ikan, kapasitas wadah, dan berat masing-masing ikan untuk menghitung total berat dan rata-rata berat ikan per wadah. Berdasarkan input: jumlah ikan 6, kapasitas wadah 4, dan berat ikan 1, 2, 3, 5, 6, 7, ikan dibagi ke dalam 2 wadah. Wadah pertama berisi ikan dengan berat total 11.00 (1, 2, 3, 5) dan rata-rata berat 2.75, sedangkan wadah kedua berisi ikan dengan berat total 13.00 (6, 7) dan rata-

rata berat 6.50. Program ini memastikan distribusi ikan ke dalam wadah sesuai kapasitas, menghitung total berat tiap wadah, dan rata-rata beratnya. Hasilnya menunjukkan pembagian ikan yang terorganisir dan perhitungan yang akurat.

III. Unguided

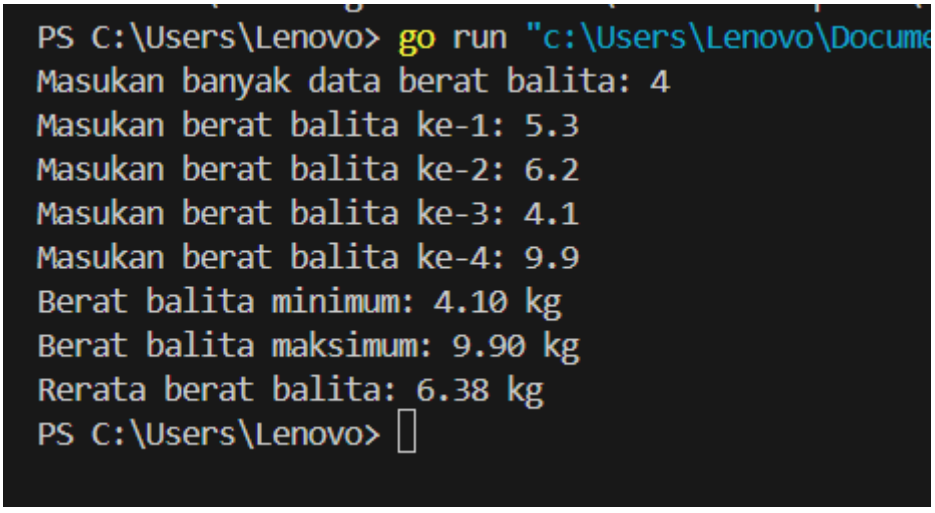
Unguided 1

Source Code

```
package main

// Destia Ananda Putra
// 2311102176

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

func hitungRerata(arrBerat []float64) float64 {
    var total float64
    for _, berat := range arrBerat {
        total += berat
    }
```

```
    return total / float64(len(arrBerat))
}

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

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

    var bMin, bMax float64
    hitungMinMax(arrBerat, &bMin, &bMax)
    rerata := hitungRerata(arrBerat)

    fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)
    fmt.Printf("Rerata berat balita: %.2f kg\n", rerata)
}
```

Output



Penjelasan

Program berat balita yang dimasukkan dianalisis untuk menentukan berat minimum, maksimum, dan rata-rata. Hasilnya, berat minimum balita adalah 4.10 kg, yang menunjukkan balita dengan berat paling ringan, sementara berat maksimum adalah 9.90 kg, yang menunjukkan balita dengan berat paling berat. Rata-rata berat balita dihitung sebesar 6.38 kg, yang memberikan gambaran umum tentang distribusi berat balita dan Program ini membantu mempermudah pengolahan data berat balita, sehingga dapat digunakan untuk mengevaluasi kondisi balita, Kondisi beratnya jauh di bawah atau di atas rata-rata. Dengan rentang berat balita berada antara 4.10 kg hingga 9.90 kg, program ini menjadi alat yang bermanfaat untuk pemantauan kesehatan di posyandu secara efisien dan akurat.