LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 10

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Oleh:

TRI PANJI UTOMO

2311102213

IF – 11- 02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

A. Pencarian Nilai Min/Max

Contoh penggunaannya misalnya pencarian file di dalam directory komputer, pencarian suatu teks di dalam sebuah dokumen, pencarian buku pada rak buku. Pertama pada modul ini akan dipelajari salah satu algoritma pencarian nilai terkecil atau terbesar pada sekumpulan data, atau biasa disebut pencarian nilai ekstrim. 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 di cek, maka nilai ekstrim yang dimiliki adalah valid.

Berikut ini adalah notasi dalam 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 < n {`
4	`    if a[i] > a[max] then`	`    if a[i] > a[max] {`
5	`        max ← i`	`        max = i`
6	`    endif`	`    }`
7	`    i ← i + 1`	`    i = i + 1`
8	`endwhile`	`}`

B. Pencarian Nilai Ekstrim pada Array Bertipe Data Dasar

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

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

Potongan program di atas sedikit berbeda dengan sebelumnya karena penggunaan indeks array pada bahasa Go di mulai dari nol atau "0" seperti penjelasan pada modul 9. Selanjutnya, pada penjelasan di awal bab 3 telah disampaikan bahwa pada pencarian yang terpenting adalah 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!

```
..  ...
5   type arrInt [2023]int
..  ...
15
16  func terkecil_2(tabInt arrInt, n int) int {
17  /* mengembalikan indeks nilai terkecil yang terdapat di dalam tabInt yang berisi
18  n bilangan bulat */
19      var idx int = 0                 // idx berisi indeks data pertama
20      var j int = 1                   // pencarian dimulai dari data berikutnya
21      for j < n {
22          if tabInt[idx] > tabInt[j] { // pengecekan apakah nilai minimum valid
23              idx = j                 // update nilai minimum dengan yang valid
24          }
25          j = j + 1
26      }
27      return idx                      // returnkan indeks nilai minimumnya
28  }
```

C. 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 dan pencarian yang dilakukan bisa mengembalikan indeks mahasiswa dengan ipk tertinggi tersebut.

```
..  ...
5   type mahasiswa struct {
..      nama, nim, kelas, jurusan string
..      ipk float64
..  }
..  type arrMhs [2023]mahasiswa
..  ...
15
16  func IPK_2(T arrMhs, n int) int {
17  /* mengembalikan indeks mahasiswa yang memiliki ipk tertinggi pada array T yang
18  berisi n mahasiswa */
19      var idx int = 0
20      var j int = 1
21      for j < n {
22          if T[idx].ipk < T[j].ipk {
23              idx = j
24          }
25          j = j + 1
26      }
27      return idx
28  }
```

II. GUIDED

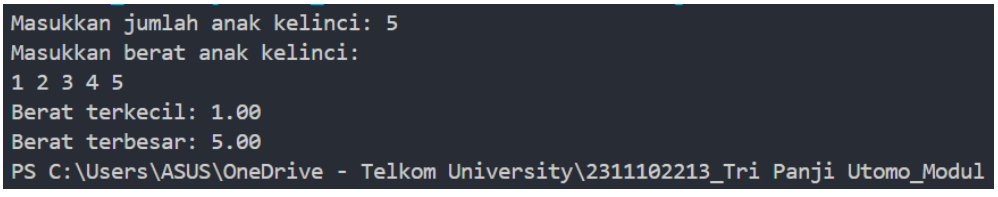
1. Guided 1

Source Code

```
package main

import (
        "fmt"
```

```
)

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

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

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

        // Inisialisasi nilai min dan max dengan elemen
pertama
        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\n", max)
}
```

Screenshot



Deskripsi:

Program ini digunakan untuk menentukan berat anak kelinci terkecil dan terbesar dari sejumlah anak kelinci yang dimasukkan oleh user. Pertama, program meminta user untuk memasukkan jumlah anak kelinci. Kemudian, user diminta untuk memasukkan berat masing-masing anak kelinci. Program

menyimpan berat tersebut dalam array berat. Setelah semua data berat dimasukkan, program melakukan iterasi untuk menemukan berat terkecil dan terbesar dari array tersebut. Terakhir, program mencetak berat terkecil dan terbesar yang ditemukan.

2. **Guided 2**

Source Code

```
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 // pembulatan ke
atas jika x tidak habis dibagi y
        totalBeratWadah := make([]float64, jumlahWadah)

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

        // 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 _, total := range totalBeratWadah {
            rataRata := total / float64(y)
            fmt.Printf("%.2f ", rataRata)
        }
        fmt.Println()
    }
```

Screenshot

```
Masukkan jumlah ikan dan kapasitas wadah: 10 1
Masukkan berat tiap ikan:
10 10 10 5 3 2 1 2 3 4 5
Total berat tiap wadah:
10.00 10.00 10.00 5.00 3.00 2.00 1.00 2.00 3.00 4.00
Rata-rata berat tiap wadah:
10.00 10.00 10.00 5.00 3.00 2.00 1.00 2.00 3.00 4.00
PS C:\Users\ASUS\OneDrive - Telkom University\2311102213_Tri Panji Utomo_Modul
```

Deskripsi:

Program ini digunakan untuk membagi sejumlah ikan ke dalam beberapa wadah dengan kapasitas yang sama, lalu menghitung total berat ikan di setiap wadah dan rata-rata berat ikan per wadah. Pertama, user memasukkan dua nilai integer yaitu jumlah total ikan dan kapasitas maksimal setiap wadah (jumlah ikan maksimum yang bisa masuk dalam satu wadah) serta pengguna diminta untuk memasukkan berat setiap ikan secara berurutan. Program akan menghitung jumlah wadah yang dibutuhkan dan membuat sebuah slice (array dinamis) bernama untuk menyimpan total berat ikan di setiap wadah dengan rumus yang sudah tertera di code, output dari program ini yaitu total berat ikan di setiap wadah serta rata-rata berat ikan di setiap wadah.

III. UNGUIDED

1. Unguided 1

Source Code

```
package main

import "fmt"

type arrBalita [100]float64

func MinMax(arrBerat arrBalita, n int) (float64, float64)
{
    min := arrBerat[0]
    max := arrBerat[0]
```

```
    for i := 1; i < n; i++ {
        if arrBerat[i] < min {
            min = arrBerat[i]
        }
        if arrBerat[i] > max {
            max = arrBerat[i]
        }
    }
    return min, max
}

func rerata(arrBerat arrBalita, n int) float64 {
    var total213 float64 = 0

    for i := 0; i < n; i++ {
        total213 += arrBerat[i]
    }
    return total213 / float64(n)
}

func main() {
    var banyak213 int
    var berat213 arrBalita

    fmt.Print("Masukkan banyak data balita: ")
    fmt.Scanln(&banyak213)

    for i := 0; i < banyak213; i++ {
        fmt.Printf("Masukan berat balita ke-%d: ", i+1)
        fmt.Scanln(&berat213[i])
    }

    min, max := MinMax(berat213, banyak213)
    avg := rerata(berat213, banyak213)

    fmt.Printf("Berat balita minimum: %.2f\n", min)
    fmt.Printf("Berat balita maximum: %.2f\n", max)
    fmt.Printf("Rerata berat balita: %.2f\n", avg)
}
```

Screenshot

```
PS C:\Users\ASUS\OneDrive - Telkom University\2311102213_Tri Panji Utomo_Modul 8> go
11102213_Tri Panji Utomo_Modul 8\Modul 10\Unguided\1\1.go"
Masukkan banyak data balita: 5
Masukan berat balita ke-1: 5
Masukan berat balita ke-2: 4
Masukan berat balita ke-3: 3
Masukan berat balita ke-4: 4
Masukan berat balita ke-5: 5
Berat balita minimum: 3.00
Berat balita maximum: 5.00
Rerata berat balita: 4.20
PS C:\Users\ASUS\OneDrive - Telkom University\2311102213_Tri Panji Utomo_Modul 8>
```

Deskripsi:

Program ini digunakan untuk menentukan minimum, maximum, dan rata-rata berat badan balita yang diinputkan oleh user. User memasukkan banyaknya jumlah balita yang akan di data, user menginputkan berat badan balita, program menjalankan fungsinya sesuai rumus yang telah tertera di code, program mencetak nilai berat minimum, maximum, dan rata-rata.