LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL 10

PENCARIAN NILAI EKSTRIM PADAHIMPUNAN DATA

Oleh:

M. AZKA HERMAWAN

2311102230

IF – 11- 02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

A. 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.

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 Nllal Ekstrlm pada Array Bertlpe 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!

```
5   type arrInt [2023]int
..  ...
15
16  func terkecil_1(tabInt arrInt, n int) int {
17  /* mengembalikan nilai terkecil yang terdapat di dalam tabInt yang berisi n
18  bilangan bulat */
19      var min int = tabInt[0]        // min berisi data pertama
20      var j int = 1                  // pencarian dimulai dari data berikutnya
21      for j < n {
22          if min > tabInt[j] {       // pengecekan apakah nilai minimum valid
23              min = tabInt[j]        // update nilai minimum dengan yang valid
24          }
25          j = j + 1
26      }
27      return min                     // returnkan nilai minimumnya
28  }
```

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 Nllal Ekstrlm pada Array Bertlpe 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.

```
..  ...
5   type mahasiswa struct {
..      nama, nim, kelas, jurusan string
..      ipk float64
..  }
..  type arrMhs [2023]mahasiswa
..  ...
15
16  func IPK_1(T arrMhs, n int) float64 {
17  /* mengembalikan ipk terkecil yang dimiliki mahasiswa pada array T yang berisi
18  n mahasiswa */
19      var tertinggi float64 = T[0].ipk
20      var j int = 1
21      for j < n {
22          if tertinggi < T[j].ipk {
23              tertinggi = T[j].ipk
24          }
25          j = j + 1
26      }
27      return tertinggi
28  }
```

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!

```
..  ...
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  }
```

Sehingga melalui algoritma di atas, identitas mahasiswa dapat diperoleh, misalnya T[idx].nama, T[idx].nim, T[idx].kelas, hingga T[idx].jurusan.

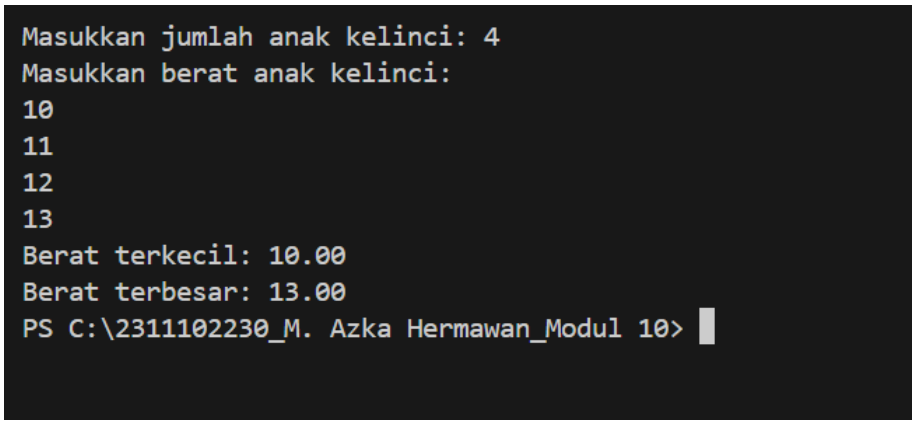
II. GUIDED

1. Guided 1

Source Code

```
package main
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])
    }

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

    for i := 0; 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 merupakan program untuk menentukan berat terkecil dan terbesar dari sekelompok anak kelinci.

Program menggunakan perulangan looping untuk menentukan berat terkecil dan terbesar pada sekelompok anak kelinci.

Program ini mengumpulkan data berat anak kelinci, kemudian menentukan dan menampilkan berat terkecil dan terbesar di antara data yang dimasukkan.

III. UNGUIDED

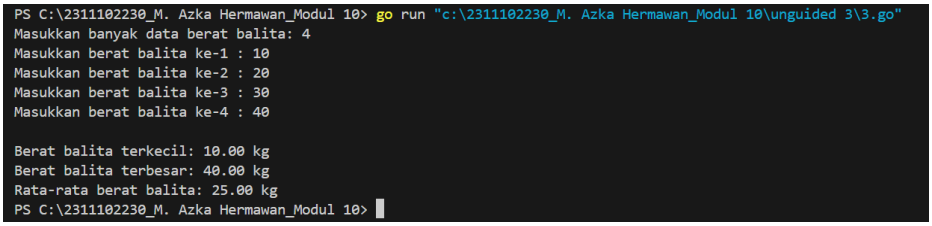
1. Unguided 3

Source Code

```
package main

import "fmt"

type arrBalita [100]float64

func main() {

    var banyakData int
    var arrBerat arrBalita
    var bMax, bMin float64

    fmt.Print("Masukkan banyak data berat balita: ")
    fmt.Scan(&banyakData)

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

    hitungMinMax(arrBerat, banyakData, &bMin, &bMax)
    rata := rerata(arrBerat, banyakData)

    fmt.Printf("\nBerat balita terkecil: %.2f kg\n", bMin)
    fmt.Printf("Berat balita terbesar: %.2f kg\n", bMax)
    fmt.Printf("Rata-rata berat balita: %.2f kg\n", rata)

}

func hitungMinMax(arrBerat arrBalita, banyakData int,
bMin, bMax *float64) {
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]

    for i := 1; i < banyakData; i++ {
        if arrBerat[i] < *bMin {
            *bMin = arrBerat[i]
```

```
        }
        if arrBerat[i] > *bMax {
            *bMax = arrBerat[i]
        }

    }
}

func rerata(arrBerat arrBalita, banyakData int) float64 {
    var totalBerat float64

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

}
```

Screenshot



Deskripsi

Program ini merupakan program untuk menentukan berat minimum, berat maksimal dan rata-rata berat pada beberapa balita yang telah pengguna inputkan.

Fungsi hitungMinMax berfungsi untuk menentukan berat balita terkecil dan terbesar. Pada fungsi ini melakukan pengecekan terhadap setiap berat balita dalam array, jika berat balita lebih kecil dari bMin, maka bMin diperbarui. Jika berat balita lebih besar dari bMax, maka bMax diperbarui.

Fungsi rerata berfungsi untuk menghitung rata-rata berat balita.

Jadi pada program ini dibuat untuk memudahkan dalam analisis sederhana terhadap data berat balita, seperti mencari nilai minimum, maksimum, dan rata-rata.