

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**

MODUL X

“PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA”



Oleh:

MUHAMMAD RAGIEL PRASTYO

2311102183

S1IF-11-02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

10.1 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.

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

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

10.2 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!

```
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  }
```

10.3 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.

```
..  ...
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. *Source Code*

```
// MUHAMMAD RAGIEL PRASTYO
// 2311102183
package main
import (
    "fmt"
)

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

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

    fmt.Println("Masukan berat anak kelinci: ")
    for i := 0; i < N; i++ {
        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\n", max)
}
```

Screenshot Output

```
PS C:\Users\USER\OneDrive\Desktop\Alpro 2> go run "c:\Users\USER\OneDrive\Desktop\Alpro 2\Muhammad Ragiel Prastyo_2311102183_Modul10\Guided\guided1.go"
Masukan jumlah anak kelinci: 6
Masukan berat anak kelinci:
2 4 6 8 10 12
Berat terkecil: 2.00
Berat terbesar: 12.00
PS C:\Users\USER\OneDrive\Desktop\Alpro 2> █
```

Penjelasan:

Program di atas menghitung berat kelinci terkecil dan terbesar. Pengguna diminta untuk memasukkan jumlah anak kelinci (N) dan berat masing-masing anak kelinci yang disimpan dalam array berat. Program kemudian menginisialisasi nilai terkecil (min) dan terbesar (max) dengan berat anak kelinci pertama. Kemudian, menggunakan perulangan untuk membandingkan setiap berat dengan nilai min dan max, memperbarui nilai jika ditemukan berat yang lebih kecil atau lebih besar. Setelah itu, program mencetak nilai terkecil dan terbesar dalam format dua desimal.

2. Source Code

```
// MUHAMMAD RAGIEL PRASTYO
// 2311102183
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)

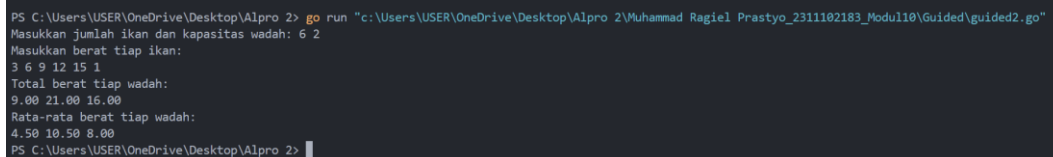
    for i := 0; i < x; i++ {
        indeksWadah := i / y
        totalBeratWadah[indeksWadah] += berat[i]
    }

    //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 Output



```

PS C:\Users\USER\OneDrive\Desktop\Alpro 2> go run "c:\Users\USER\OneDrive\Desktop\Alpro 2\Muhammad Ragiel Prastyo_2311102183_Modul10\Guided\guided2.go"
Masukkan jumlah ikan dan kapasitas wadah: 6 2
Masukkan berat tiap ikan:
3 6 9 12 15 1
Total berat tiap wadah:
9.00 21.00 16.00
Rata-rata berat tiap wadah:
4.50 10.50 8.00
PS C:\Users\USER\OneDrive\Desktop\Alpro 2>

```

Penjelasan:

Program di atas mengelompokkan ikan ke dalam wadah dengan kapasitas yang berbeda dan menghitung total dan rata-rata berat tiap wadah. Ini dilakukan dengan memasukkan jumlah ikan (x), kapasitas wadah (y), dan berat setiap ikan. Setelah menghitung jumlah wadah yang dibutuhkan, program

mendistribusikan berat ikan ke masing-masing wadah, dan kemudian menghitung total berat dan rata-rata berat untuk setiap wadah.

III. UNGUIDED

1. Pos Pelayanan Terpadu (posyandu) sebagai tempat pelayanan kesehatan perlu mencatat data berat balita (dalam kg). Petugas akan memasukkan data tersebut ke dalam array. Dari data yang diperoleh akan dicari berat balita terkecil, terbesar, dan reratanya.

Source Code

```
// MUHAMMAD RAGIEL PRASTYO
// 2311102183
package main
import (
    "fmt"
)

type arrBalita [100]float64

// Subprogram untuk menghitung berat minimum dan
maksimum
func hitungMinMax(arrBerat arrBalita, n int, bMin, bMax
*float64) {
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]
    for i := 1; i < n; i++ {
        if arrBerat[i] < *bMin {
            *bMin = arrBerat[i]
        }
        if arrBerat[i] > *bMax {
            *bMax = arrBerat[i]
        }
    }
}

// Subprogram untuk menghitung rata-rata berat balita
func rataRata(arrBerat arrBalita, n int) float64 {
    var total float64
    for i := 0; i < n; i++ {
        total += arrBerat[i]
    }
}
```

```

    }
    return total / float64(n)
}

func main() {
    var n int
    var arrBerat arrBalita
    var bMin, bMax float64

    // Input jumlah balita
    fmt.Print("Masukkan banyak data berat balita: ")
    fmt.Scan(&n)

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

    // Hitung berat minimum, maksimum, dan rata-rata
    hitungMinMax(arrBerat, n, &bMin, &bMax)
    rata := rataRata(arrBerat, n)

    // Output hasil
    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", rata)
}

```

Screenshot Output

```

PS C:\Users\USER\OneDrive\Desktop\Alpro 2> go run "c:\Users\USER\OneDrive\Desktop\Alpro 2\Muhammad Ragiel Prastyo_2311102183_Modul10\Unguided\unguided1.go"
Masukkan banyak data berat balita: 9
Masukkan berat balita ke-1: 2.3
Masukkan berat balita ke-2: 2.0
Masukkan berat balita ke-3: 5.0
Masukkan berat balita ke-4: 3.2
Masukkan berat balita ke-5: 3.9
Masukkan berat balita ke-6: 5.2
Masukkan berat balita ke-7: 5.5
Masukkan berat balita ke-8: 9.2
Masukkan berat balita ke-9: 3.3
Berat balita minimum: 2.00 kg
Berat balita maksimum: 9.20 kg
Rata-rata berat balita: 4.40 kg
PS C:\Users\USER\OneDrive\Desktop\Alpro 2>

```

Penjelasan:

Program di atas menghitung berat balita minimum, maksimum, dan rata-rata pengguna. Pengguna memberikan jumlah balita dan berat yang masing-masing disimpan dalam array `arrBerat`. Subprogram `hitungMinMax` menghitung berat terkecil dan terbesar, dan subprogram `rataRata` menghitung berat rata-rata. Di belakang koma, angka dua menunjukkan hasil perhitungan: minimum, maksimum, dan rata-rata. Program ini membantu menganalisis data berat balita dengan efektif.