

LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL 10
PENCARIAN NILAI EXSTRIM PADA HIMPUNAN DATA



Oleh:

YASVIN

SYAHGANA

2311102065

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.

Contohnya dalam dunia nyata sangat beragam. Salah satu contohnya adalah mencari file di dalam direktori komputer, mencari teks dalam dokumen, mencari buku di rak buku, dan sebagainya. Salah satu algoritma pencarian nilai terkecil atau terbesar pada sekumpulan data—juga dikenal sebagai pencarian nilai ekstrim—akan dibahas pertama kali dalam modul ini.

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, notics lob 3

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	}

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 }
```

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

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)
}
```

Screenshot output

```
PS E:\LAPORAN PRAKTIKUM\YASVIN SYAHGANA_2311102065_Laprak 10> go run "e:\LAPORAN PRAKTIKUM\YASVIN SYAHGANA_2311102065_Laprak 10\guided1.go"
masukkan jumlah anak kelinci: 2
masukkan berat anak kelinci: 5
masukkan berat anak kelinci: 4
Berat terkecil: 4.00
Berat terbesar: 5.00
PS E:\LAPORAN PRAKTIKUM\YASVIN SYAHGANA_2311102065_Laprak 10> █
```

Deskripsi program

Program di atas menentukan berat terkecil dan terbesar dari anak-anak kelinci. Setelah pengguna memasukkan jumlah dan berat tiap anak kelinci, program membandingkan setiap berat untuk menemukan nilai terkecil dan terbesar, lalu menampilkannya dalam dua desimal.

Guided 2

```
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)

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

    fmt.Println("total berat tiap wadah: ")
    for _, total := range totalBeratWadah {
```

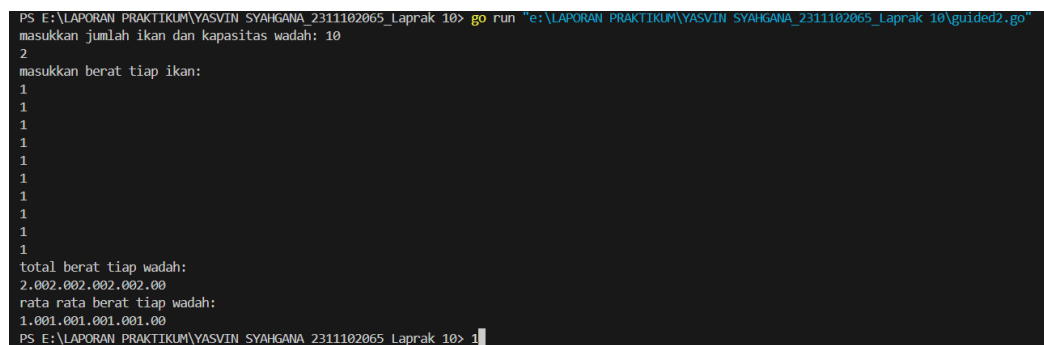
```

        fmt.Printf("%.2f", total)
    }
    fmt.Println()

    fmt.Println("rata rata berat tiap wadah: ")
    for _, total := range totalBeratWadah {
        ratarata := total / float64(y)
        fmt.Printf("%.2f", ratarata)
    }
    fmt.Println()
}

```

Screenshot output



```

PS E:\LAPORAN PRAKTIKUM\YASVIN SYAHGANA 2311102065_Laprak 10> go run "e:\LAPORAN PRAKTIKUM\YASVIN SYAHGANA 2311102065_Laprak 10\guided2.go"
masukkan jumlah ikan dan kapasitas wadah: 10
2
masukkan berat tiap ikan:
1
1
1
1
1
1
1
1
1
1
1
1
total berat tiap wadah:
2.002.002.002.002.00
rata rata berat tiap wadah:
1.001.001.001.001.00
PS E:\LAPORAN PRAKTIKUM\YASVIN SYAHGANA 2311102065_Laprak 10>

```

Deskripsi program

Program ini menghitung berat ikan total dan rata-rata di setiap wadah berdasarkan jumlah ikan dan kapasitas wadah yang dimasukkan pengguna. Setelah berat ikan didistribusikan ke masing-masing wadah, program menampilkan berat total dan rata-rata per wadah dengan dua angka desimal.

III. UNGUIDED

Soal Modul 10

Source code no 3

```
package main

import "fmt"

type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita, bMin *float64, bMax *float64,
n int) {
    for i := 1; i < n; i++ {
        if arrBerat[i] < *bMin {
            fmt.Print(arrBerat[i])
            *bMin = arrBerat[i]
        }
        if arrBerat[i] > *bMax {
            *bMax = arrBerat[i]
        }
    }
}

func rerata(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
    fmt.Print("Masukkan banyak data berat balita : ")
    fmt.Scan(&n)

    if n > 100 {
        fmt.Print("Ukuran terlalu besar. Harus kurang dari atau sama
dengan 100.")
    }
}
```



```

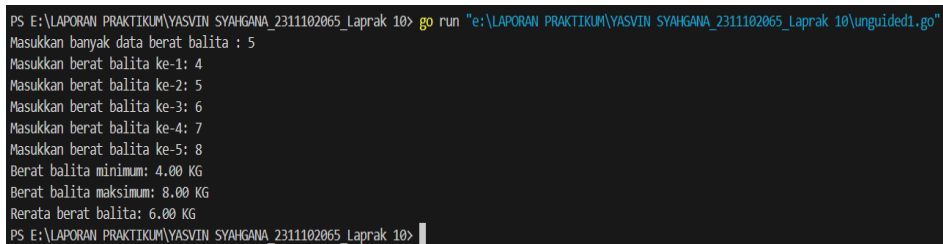
for i := 1; i <= n; i++ {
    fmt.Printf("Masukkan berat balita ke-%v: ", i)
    fmt.Scan(&arrBerat[i-1])
}

var min, max float64 = arrBerat[0], arrBerat[0]
hitungMinMax(arrBerat, &min, &max, n)
fmt.Printf("Berat balita minimum: %.2f KG\n", min)
fmt.Printf("Berat balita maksimum: %.2f KG\n", max)
fmt.Printf("Rerata berat balita: %.2f KG\n", rerata(arrBerat, n))
}

```

Screenshot output

Deskripsi



```

PS E:\LAPORAN PRAKTIKUM\YASVIN SYAHGANA_2311102065_Laprak 10> go run "e:\LAPORAN PRAKTIKUM\YASVIN SYAHGANA_2311102065_Laprak 10\unguided1.go"
Masukkan banyak data berat balita : 5
Masukkan berat balita ke-1: 4
Masukkan berat balita ke-2: 5
Masukkan berat balita ke-3: 6
Masukkan berat balita ke-4: 7
Masukkan berat balita ke-5: 8
Berat balita minimum: 4.00 KG
Berat balita maksimum: 8.00 KG
Rerata berat balita: 6.00 KG
PS E:\LAPORAN PRAKTIKUM\YASVIN SYAHGANA_2311102065_Laprak 10>

```

Program ini menghitung berat balita minimum, maksimum, dan rata-rata dari sekumpulan data. Pengguna memasukkan jumlah data dan berat balita, lalu program menggunakan fungsi minMax untuk menghitung berat terkecil dan terbesar, dan fungsi rataRata untuk menghitung rata-rata berat balita. Hasilnya ditampilkan dalam format dua angka desimal.