

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 10
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA**



**Universitas
Telkom**

Oleh:

HENDWI SAPUTRA

2311102218

IF-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 Ekstrem pada Array Ber tipe 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 Ekstrem pada Array Bertipe Data Terstruktur

Pada kasus yang lebih kompleks pencarian ekstrem 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 terbesar 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 I

Source Code

```
package main

import "fmt"

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

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

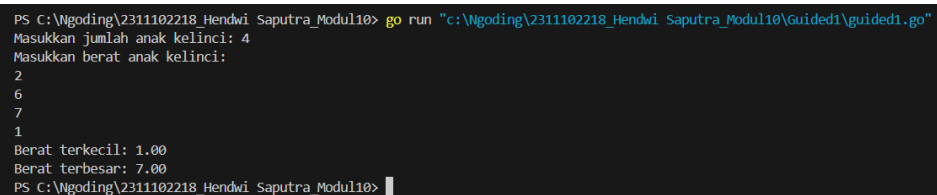
    fmt.Println("Masukkan berat anak kelinci:")
    for i := 0; i < N; i++ {
        fmt.Scan(&berat[i])
    }

    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



```
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul10> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul10\Guided1\guided1.go"
Masukkan jumlah anak kelinci: 4
Masukkan berat anak kelinci:
2
6
7
1
Berat terkecil: 1.00
Berat terbesar: 7.00
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul10> █
```

Deskripsi:

Program ini menentukan berat anak kelinci terkecil dan terbesar dari sejumlah anak kelinci yang dimasukkan oleh pengguna. Pertama, program meminta pengguna untuk memasukkan jumlah anak kelinci. Kemudian, pengguna 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.

GUIDED II

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]
    }

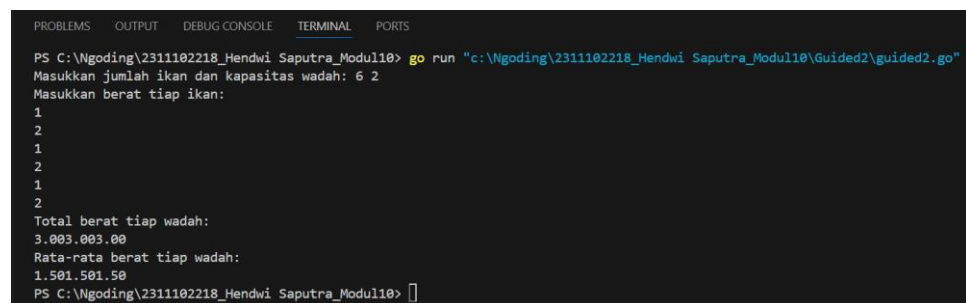
    // Output total berat tiap wadah
    fmt.Println("Total berat tiap wadah:")
    for _, total := range totalBeratWadah {
        fmt.Printf("%.2f", total)
    }
    fmt.Println()
}
```

```

// 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



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul10> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul10\Guided2\guided2.go"
Masukkan jumlah ikan dan kapasitas wadah: 6 2
Masukkan berat tiap ikan:
1
2
1
2
1
2
Total berat tiap wadah:
3.003.003.00
Rata-rata berat tiap wadah:
1.501.501.50
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul10> 

```

Deskripsi:

Program ini menggunakan searching untuk menghitung total berat dan rata-rata berat ikan di tiap wadah berdasarkan jumlah ikan dan kapasitas wadah yang dimasukkan oleh pengguna. Pengguna diminta memasukkan jumlah ikan dan kapasitas wadah, kemudian berat masing-masing ikan. Program menghitung jumlah wadah yang dibutuhkan dan total berat ikan di tiap wadah. Setelah itu, program mencetak total berat dan rata-rata berat ikan untuk setiap wadah. Rata-rata berat dihitung dengan membagi total berat ikan dalam wadah dengan kapasitas wadah.

III. UNGUIDED

UNGUIDED I

Source Code

```
package main

import "fmt"

type arrBalita [100]float64

func hitungMinMax(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 total float64 = 0

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

func main() {
    var banyakBalita int
    var beratBalita arrBalita

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

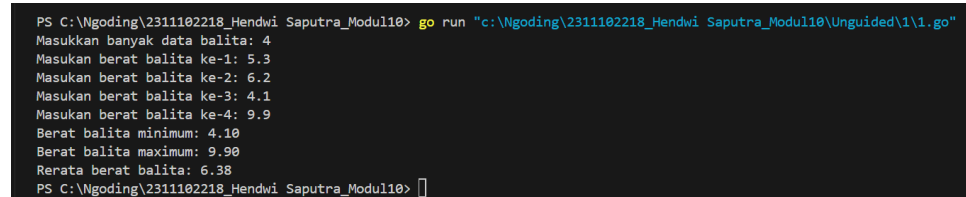
    for i := 0; i < banyakBalita; i++ {
        fmt.Printf("Masukan berat balita ke-%d: ", i+1)
        fmt.Scanln(&beratBalita[i])
    }

    min, max := hitungMinMax(beratBalita, banyakBalita)
    avg := rerata(beratBalita, banyakBalita)

    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:\Ngoding\2311102218_Hendwi Saputra_Modul10> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul10\Unguided\1\1.go"  
Masukkan banyak data balita: 4  
Masukan berat balita ke-1: 5.3  
Masukan berat balita ke-2: 6.2  
Masukan berat balita ke-3: 4.1  
Masukan berat balita ke-4: 9.9  
Berat balita minimum: 4.10  
Berat balita maximum: 9.90  
Rerata berat balita: 6.38  
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul10> 
```

Deskripsi:

Program ini merupakan program searching untuk menghitung berat minimum, berat maksimum, dan rata-rata berat balita dari sejumlah data yang dimasukkan oleh pengguna. Pertama, program meminta pengguna untuk memasukkan jumlah data balita dan kemudian memasukkan berat masing-masing balita. Setelah data dimasukkan, program menggunakan func `hitungMinMax` untuk menemukan berat minimum dan maksimum dari array berat balita. Selanjutnya, func `rerata` menghitung rata-rata berat balita dengan menjumlahkan semua nilai berat dan membaginya dengan jumlah data balita. Akhirnya, program mencetak hasil perhitungan berat minimum, maksimum, dan rata-rata.