

**LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2**

**MODUL X
PENCARIAN NILAI EKSTRIM**



Oleh:

Mansyuroh

NIM:

2311102234

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2024

I. DASAR TEORI

A. Ide Pencarian Nilai Max/Min

Pencarian adalah suatu proses yang umum 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.

Ide algoritma sederhana, karena data harus diproses secara sekuensial, maka nilai atau indeks ke nilai maksimum dari data yang telah diproses disimpan sampai algoritma tersebut berakhir adalah nilai maksimum yang dicari. Algoritma secara umum :

- 1) Jadikan nilai pertama sebagai nilai ekstrim
- 2) Lakukan validasi nilai ekstrim dari data kedua hingga data terakhir.

Apabila nilai ekstrim tidak valid, maka update nilai ekstrim 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 | } |

B. 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” . Maka program di atas dapat dimodifikasi menjadi seperti 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 Ekstrem pada Array Bertipe Data Terstruktur

Pada kasus yang lebih kompleks pencarian nilai ekstrem dapat juga dilakukan, misalnya mencari dapat mahasiswa dengan nilai terbesar, mencari lagu dengan durasi terlama, mencari pembalap yang memiliki catatan waktu balap tercepat, dan sebagainya. Sebagai contoh misal 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 diatas, maka akan diperoleh nilai ipk tertinggi, tetapi kita tidak akan memperoleh identitas mahasiswa dengan ipk tertinggi tersebut. Maka program dapat dimodifikasi menjadi :

```
.. ..  
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

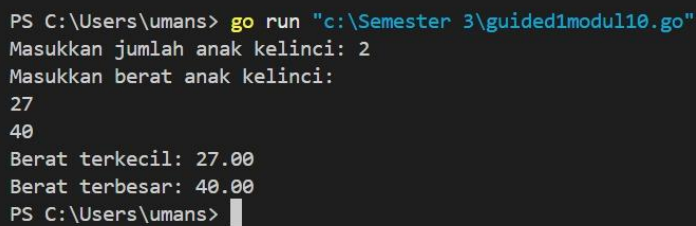
1. Source code

```
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 := 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 program



```
PS C:\Users\umans> go run "c:\Semester 3\guided1modul10.go"
Masukkan jumlah anak kelinci: 2
Masukkan berat anak kelinci:
27
40
Berat terkecil: 27.00
Berat terbesar: 40.00
PS C:\Users\umans> █
```

Deskripsi program

Program ini berfungsi untuk menghitung berat terkecil dan terbesar dari sejumlah anak kelinci berdasarkan input pengguna. Berikut penjelasannya:

1. **Input:** Program meminta jumlah anak kelinci (N) dan berat masing-masing anak kelinci (dalam array berat).
2. **Proses:** Program menentukan berat terkecil (min) dan terbesar (max) menggunakan perulangan untuk membandingkan setiap berat yang diinput.
3. **Output:** Program mencetak berat terkecil dan terbesar dengan format dua angka desimal.

2. Source code

```
package main
import (
    "fmt"
)

func main() {
    var x, y int
    fmt.Printf("masukkan jumlah ikan dan kapasitas wadah: ")
    fmt.Scan(&x, &y)

    berat := make([]float64, x)
    fmt.Printf("masukkan berat tiap ikan: ")
    for i := 0; i < x; i++ {
        fmt.Scan(&berat[i])
    }

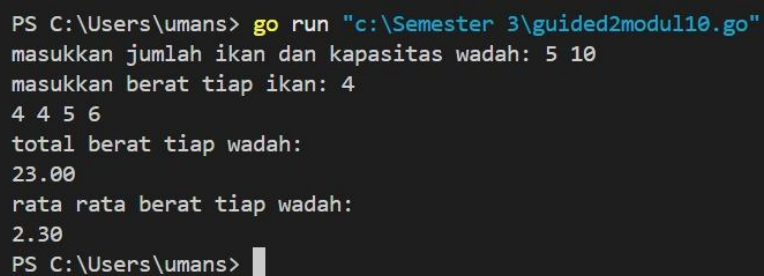
    jumlahWadah := (x + y - 1) / y
    totalBeratWadah := make([]float64, jumlahWadah)

    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 program



```
PS C:\Users\umans> go run "c:\Semester 3\guided2modul10.go"
masukkan jumlah ikan dan kapasitas wadah: 5 10
masukkan berat tiap ikan: 4
4 4 5 6
total berat tiap wadah:
23.00
rata rata berat tiap wadah:
2.30
PS C:\Users\umans> █
```

Program ini berfungsi untuk menghitung total dan rata-rata berat ikan per wadah berdasarkan jumlah ikan dan kapasitas wadah. Berikut penjelasannya:

1. **Input:** Pengguna memasukkan jumlah ikan (x), kapasitas wadah (y), dan berat masing-masing ikan.
2. **Proses:** Total wadah dihitung dengan rumus $\text{jumlah wadah} = (\text{jumlah ikan} + \text{kapasitas} - 1) / \text{kapasitas}$. Berat ikan didistribusikan ke setiap wadah, lalu total berat setiap wadah dihitung. Rata-rata berat ikan per wadah dihitung dengan membagi total berat wadah dengan kapasitas.
3. **Output:** Program mencetak total berat dan rata-rata berat setiap wadah dalam format dua angka desimal.

III. UNGUIDED

1. Source code

```
package main

import (
    "fmt"
)

func main() {
    var jumlahBalita234 int
    var beratBalita234 []float64
    // Meminta input jumlah balita
    fmt.Print("Jumlah balita: ")
    fmt.Scanln(&jumlahBalita234)
    beratBalita234 = make([]float64, jumlahBalita234)
    // Meminta input berat masing-masing balita
    for i := 0; i < jumlahBalita234; i++ {
        fmt.Printf("Berat balita ke-%d: ", i+1)
        fmt.Scanln(&beratBalita234[i])
    }

    beratTerkecil234 := beratBalita234[0]
```

```

beratTerbesar234 := beratBalita234[0]
totalBerat234 := 0.0
indeksTerkecil := 0
indeksTerbesar := 0
for i, berat := range beratBalita234 {
    if berat < beratTerkecil234 {
        beratTerkecil234 = berat
        indeksTerkecil = i
    }
    if berat > beratTerbesar234 {
        beratTerbesar234 = berat
        indeksTerbesar = i
    }
    totalBerat234 += berat
}

rataRata234 := totalBerat234 / float64(jumlahBalita234)

// Menampilkan hasil
fmt.Printf("\nBerat balita minimum: %.2f kg (balita ke-%d)\n",
beratTerkecil234, indeksTerkecil+1)

fmt.Printf("Berat balita maksimum: %.2f kg (balita ke-%d)\n",
beratTerbesar234, indeksTerbesar+1)

fmt.Printf("Rata-rata berat balita: %.2f kg\n", rataRata234)
}

```

Screenshot program

```

PS C:\Users\umans> go run "c:\Semester 3\unguidedmodul10.go"
Jumlah balita: 4
Berat balita ke-1: 10
Berat balita ke-2: 15
Berat balita ke-3: 10
Berat balita ke-4: 16

Berat balita minimum: 10.00 kg (balita ke-1)
Berat balita maksimum: 16.00 kg (balita ke-4)
Rata-rata berat balita: 12.75 kg
PS C:\Users\umans>

```


Deskripsi program

Program ini berfungsi untuk menghitung berat minimum, maksimum, dan rata-rata dari sejumlah balita berdasarkan input pengguna. Berikut penjelasannya:

1. Input: Pengguna memasukkan jumlah balita dan berat masing-masing balita.

2. Proses:

- Program mencari berat minimum beserta indeksinya.
- Program mencari berat maksimum beserta indeksinya.
- Total berat dihitung untuk memperoleh rata-rata berat balita.

3. Output:

Berat minimum dan balita beberapa yang memilikinya.

Berat maksimum dan balita beberapa yang memilikinya.

Rata-rata berat semua balita dalam format dua angka desimal.