

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XI  
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA**



**Disusun Oleh :**

**Wafiq Nur Azizah / 2311102270**

**S1IF-11-05**

**Dosen Pengampu :**

**Arif Amrulloh, S.Kom., M.Kom.**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

Pencarian nilai ekstrem (minimum dan maksimum) pada himpunan data merupakan salah satu operasi dasar dalam pemrosesan data. Operasi ini penting dalam berbagai aplikasi, seperti analisis statistik, pengambilan keputusan, dan pengembangan algoritma. Dalam bahasa pemrograman Golang (Go), proses pencarian nilai ekstrem dapat dilakukan dengan efisien menggunakan struktur data sederhana dan algoritma iteratif.

Pencarian nilai ekstrem biasanya dilakukan dengan memindai setiap elemen dalam himpunan data secara berurutan (iterasi). Pada setiap langkah, elemen tersebut dibandingkan dengan nilai minimum atau maksimum yang telah ditemukan sejauh ini. Jika ditemukan elemen yang lebih kecil atau lebih besar, nilai tersebut diperbarui. Pendekatan ini memiliki kompleksitas waktu sebesar  $O(n)$ , di mana  $n$  adalah jumlah elemen dalam himpunan data.

### a. Struktur Data dalam Golang

Dalam Golang, himpunan data biasanya diwakili dengan tipe data array atau slice. Slice lebih sering digunakan karena fleksibilitasnya dalam menambah atau menghapus elemen. Untuk melakukan pencarian nilai ekstrem, slice dapat diiterasi menggunakan perulangan sederhana seperti `for`.

### b. Implementasi Dasar Pencarian Nilai Ekstrem di Golang

Kode berikut menunjukkan implementasi pencarian nilai minimum dan maksimum pada slice di Golang:

```
package main

import "fmt"

func findExtremes(data []int) (int, int) {
    if len(data) == 0 {
        panic("Himpunan data tidak boleh kosong")
    }
    min, max := data[0], data[0]
    for _, value := range data {
        if value < min {
            min = value
        }
        if value > max {
            max = value
        }
    }
}
```

```

    }
    return min, max
}

func main() {
    data := []int{10, 25, -3, 56, 7, 19}
    min, max := findExtremes(data)
    fmt.Printf("Nilai minimum: %d, Nilai maksimum: %d\n", min,
max)
}

```

Dalam implementasi di atas, kita menangani kasus di mana slice kosong dengan menggunakan fungsi `panic` untuk menghentikan eksekusi program. Penanganan kasus tepi lain, seperti slice yang hanya memiliki satu elemen, diatasi secara alami karena nilai minimum dan maksimum akan sama dengan elemen tersebut.

c. Peningkatan Efisiensi untuk Data Besar

Jika himpunan data sangat besar, penting untuk memastikan bahwa algoritma tetap efisien. Misalnya, jika data tersimpan di file atau aliran data yang tidak dapat dimuat ke memori sekaligus, kita dapat menggunakan pendekatan iteratif berbasis streaming untuk membaca dan memproses data satu per satu.

Pencarian nilai ekstrem tidak hanya berlaku untuk tipe data numerik, tetapi juga untuk tipe data lain seperti string atau objek kompleks, asalkan terdapat kriteria perbandingan. Dalam Golang, hal ini dapat dilakukan dengan menggunakan fungsi pembandingan atau metode tertentu. Pencarian nilai ekstrem adalah operasi penting yang mudah diimplementasikan di Golang. Praktik terbaik mencakup penanganan kasus slice, pengujian ekstensif, dan penggunaan algoritma yang efisien untuk memastikan performa yang optimal pada berbagai ukuran data. Memahami dasar ini memberikan fondasi yang kuat untuk mengembangkan aplikasi yang lebih kompleks.

## II. UNGUIDED

### 1. Soal Studi Case

Sebuah program digunakan untuk mendata berat anak kelinci yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat anak kelinci yang akan dijual.

**Masukan** terdiri dari sekumpulan bilangan, yang mana bilangan pertama adalah bilangan bulat N yang menyatakan banyaknya anak kelinci yang akan ditimbang beratnya. Selanjutnya, N bilangan riil berikutnya adalah berat dari anak kelinci yang akan dijual.

**Keluaran** terdiri dari dua buah bilangan riil yang menyatakan berat kelinci terkecil dan terbesar.

#### Sourcecode

```
package main

import "fmt"

type kelinci [1000]float64

func isiArray(a *kelinci, n int) {
    for i := 0; i < n; i++ {
        fmt.Printf("Berat kelinci ke-%d: ", i+1)
        fmt.Scan(&a[i])
    }
}

func cariTerkecil(a kelinci, n int) float64 {
    terkecil := a[0]
    for i := 1; i < n; i++ {
        if terkecil > a[i] {
            terkecil = a[i]
        }
    }
    return terkecil
}

func cariTerbesar(a kelinci, n int) float64 {
    terbesar := a[0]
    for i := 1; i < n; i++ {
        if terbesar < a[i] {
            terbesar = a[i]
        }
    }
    return terbesar
}
```

```

func main() {
    var berat kelinci
    var n int

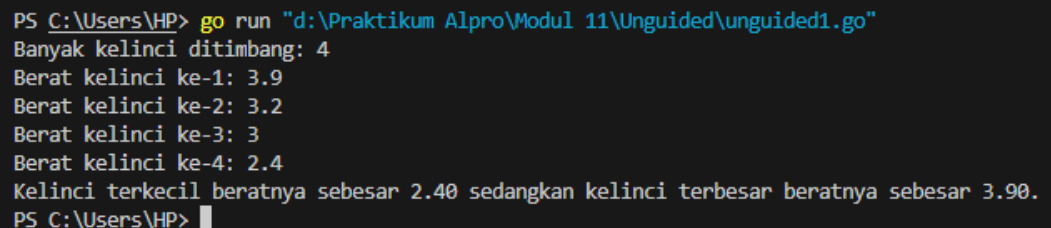
    fmt.Print("Banyak kelinci ditimbang: ")
    fmt.Scan(&n)

    isiArray(&berat, n)

    fmt.Printf("Kelinci terkecil beratnya sebesar %.2f
sedangkan kelinci terbesar beratnya sebesar %.2f.\n",
        cariTerkecil(berat, n), cariTerbesar(berat, n))
}

```

## Screenshoot Output



```

PS C:\Users\HP> go run "d:\Praktikum Alpro\Modul 11\Unguided\unguided1.go"
Banyak kelinci ditimbang: 4
Berat kelinci ke-1: 3.9
Berat kelinci ke-2: 3.2
Berat kelinci ke-3: 3
Berat kelinci ke-4: 2.4
Kelinci terkecil beratnya sebesar 2.40 sedangkan kelinci terbesar beratnya sebesar 3.90.
PS C:\Users\HP>

```

## Deskripsi Program

Program Go ini bertujuan untuk menginput, memproses, dan menampilkan data berat badan kelinci dalam sebuah kelompok. Program dimulai dengan meminta pengguna menentukan jumlah kelinci yang akan ditimbang, kemudian memasukkan berat badan masing-masing kelinci. Setelah semua data dimasukkan, program akan mencari dan menampilkan berat badan kelinci yang paling ringan dan paling berat.

Beberapa fungsi utama dalam program ini adalah `isiArray`, yang digunakan untuk mengisi array dengan data berat badan kelinci berdasarkan input pengguna. Fungsi `cariTerkecil` bertugas mencari berat badan terkecil dengan membandingkan setiap elemen dalam array sementara `cariTerbesar` digunakan untuk mencari berat badan terbesar dengan cara yang sama. Jadi, program dapat memberikan informasi tentang variasi berat badan kelinci dari kelompok tersebut.

## 2. Soal Studi Case

Sebuah program digunakan untuk menentukan tarif ikan yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat ikan yang akan dijual.

**Masukan** terdiri dari dua baris, yang mana baris pertama terdiri dari dua bilangan bulat x dan y. Bilangan x menyatakan banyaknya ikan yang akan dijual, sedangkan y adalah banyaknya ikan yang akan dimasukkan ke dalam wadah. Baris kedua terdiri dari sejumlah x bilangan riil yang menyatakan banyaknya ikan yang akan dijual.

**Keluaran** terdiri dari dua baris. Baris pertama adalah kumpulan bilangan riil yang menyatakan total berat ikan di setiap wadah (jumlah wadah tergantung pada nilai x dan y, urutan ikan yang dimasukkan ke dalam wadah sesuai urutan pada masukan baris ke-2). Baris kedua adalah sebuah bilangan riil yang menyatakan berat rata-rata ikan di setiap wadah.

### Sourcecode

```
package main

import "fmt"

type ikan [1000]float64

func dataikan(b *ikan, x int) {
    for i := 0; i < x; i++ {
        fmt.Printf("Masukkan berat ikan ke-%d: ", i+1)
        fmt.Scan(&b[i])
    }
}

func totalBeratDanRataRata(b *ikan, x int, y int)
([]float64, []float64) {
    var totalBerat []float64
    var rataRata []float64

    totalWadah := (x + y - 1) / y

    for wadah := 0; wadah < totalWadah; wadah++ {
        start := wadah * y
        end := start + y
        if end > x {
            end = x
        }

        var total float64
        for i := start; i < end; i++ {
            total += b[i]
```

```

    }
    totalBerat = append(totalBerat, total)

    jumlahIkan := end - start
    rataRata = append(rataRata,
total/float64(jumlahIkan))
    }

    return totalBerat, rataRata
}

func main() {
    var ikanData ikan
    var x, y int

    fmt.Print("Masukkan jumlah ikan: ")
    fmt.Scan(&x)
    fmt.Print("Masukkan kapasitas per wadah: ")
    fmt.Scan(&y)

    fmt.Println("Silahkan masukkan berat ikan!")
    dataikan(&ikanData, x)

    totalBerat, rataRata :=
totalBeratDanRataRata(&ikanData, x, y)

    fmt.Println("Total berat ikan di setiap wadah:")
    for _, berat := range totalBerat {
        fmt.Printf("%.2f ", berat)
    }
    fmt.Println()

    fmt.Println("Rata-rata berat ikan di setiap wadah:")
    for _, rata := range rataRata {
        fmt.Printf("%.2f ", rata)
    }
    fmt.Println()
}

```

## Screenshoot Output

```
PS C:\Users\HP> go run "d:\Praktikum Alpro\Modul 11\Unguided\unguided2.go"
Masukkan jumlah ikan: 4
Masukkan kapasitas per wadah: 2
Silahkan masukkan berat ikan!
Masukkan berat ikan ke-1: 2.1
Masukkan berat ikan ke-2: 3.4
Masukkan berat ikan ke-3: 2.7
Masukkan berat ikan ke-4: 1.8
Total berat ikan di setiap wadah:
5.50 4.50
Rata-rata berat ikan di setiap wadah:
2.75 2.25
PS C:\Users\HP> █
```

## Deskripsi Program

Program go di atas dirancang untuk membantu pengguna menghitung total berat dan rata-rata berat ikan dalam beberapa wadah berdasarkan jumlah ikan dan kapasitas maksimum wadah yang ditentukan. Pengguna diminta memasukkan berat masing-masing ikan, lalu program akan membagi ikan-ikan tersebut ke dalam wadah sesuai kapasitas yang ditentukan. Setelah itu, program akan menghitung total berat dan rata-rata berat ikan di setiap wadah, kemudian menampilkan hasilnya.

Fungsi utama program ini adalah `dataikan` yang berfungsi untuk membaca berat ikan dari input pengguna dan menyimpannya ke dalam array. Selanjutnya, fungsi `totalBeratDanRataRata` menghitung total berat dan rata-rata berat ikan di setiap wadah. Fungsi ini menggunakan metode pembagian data berdasarkan indeks array untuk menentukan batas awal dan akhir setiap wadah sehingga data ikan dapat terbagi secara proporsional.



### 3. Soal Studi Case

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.

Buatlah program dengan spesifikasi subprogram sebagai berikut:

```
type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita; bMin, bMax *float64) {
  /* I.S. Terdefinisi array dinamis arrBerat
  Proses: Menghitung berat minimum dan maksimum dalam array
  F.S. Menampilkan berat minimum dan maksimum balita */
  ...
}

function rerata (arrBerat arrBalita) real {
  /* menghitung dan mengembalikan rerata berat balita dalam array */
  ...
}
```

Perhatikan sesi interaksi pada contoh berikut ini (teks bergaris bawah adalah input/read)

```
Masukan banyak data berat 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 kg
Berat balita maksimum: 9.90 kg
Rerata berat balita: 6.38 kg
```

### Sourcecode

```
package main

import "fmt"

type arrBerat [100]float64

func hitungMinMax(arr arrBerat, n int, Min, Max *float64) {
  {
    *Min = arr[0]
    *Max = arr[0]

    for i := 1; i < n; i++ {
      if arr[i] < *Min {
```

```

        *Min = arr[i]
    }
    if arr[i] > *Max {
        *Max = arr[i]
    }
}

func rerata(arr arrBerat, n int) float64 {
    var total float64
    for i := 0; i < n; i++ {
        total += arr[i]
    }
    return total / float64(n)
}

func main() {
    var berat arrBerat
    var n int
    var min, max float64

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

    if n <= 0 || n > 100 {
        fmt.Println("Jumlah data tidak valid! Harus antara
1 hingga 100.")
        return
    }

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

    hitungMinMax(berat, n, &min, &max)
    rata := rerata(berat, n)

    fmt.Printf("Berat balita minimum: %.2f kg\n", min)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", max)
    fmt.Printf("Rata-rata berat balita: %.2f kg\n", rata)
}

```

## Screenshoot Output

```
PS C:\Users\HP> go run "d:\Praktikum Alpro\Modul 11\Unguided\unguided3.go"
Masukkan banyak data berat balita: 4
Masukkan berat balita ke-1: 3.4
Masukkan berat balita ke-2: 3.8
Masukkan berat balita ke-3: 3.1
Masukkan berat balita ke-4: 3
Berat balita minimum: 3.00 kg
Berat balita maksimum: 3.80 kg
Rata-rata berat balita: 3.32 kg
PS C:\Users\HP> █
```

## Deskripsi Program

Program Go ini di atas untuk mengolah data berat badan balita dan memberikan informasi seperti berat badan terkecil, terbesar, serta rata-rata dari data yang dimasukkan pengguna. Pengguna akan diminta memasukkan jumlah balita dan berat badan masing-masingnya, lalu program akan menghitung nilai minimum, maksimum, dan rata-rata berat badan. Fungsi utama dalam program ini adalah `hitungMinMax` yang digunakan untuk mencari berat badan terkecil dan terbesar dari data yang ada. Kemudian, ada fungsi `rerata` untuk menghitung rata-rata berat badan dari balita-balita tersebut. Setelah dihitung, data akan langsung ditampilkan.

### III. DAFTAR PUSTAKA

- Novalagung. (n.d.). Fungsi dan closure dalam Go. Dasar Pemrograman Golang. Diakses pada 24 November 2024, dari <https://dasarpemrogramangolang.novalagung.com/A-fungsi-closure.html>
- GeeksforGeeks. (n.d.). Slices in Golang. Retrieved November 24, 2024, from <https://www.geeksforgeeks.org/slices-in-golang/>
- The Legend. (n.d.). Best praktis basic Golang - Part 1. Medium. Diakses pada 24 November 2024, dari <https://medium.com/the-legend/best-praktis-basic-golang-part-1-2cf8111a695d>