

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 11**

**PENCARIAN NILAI KSTRIM PADA HIMPUNAN DATA**



**Disusun Oleh :**

**Zahra Tsurroya Poetri / 231110217**

**IF 11 - 05**

**Dosen Pengampu :**

**Arif Amrulloh**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS 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. Contoh penggunaannya dalam kehidupan nyata sangat beragam, misalnya-pencarian file di dalam directory komputer, pencarian suatu teks di dalam sebuah dokumen, pencarian bulu 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 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	max ← 1	max = 0
2	i ← 2	i = 1
3	while i ≤ n do	for i < n {
4	if a[i] > a[max] then	if a[i] > a[max] {
5	max ← i	max = i
6	endif	}
7	i ← 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. UNGUIDED

### 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"
)

// Mendeklarasikan tipe data beratKelinci dengan array
berkapasitas 1000
type beratKelinci [1000]float64

func main() {
    // Mendeklarasikan variabel n untuk menyimpan
    jumlah kelinci
    var n int
    // Mendeklarasikan variabel berat bertipe array
    beratKelinci untuk menyimpan berat kelinci
    var berat beratKelinci

    // Menampilkan pesan untuk meminta input jumlah
    anak kelinci
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&n)

    // Memastikan jumlah anak kelinci yang diinputkan
    harus di antara 1 dan 1000
    if n <= 0 || n > 1000 {
        // Jika jumlah kelinci tidak valid, tampilkan
        pesan error dan hentikan eksekusi program
        fmt.Println("Input tidak valid. Jumlah anak
        kelinci harus antara 1 dan 1000.")
        return
    }

    // Menampilkan pesan untuk meminta input berat anak
```

```

kelinci
    fmt.Println("Masukkan berat anak kelinci:")
    // Loop untuk menerima input berat kelinci sesuai
    dengan jumlah n
    for i := 0; i < n; i++ {
        // Menampilkan pesan untuk meminta berat anak
        kelinci ke-i
        fmt.Printf("Berat anak kelinci ke-%d: ", i+1)
        // Menerima input berat anak kelinci ke-i
        fmt.Scan(&berat[i])
    }

    // Inisialisasi nilai awal untuk pencarian berat
    terkecil (min) dan terbesar (max)
    minBerat := berat[0]
    maxBerat := berat[0]

    // Melakukan iterasi untuk mencari berat terkecil
    dan terbesar
    for i := 1; i < n; i++ {
        // Jika berat kelinci ke-i lebih kecil dari
        minBerat, perbarui minBerat
        if berat[i] < minBerat {
            minBerat = berat[i]
        }
        // Jika berat kelinci ke-i lebih besar dari
        maxBerat, perbarui maxBerat
        if berat[i] > maxBerat {
            maxBerat = berat[i]
        }
    }

    // Menampilkan hasil berat terkecil dan terbesar
    fmt.Printf("Berat terkecil: %.2f\n", minBerat)
    fmt.Printf("Berat terbesar: %.2f\n", maxBerat)
}

```

## Screenshoot Output

```
PS D:\kuliah\sem 3\Praktikum ALPRO 2\Modul 11> go run
Masukkan jumlah anak kelinci:
5
Masukkan berat anak kelinci:
2.1 1.9 3 2.7 2.5
Berat terkecil: 1.90
Berat terbesar: 3.00
PS D:\kuliah\sem 3\Praktikum ALPRO 2\Modul 11> █
```

## Deskripsi Program

Program tersebut merupakan program menggunakan tipe data dasar untuk menghitung berat terkecil (`minBerat`) dan terbesar (`maxBerat`) dari jumlah anak kelinci dan beratnya yang diinputkan oleh pengguna dengan nilai N. Pengguna diminta menginputkan nilai N diantara 1 dan 1000 sesuai dengan kapasitas `beratKelinci` yaitu 1000. Jika pengguna menginputkan di luar kapasitas `beratKelinci`, maka program akan menampilkan pesan tidak valid.

## Unguided 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"
)

// Mendeklarasikan tipe data struct 'wadah' untuk
menampung total berat ikan per wadah
type wadah struct {
    total_beratIkan float64
}

func main() {
    // Mendeklarasikan dua variabel x dan y, untuk
    jumlah ikan dan kapasitas wadah
    var x, y int
    fmt.Print("Masukkan jumlah ikan dan kapasitas
wadah: ") // Menampilkan pesan untuk input
    fmt.Scan(&x, &y) // Membaca input jumlah ikan dan
    kapasitas wadah

    // Memastikan jumlah ikan yang diinputkan harus
    antara 1 sampai 1000 dan kapasitas wadah lebih besar dari
    0
    if x <= 0 || x > 1000 || y <= 0 {
        fmt.Println("Input tidak valid. Jumlah ikan
        harus antara 1 dan 1000. Kapasitas wadah harus lebih
```

```

    besar dari 0") // Menampilkan pesan error
        return // Menghentikan program jika input
    tidak valid
    }

    // Array untuk menampung berat ikan, kapasitas
    maksimal 1000 ikan
    var beratIkan [1000]float64

    // Meminta pengguna untuk memasukkan berat ikan
    satu per satu
    fmt.Println("Masukkan berat ikan: ")
    for i := 0; i < x; i++ {
        fmt.Scan(&beratIkan[i]) // Membaca berat
    ikan dari input pengguna
    }

    // Menghitung jumlah wadah yang diperlukan
    jumlahWadah := (x + y - 1) / y

    // Array untuk menampung wadah, kapasitas maksimal
    1000 wadah
    var wadah [1000]wadah

    // Mendistribusikan berat ikan ke wadah sesuai
    dengan kapasitas wadah
    for i := 0; i < x; i++ {
        wadahIndex := i / y // Menentukan indeks
    wadah berdasarkan indeks ikan
        wadah[wadahIndex].total_beratIkan +=
    beratIkan[i] // Menambahkan berat ikan ke total berat
    wadah yang sesuai
    }

    // Menampilkan total berat per wadah
    fmt.Println("Total berat per wadah: ")

```

```

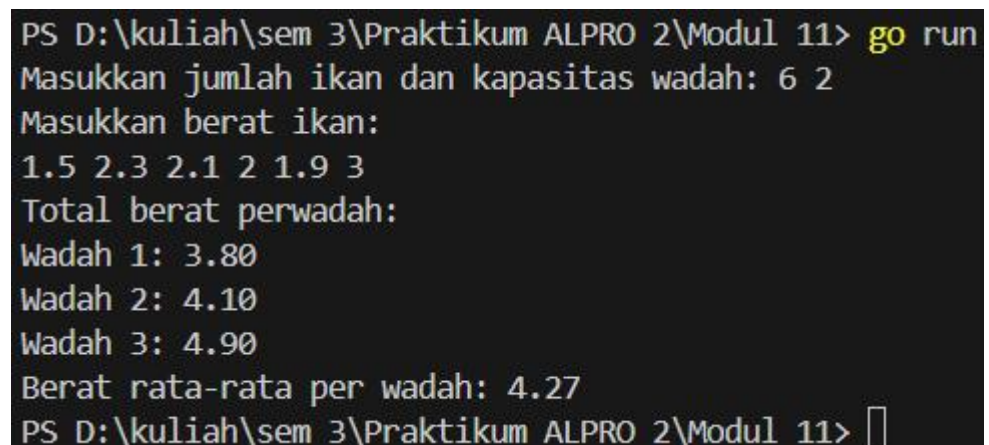
        for i := 0; i < jumlahWadah; i++ {
            fmt.Printf("Wadah %d: %.2f\n", i+1,
wadah[i].total_beratIkan) // Menampilkan total berat
ikan dalam wadah ke-i
        }

        // Menghitung total berat ikan untuk menghitung
rata-rata
        total_beratIkan := 0.0
        for i := 0; i < jumlahWadah; i++ {
            total_beratIkan += wadah[i].total_beratIkan
// Menambahkan total berat wadah ke total keseluruhan
        }

        // Menghitung rata-rata berat per wadah
        rRata := total_beratIkan / float64(jumlahWadah)
        fmt.Printf("Berat rata-rata per wadah: %.2f\n",
rRata) // Menampilkan berat rata-rata per wadah
    }

```

### Screenshoot Output



```

PS D:\kuliah\sem 3\Praktikum ALPRO 2\Modul 11> go run
Masukkan jumlah ikan dan kapasitas wadah: 6 2
Masukkan berat ikan:
1.5 2.3 2.1 2 1.9 3
Total berat perwadah:
Wadah 1: 3.80
Wadah 2: 4.10
Wadah 3: 4.90
Berat rata-rata per wadah: 4.27
PS D:\kuliah\sem 3\Praktikum ALPRO 2\Modul 11> 

```

### Deskripsi Program

Program tersebut merupakan program untuk menentukan tarif ikan yang akan dijual ke pasar menggunakan tipe data terstruktur yang disimpan

dalam wadah dengan kapasitas 1000 untuk menampung data berat ikan yang akan dijual. Pengguna diminta untuk menginputkan jumlah ikan ( $x$ ) dan kapasitas wadah yang dapat dimasukkan ikan ( $y$ ). Kemudian hasil output akan menampilkan total berat (`total_beratIkan`) per wadah yang telah dihitung sesuai dengan inputan pengguna dan menampilkan rata-ratanya (`rRata`). Jika inputan tidak sesuai dengan kapasitas 1000 dalam wadah, maka program akan menampilkan pesan tidak valid.

### Unguided 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"
)

// Mendeklarasikan tipe data array 'arrBalita' untuk
menampung berat balita, kapasitas maksimal 100
type arrBalita [100]float64

// Fungsi untuk menghitung nilai minimum dan maksimum
dari array berat balita
func hitungMinMax(arrBerat arrBalita, n int) (float64,
float64) {
    // Inisialisasi nilai minimum dan maksimum
    bMin := arrBerat[0]
    bMax := arrBerat[0]

    // Melakukan iterasi untuk mencari nilai minimum
dan maksimum
    for i := 1; i < n; i++ {
        // Menentukan jika nilai array lebih kecil
dari nilai minimum
        if arrBerat[i] < bMin {
            bMin = arrBerat[i]
        }
        if arrBerat[i] > bMax {
            bMax = arrBerat[i]
        }
    }
    return bMin, bMax
}
```

```

    }

    // Menentukan jika nilai array lebih besar
    dari nilai maksimum
    if arrBerat[i] > bMax {
        bMax = arrBerat[i]
    }
}

// Mengembalikan nilai minimum dan maksimum
return bMin, bMax
}

// Fungsi untuk menghitung rata-rata dari array berat
balita
func rerata(arrBerat arrBalita, n int) float64 {
    var total float64
    // Menjumlahkan semua nilai dalam array
    for i := 0; i < n; i++ {
        total += arrBerat[i]
    }
    // Mengembalikan nilai rata-rata
    return total / float64(n)
}

func main() {
    var n int // Deklarasi variabel untuk
    jumlah data balita
    var berat arrBalita // Deklarasi array untuk
    menyimpan berat balita

    // Meminta input jumlah data berat balita
    fmt.Print("Masukan banyak data berat balita: ")
    fmt.Scan(&n)

    // Memastikan jumlah data yang diinputkan berada
    antara 1 dan 100
    if n <= 0 || n > 100 {

```

```

        fmt.Println("JInput tidak valid. Jumlah harus
antara 1 dan 100") // Menampilkan pesan jika input tidak
valid

        return // Menghentikan program
    }

    // Meminta input berat balita satu per satu
    for i := 0; i < n; i++ {
        fmt.Printf("Masukan berat balita ke-%d: ",
i+1) // Menampilkan pesan untuk input berat balita ke-i
        fmt.Scan(&berat[i]) // Membaca berat balita
ke-i
    }

    // Menghitung nilai minimum dan maksimum berat
balita
    bMin, bMax := hitungMinMax(berat, n)

    // Menghitung rata-rata berat balita
    rataRata := rerata(berat, n)

    // Menampilkan hasil perhitungan
    fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)
// Menampilkan berat balita minimum
    fmt.Printf("Berat balita maksimum: %.2f kg\n",
bMax) // Menampilkan berat balita maksimum
    fmt.Printf("Rerata berat balita: %.2f kg\n",
rataRata) // Menampilkan rata-rata berat balita
}

```

## Screenshoot Output

```
PS D:\kuliah\sem 3\Praktikum ALPRO 2\Modul 11> go run
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
PS D:\kuliah\sem 3\Praktikum ALPRO 2\Modul 11> █
```

## Deskripsi Program

Program tersebut merupakan program menggunakan tipe data terstruktur untuk mendata berat balita pada posyandu yang disimpan dalam `arrBalita` dengan kapasitas 100. Program memina pengguna memasukkan banyak data balita dan berat balita sesuai banyaknya data. Kemudian program mengeksekusi untuk menentukan hasil akhir yaitu berat balita minimum (`bMin`), maksimum (`bMax`) dan reratanya (`rataRata`) dalam bentuk kg. Jika pengguna menginputkan tidak sesuai dengan kapasitas array, maka program akan menampilkan pesan tidak valid.



### **Daftar Pustaka**

- [1] Susilowati, A. Zahara, A. (2024). *Modul 11 Praktikum Algoritma Pemrograman 2*. Program Studi Teknik Informatika, Telkom University Purwokerto.