

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL XI

PENCARIAN NILAI EXTREME PADA HIMPUNAN DATA



Disusun Oleh :

Nia Novela Ariandini / 2311102057

IF-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 dalam suatu himpunan data adalah salah satu proses penting dalam analisis data yang bertujuan untuk menemukan nilai terendah (minimum) atau tertinggi (maksimum) dalam kumpulan data tersebut. Nilai ekstrem ini memainkan peran penting dalam memahami distribusi data dan mendeteksi nilai yang tidak biasa atau outliers. Proses ini sering digunakan dalam berbagai aplikasi, termasuk analisis statistik, optimasi, pembelajaran mesin, dan pemrosesan data.

1. Pengertian Nilai Ekstrem

Nilai Minimum : Nilai minimum dalam suatu himpunan data adalah elemen yang memiliki nilai terkecil jika dibandingkan dengan semua elemen lainnya dalam kumpulan data tersebut. Nilai ini dapat digunakan untuk memahami batas bawah dari sebuah dataset dan memberikan gambaran tentang titik terendah yang ada dalam data.

Nilai Maksimum : Nilai maksimum dalam suatu himpunan data adalah elemen yang memiliki nilai terbesar jika dibandingkan dengan semua elemen lainnya dalam kumpulan data tersebut. Nilai ini memberikan gambaran tentang titik tertinggi yang ada dalam data.

Contoh :

- Pada kumpulan data {3, 5, 1, 9, 7}, nilai minimum adalah 1, dan nilai maksimum adalah 9.
- Pada kumpulan data {20, 15, 30, 10, 25}, nilai minimum adalah 10, dan nilai maksimum adalah 30.

2. Proses Pencarian Nilai Ekstrem

Pencarian nilai ekstrem biasanya dilakukan dengan menggunakan algoritma yang mengiterasi seluruh elemen dalam dataset untuk membandingkan setiap elemen dengan nilai ekstrem sementara yang telah ditemukan. Proses ini melibatkan langkah-langkah sebagai berikut :

- **Inisialisasi :** Langkah pertama adalah menginisialisasi nilai ekstrem (baik minimum atau maksimum) dengan elemen pertama dalam dataset.
- **Iterasi :** Selanjutnya, algoritma akan mengiterasi melalui setiap elemen dalam dataset, membandingkannya dengan nilai ekstrem sementara yang telah ditemukan. Jika ditemukan nilai yang lebih

kecil dari nilai minimum yang ada atau lebih besar dari nilai maksimum, maka nilai ekstrem akan diperbarui.

- Hasil : Setelah seluruh elemen diperiksa, nilai ekstrem yang tersisa adalah nilai minimum atau maksimum yang dicari.

3. Algoritma Pencarian Nilai Ekstrem

Ada beberapa cara untuk mencari nilai ekstrem dalam dataset, baik dengan menggunakan metode iteratif sederhana atau menggunakan algoritma yang lebih canggih. Salah satu cara umum untuk mencari nilai ekstrem adalah sebagai berikut :

- Pencarian Nilai Minimum : Untuk mencari nilai minimum, mulai dengan mengasumsikan elemen pertama adalah nilai minimum, lalu periksa elemen lainnya. Setiap kali menemukan elemen yang lebih kecil dari nilai minimum yang ditemukan, perbarui nilai minimum tersebut.
- Pencarian Nilai Maksimum: Untuk mencari nilai maksimum, mulai dengan mengasumsikan elemen pertama adalah nilai maksimum, lalu periksa elemen lainnya. Setiap kali menemukan elemen yang lebih besar dari nilai maksimum yang ditemukan, perbarui nilai maksimum tersebut.

II. UNGUIDED

1. UNGUIDED 1

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"
    "math"
)

func main() {
    var jumlah int
    var weights [1000]float64

    //input banyaknya anak kelinci yang akan ditimbang
    fmt.Print("Masukkan jumlah anak kelinci : ")
    fmt.Scan(&jumlah)

    //input berat anak kelinci
    fmt.Print("Masukkan berat anak kelinci : ")
    for i := 0; i < jumlah; i++ {
        fmt.Scan(&weights[i])
    }

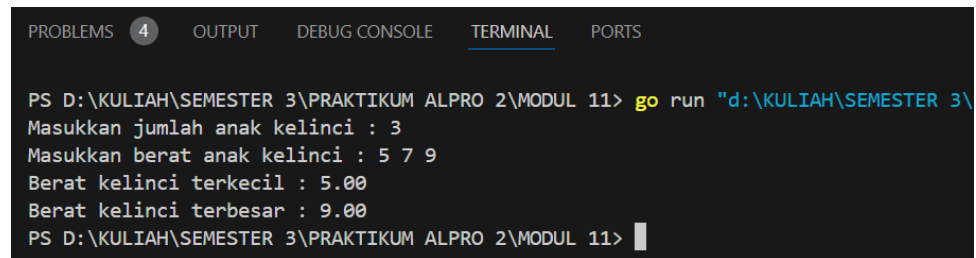
    //menentukan berat terkecil dan terbesar
    min := math.MaxFloat64
    max := -math.MaxFloat64

    for i := 0; i < jumlah; i++ {
        if weights[i] < min {
            min = weights[i] //memperbarui nilai terkecil
        }
        if weights[i] > max {
            max = weights[i] //memperbarui nilai terbesar
        }
    }

    //output berat kelinci terkecil dan terbesar
```

```
    fmt.Printf("Berat kelinci terkecil : %.2f\n", min)
    fmt.Printf("Berat kelinci terbesar : %.2f\n", max)
}
```

Screenshoot Output

A screenshot of a code editor's terminal window. The terminal shows the execution of a Go program. The prompt is 'PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 11>'. The user enters 'go run "d:\KULIAH\SEMESTER 3\' followed by several lines of input: 'Masukkan jumlah anak kelinci : 3', 'Masukkan berat anak kelinci : 5 7 9', and the program outputs 'Berat kelinci terkecil : 5.00' and 'Berat kelinci terbesar : 9.00'. The prompt returns to 'PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 11>'.

```
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 11> go run "d:\KULIAH\SEMESTER 3\
Masukkan jumlah anak kelinci : 3
Masukkan berat anak kelinci : 5 7 9
Berat kelinci terkecil : 5.00
Berat kelinci terbesar : 9.00
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 11> |
```

Deskripsi Program

Program ini untuk mencatat berat sejumlah anak kelinci dan menentukan berat terkecil serta terbesar di antara data yang dimasukkan. Program ini dirancang untuk menerima input berupa jumlah anak kelinci dan berat masing-masing anak kelinci. Hasil akhirnya adalah menampilkan berat anak kelinci terkecil dan terbesar dalam format desimal dengan dua angka di belakang koma.

Algoritma dan Cara Program Berfungsi : Program dimulai dengan meminta input jumlah anak kelinci yang akan ditimbang, yang disimpan dalam variabel `jumlah`. Selanjutnya, program meminta pengguna untuk memasukkan berat badan setiap anak kelinci satu per satu, yang kemudian disimpan dalam array `weights`. Untuk menemukan berat terkecil dan terbesar, program menggunakan dua variabel, `min` dan `max`, yang diinisialisasi masing-masing dengan nilai maksimum (`math.MaxFloat64`) dan nilai minimum (`-math.MaxFloat64`). Melalui perulangan, program memeriksa setiap elemen dalam array. Jika berat anak kelinci lebih kecil dari `min`, maka `min` diperbarui dengan nilai tersebut. Sebaliknya, jika berat anak kelinci lebih besar dari `max`, maka `max` diperbarui. Setelah semua data diproses, program mencetak hasil berupa berat anak kelinci terkecil dan terbesar dengan format desimal dua angka. Program ini dirancang efisien dengan kompleksitas waktu sebesar $O(n)$ karena hanya membutuhkan satu iterasi melalui array untuk menentukan nilai minimum dan maksimum. Output akhir membantu pengguna mengetahui berat badan terkecil dan terbesar di antara anak kelinci yang ditimbang.

2. UNGUIDED 2

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 dimasukan ke dalam wadah. Baris kedua terdiri dari sejumlah x bilangan riil 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 dimasukan 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"

func main() {
    var x, y int

    //input jumlah ikan dan jumlah ikan per wadah
    fmt.Print("Masukkan jumlah ikan dan jumlah ikan per wadah : ")
    fmt.Scan(&x, &y)

    //menyimpan berat ikan dalam array
    var beratIkan [1000]float64
    fmt.Print("Masukkan berat ikan : ")
    for i := 0; i < x; i++ {
        fmt.Scan(&beratIkan[i])
    }

    //menghitung total berat ikan di setiap wadah
    var totalBerat []float64
    var beratWadah float64

    for i := 0; i < x; i++ {
        beratWadah += beratIkan[i] //tambahkan berat ikan ke wadah saat ini

        //jika sudah cukup banyak ikan dalam wadah (y ikan), simpan total berat wadah
        if (i+1)%y == 0 || i == x-1 {
            totalBerat = append(totalBerat, beratWadah)
            beratWadah = 0 //reset berat wadah setelah memasukkan ikan
        }
    }
}
```

```

    }
}

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

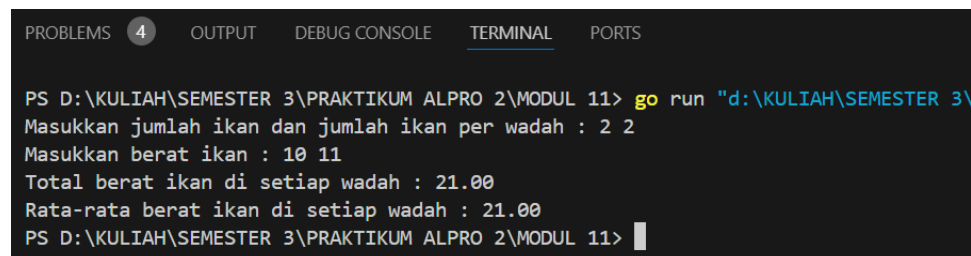
//menghitung rata-rata berat ikan per wadah
rataRata := 0.0
for _, berat := range totalBerat {
    rataRata += berat
}

rataRata /= float64(len(totalBerat))

//output : rata-rata berat ikan per wadah
fmt.Printf("Rata-rata berat ikan di setiap wadah :
%.2f\n", rataRata)
}

```

Screenshoot Output



```

PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 11> go run "d:\KULIAH\SEMESTER 3\
Masukkan jumlah ikan dan jumlah ikan per wadah : 2 2
Masukkan berat ikan : 10 11
Total berat ikan di setiap wadah : 21.00
Rata-rata berat ikan di setiap wadah : 21.00
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 11>

```

Deskripsi Program

Program ini adalah sebuah aplikasi yang digunakan untuk mencatat berat sejumlah ikan, menghitung total berat ikan di setiap wadah, serta menghitung rata-rata berat ikan per wadah.

Algoritma dan Cara Program Berfungsi : Program dimulai dengan meminta pengguna memasukkan jumlah total ikan (x) dan kapasitas jumlah ikan per wadah (y). Selanjutnya, pengguna diminta untuk memasukkan berat masing-masing ikan, yang disimpan dalam array beratIkan. Program kemudian menggunakan perulangan untuk menghitung total berat ikan di setiap wadah. Dalam proses ini, program menambahkan berat ikan ke wadah yang sedang diisi hingga jumlah ikan di wadah tersebut mencapai kapasitas maksimum (y) atau jika semua ikan telah dimasukkan. Setelah itu, total berat setiap wadah disimpan dalam array totalBerat, dan berat wadah

direset untuk menghitung wadah berikutnya. Setelah seluruh ikan dikelompokkan ke dalam wadah, program menampilkan total berat ikan di setiap wadah. Kemudian, program menghitung rata-rata berat ikan per wadah dengan menjumlahkan seluruh berat di array totalBerat dan membagi total tersebut dengan jumlah wadah. Hasil akhir berupa total berat ikan di setiap wadah serta rata-rata berat ikan per wadah ditampilkan dalam format dua angka desimal.

3. UNGUIDED 3

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. Dan 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"
)

//fungsi untuk menghitung berat minimum dan maksimum
```



```

func hitungMinMax(arrBerat []float64, bMin *float64, bMax
*float64) {
    //inisialisasi bMin dan bMax dengan nilai elemen
pertama
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]
    for _, berat := range arrBerat {
        if berat < *bMin {
            *bMin = berat
        }
        if berat > *bMax {
            *bMax = berat
        }
    }
}

//fungsi untuk menghitung rata-rata berat balita
func hitungRerata(arrBerat []float64) float64 {
    var total float64
    for _, berat := range arrBerat {
        total += berat
    }
    return total / float64(len(arrBerat))
}

func main() {
    var n int
    fmt.Print("Masukkan banyak data berat balita : ")
    fmt.Scan(&n)

    arrBerat := make([]float64, n) //membuat array dengan
panjang n

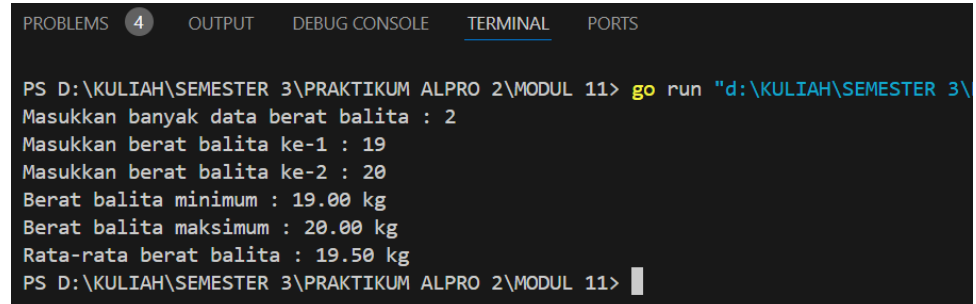
    //input data berat balita
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan berat balita ke-%d : ", i+1)
        fmt.Scan(&arrBerat[i])
    }

    //hitung berat minimum, maksimum, dan rata-rata
    var bMin, bMax float64
    hitungMinMax(arrBerat, &bMin, &bMax)
    rerata := hitungRerata(arrBerat)

    //tampilkan hasil
    fmt.Printf("Berat balita minimum : %.2f kg\n", bMin)
    fmt.Printf("Berat balita maksimum : %.2f kg\n", bMax)
    fmt.Printf("Rata-rata berat balita : %.2f kg\n",
rerata)
}

```

Screenshoot Output

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active. The prompt is 'PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 11>'. The user enters 'go run "d:\KULIAH\SEMESTER 3\' followed by several lines of input: 'Masukkan banyak data berat balita : 2', 'Masukkan berat balita ke-1 : 19', and 'Masukkan berat balita ke-2 : 20'. The program outputs: 'Berat balita minimum : 19.00 kg', 'Berat balita maksimum : 20.00 kg', and 'Rata-rata berat balita : 19.50 kg'. The prompt returns to 'PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 11>'.

```
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 11> go run "d:\KULIAH\SEMESTER 3\
Masukkan banyak data berat balita : 2
Masukkan berat balita ke-1 : 19
Masukkan berat balita ke-2 : 20
Berat balita minimum : 19.00 kg
Berat balita maksimum : 20.00 kg
Rata-rata berat balita : 19.50 kg
PS D:\KULIAH\SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 11>
```

Deskripsi Program

Program ini adalah aplikasi yang dirancang untuk mencatat data berat badan sejumlah balita, kemudian menghitung dan menampilkan berat minimum, maksimum, serta rata-rata berat balita tersebut.

Algoritma dan Cara Program Berfungsi : Program dimulai dengan meminta pengguna untuk memasukkan jumlah balita yang akan dicatat. Selanjutnya, program meminta berat badan masing-masing balita secara berurutan dan menyimpannya dalam array dinamis `arrBerat`. Setelah semua data berat dimasukkan, program menggunakan dua fungsi, yaitu `hitungMinMax` untuk menentukan berat minimum dan maksimum, serta `hitungRerata` untuk menghitung rata-rata berat balita. Fungsi `hitungMinMax` bekerja dengan memeriksa setiap elemen dalam array. Awalnya, variabel `bMin` dan `bMax` diinisialisasi dengan nilai elemen pertama array. Kemudian, setiap elemen dibandingkan dengan `bMin` dan `bMax`. Jika ada elemen yang lebih kecil dari `bMin`, nilai `bMin` diperbarui; sebaliknya, jika ada elemen yang lebih besar dari `bMax`, nilai `bMax` diperbarui. Fungsi `hitungRerata` menghitung rata-rata dengan menjumlahkan semua elemen dalam array dan membagi totalnya dengan jumlah elemen dalam array. Setelah menghitung semua nilai, program menampilkan hasilnya, yaitu berat minimum, berat maksimum, dan rata-rata berat balita dalam format dua angka desimal.

III. KESIMPULAN

Pencarian nilai ekstrem pada himpunan data merupakan konsep penting dalam analisis data yang bertujuan untuk menentukan nilai minimum dan maksimum dari sekumpulan data tertentu. Di mana data cukup dilalui satu kali untuk menemukan nilai minimum dan maksimum. Prosesnya dimulai dengan menginisialisasi variabel untuk menyimpan nilai ekstrem, biasanya menggunakan elemen pertama dari data atau nilai awal ekstrem default. Selanjutnya, setiap elemen dalam data dibandingkan dengan nilai minimum dan maksimum saat ini, dan nilainya diperbarui jika diperlukan. Dengan memanfaatkan struktur data seperti array atau slice, implementasi algoritma ini menjadi sederhana dan terorganisasi. Fungsi yang modular, seperti untuk menghitung nilai ekstrem dan rata-rata, membuat kode lebih mudah dipahami dan dapat digunakan kembali. Pencarian nilai ekstrem tidak hanya membantu dalam memahami distribusi data, tetapi juga sering dikombinasikan dengan metrik lain seperti rata-rata untuk analisis yang lebih mendalam.

IV. REFERENSI

- [1] Modul XI Praktikum Algoritma dan Pemrograman 2
- [2] https://www.petanikode.com/go-untuk-pemula/#google_vignette