

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



Oleh:

NAUFAL THORIQ MUZHAFAR

2311102078

IF-11-02

S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Dalam analisis data, pencarian nilai ekstrem (minimum dan maksimum) merupakan salah satu proses penting untuk menentukan batas-batas dalam suatu himpunan data. Nilai ekstrem memberikan informasi tentang elemen terkecil dan terbesar dalam data, yang dapat digunakan untuk berbagai tujuan seperti analisis statistik, visualisasi data, atau pengambilan keputusan.

Definisi Nilai Ekstrim

Nilai ekstrem dari sebuah himpunan data adalah:

1. **Nilai Minimum:** Elemen terkecil dalam himpunan data.
2. **Nilai Maksimum:** Elemen terbesar dalam himpunan data.

Misalnya, untuk data 4,8,15,16,23,42, 8, 15, 16, 23, 42, 8,15,16,23,42:

- Nilai minimum adalah 4.
- Nilai maksimum adalah 42.

Penerapan Pencarian Nilai Ekstrim

Pencarian nilai ekstrem biasanya dilakukan melalui iterasi elemen dalam himpunan data. Proses ini sering melibatkan perbandingan berulang untuk menentukan apakah suatu elemen lebih kecil dari nilai minimum atau lebih besar dari nilai maksimum yang sudah ditemukan.

Dalam Golang, pencarian nilai ekstrem dilakukan dengan langkah-langkah berikut:

1. **Inisialisasi Nilai Awal:** Tetapkan nilai awal minimum dan maksimum. Biasanya, nilai awal diambil dari elemen pertama himpunan data.
2. **Iterasi Elemen:** Gunakan struktur kontrol seperti for loop untuk membandingkan setiap elemen dalam himpunan data.
3. **Perbandingan Nilai:**
 - Jika elemen saat ini lebih kecil dari nilai minimum, perbarui nilai minimum.
 - Jika elemen saat ini lebih besar dari nilai maksimum, perbarui nilai maksimum.

```
package main

import (
    "fmt"
)

func findExtremes(data []int) (int, int) {
    if len(data) == 0 {
        panic("Himpunan data kosong")
    }

    // Inisialisasi nilai awal minimum dan maksimum
    min := data[0]
    max := data[0]

    // Iterasi melalui elemen-elemen dalam data
    for _, value := range data {
        if value < min {
            min = value // Perbarui nilai minimum
        }
        if value > max {
            max = value // Perbarui nilai maksimum
        }
    }

    return min, max
}
```

```
func main() {  
    data := []int{4, 8, 15, 16, 23, 42}  
    min, max := findExtremes(data)  
    fmt.Printf("Nilai minimum: %d\n", min)  
    fmt.Printf("Nilai maksimum: %d\n", max)  
}
```

Penjelasan Program

Fungsi findExtremes:

- Menerima input berupa slice berisi data integer.
- Mengembalikan nilai minimum dan maksimum.

Inisialisasi:

- Nilai awal min dan max diambil dari elemen pertama data.

Iterasi:

- Menggunakan pernyataan `for _, value := range data` untuk membandingkan setiap elemen.

Perbandingan:

- Memperbarui nilai min jika elemen lebih kecil dari min.
- Memperbarui nilai max jika elemen lebih besar dari max.

II. GUIDED GUIDED 1

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.

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])
    }

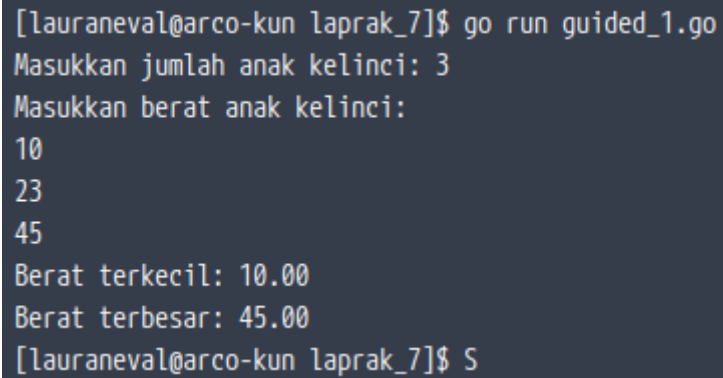
    //Inisialisasi nilai min dan max dengan elemen
    pertama
    min := berat[0]
    max := berat[0]

    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)
}
```

Screenshoot Output

A screenshot of a terminal window showing the execution of a Go program. The prompt is [lauraneval@arco-kun laprak_7]\$ and the command is go run guided_1.go. The program prompts for the number of kittens (Masukkan jumlah anak kelinci: 3) and then for the weight of each kitten (Masukkan berat anak kelinci:). Three weights are entered: 10, 23, and 45. The program then outputs the minimum weight (Berat terkecil: 10.00) and the maximum weight (Berat terbesar: 45.00). The prompt returns to [lauraneval@arco-kun laprak_7]\$ S.

```
[lauraneval@arco-kun laprak_7]$ go run guided_1.go
Masukkan jumlah anak kelinci: 3
Masukkan berat anak kelinci:
10
23
45
Berat terkecil: 10.00
Berat terbesar: 45.00
[lauraneval@arco-kun laprak_7]$ S
```

Penjelasan Program

Program ini menghitung berat terkecil dan terbesar dari sejumlah anak kelinci. Pengguna memasukkan jumlah anak kelinci (N) dan berat masing-masing yang disimpan dalam array. Nilai awal *min* dan *max* diinisialisasi dengan berat anak kelinci pertama, kemudian dibandingkan dengan berat lainnya melalui perulangan untuk memperbarui nilai *min* dan *max*. Akhirnya, program mencetak berat terkecil dan terbesar dalam format dua angka desimal.

GUDED 2

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 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 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.

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()
}
```


Screenshoot Output

```
[lauraneval@arco-kun laprak_7]$ go run guided_2.go
Masukkan jumlah ikan dan kapasitas wadah: 3 5
Masukkan berat tiap ikan:
10 20 30
Total berat tiap wadah:
60.00
Rata-rata berat tiap wadah:
12
[lauraneval@arco-kun laprak_7]$ S_
```

Penjelasan Program

Program ini menghitung total dan rata-rata berat ikan di setiap wadah berdasarkan jumlah ikan (x) dan kapasitas wadah (y). Pengguna memasukkan berat setiap ikan, lalu program menentukan jumlah wadah yang diperlukan dengan pembulatan ke atas. Berat ikan dimasukkan ke wadah sesuai indeks, dan total berat tiap wadah dihitung. Program mencetak total berat tiap wadah dan rata-rata berat per wadah dengan hasil dalam format desimal.

III. UNGUIDED

UNGUIDED 1

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
```

Source Code

```
package main

import "fmt"

type arrBalita [100]float64
```

```

func hitungMinMax(arrBerat arrBalita, n int, bMin, bMax
*float64) {

    *bMin = arrBerat[0]

    *bMax = arrBerat[0]

    for i := 1; i < n; i++ {

        if arrBerat[i] < *bMin {

            *bMin = arrBerat[i]

        }

        if arrBerat[i] > *bMax {

            *bMax = arrBerat[i]

        }

    }

}

func hitungRerata(arrBerat arrBalita, n int) float64 {

    total := 0.0

    for i := 0; i < n; i++ {

        total += arrBerat[i]

    }

    return total / float64(n)

}

func main() {

    var n int

    var berat arrBalita

    var min, max float64

```

```

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

        fmt.Scan(&n)

        for i := 0; i < n; i++ {

            fmt.Printf("Masukkan berat balita ke-%d: ",
i+1)

            fmt.Scan(&berat[i])

        }

        hitungMinMax(berat, n, &min, &max)

        rerata := hitungRerata(berat, n)

        fmt.Printf("Berat balita minimum: %.2f kg\n", min)

        fmt.Printf("Berat balita maksimum: %.2f kg\n",
max)

        fmt.Printf("Rerata berat balita: %.2f
kg\n", rerata)

    }

```

Screenshot Output

```

[lauraneval@arco-kun laprak_7]$ go run unguided_1.go
Masukkan banyak data berat balita: 4
Masukkan berat balita ke-1: 5.3
Masukkan berat balita ke-2: 6.2
Masukkan berat balita ke-3: 4.1
Masukkan berat balita ke-4: 9.9
Berat balita minimum: 4.10 kg
Berat balita maksimum: 9.90 kg
Rerata berat balita: 6.38 kg
[lauraneval@arco-kun laprak_7]$

```

Deskripsi Program

Program di atas dirancang untuk menganalisis data berat balita yang dimasukkan oleh pengguna. Pertama, pengguna diminta untuk memasukkan jumlah data balita (n) dan berat masing-masing balita, yang disimpan dalam array `arrBalita`. Program menggunakan dua fungsi utama. **hitungMinMax**: Fungsi ini menentukan berat balita terkecil (\min) dan terbesar (\max) dengan membandingkan setiap elemen array. Nilai awal \min dan \max diinisialisasi dengan elemen pertama array, lalu diperbarui selama iterasi jika ditemukan nilai yang lebih kecil atau lebih besar. **hitungRerata**: Fungsi ini menghitung rata-rata berat balita dengan menjumlahkan seluruh elemen array dan membaginya dengan jumlah data (n). Setelah fungsi-fungsi tersebut dipanggil, program menampilkan hasil berupa berat balita minimum, maksimum, dan rata-rata dalam format desimal dua angka. Dengan cara ini, data berat balita dapat dianalisis secara sederhana dan efisien.