

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL 10

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Disusun Oleh :

Aji Tri Prasetyo / 2311102064

IF-11-02

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pencarian nilai ekstrem dalam himpunan data adalah proses untuk menemukan nilai maksimum (nilai tertinggi) dan minimum (nilai terendah) dari sekumpulan elemen data. Proses ini sering digunakan dalam berbagai aplikasi seperti analisis statistik, optimasi, dan pemrosesan data.

1. Konsep Dasar Nilai Ekstrem

Nilai Minimum: Elemen dengan nilai terkecil dalam suatu himpunan data.

Nilai Maksimum: Elemen dengan nilai terbesar dalam suatu himpunan data.

2. Metode Pencarian Nilai Ekstrem

Pencarian nilai ekstrem dapat dilakukan dengan algoritma sederhana:

Inisialisasi nilai minimum dan maksimum dengan elemen pertama dalam himpunan. Iterasi melalui setiap elemen dalam himpunan: Jika elemen lebih kecil dari nilai minimum saat ini, perbarui nilai minimum., Jika elemen lebih besar dari nilai maksimum saat ini, perbarui nilai maksimum.

Kompleksitas waktu algoritma ini adalah $O(n)$, di mana n adalah jumlah elemen dalam himpunan.

3. Implementasi dalam Golang

Golang adalah bahasa pemrograman statically typed yang efisien untuk pengolahan data. Untuk mencari nilai ekstrem, kita memanfaatkan tipe data slice karena mendukung penyimpanan elemen dengan ukuran dinamis.

II. GUIDED

1. Guided 1

Sourcecode

```
package main

import "fmt"

func main() {
    var N int
    var berat [1000]float64

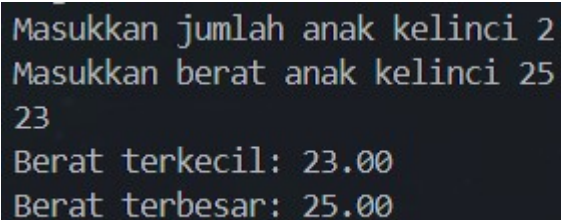
    fmt.Print("Masukkan jumlah anak kelinci ")
    fmt.Scan(&N)
    fmt.Print("Masukkan berat anak kelinci ")
    for i := 0; i < N; i++ {
        fmt.Scan(&berat[i])
    }

    min := berat[0]
    max := berat[0]

    for i := 0; 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 output of the Go program. The text is as follows:

```
Masukkan jumlah anak kelinci 2
Masukkan berat anak kelinci 25
23
Berat terkecil: 23.00
Berat terbesar: 25.00
```

Deskripsi Program

Array berat: Sebuah array dengan kapasitas maksimum 1000 elemen yang digunakan untuk menyimpan berat masing-masing anak kelinci. Tipe datanya adalah float64, yang memungkinkan penyimpanan nilai desimal. Program meminta pengguna untuk memasukkan: Jumlah anak kelinci (N). Berat masing-masing anak kelinci yang disimpan dalam array berat.

2. Guided 1

Sourcecode

```
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
    totalBeratWadah := make([]float64, jumlahWadah)

    for i := 0; i < x; i++ {
        indeksWadah := i / y
        totalBeratWadah[indeksWadah] += berat[i]
    }

    fmt.Print("Berat total wadah: ")
    for _, total := range totalBeratWadah {
        fmt.Printf("%.2f ", total)
    }
    fmt.Println()

    fmt.Print("Rata-rata berat tiap wadah: ")
    for _, total := range totalBeratWadah {
        rataRata := total / float64(y)
        fmt.Printf("%.2f ", rataRata)
    }
    fmt.Println()
}
```

```
}
```

Screenshoot Output

```
Masukkan jumlah ikan dan kapasitas wadah: 4 20
Masukkan berat tiap ikan:
2
2
3
4
Berat total wadah: 11.00
Rata-rata berat tiap wadah: 0.55
```

Deskripsi Program

Program menerima jumlah ikan (x) dan kapasitas wadah (y). Berat setiap ikan dimasukkan ke dalam slice berat. Program menghitung jumlah wadah yang diperlukan. Berat setiap ikan dialokasikan ke wadah yang sesuai, lalu total berat tiap wadah dihitung. Program mencetak: Berat total setiap wadah, Rata-rata berat ikan per wadah.

III. UNGUIDED

Unguided 1

Sourcecode

```
package main
import(
    "fmt"
)
//2311102064 Aji Tri Prasetyo
type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita, x int,bMin, bMax
*float64){
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]
    for i := 1; i < x; i++ {
        if arrBerat[i] > *bMax {
            *bMax = arrBerat[i]
        }
        if arrBerat[i] < *bMin {
            *bMin = arrBerat[i]
        }
    }
}

func rerata(arrBerat arrBalita, x int) float64{
    sum:= 0.0
    for i := 0; i < x; i++ {
        sum += arrBerat[i]
    }
    return sum/ (float64(x))
}

func main(){
    var x int
    var arrBalita[100] float64
    var bMin,bMax float64

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

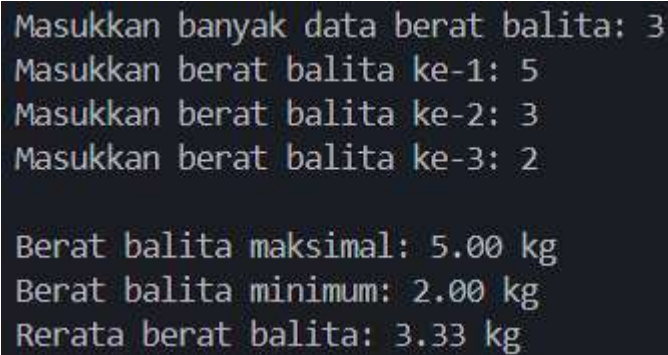
    for i := 0; i < x; i++ {
        fmt.Printf("Masukkan berat balita ke-%d: ",i+1)
        fmt.Scan(&arrBalita[i])
    }
    fmt.Println()

    hitungMinMax(arrBalita,x,&bMin,&bMax)
    avg := rerata(arrBalita,x)

    fmt.Printf("Berat balita maksimal: %.2f kg\n", bMax)
    fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)
```

```
    fmt.Printf("Rerata berat balita: %.2f kg", avg )  
}
```

Screenshoot Output



```
Masukkan banyak data berat balita: 3  
Masukkan berat balita ke-1: 5  
Masukkan berat balita ke-2: 3  
Masukkan berat balita ke-3: 2  
  
Berat balita maksimal: 5.00 kg  
Berat balita minimum: 2.00 kg  
Rerata berat balita: 3.33 kg
```

Deskripsi Program

Fungsi rerata Parameter: arrBerat: Array berat balita. x: Jumlah elemen yang digunakan dalam array. Proses: Menghitung jumlah seluruh elemen dalam array (sum). Membagi sum dengan jumlah elemen (x) untuk mendapatkan rata-rata. Hasil: Mengembalikan rata-rata berat balita dalam bentuk float64.