

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 11
NILAI EKSTREM**



Disusun Oleh :
Mohammad Alfian Naraya / 2311102170
S1-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

A. 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 computer, pencarian suatu teks di dalam sebuah dokumen, pencarian buku pada rak buku, dan contoh lainnya.

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 sebagai berikut :

- Jadikan data pertama sebagai nilai ekstrim
- Lakukan validasi nilai ekstrim dari data kedua hingga data terakhir
- Apabila semua data telah dicek, maka nilai ekstrim yang dimiliki adalah valid.

B. Pencarian Nilai Ekstrim pada Array Bertipe Data Dasar

Misalnya terdefinisi sebuah array of integer dengan kapasitas 2023, dan array berisi sejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut.

C. Pencarian Nilai Ekstrim pada Array Bertipe Data Terstruktur Pada

kasus yang lebih kompleks pencarian ekstrim dapat juga dilakukan, misalnya mencari mahasiswa dengan nilai terbesar, mencari lagu dengan durasi terlama, mencari pembalap yang memiliki catatan waktu balap tercepat dan sebagainya.

II. UNGUIDED

1. 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"
)

func main() {
    var beratKelinci []float64

    var jumlahKelinci int
    fmt.Print("Masukkan jumlah anak kelinci
yang akan ditimbang: ")
    fmt.Scan(&jumlahKelinci)

    if jumlahKelinci <= 0 || jumlahKelinci >
1000 {
        fmt.Println("Jumlah kelinci harus
antara 1-1000")
        return
    }

    beratKelinci = make([]float64,
jumlahKelinci)

    fmt.Println("Masukkan berat masing-masing
kelinci:")
    for i := 0; i < jumlahKelinci; i++ {
```

```

        fmt.Printf("Berat kelinci ke-%d: ",
i+1)
        fmt.Scan(&beratKelinci[i])
    }

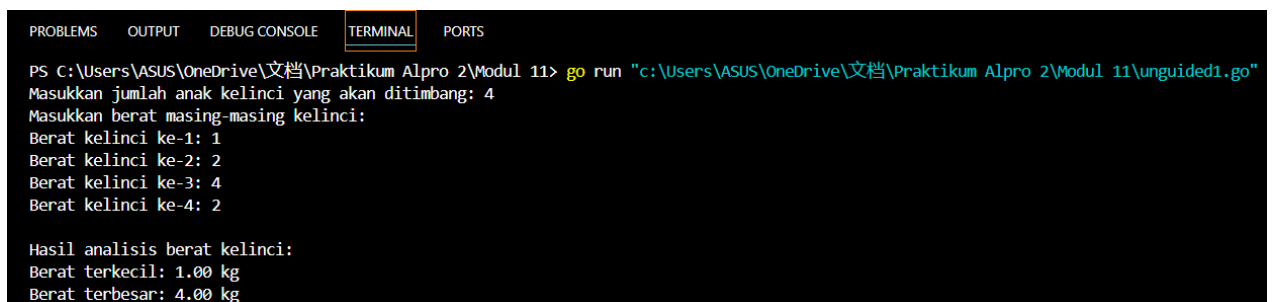
    beratMin, beratMax := beratKelinci[0],
beratKelinci[0]

    for _, berat := range beratKelinci {
        if berat < beratMin {
            beratMin = berat
        }
        if berat > beratMax {
            beratMax = berat
        }
    }

    fmt.Printf("\nHasil analisis berat
kelinci:\n")
    fmt.Printf("Berat terkecil: %.2f kg\n",
beratMin)
    fmt.Printf("Berat terbesar: %.2f kg\n",
beratMax)
}

```

Screenshoot Output



The screenshot shows a terminal window with the following output:

```

PS C:\Users\ASUS\OneDrive\文档\Praktikum Alpro 2\Modul 11> go run "c:\Users\ASUS\OneDrive\文档\Praktikum Alpro 2\Modul 11\unguided1.go"
Masukkan jumlah anak kelinci yang akan ditimbang: 4
Masukkan berat masing-masing kelinci:
Berat kelinci ke-1: 1
Berat kelinci ke-2: 2
Berat kelinci ke-3: 4
Berat kelinci ke-4: 2

Hasil analisis berat kelinci:
Berat terkecil: 1.00 kg
Berat terbesar: 4.00 kg

```

Deskripsi Program :

Program ini membaca jumlah dan berat badan beberapa kelinci, lalu menganalisis berat terkecil dan terbesar di antaranya. Pengguna diminta untuk memasukkan jumlah kelinci dan berat masing-masing. Program kemudian mencari nilai minimum dan maksimum dari berat tersebut dan menampilkan hasilnya dengan format dua desimal.

2. 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 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 beratIkan []float64

    var jumlahIkan int
    fmt.Print("Masukkan jumlah ikan (x): ")
    fmt.Scan(&jumlahIkan)
```

```

var kapasitasWadah int
fmt.Print("Masukkan kapasitas wadah (y): ")
fmt.Scan(&kapasitasWadah)

if jumlahIkan <= 0 || kapasitasWadah <= 0 {
    fmt.Println("Jumlah ikan dan kapasitas
wadah harus lebih dari 0.")
    return
}

beratIkan = make([]float64, jumlahIkan)
fmt.Println("Masukkan berat ikan:")
for i := 0; i < jumlahIkan; i++ {
    fmt.Printf("Berat ikan ke-%d: ", i+1)
    fmt.Scan(&beratIkan[i])
}

jumlahWadah := (jumlahIkan + kapasitasWadah
- 1) / kapasitasWadah

beratWadah := make([]float64, jumlahWadah)
for i := 0; i < jumlahIkan; i++ {
    wadahKe := i / kapasitasWadah
    beratWadah[wadahKe] += beratIkan[i]
}

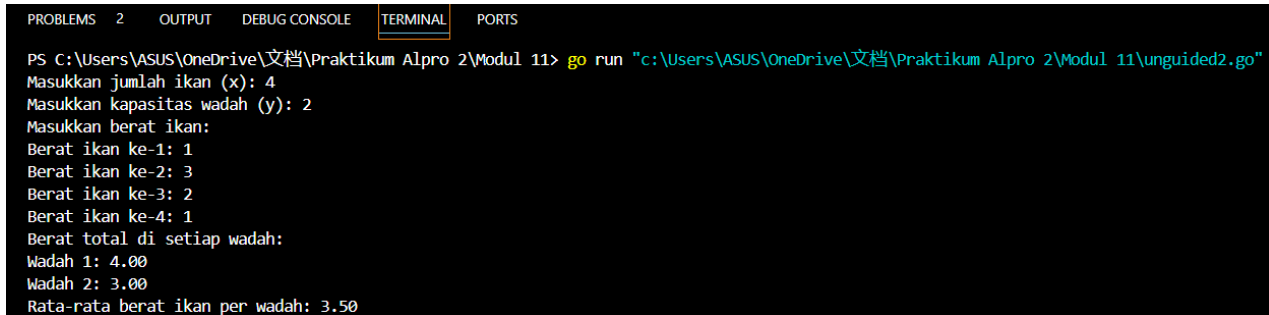
fmt.Println("Berat total di setiap wadah:")
for i, berat := range beratWadah {
    fmt.Printf("Wadah %d: %.2f\n", i+1,
berat)
}

var totalBerat float64
for _, berat := range beratWadah {
    totalBerat += berat
}

fmt.Printf("Rata-rata berat ikan per wadah:
%.2f\n", totalBerat/float64(jumlahWadah))
}

```

Screenshoot Output

A screenshot of a Go IDE's terminal window. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active. The command prompt shows the user running a Go program. The output of the program is displayed below the command prompt.

```
PS C:\Users\ASUS\OneDrive\文档\Praktikum Alpro 2\Modul 11> go run "c:\Users\ASUS\OneDrive\文档\Praktikum Alpro 2\Modul 11\unguided2.go"
Masukkan jumlah ikan (x): 4
Masukkan kapasitas wadah (y): 2
Masukkan berat ikan:
Berat ikan ke-1: 1
Berat ikan ke-2: 3
Berat ikan ke-3: 2
Berat ikan ke-4: 1
Berat total di setiap wadah:
Wadah 1: 4.00
Wadah 2: 3.00
Rata-rata berat ikan per wadah: 3.50
```

Deskripsi Program :

Program ini menghitung berat total ikan di setiap wadah dan rata-rata berat ikan per wadah. Pengguna memasukkan jumlah ikan, kapasitas wadah, dan berat masing-masing ikan. Program kemudian mendistribusikan ikan ke beberapa wadah berdasarkan kapasitasnya, menghitung berat total setiap wadah, dan menampilkan rata-rata berat per wadah.

3. 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. 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"
    "math"
)

func hitungStatistik(arrBerat []float64)
(float64, float64, float64) {
    min := math.MaxFloat64
    max := -math.MaxFloat64
    total := 0.0

    for _, berat := range arrBerat {
        if berat < min {
            min = berat
        }
        if berat > max {
            max = berat
        }
        total += berat
    }

    rerata := total / float64(len(arrBerat))
    return min, max, rerata
}

func main() {
    var jumlahData int

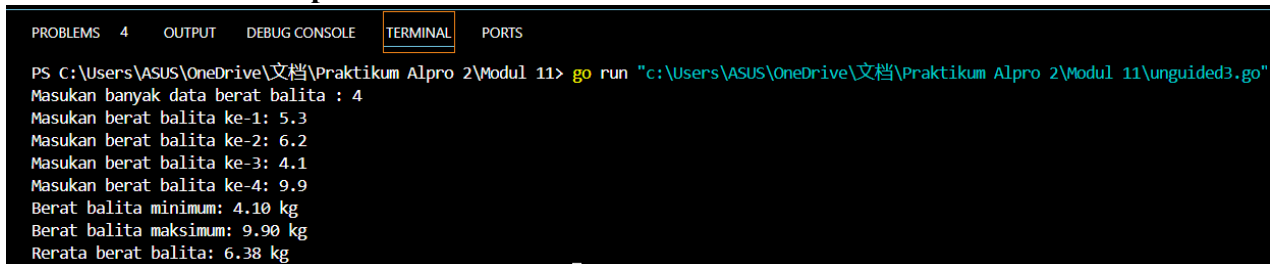
    fmt.Print("Masukan banyak data berat balita
: ")
    fmt.Scan(&jumlahData)

    beratBalita := make([]float64, jumlahData)

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

```
        beratMin, beratMax, rerataBalita :=  
hitungStatistik(beratBalita)  
  
        fmt.Printf("Berat balita minimum: %.2f  
kg\n", beratMin)  
        fmt.Printf("Berat balita maksimum: %.2f  
kg\n", beratMax)  
        fmt.Printf("Rerata berat balita: %.2f kg\n",  
rerataBalita)  
    }
```

Screenshot Output



The screenshot shows a terminal window with the following output:

```
PS C:\Users\ASUS\OneDrive\文档\Praktikum Alpro 2\Modul 11> go run "c:\Users\ASUS\OneDrive\文档\Praktikum Alpro 2\Modul 11\unguided3.go"  
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
```

Deskripsi Program :

Program ini, ditulis dalam bahasa Go, menghitung statistik berat balita (minimum, maksimum, dan rata-rata). Pengguna memasukkan jumlah data dan berat balita satu per satu.

Program menggunakan fungsi `hitungMinMax` untuk mencari berat minimum dan maksimum dengan pointer, serta fungsi `rerata` untuk menghitung rata-rata. Hasil akhirnya ditampilkan dalam format berat minimum, maksimum, dan rata-rata.

III. KESIMPULAN

Pencarian nilai ekstrem, seperti nilai maksimum dan minimum dalam sebuah dataset, dapat dilakukan dengan cara sederhana menggunakan array dasar. Namun, untuk meningkatkan efisiensi, kita juga bisa memanfaatkan struktur data khusus. Pemilihan metode yang tepat tergantung pada kebutuhan dan karakteristik dataset yang dianalisis.

Jika dataset kecil dan tidak memerlukan pengolahan kompleks, array dasar mungkin sudah memadai. Namun, untuk dataset yang lebih besar atau ketika performa sangat penting, struktur data canggih seperti pohon biner atau heap dapat memberikan keunggulan signifikan. Struktur data ini tidak hanya mempercepat pencarian nilai ekstrem, tetapi juga memfasilitasi operasi lain, seperti penyisipan dan penghapusan data, dengan lebih efisien.

Oleh karena itu, penting bagi pengembang dan analis data untuk mempertimbangkan ukuran dan sifat dataset, serta tujuan analisis, saat memilih metode pencarian nilai ekstrem. Dengan pendekatan yang tepat, kita dapat mengoptimalkan kinerja aplikasi dan mendapatkan hasil analisis yang lebih cepat dan akurat.

IV. REFERENSI

[1] Modul 11 Praktikum Algoritma 2