

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL XI

PENCARIAN NILAI EXTREME PADA HIMPUNAN DATA



Disusun Oleh :

Muhammad Hamzah Haifan Ma'ruf

2311102091

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 pada himpunan data bertujuan untuk menemukan nilai terkecil (minimum) dan terbesar (maksimum) dari kumpulan data. Nilai minimum menunjukkan batas bawah, sedangkan nilai maksimum menunjukkan batas atas dari data. Dalam berbagai bidang, seperti statistik, ilmu data, dan ekonomi, pencarian nilai ekstrem membantu memahami sebaran data dan batas-batasnya.

Metode yang umum digunakan untuk pencarian nilai ekstrem adalah iterasi linear (one-pass scan), di mana seluruh elemen data diiterasi satu per satu. Metode ini cocok untuk himpunan data dengan ukuran kecil atau tetap karena efisien dan sederhana. Pada metode ini, elemen pertama diasumsikan sebagai nilai minimum dan maksimum awal. Kemudian, setiap elemen berikutnya dibandingkan dengan nilai minimum dan maksimum saat ini, dan jika ditemukan nilai yang lebih kecil atau lebih besar, maka nilai tersebut diperbarui. Metode ini memiliki kompleksitas waktu $O(n)$, yang optimal untuk pencarian minimum dan maksimum dalam array berukuran n .

Dalam bahasa Go, pencarian nilai ekstrem dapat diimplementasikan menggunakan fungsi yang menerima array sebagai input dan mengembalikan nilai minimum dan maksimum. Misalnya, fungsi `findMinMax` dalam Go menerima array `float64`, kemudian menggunakan iterasi linear untuk menentukan nilai minimum dan maksimum array tersebut. Go memiliki beberapa keunggulan untuk implementasi algoritma ini, yaitu efisiensi memori dan konsistensi kinerja, sehingga cocok untuk bekerja dengan data dalam jumlah besar. Selain itu, sintaks Go yang sederhana memungkinkan penulisan kode yang ringkas dan mudah dipahami, mengurangi kemungkinan kesalahan dalam pemrograman.

Pencarian nilai ekstrem ini juga memiliki banyak aplikasi praktis, misalnya dalam analisis data untuk mengidentifikasi rentang atau pola, dalam pemantauan kinerja sistem untuk menentukan batas penggunaan sumber daya, dan dalam pengendalian mutu di industri untuk memastikan bahwa parameter produk tetap dalam rentang yang dapat diterima. Dengan memahami konsep, metode, dan implementasi pencarian nilai ekstrem, kita dapat memperoleh wawasan lebih lanjut dari data dan menerapkannya dalam berbagai konteks aplikasi.

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 n int
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&n)

    if n > 1000 {
        fmt.Println("Jumlah anak kelinci tidak boleh lebih dari 1000")
        return
    }

    var berat float64
    minBerat := math.MaxFloat64
    maxBerat := -math.MaxFloat64

    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan berat anak kelinci ke-%d: ",
i+1)
        fmt.Scan(&berat)

        if berat < minBerat {
            minBerat = berat
        }
        if berat > maxBerat {
            maxBerat = berat
        }
    }
```

```

    }

    fmt.Printf("Berat terkecil: %.2f\n", minBerat)
    fmt.Printf("Berat terbesar: %.2f\n", maxBerat)
}

```

Screenshoot Output

The screenshot shows a Go IDE with the following code in `Ungu1.go`:

```

1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     var n int
10    fmt.Print("Masukkan jumlah anak kelinci: ")
11    fmt.Scan(&n)
12
13    if n > 1000 {
14        fmt.Println("Jumlah anak kelinci tidak boleh lebih dari 1000")
15    }
16 }

```

The terminal output shows the program execution:

```

PS D:\Semester 3\Praktikum Alpro 2\Modul 11> go run "d:\Semester 3\Praktikum Alpro 2\Modul 11\Ungu1.go"
Masukkan jumlah anak kelinci: 3
Masukkan berat anak kelinci ke-1: 5
Masukkan berat anak kelinci ke-2: 4
Masukkan berat anak kelinci ke-3: 6
Berat terkecil: 4.00
Berat terbesar: 6.00
PS D:\Semester 3\Praktikum Alpro 2\Modul 11>

```

Deskripsi Program

Program ini bertujuan untuk mencari berat minimum dan maksimum dari sejumlah anak kelinci yang akan dijual. Program dimulai dengan meminta pengguna memasukkan jumlah anak kelinci yang ingin didata. Jika jumlah anak kelinci melebihi 1000, program akan menghentikan proses dan menampilkan pesan bahwa jumlah tersebut tidak boleh lebih dari 1000. Setelah jumlah yang valid diterima, program meminta pengguna untuk memasukkan berat setiap anak kelinci satu per satu.

Untuk menentukan nilai ekstrem, program menggunakan dua variabel, yaitu `minBerat` yang diinisialisasi dengan nilai maksimum `math.MaxFloat64` dan `maxBerat` yang diinisialisasi dengan nilai minimum `-math.MaxFloat64`. Setiap kali berat anak kelinci dimasukkan, program akan memeriksa apakah berat tersebut lebih kecil dari `minBerat` atau lebih besar dari `maxBerat`. Jika iya, program akan memperbarui nilai `minBerat` atau `maxBerat` sesuai dengan kondisi tersebut. Setelah semua data berat anak kelinci dimasukkan, program akan menampilkan berat terkecil dan terbesar dalam format desimal dengan dua angka di belakang koma.

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
    fmt.Print("Masukkan jumlah ikan yang akan dijual (x)
dan jumlah ikan per wadah (y): ")
    fmt.Scan(&x, &y)

    if x > 1000 {
        fmt.Println("Jumlah ikan tidak boleh lebih dari
1000")
        return
    }

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

    wadah := (x + y - 1) / y
    totalBeratPerWadah := make([]float64, wadah)
    for i := 0; i < x; i++ {
        totalBeratPerWadah[i/y] += ikan[i]
    }
}
```

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

var totalBerat float64
for _, total := range totalBeratPerWadah {
    totalBerat += total
}

rataRata := totalBerat / float64(wadah)
fmt.Printf("Berat rata-rata ikan di setiap wadah:
%.2f\n", rataRata)
}
```

Screenshot Output

The screenshot shows a Go IDE with a file named 'Modul 11'. The code is as follows:

```

7 func main() {
19     for i := 0; i < X; i++ {
20         fmt.Printf("Berat ikan ke- %d: ", i+1)
21         fmt.Scan(&ikan[i])
22     }
23
24     wadah := (x + y - 1) / y
25     totalBeratPerWadah := make([]float64, wadah)
26     for i := 0; i < X; i++ {
27         totalBeratPerWadah[i/y] += ikan[i]
28     }
29
30     fmt.Println("Total berat di setiap wadah:")
31     for i, total := range totalBeratPerWadah {
32         fmt.Printf("Wadah %d: %.2f\n", i+1, total)
33     }

```

The output in the terminal is:

```

Masukkan jumlah ikan yang akan dijual (x) dan jumlah ikan per wadah (y): 2 2
Masukkan berat setiap ikan:
Berat ikan ke-1: 5
Berat ikan ke-2: 3
Total berat di setiap wadah:
Wadah 1: 8.00
Berat rata-rata ikan di setiap wadah: 8.00
PS D:\Semester 3\Praktikum Alpro 2\Modul 11>

```

Deskripsi Program

Program ini bertujuan untuk menghitung total berat ikan di setiap wadah dan rata-rata berat ikan per wadah berdasarkan jumlah ikan dan jumlah ikan yang dapat dimuat di setiap wadah. Program dimulai dengan meminta pengguna untuk memasukkan jumlah ikan yang akan dijual (x) dan kapasitas maksimal ikan per wadah (y). Jika jumlah ikan lebih dari 1000, program akan menampilkan pesan bahwa jumlah ikan tidak boleh melebihi batas tersebut dan menghentikan proses.

Selanjutnya, program membuat array ikan untuk menyimpan berat masing-masing ikan dan meminta pengguna memasukkan berat tiap ikan satu per satu. Setelah itu, program menghitung jumlah wadah yang diperlukan berdasarkan jumlah ikan dan kapasitas per wadah, lalu membuat array

totalBeratPerWadah untuk menyimpan total berat ikan di setiap wadah. Program kemudian mendistribusikan ikan ke dalam wadah dengan menambahkan berat setiap ikan ke wadah yang sesuai, berdasarkan indeks wadah yang dihitung menggunakan pembagian indeks ikan dengan kapasitas y.

Setelah proses distribusi selesai, program menampilkan total berat ikan di setiap wadah. Terakhir, program menghitung total keseluruhan berat ikan dan rata-rata berat ikan per wadah, lalu menampilkan hasil rata-rata tersebut dalam format desimal dengan dua angka di belakang koma. Program ini membantu menghitung berat dan mendistribusikan ikan secara merata ke dalam beberapa wadah sesuai kapasitas yang diinginkan.

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"
)

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 rata(arrBerat arrBalita, n int) float64 {
    var total float64
    for i := 0; i < n; i++ {
        total += arrBerat[i]
    }
    return total / float64(n)
}

func main() {
    var n int
    var arrBerat arrBalita
    var bMin, bMax 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(&arrBerat[i])
    }

    hitungMinMax(arrBerat, n, &bMin, &bMax)

    rataRata := rata(arrBerat, n)

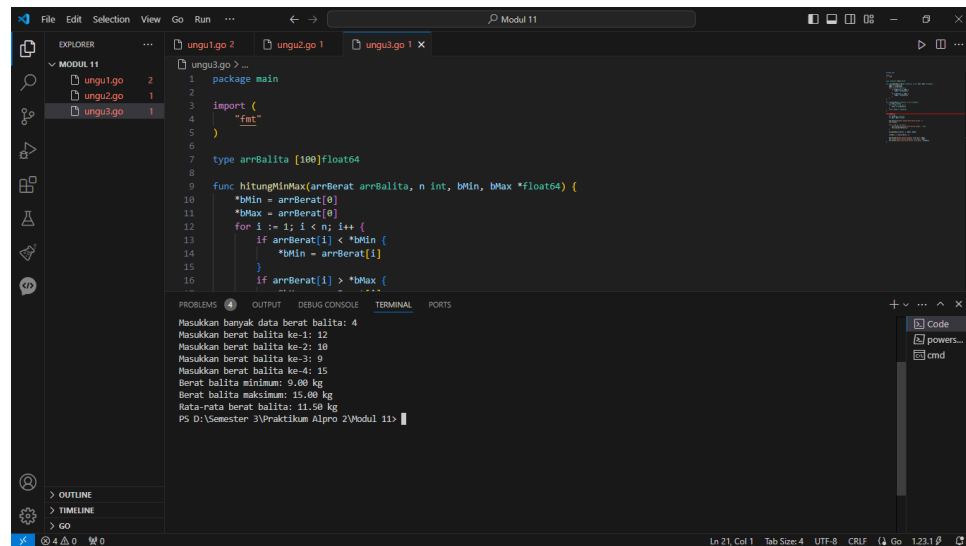
    fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)
    fmt.Printf("Rata-rata berat balita: %.2f kg\n",
rataRata)
}

```



```
}
```

Screenshoot Output



```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 type arrBalita [100]float64
8
9 func hitungMinMax(arrBerat arrBalita, n int, bMin, bMax *float64) {
10     *bMin = arrBerat[0]
11     *bMax = arrBerat[0]
12     for i := 1; i < n; i++ {
13         if arrBerat[i] < *bMin {
14             *bMin = arrBerat[i]
15         }
16         if arrBerat[i] > *bMax {
17             *bMax = arrBerat[i]
18         }
19     }
20 }
21
22 func rataRata(arrBerat arrBalita, n int) float64 {
23     sum := 0.0
24     for i := 0; i < n; i++ {
25         sum += arrBerat[i]
26     }
27     return sum / float64(n)
28 }
29
30 func main() {
31     n := 4
32     arrBerat := arrBalita{}
33     for i := 0; i < n; i++ {
34         fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
35         var b float64
36         fmt.Scanln(&b)
37         arrBerat[i] = b
38     }
39     bMin := 0.0
40     bMax := 0.0
41     hitungMinMax(arrBerat, n, &bMin, &bMax)
42     rata := rataRata(arrBerat, n)
43     fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)
44     fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)
45     fmt.Printf("Rata-rata berat balita: %.2f kg\n", rata)
46 }
```

Masukkan banyak data berat balita: 4
Masukkan berat balita ke-1: 12
Masukkan berat balita ke-2: 10
Masukkan berat balita ke-3: 9
Masukkan berat balita ke-4: 15
Berat balita minimum: 9.00 kg
Berat balita maksimum: 15.00 kg
Rata-rata berat balita: 11.50 kg
PS D:\Semester 3\Praktikum Alpro 2\Modul 11>

Deskripsi Program

Program ini dirancang untuk mengelola data berat badan balita di sebuah pos pelayanan kesehatan. Program ini memungkinkan pengguna untuk memasukkan sejumlah data berat badan balita, kemudian menghitung berat minimum, berat maksimum, dan rata-rata dari data yang dimasukkan. Pertama, program meminta pengguna memasukkan jumlah data berat badan balita yang akan dicatat. Kemudian, pengguna diminta untuk memasukkan berat masing-masing balita satu per satu. Data berat ini disimpan dalam array `arrBalita` dengan kapasitas maksimal 100 elemen.

Program memiliki dua fungsi utama, yaitu `hitungMinMax` dan `rata`. Fungsi `hitungMinMax` menerima array `arrBalita`, jumlah data `n`, serta dua pointer `bMin` dan `bMax` untuk menyimpan nilai minimum dan maksimum. Fungsi ini membandingkan setiap elemen dalam array untuk menemukan nilai terkecil dan terbesar, kemudian menyimpannya di `bMin` dan `bMax`. Fungsi kedua, `rata`, menghitung rata-rata berat badan balita dengan menjumlahkan semua data dalam array `arrBalita` dan membaginya dengan jumlah data `n`.

Setelah fungsi-fungsi tersebut dijalankan, program menampilkan hasil berupa berat minimum, berat maksimum, dan rata-rata berat balita dalam format desimal dengan dua angka di belakang koma. Program ini membantu dalam pemantauan berat badan balita dan memberikan informasi dasar terkait rentang dan distribusi data berat badan di pos pelayanan kesehatan.

III. KESIMPULAN

Kesimpulannya, pencarian nilai ekstrem dan perhitungan rata-rata dalam himpunan data adalah konsep penting dalam pengelolaan data yang digunakan di berbagai bidang, termasuk dalam pemantauan kesehatan, analisis data, dan pengambilan keputusan. Melalui implementasi program berbasis bahasa Go, kita dapat dengan mudah menemukan nilai minimum, maksimum, dan rata-rata dari sekumpulan data, seperti data berat badan anak kelinci, ikan, atau balita. Metode pencarian ekstrem yang menggunakan iterasi satu per satu efisien untuk menemukan nilai terkecil dan terbesar, sedangkan perhitungan rata-rata memberikan gambaran umum tentang sebaran data.

Bahasa Go menyediakan sintaks sederhana dan efisien untuk bekerja dengan array dan operasi matematika dasar, yang mempermudah pemrograman algoritma pencarian dan penghitungan statistik dasar. Penggunaan array dan variabel pointer pada Go juga memberikan fleksibilitas dalam memproses dan memanipulasi data dalam jumlah besar. Dengan pemahaman ini, kita dapat mengaplikasikan teknik-teknik tersebut dalam berbagai skenario untuk mendapatkan informasi yang relevan dan bermanfaat dari data yang dianalisis.

IV. REFERENSI

- [1] Modul 11 Praktikum Algoritma dan Pemrograman 2
- [2] GeeksforGeeks. (n.d.). *GeeksforGeeks*. <https://www.geeksforgeeks.org/>