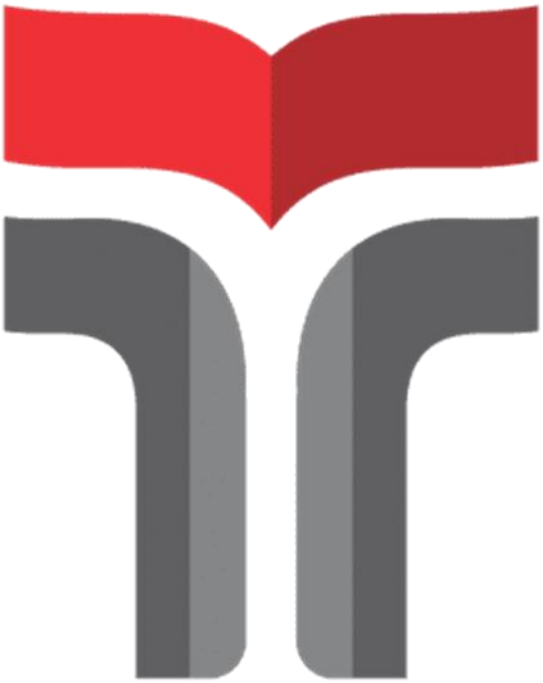
LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL 10

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Oleh:

TSAQIF KANZ AHMAD
2311102075
IF-11-02

S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024

I. DASAR TEORI

Pencarian nilai ekstrem (baik nilai minimum maupun maksimum) pada suatu himpunan data adalah konsep dasar yang penting, terutama ketika bekerja dengan data numerik dalam konteks statistik, data science, ataupun aplikasi lain yang membutuhkan analisis data. Berikut adalah dasar teori mengenai pencarian nilai ekstrem pada himpunan data:

1. Konsep Nilai Ekstrem

- Nilai Minimum: Nilai minimum dalam suatu himpunan data adalah elemen yang memiliki nilai terkecil di antara elemen-elemen lain dalam himpunan tersebut.

- Nilai Maksimum: Nilai maksimum adalah elemen yang memiliki nilai terbesar di antara elemen-elemen lain.

Mencari nilai ekstrem ini sering digunakan dalam berbagai aplikasi, misalnya untuk mengetahui batas bawah atau atas dari data, memfilter nilai ekstrem untuk analisis outlier, atau dalam algoritma yang membutuhkan penentuan batas data.

2. Pendekatan Algoritmik dalam Pencarian Nilai Ekstrem

Algoritma untuk mencari nilai ekstrem dalam sebuah himpunan data linear (seperti array atau slice di Golang) cukup sederhana:

- Inisialisasi nilai awal sebagai nilai ekstrem sementara (misalnya elemen pertama).
- Iterasi melalui seluruh elemen dalam himpunan data.
- Bandingkan setiap elemen dengan nilai ekstrem sementara:
  - Jika mencari nilai minimum, perbarui nilai ekstrem sementara jika elemen saat ini lebih kecil.
  - Jika mencari nilai maksimum, perbarui nilai ekstrem sementara jika elemen saat ini lebih besar.
- Setelah iterasi selesai, nilai ekstrem sementara tersebut adalah nilai minimum atau maksimum dalam himpunan data.

Kompleksitas algoritma ini adalah *O(n),* di mana n adalah jumlah elemen dalam himpunan data, karena setiap elemen hanya perlu diperiksa sekali.

3. Pencarian Nilai Ekstrem pada Array dengan Tipe Data Dasar

Pada pencarian nilai ekstrem pada array tipe dasar dengan tipe data dasar mengandung elemen-elemen dengan jenis data primitif seperti integer, float, string, atau karakter. Pada jenis data ini, pencarian nilai ekstrem lebih sederhana karena setiap elemen array memiliki tipe data yang langsung dapat dibandingkan.

- **Algoritma Pencarian Nilai Minimum dan Maksimum pada array tipe dasar**

  Proses pencarian nilai minimum atau maksimum pada array dengan tipe data dasar dapat dilakukan dengan langkah-langkah berikut:

  1. Inisialisasi variabel untuk menyimpan nilai ekstrem (minimum atau maksimum), biasanya dengan nilai dari elemen pertama array.

2. Iterasi melalui seluruh elemen array.
3. Bandingkan setiap elemen dengan nilai ekstrem yang tersimpan:
   - Jika mencari nilai minimum, perbarui nilai ekstrem jika elemen saat ini lebih kecil dari nilai ekstrem sementara.
   - Jika mencari nilai maksimum, perbarui nilai ekstrem jika elemen saat ini lebih besar dari nilai ekstrem sementara.
4. Setelah iterasi selesai, variabel nilai ekstrem menyimpan nilai minimum atau maksimum dari array tersebut.

Contoh program Golang untuk mencari nilai minimum dan maksimum pada array bertipe integer:

```
package main

import (
    "fmt"
)

func findMin(arr []int) int {
    min := arr[0]
    for _, value := range arr {
        if value < min {
            min = value
        }
    }
    return min
}

func findMax(arr []int) int {
    max := arr[0]
    for _, value := range arr {
        if value > max {
            max = value
        }
    }
    return max
}

func main() {
    arr := []int{10, 20, 5, 3, 25, 15}
    fmt.Println("Nilai Minimum:", findMin(arr))
    fmt.Println("Nilai Maksimum:", findMax(arr))
}
```

Pada contoh diatas fungsi **findMin** mengiterasi elemen array untuk mencari nilai terkecil dan mengembalikannya sementara fungsi **findMax** sama seperti **findMin** tetapi untuk mencari nilai terbesar serta mengembalikannya.

4. Pencarian Nilai Ekstrem pada Array dengan Tipe Data Struktur

Pada pencarian nilai ekstrim pada array yang bertipe data struktur (struct) adalah array yang elemennya terdiri dari sekumpulan nilai dengan beberapa atribut atau field. Setiap elemen struktur bisa memiliki beberapa jenis data yang berbeda. Oleh karena itu, pencarian nilai ekstrem pada array struct memerlukan sedikit penyesuaian: kita perlu menentukan field mana yang menjadi basis perbandingan.

- Algoritma Pencarian Nilai Minimum dan Maksimum pada Array Struct

Langkah-langkahnya mirip dengan tipe data dasar, tetapi dengan fokus pada field tertentu:

1. Tentukan field dalam struct yang akan digunakan sebagai dasar perbandingan.
2. Inisialisasi variabel untuk menyimpan nilai ekstrem berdasarkan nilai pada field tersebut dari elemen pertama array.
3. Iterasi melalui elemen-elemen array, bandingkan nilai dari field yang dipilih di setiap elemen dengan nilai ekstrem sementara:
   - Perbarui nilai ekstrem jika nilai pada field saat ini lebih kecil (untuk pencarian minimum) atau lebih besar (untuk pencarian maksimum) dari nilai ekstrem sementara.
4. Setelah iterasi selesai, variabel nilai ekstrem berisi nilai minimum atau maksimum dari array berdasarkan field yang dipilih.

Contoh Implementasi golang untuk mencari nilai minimum dan maksimum dari sebuah array yang bertipe struct:

```
package main

import (
    "fmt"
)

type Product struct {
    Name  string
    Price float64
}

func findMinPrice(products []Product) Product {
    minProduct := products[0]
    for _, product := range products {
        if product.Price < minProduct.Price {
            minProduct = product
        }
    }
    return minProduct
}

func findMaxPrice(products []Product) Product {
    maxProduct := products[0]
```

```
    for _, product := range products {
        if product.Price > maxProduct.Price {
            maxProduct = product
        }
    }
    return maxProduct
}

func main() {
    products := []Product{
        {"Product A", 100.0},
        {"Product B", 75.5},
        {"Product C", 150.0},
        {"Product D", 90.0},
    }

    minProduct := findMinPrice(products)
    maxProduct := findMaxPrice(products)

    fmt.Println("Produk dengan harga minimum:", minProduct.Name,
"-", minProduct.Price)
    fmt.Println("Produk dengan harga maksimum:",
maxProduct.Name, "-", maxProduct.Price)
}
```

Pada contoh diatas terdapat **struct product** memiliki dua field yaitu **name** (nama produk) dan **price** (harga produk) serta Fungsi **findMinPrice** untuk mencari produk dengan harga terendah, sedangkan **findMaxPrice** untuk mencari produk dengan harga tertinggi.

5. Aplikasi Nilai Ekstrem dalam Pemrograman

Pencarian nilai ekstrem digunakan dalam:

- **Pengolahan Data**: Menentukan batas data, seperti dalam statistik deskriptif.
- **Machine Learning dan AI**: Digunakan dalam analisis data untuk normalisasi atau penghapusan outlier.
- **Optimalisasi Algoritma**: Dalam pencarian solusi optimal atau suboptimal.

Pencarian nilai ekstrem adalah langkah dasar namun krusial dalam analisis data yang lebih lanjut dalam pemrograman Golang dan aplikasi-aplikasi lainnya.

II. 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.

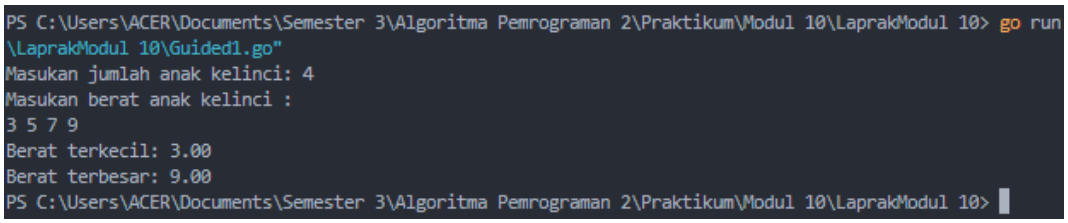
   Masukan terdiri dari sekumpulan bilangan, yang mana bilangan pertama adalah bilangan bulat N yang menyatakan banyaknya anak kelinci yang akan ditimbang beratnya. Selanjutnya N bilangan riil berikutnya adalah berat dari anak kelinci yang akan dijual.

   Keluaran terdiri dari dua buah bilangan riil yang menyatakan berat kelinci terkecil dan terbesar.

SOURCE CODE

```go
package main

import "fmt"

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

    fmt.Print("Masukan jumlah anak kelinci: ")
    fmt.Scan(&N)

    fmt.Println("Masukan 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)
}
```

SCREENSHOT OUTPUT :



PENJELASAN PROGRAM :

Program tersebut adalah program untuk mencari berat terkecil dan terbesar dari sejumlah anak kelinci berdasarkan data berat yang dimasukkan oleh pengguna. Pertama program meminta input jumlah anak kelinci (N) dan berat masing-masing kelinci yang tersimpan dalam array berat. Lalu program menginisialisasi nilai minimum dan maksimum yang ada jika ditemukan berat yang lebih kecil dari minimum atau lebih besar dari maksimum, nilai tersebut akan diperbarui. Pada akhirnya program menampilkan berat terkecil dan terbesar dari seluruh anak kelinci dengan format dua angka desimal.

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

```go
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.Println("total berat tiap wadah: ")
    for _, total := range totalBeratWadah {
        fmt.Printf("%.2f", total)
```

```
    }
    fmt.Println()

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

SCREENSHOT OUTPUT :

```
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Modul 10\LaprakModul 10> go run
\LaprakModul 10\Guided2.go"
masukkan jumlah ikan dan kapasitas wadah: 5
2 3 1 1 1
masukkan berat tiap ikan:
4 2 3 1 2
total berat tiap wadah:
4.002.004.00
rata rata berat tiap wadah:
2.001.002.00
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Modul 10\LaprakModul 10>
```

PENJELASAN PROGRAM :

Program tersebut adalah Program menghitung total berat ikan yang dimasukan kedalam wadah berdasarkan kapasitas wadah yang ditentukan. Pertama program meminta input untuk jumlah ikan (x) dan kapasitas wadah (y), setelah itu program akan meminta user menginput berat masing-masing ikan satu-persatu. Berat ikan ini akan disimpan dalam sebuah array yang ukurannya sesuai dengan jumlah total ikan yang dimasukan dari awal. Setelah semua data berat ikan tersimpan, program kemudian akan menghitung berapa wadah yang dibutuhkan menggunakan rumus pembagian pembulatan ke atas (x+y-1) dibagi kapasitas. Program lalu mendistribusikan ikan-ikan tersebut kedalam wadah-wadah yang ditentukan secara berurutan. Setelah itu program menampilkan total berat ikan dan rata-rata berat untuk setiap wadah dengan format dua angka desimal.

III. UNGUIDED

1. Pos Pelayanan Terpadu (posyandu) sebagai tempat kesehatan pelayanan kesehatan perlu mencatat data berat balita (dalam kg). Petugas akan memasukan data tersebut kedalam 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

```go
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 rataRata(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 berat arrBalita
    var bMin, bMax, rataan float64

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

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

    hitungMinMax(berat, n, &bMin, &bMax)
    rataan = rataRata(berat, n)
```

```
    fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)
    fmt.Printf("Rerata berat balita: %.2f kg\n", rataan)
}
```

SCREENSHOT OUTPUT

```
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Modul 10\LaprakModul 10> go run
\LaprakModul 10\Unguided1.go"
Masukan banyak data berat balita: 4
Masukan berat balita ke-1: 3.2
Masukan berat balita ke-2: 3.4
Masukan berat balita ke-3: 2.9
Masukan berat balita ke-4: 3.0
Berat balita minimum: 2.90 kg
Berat balita maksimum: 3.40 kg
Rerata berat balita: 3.12 kg
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Modul 10\LaprakModul 10>
```

PENJELASAN PROGRAM

Program ini bertujuan untuk membantu posyandu mencatat, mengolah, dan menganalisis data berat balita. Data disimpan dalam array bertipe arrBalita yang dapat menampung hingga 100 elemen float64. Dalam program ini, ada dua fungsi yaitu hitungMinMax dan rataRata. Fungsi pertama menghitung nilai minimum dan maksimum dari data berat balita dengan membandingkan setiap elemen dalam array dengan pointer untuk menyimpan hasilnya. Fungsi kedua menghitung berat balita rata-rata dengan menjumlahkan semua elemen dalam array, kemudian membaginya dengan jumlah data yang dimasukkan. Dalam fungsi main, program meminta nilai jumlah data dan berat balita, dan kemudian menggunakan kedua fungsi untuk memproses data. Hasilnya menampilkan berat minimum, maksimum, dan rata-rata, sehingga lebih mudah untuk menilai kesehatan balita.