LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 11

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Disusun Oleh :

MAULANA GHANI ROLANDA | 2311102012

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

10.1 Ide Pencarian Nilai MaxlMin

Pencarian adalah suatu proses yang lazim dilakukan di datam kehidupan sehari-hari. ContoH penggunaannya dalam kehidupan nyata sangat beragam, misalnya pencarian file di dalam directory komputer, pencarian suatu teks di dalam sebuah dokumen, pencarian buku pada rak buku, dan contoh lainnya. Pertama pada modul ini akan dipelajari salah satu algoritma pencarian nilai terkecil atau terbesar pada sekumpulan data, atau biasa disebut pencarian nilai ekstrim. 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 adalah sebagai berikut:

I) Jadikan data pertama sebagai nilai ekstrim

2) Lakukan validasi nilai ekstrim dari data kedua hingga data terakhir. Apabila nilai ekstrim tidak valid, maka update nilai ekstrims tersebut dengan data yang dicek.

3) Apabila semua data telah dicek, maka nilai ekstrim yang dimiliki adalah valid.

Berikut ini adalah notasi dalam pseudocode dan bahasa Go, misalnya untuk pencarian nilai terbesar atau maksimum:

	Notasi Algoritma	Notasi dalam bahasa Go
1	`max ← 1`	`max = 0`
2	`i ← 2`	`i = 1`
3	`while i <= n do`	`for i < n {`
4	`    if a[i] > a[max] then`	`    if a[i] > a[max] {`
5	`        max ← i`	`        max = i`
6	`    endif`	`    }`
7	`    i ← i + 1`	`    i = i + 1`
8	`endwhile`	`}`

10.2 Pencarlan Nilai Ekstrim pada Array Bet-tipe Data Dasar

Misalnya terdefinisi sebuah array of integer dengan kapasitas 2023, dan array terisi sejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut. Perhatikan potongan program dalam bahasa Go berikut ini!

```
5   type arrInt [2023]int
..  ...
15
16  func terkecil_1(tabInt arrInt, n int) int {
17  /* mengembalikan nilai terkecil yang terdapat di dalam tabInt yang berisi n
18  bilangan bulat */
19      var min int = tabInt[0]        // min berisi data pertama
20      var j int = 1                  // pencarian dimulai dari data berikutnya
21      for j < n {
22          if min > tabInt[j] {       // pengecekan apakah nilai minimum valid
23              min = tabInt[j]        // update nilai minimum dengan yang valid
24          }
25          j = j + 1
26      }
27      return min                     // returnkan nilai minimumnya
28  }
```

Potongan program di atas sedikit berbeda dengan sebelumnya karena penggunaan indeks array pada bahasa Go di mulai dari nol atau "O" seperti penjelasan pada modul 9. Selanjutnya, pada penjelasan di awal bab 3 telah disampaikan bahwa pada pencarian yang terpenting adalah posisi atau indeks dari nilai yang dicari dalam kumpulan data atau array. Oleh karena itu modifikasi pada program di atas dapat dilihat pada potongan program berikut ini!

```
..  ...
5   type arrInt [2023]int
..  ...
15
16  func terkecil_2(tabInt arrInt, n int) int {
17  /* mengembalikan indeks nilai terkecil yang terdapat di dalam tabInt yang berisi
18  n bilangan bulat */
19      var idx int = 0                // idx berisi indeks data pertama
20      var j int = 1                  // pencarian dimulai dari data berikutnya
21      for j < n {
22          if tabInt[idx] > tabInt[j] { // pengecekan apakah nilai minimum valid
23              idx = j                // update nilai minimum dengan yang valid
24          }
25          j = j + 1
26      }
27      return idx                     // returnkan indeks nilai minimumnya
28  }
```

10.3 Pencarian Nilai Ekstrim pada Array Bertipe Data Terstruktur

Pada kasus yang lebih kompleks pencarian ekstrim dapat juga dilakukan, misalnya mencari data mahasiswa dengan nilai terbesar, mencari lagu dengan durasi terlama, mencari pembalap yang memiliki catatan waktu balap tercepat, dan sebagainya. Sebagai contoh misalnya terdapat array yang digunakan untuk menyimpan data mahasiswa, kemudian terdapat fungsi IPK yang digunakan untuk mencari data mahasiswa dengan IPK tertinggi.

```
..  ...
5   type mahasiswa struct {
..      nama, nim, kelas, jurusan string
..      ipk float64
..  }
..  type arrMhs [2023]mahasiswa
..  ...
15
16  func IPK_1(T arrMhs, n int) float64 {
17  /* mengembalikan ipk terkecil yang dimiliki mahasiswa pada array T yang berisi
18  n mahasiswa */
19      var tertinggi float64 = T[0].ipk
20      var j int = 1
21      for j < n {
22          if tertinggi < T[j].ipk {
23              tertinggi = T[j].ipk
24          }
25          j = j + 1
26      }
27      return tertinggi
28  }
```

Apabila diperhatikan potongan program di atas, maka kita akan memperoleh ipk tertinggi, tetapi kita tidak memperoleh identitas mahasiswa dengan ipk tertinggi tersebut. Maka seperti penjelasan yang sudah diberikan sebelumnya, maka pencarian yang dilakukan bisa mengembalikan indeks mahasiswa dengan ipk tertinggi tersebut. Berikut ini adalah modifikasinya!

```
..  ...
5   type mahasiswa struct {
..      nama, nim, kelas, jurusan string
..      ipk float64
..  }
..  type arrMhs [2023]mahasiswa
..  ...
15
16  func IPK_2(T arrMhs, n int) int {
17  /* mengembalikan indeks mahasiswa yang memiliki ipk tertinggi pada array T yang
18  berisi n mahasiswa */
19      var idx int = 0
20      var j int = 1
21      for j < n {
22          if T[idx].ipk < T[j].ipk {
23              idx = j
24          }
25          j = j + 1
26      }
27      return idx
28  }
```

Sehingga melalui algoritma di atas, identitas mahasiswa dapat diperoleh, misalnya T[idx] .nama,T [idx] . nim, T [idx] . kelas, hingga T[idx] . jurusan.

II. GUIDED

1. Sourcecode

```
//2311102012
package main

import "fmt"

// Mendeklarasikan tipe data array arrInt dengan panjang
2023
type arrInt [2023]int

// Fungsi untuk mencari indeks elemen terkecil dalam array
func terkecil(tabInt arrInt, n int) int {
    var idx int = 0  // idx menyimpan indeks elemen terkecil
    var j int = 1
    for j < n {
        if tabInt[idx] > tabInt[j] {
            idx = j  // Simpan indeks j jika elemen di
indeks j lebih kecil
        }
        j = j + 1
    }
    return idx
}

// Fungsi main untuk menguji fungsi terkecil
func main() {
    var n int
    var tab arrInt

    // Meminta input jumlah elemen array
    fmt.Print("Masukkan jumlah elemen (maks 2023): ")
    fmt.Scan(&n)

    // Validasi input jumlah elemen
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah elemen harus antara 1 dan
2023.")
        return
    }

    // Memasukkan elemen-elemen array
    fmt.Println("Masukkan elemen-elemen array:")
    for i := 0; i < n; i++ {
```

```
        fmt.Print("Elemen ke-", i+1, ": ")
        fmt.Scan(&tab[i])
    }

    // Memanggil fungsi terkecil untuk menemukan indeks
elemen terkecil
    idxMin := terkecil(tab, n)

    // Menampilkan nilai dan indeks terkecil
    fmt.Println("Nilai terkecil dalam array adalah:",
tab[idxMin], "pada indeks:", idxMin)
}
```

Screenshoot Output

```
PS C:\olan\KULYEAH\LaPrak Alpro 2\2311102012-MaulanaGhaniRolanda-M11> go run "c:\olan\KULYEAH\LaPrak Alpro 2\2311102012-MaulanaGhaniRolanda-M11\guided1.go"
Masukkan jumlah elemen (maks 2023): 4
Masukkan elemen-elemen array:
Elemen ke-1: 1
Elemen ke-2: 2
Elemen ke-3: 3
Elemen ke-4: 4
Nilai terkecil dalam array adalah: 1 pada indeks: 0
PS C:\olan\KULYEAH\LaPrak Alpro 2\2311102012-MaulanaGhaniRolanda-M11> go run "c:\olan\KULYEAH\LaPrak Alpro 2\2311102012-MaulanaGhaniRolanda-M11\guided1.go"
Masukkan jumlah elemen (maks 2023): 4
Masukkan elemen-elemen array:
Elemen ke-1: 4
Elemen ke-2: 3
Elemen ke-3: 2
Elemen ke-4: 1
Nilai terkecil dalam array adalah: 1 pada indeks: 3
PS C:\olan\KULYEAH\LaPrak Alpro 2\2311102012-MaulanaGhaniRolanda-M11>
```

Deskripsi Program

Program ini bertujuan untuk mencari nilai terkecil dalam sebuah array beserta indeksnya. Program ditulis dalam bahasa Go dan menggunakan tipe data array untuk menyimpan elemen-elemen integer.

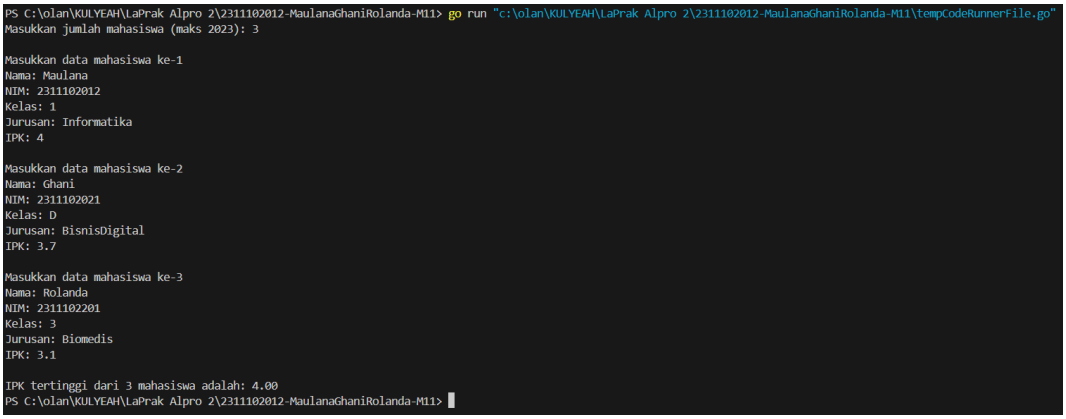
2. Source Code

```
//2311102012
package main

import "fmt"

// Definisi struct mahasiswa dengan atribut nama, nim, kelas,
jurusan, dan ipk
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk                       float64
```

```
}

// Definisi tipe data array mahasiswa dengan kapasitas maksimal 2023
type arrMhs [2023]mahasiswa

// Fungsi untuk mencari IPK tertinggi dalam array mahasiswa
func ipk(T arrMhs, n int) float64 {
    var tertinggi float64 = T[0].ipk
    var j int = 1
    for j < n {
        if tertinggi < T[j].ipk {
            tertinggi = T[j].ipk
        }
        j = j + 1
    }
    return tertinggi
}

// Fungsi main untuk mengisi data mahasiswa dan mencari IPK
tertinggi
func main() {
    var n int
    var dataMhs arrMhs

    // Meminta input jumlah mahasiswa
    fmt.Print("Masukkan jumlah mahasiswa (maks 2023): ")
    fmt.Scan(&n)

    // Validasi jumlah mahasiswa yang dimasukkan
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah mahasiswa harus antara 1 dan 2023.")
        return
    }

    // Mengisi data mahasiswa
    for i := 0; i < n; i++ {
        fmt.Printf("\nMasukkan data mahasiswa ke-%d\n", i+1)
        fmt.Print("Nama: ")
        fmt.Scan(&dataMhs[i].nama)
        fmt.Print("NIM: ")
        fmt.Scan(&dataMhs[i].nim)
        fmt.Print("Kelas: ")
        fmt.Scan(&dataMhs[i].kelas)
        fmt.Print("Jurusan: ")
        fmt.Scan(&dataMhs[i].jurusan)
```

```
        fmt.Print("IPK: ")
        fmt.Scan(&dataMhs[i].ipk)
    }

    // Mencari dan menampilkan IPK tertinggi
    tertinggi := ipk(dataMhs, n)
    fmt.Printf("\nIPK tertinggi dari %d mahasiswa adalah: %.2f\n",
n, tertinggi)


}
```

Screenshot Output



Deskripsi Program

Program ini ditulis dalam bahasa Go dengan tujuan untuk mengelola data mahasiswa, seperti nama, NIM, kelas, jurusan, dan IPK, serta mencari nilai IPK tertinggi dari sejumlah mahasiswa. Program menggunakan tipe data struct untuk merepresentasikan data mahasiswa dan array untuk menyimpan data mahasiswa dalam jumlah banyak..

III. UNGUIDED

Study Case 1

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.

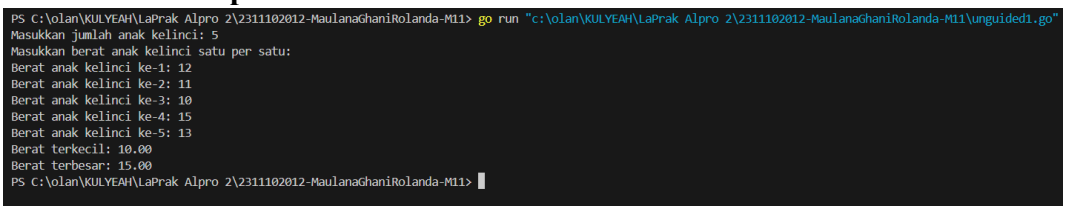
   Keluaran terdiri dari dua buah bilangan riil yang menyatakan berat kelinci terkecil dan terbesar.

Sourcecode

```go
//2311102012
package main

import (
    "fmt"
    "math"
)

func main() {
    // Deklarasi variabel
    var n int
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&n)

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

    berat := make([]float64, n) // Array untuk menampung berat anak kelinci
    fmt.Println("Masukkan berat anak kelinci satu per satu:")

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

```go
    // Inisialisasi nilai min dan max
    minBerat := math.MaxFloat64
    maxBerat := -math.MaxFloat64

    // Cari berat terkecil dan terbesar
    for _, b := range berat {
        if b < minBerat {
            minBerat = b
        }
        if b > maxBerat {
            maxBerat = b
        }
    }

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

Screenshoot Output



Deskripsi Program

Program ini ditulis dalam bahasa Go dan bertujuan untuk membaca data berat anak kelinci, lalu mencari berat terkecil dan berat terbesar dari data tersebut. Program membatasi jumlah data maksimal 1000 dan memanfaatkan array dinamis untuk menyimpan berat setiap anak kelinci.

Study Case 2

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.

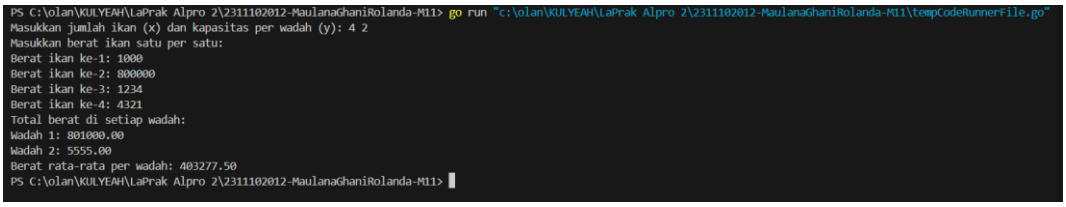
Keluaran terdiri dari dua baris. Baris pertama adalah kumpulan bilangan riil yang menyatakan total berat ikan di setiap wadah (jumlah wadah tergantung pada nilai x dan y, urutan ikan yang dimasukan ke dalam wadah sesuai urutan pada masukan baris ke-2). Baris kedua adalah sebuah bilangan riil yang menyatakan berat rata-rata ikan di setiap wadah.

Source Code

```
//2311102012
package main

import (
    "fmt"
)

func main() {
    // Deklarasi variabel
    var x, y int
    fmt.Print("Masukkan jumlah ikan (x) dan kapasitas per wadah (y):
")
    fmt.Scan(&x, &y)

    // Validasi input
    if x <= 0 || y <= 0 || x > 1000 {
```

```
        fmt.Println("Jumlah ikan atau kapasitas wadah tidak valid.")
        return
    }

    // Deklarasi array untuk berat ikan
    beratIkan := make([]float64, x)
    fmt.Println("Masukkan berat ikan satu per satu:")

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

    // Hitung jumlah wadah yang dibutuhkan
    jumlahWadah := (x + y - 1) / y // Pembulatan ke atas
    totalBeratWadah := make([]float64, jumlahWadah)

    // Distribusi ikan ke wadah
    for i := 0; i < x; i++ {
        wadahIndex := i / y
        totalBeratWadah[wadahIndex] += beratIkan[i]
    }

    // Hitung rata-rata berat per wadah
    var totalBeratSemua float64
    for _, total := range totalBeratWadah {
        totalBeratSemua += total
    }
    rataRata := totalBeratSemua / float64(jumlahWadah)

    // Output hasil
    fmt.Println("Total berat di setiap wadah:")
    for i, berat := range totalBeratWadah {
        fmt.Printf("Wadah %d: %.2f\n", i+1, berat)
    }
    fmt.Printf("Berat rata-rata per wadah: %.2f\n", rataRata)
}
```

Screenshot Output



Deskripsi Program

Program ini ditulis dalam bahasa Go dan dirancang untuk mendistribusikan ikan ke dalam beberapa wadah berdasarkan kapasitas maksimum setiap wadah. Program menghitung total berat ikan di setiap wadah dan berat rata-rata per wadah.

Study Case 3

3) 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**)

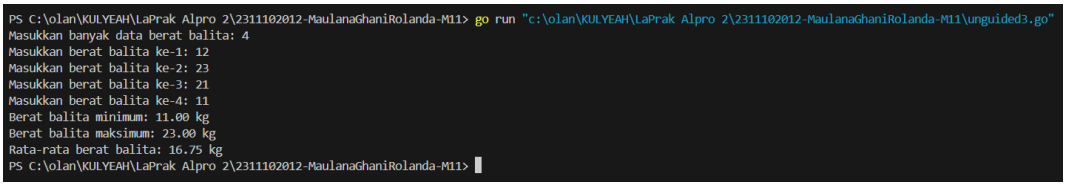
```
Masukan banyak data berat balita : 4
Masukan berat balita ke-1: 5.3
Masukan berat balita ke-2: 6.2
Masukan berat balita ke-3: 4.1
Masukan berat balita ke-4: 9.9
Berat balita minimum: 4.10 kg
Berat balita maksimum: 9.90 kg
Rerata berat balita: 6.38 kg
```

Source Code

```go
//2311102012
package main

import (
    "fmt"
    "math"
)

// Definisi array balita
type arrBalita [100]float64

// Fungsi untuk menghitung minimum dan maksimum
func hitungMinMax(arrBerat arrBalita, n int, bMin, bMax *float64) {
    *bMin = math.MaxFloat64
    *bMax = -math.MaxFloat64
    for i := 0; i < n; i++ {
        if arrBerat[i] < *bMin {
            *bMin = arrBerat[i]
        }
        if arrBerat[i] > *bMax {
            *bMax = arrBerat[i]
        }
    }
}

// Fungsi untuk menghitung rata-rata
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 arr arrBalita
    var bMin, bMax float64

    // Input jumlah data balita
    fmt.Print("Masukkan banyak data berat balita: ")
    fmt.Scan(&n)
```

```go
    // Validasi input
    if n <= 0 || n > 100 {
        fmt.Println("Jumlah data harus antara 1 dan 100.")
        return
    }

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

    // Hitung nilai minimum, maksimum, dan rata-rata
    hitungMinMax(arr, n, &bMin, &bMax)
    rataRata := rata(arr, n)

    // Output hasil
    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)
}
```

Screenshot Program



Deskripsi Program

Program ini ditulis dalam bahasa Go dan dirancang untuk mengolah data berat balita. Program membaca data berat beberapa balita, kemudian menghitung berat minimum, berat maksimum, dan berat rata-rata dari data yang diinputkan.