LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XI

NILAI EXTRIM

Disusun Oleh :
Wisnu Rananta Raditya Putra / 2311102013
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

Ide 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 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:

1) 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.

Pencarian Nilai Ekstrim pada Array Bertipe 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 Co 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  }
```

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[idxl.kelas, hingga T[idx].jurusan.

II. GUIDED

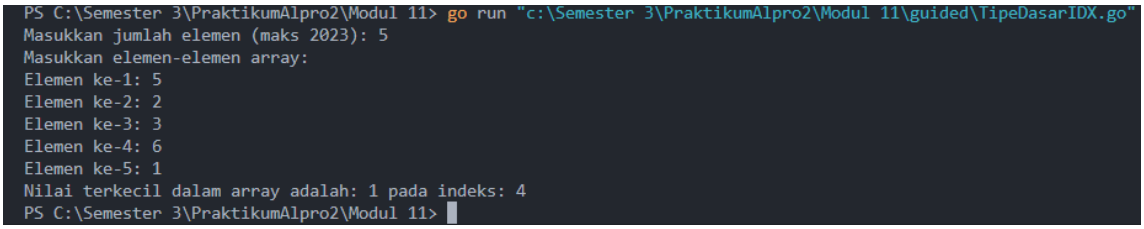
Guided 1

Source Code

```
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)
}
```

Screenshots Output:



Deskripsi:

Program ini ditulis dalam bahasa Go untuk mencari nilai terkecil dalam sebuah array sekaligus indeksnya. Tipe data arrInt digunakan untuk mendefinisikan array dengan panjang maksimum 2023. Fungsi terkecil menghitung indeks elemen terkecil dengan membandingkan elemen satu per satu. Pada fungsi main, pengguna diminta menentukan jumlah elemen (dengan validasi 1-2023) dan memasukkan nilai-nilai array. Setelah itu, program memanggil fungsi terkecil, menampilkan nilai terkecil, dan indeksnya di array.

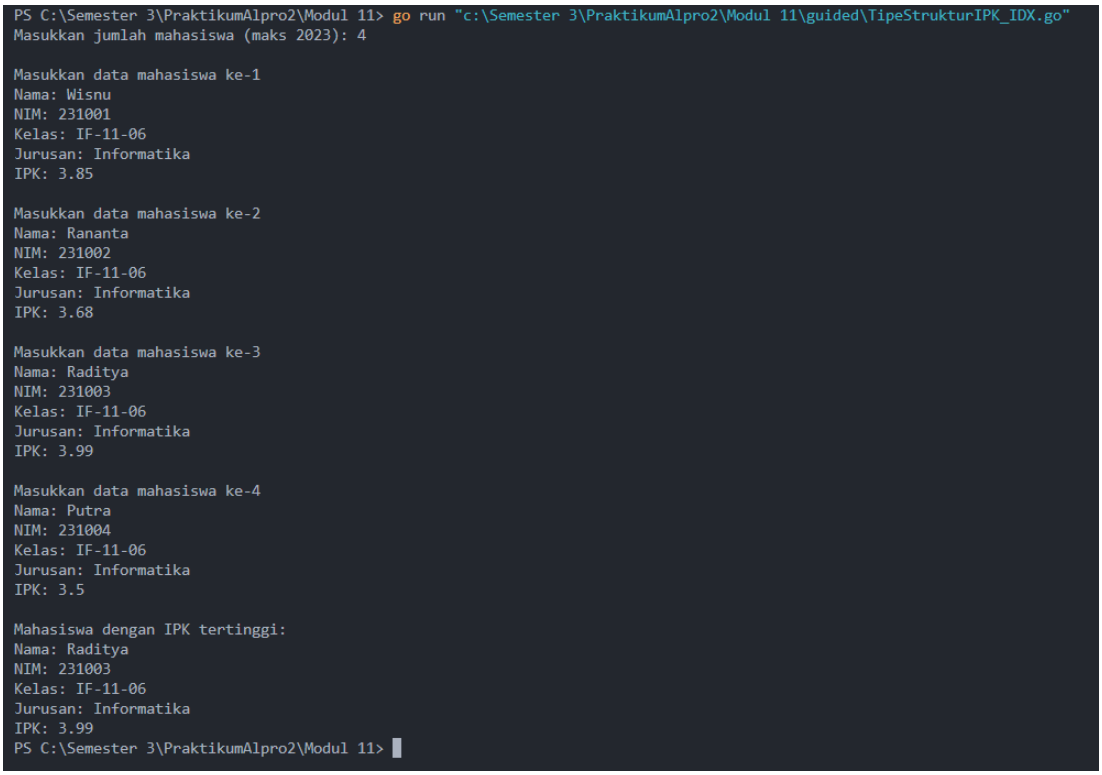
Guided 2

Source Code

```
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 indeks IPK tertinggi dalam array mahasiswa
func indeksIPKTertinggi(T arrMhs, n int) int {
    var idx int = 0  // Inisialisasi indeks IPK tertinggi pada indeks
pertama
    for j := 1; j < n; j++ {
```

```
        if T[idx].ipk < T[j].ipk {
            idx = j // Update indeks jika ditemukan IPK yang lebih tinggi
        }
    }
    return idx
}

// Fungsi main untuk mengisi data mahasiswa dan mencari IPK tertinggi
beserta indeksnya
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)
    }

    // Mendapatkan indeks IPK tertinggi
    idxTertinggi := indeksIPKTertinggi(dataMhs, n)

    // Menampilkan data mahasiswa dengan IPK tertinggi
    fmt.Printf("\nMahasiswa dengan IPK tertinggi:\n")
    fmt.Printf("Nama: %s\n", dataMhs[idxTertinggi].nama)
    fmt.Printf("NIM: %s\n", dataMhs[idxTertinggi].nim)
    fmt.Printf("Kelas: %s\n", dataMhs[idxTertinggi].kelas)
    fmt.Printf("Jurusan: %s\n", dataMhs[idxTertinggi].jurusan)
    fmt.Printf("IPK: %.2f\n", dataMhs[idxTertinggi].ipk)
}
```

Screenshots Output:



Deskripsi:

Program ini ditulis dalam bahasa Go untuk mencari mahasiswa dengan IPK tertinggi dari data yang dimasukkan. Pengguna diminta memasukkan jumlah mahasiswa (maksimal 2023) dan data setiap mahasiswa, seperti nama, NIM, kelas, jurusan, serta IPK. Setelah itu, program mencari mahasiswa dengan IPK tertinggi dan menampilkan informasi lengkapnya. Program ini mempermudah seleksi mahasiswa terbaik berdasarkan IPK.

III. UNGUIDED

Unguided 1

Study 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.

Source Code:

```
//WISNU RANANTA RADITYA PUTRA (2311102013) IF-11-06
package main

import "fmt"

func main4() {
    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
    }

    weights := make([]float64, N)
    fmt.Println("Masukkan berat anak kelinci:")
    for i := 0; i < N; i++ {
        fmt.Scan(&weights[i])
    }

    minWeight, maxWeight := weights[0], weights[0]

    for _, weight := range weights {
        if weight < minWeight {
            minWeight = weight
        }
        if weight > maxWeight {
            maxWeight = weight
        }
    }

    fmt.Printf("Berat kelinci terkecil: %.2f\n", minWeight)
    fmt.Printf("Berat kelinci terbesar: %.2f\n", maxWeight)
}
```

Screenshots Output:

```
PS C:\Semester 3\PraktikumAlpro2\Modul 11> go run "c:\Semester 3\PraktikumAlpro2\Modul 11\unguided\unguided1.go"
Masukkan jumlah anak kelinci: 4
Masukkan berat anak kelinci:
3
2
7
6
Berat kelinci terkecil: 2.00
Berat kelinci terbesar: 7.00
PS C:\Semester 3\PraktikumAlpro2\Modul 11>
```

Deskripsi:

Program ini merupakan program sederhana yang menggunakan bahasa Go untuk menghitung berat anak kelinci terkecil dan terbesar dari sejumlah data berat yang dimasukkan oleh pengguna. Pengguna diminta untuk memasukkan jumlah anak kelinci terlebih dahulu (harus antara 1 dan 1000), lalu memasukkan berat masing-masing anak kelinci satu per satu. Program akan memproses data tersebut untuk menemukan dan menampilkan berat teringan dan terberat dengan format angka desimal dua digit. Jika jumlah anak kelinci yang dimasukkan tidak sesuai, program akan memberikan pesan peringatan dan langsung berhenti.

Unguided 2

Soal Study 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.

Source Code

```go
//WISNU RANANTA RADITYA PUTRA (2311102013) IF-11-06
package main

import "fmt"

const maxCapacity = 1000

func calculateWeight(x int, y int, weights_2311102013
[maxCapacity]float64) ([]float64, float64) {
    numContainers := (x + y - 1) / y
    containerWeights := make([]float64, numContainers)

    for i := 0; i < x; i++ {
        containerIndex := i / y
        containerWeights[containerIndex] += weights_2311102013[i]
    }

    totalWeight := 0.0
    for _, weight := range containerWeights {
        totalWeight += weight
    }
```

```
    averageWeight := totalWeight / float64(numContainers)

    return containerWeights, averageWeight
}

func main() {
    var x, y int
    var weights [maxCapacity]float64

    fmt.Print("Masukkan jumlah ikan (x) dan kapasitas wadah (y): ")
    fmt.Scan(&x, &y)

    if x > maxCapacity {
        fmt.Printf("Jumlah ikan melebihi kapasitas maksimum %d\n",
maxCapacity)
        return
    }

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

    containerWeights, averageWeight := calculateWeight(x, y, weights)

    fmt.Println("\nHASIL PERHITUNGAN IKAN")
    fmt.Println("Total berat ikan di setiap wadah adalah sebagai
berikut:")
    for i, weight := range containerWeights {
        fmt.Printf("Wadah %d: %.2f kg\n", i+1, weight)
    }

    fmt.Printf("\nBerat rata-rata ikan di setiap wadah adalah %.2f kg\n",
averageWeight)
}
```

Screenshots Output:

```
Berat rata-rata ikan di setiap wadah adalah 0.00 kg
PS C:\Semester 3\PraktikumAlpro2\Modul 11> go run "c:\Semester 3\PraktikumAlpro2\Modul 11\unguided\unguided2.go"
Masukkan jumlah ikan (x) dan kapasitas wadah (y): 6 2
Masukkan berat ikan sebanyak 6:
Berat ikan ke-1: 7
Berat ikan ke-2: 3
Berat ikan ke-3: 4
Berat ikan ke-4: 5
Berat ikan ke-5: 6
Berat ikan ke-6: 8

HASIL PERHITUNGAN IKAN
Total berat ikan di setiap wadah adalah sebagai berikut:
Wadah 1: 10.00 kg
Wadah 2: 9.00 kg
Wadah 3: 14.00 kg

Berat rata-rata ikan di setiap wadah adalah 11.00 kg
PS C:\Semester 3\PraktikumAlpro2\Modul 11>
```

Deskripsi:

Program ini merupakan program sederhana yang menggunakan bahasa Go untuk membagi berat ikan ke dalam beberapa wadah berdasarkan jumlah ikan dan kapasitas wadah yang ditentukan pengguna. Pengguna diminta memasukkan jumlah ikan (*x*), kapasitas wadah (*y*), dan berat setiap ikan satu per satu. Program kemudian menghitung total berat ikan di setiap wadah serta rata-rata berat ikan per wadah. Jika jumlah ikan melebihi kapasitas maksimum yang ditentukan (1000), program akan memberikan peringatan dan berhenti. Setelah selesai, program menampilkan total berat di setiap wadah dan rata-rata beratnya dengan format dua angka desimal.

Unguided 3

Soal Study Case:

Pos Pelayanan Terpadu (posyandu) sebagai tempat pelayanan kesehatan perlu mencatat data berat balita (dalam kg). Petugas altan memasukkan data tersebut ke dalam array. Dari data yang diperoles akan dicari berat balita terbecil, terbesar, dan reratanya. Buatlah program dengan spesifikasi subprogram sebagai berikut

```go
//WISNU RANANTA RADITYA PUTRA (2311102013) IF-11-06
package main

import (
    "fmt"
)

type arrBalita [100]float64

func hitungMinMax(arr arrBalita, n int, min_2311102013, max_2311102013
*float64) {
    *min_2311102013 = arr[0]
    *max_2311102013 = arr[0]
    for i := 1; i < n; i++ {
        if arr[i] < *min_2311102013 {
            *min_2311102013 = arr[i]
        }
```

```go
        if arr[i] > *max_2311102013 {
            *max_2311102013 = arr[i]
        }
    }
}
func rataRata(arr arrBalita, n int) float64 {
    var total float64 = 0
    for i := 0; i < n; i++ {
        total += arr[i]
    }
    return total / float64(n)
}
func main() {
    var n int
    var berat arrBalita
    var min, max 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(&berat[i])
    }
    hitungMinMax(berat, n, &min, &max)
    rata := rataRata(berat, n)
    // Output hasil
    fmt.Printf("Berat balita minimum: %.2f kg\n", min)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", max)
    fmt.Printf("Rata-rata berat balita: %.2f kg\n", rata)
}
```

Screenshots Output:

```
PS C:\Semester 3\PraktikumAlpro2\Modul 11> go run "c:\Semester 3\PraktikumAlpro2\Modul 11\unguided\unguided3.go"
Masukkan banyak data berat balita: 3
Masukkan berat balita ke-1: 12
Masukkan berat balita ke-2: 20
Masukkan berat balita ke-3: 19
Berat balita minimum: 12.00 kg
Berat balita maksimum: 20.00 kg
Rata-rata berat balita: 17.00 kg
PS C:\Semester 3\PraktikumAlpro2\Modul 11>
```

Deskripsi:

Program ini merupakan program sederhana yang menggunakan bahasa Go untuk mengolah data berat balita. Pengguna diminta memasukkan jumlah data berat balita, kemudian menginput berat masing-masing balita satu per satu. Program akan menghitung berat balita terkecil (minimum), terbesar (maksimum), dan rata-rata berat balita. Semua hasil akan ditampilkan dalam format angka desimal dengan dua digit di belakang koma. Program ini membantu menganalisis data berat balita secara sederhana dan efisien.