

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL 11

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Disusun Oleh :

Aby Hakim Al Yasiry Faozi/2311102208

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pencarian nilai ekstrim adalah proses untuk menemukan nilai maksimum dan nilai minimum dalam suatu himpunan data. Proses ini penting dalam berbagai bidang, seperti analisis data, statistik, dan ilmu komputer, karena nilai ekstrim dapat memberikan informasi penting tentang batas dan rentang data.\

Definisi Nilai Ekstrim

Nilai ekstrim dalam konteks ini adalah:

- Nilai maksimum: Nilai tertinggi dalam himpunan data.
- Nilai minimum: Nilai terendah dalam himpunan data.

Manfaat Pencarian Nilai Ekstrim

1. Analisis Data: Dengan mengetahui nilai tertinggi dan terendah, kita bisa mendapatkan gambaran tentang sebaran dan variabilitas data.
2. Deteksi Anomali: Nilai ekstrim sering kali mengindikasikan adanya anomali atau outlier dalam data, yang mungkin perlu dianalisis lebih lanjut.
3. Pengambilan Keputusan: Mengetahui batas maksimum dan minimum dapat membantu dalam pengambilan keputusan yang lebih baik, misalnya dalam penentuan threshold atau batasan dalam sistem kontrol.

Metode Pencarian Nilai Ekstrim

Beberapa metode umum untuk mencari nilai ekstrim meliputi:

1. Pencarian Linear: Metode ini melibatkan iterasi melalui semua elemen dalam himpunan data untuk membandingkan dan menemukan nilai maksimum dan minimum. Ini adalah metode paling sederhana dan mudah dipahami.
2. Algoritma Pembagian dan Penaklukan: Metode ini membagi himpunan data menjadi sub-himpunan yang lebih kecil, menemukan nilai ekstrim di setiap sub-himpunan, dan kemudian menggabungkan hasilnya untuk menemukan nilai ekstrim global. Metode ini lebih efisien untuk himpunan data yang sangat besar.

3. Penggunaan Struktur Data Khusus: Struktur data seperti heap atau binary search tree dapat digunakan untuk mempercepat pencarian nilai ekstrim dalam himpunan data.

II. GUIDED

1. Soal Studi Case

Sourcecode

```
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

```

akim\Kuliah\Semester 3\prak alpro > go run "c:\Aby H
akim\Kuliah\Semester 3\prak alpro 2\modul11\guided1\
guided1.go"
Masukkan jumlah elemen (maks 2023): 2
Masukkan elemen-elemen array:
Elemen ke-1: 4
Elemen ke-2: 5
Nilai terkecil dalam array adalah: 4 pada indeks: 0
PS C:\Aby Hakim\Kuliah\Semester 3\prak alpro 2>

```

Deskripsi Program

1. Deklarasi Tipe Data Array

- Kode dimulai dengan mendeklarasikan tipe data array `arrInt` yang memiliki panjang tetap 2023. Ini berarti array ini hanya bisa menampung 2023 elemen integer.

2. Fungsi untuk Mencari Indeks Elemen Terkecil

- Sebuah fungsi bernama `terkecil` didefinisikan untuk mencari indeks elemen terkecil dalam array.
- Fungsi ini menerima dua parameter: array `tabInt` dan jumlah elemen `n`.
- Pada awalnya, `idx` diinisialisasi dengan 0, yang berarti indeks elemen pertama dianggap sebagai elemen terkecil.
- Fungsi kemudian membandingkan elemen pertama dengan elemen lainnya dalam loop. Jika ditemukan elemen yang lebih kecil, indeks `idx` diperbarui.

- Fungsi mengembalikan indeks dari elemen terkecil.

3. Fungsi main untuk Uji Coba

- Fungsi main digunakan untuk menguji fungsi terkecil.
- Pertama, jumlah elemen n diinput oleh pengguna.
- Ada validasi untuk memastikan bahwa jumlah elemen n berada dalam rentang 1 sampai 2023. Jika tidak, program mengeluarkan pesan kesalahan dan berhenti.
- Pengguna kemudian diminta memasukkan nilai-nilai elemen array satu per satu.
- Setelah semua nilai dimasukkan, fungsi terkecil dipanggil untuk menemukan indeks elemen terkecil dalam array.
- Terakhir, program menampilkan nilai terkecil dan indeksnya.

2. Soal Studi Case

Sourcecode

```
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)
    }

```

Screenshoot Output

```
akim\Kuliah\Semester 3\prak alpro 2> go run "c:\Aby
Hakim\Kuliah\Semester 3\prak alpro 2\modul11\guided2
\guided2.go"
Masukkan jumlah mahasiswa (maks 2023): 2

Masukkan data mahasiswa ke-1
Nama: aby
NIM: 209
Kelas: 06
Jurusan: informatika
IPK: 4.0

Masukkan data mahasiswa ke-2
Nama: hakim
NIM: 208
Kelas: 05
Jurusan: informatika
IPK: 3.9

IPK tertinggi dari 2 mahasiswa adalah: 4.00
PS C:\Aby Hakim\Kuliah\Semester 3\prak alpro 2> █
```

Deskripsi Program

1. Definisi Struct Mahasiswa

- Kode mendefinisikan sebuah struct bernama mahasiswa yang memiliki atribut-atribut seperti nama, nim, kelas, jurusan, dan ipk. Struct ini digunakan untuk menyimpan data terkait mahasiswa.

2. Deklarasi Array Mahasiswa

- Kode mendeklarasikan tipe data array arrMhs yang dapat menampung hingga 2023 elemen bertipe mahasiswa. Ini berarti array ini dapat menyimpan data dari maksimal 2023 mahasiswa.

3. Fungsi untuk Mencari IPK Tertinggi

- Sebuah fungsi bernama ipk didefinisikan untuk mencari nilai IPK tertinggi dalam array mahasiswa.

- Fungsi ini menerima dua parameter: array mahasiswa T dan jumlah elemen n.
- Pada awalnya, nilai IPK tertinggi diinisialisasi dengan IPK dari mahasiswa pertama dalam array.
- Fungsi kemudian membandingkan IPK dari mahasiswa pertama dengan IPK dari mahasiswa lainnya dalam loop. Jika ditemukan IPK yang lebih tinggi, nilai IPK tertinggi diperbarui.
- Fungsi mengembalikan nilai IPK tertinggi setelah selesai membandingkan semua elemen dalam array.

III. UNGUIDED

1. Soal Studi Case

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var jumlahKelinci int
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&jumlahKelinci)

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

    beratKelinci := make([]float64, jumlahKelinci)

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

    terkecil := beratKelinci[0]
    terbesar := beratKelinci[0]

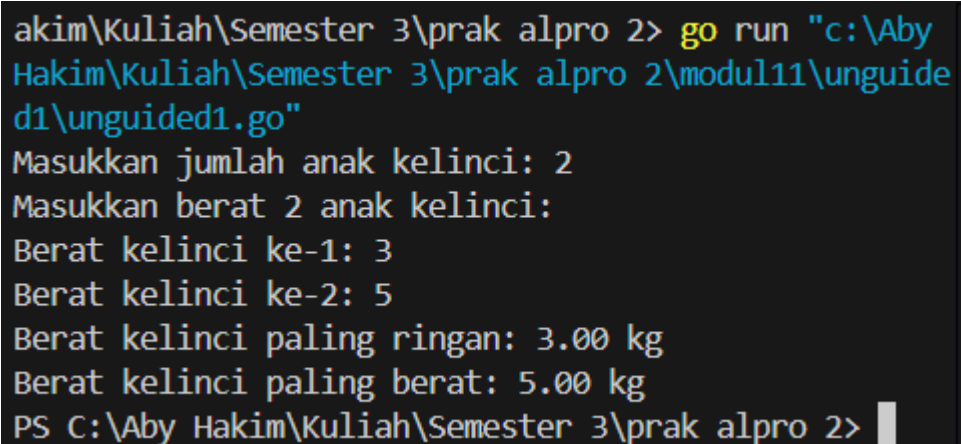
    for i := 1; i < jumlahKelinci; i++ {
        if beratKelinci[i] < terkecil {
            terkecil = beratKelinci[i]
        }
    }
}
```



```
        if beratKelinci[i] > terbesar {
            terbesar = beratKelinci[i]
        }
    }

    fmt.Printf("Berat kelinci paling ringan: %.2f kg\n",
terkecil)
    fmt.Printf("Berat kelinci paling berat: %.2f kg\n",
terbesar)
}
```

Screenshoot Output



```
akim\Kuliah\Semester 3\prak alpro 2> go run "c:\Aby
Hakim\Kuliah\Semester 3\prak alpro 2\modul11\unguide
d1\unguided1.go"
Masukkan jumlah anak kelinci: 2
Masukkan berat 2 anak kelinci:
Berat kelinci ke-1: 3
Berat kelinci ke-2: 5
Berat kelinci paling ringan: 3.00 kg
Berat kelinci paling berat: 5.00 kg
PS C:\Aby Hakim\Kuliah\Semester 3\prak alpro 2> █
```

Deskripsi Program

1. Deklarasi dan Inisialisasi

- Program dimulai dengan mendeklarasikan variabel untuk menyimpan jumlah anak kelinci.
- Pengguna diminta untuk memasukkan jumlah anak kelinci melalui input, dan program memvalidasi input tersebut untuk memastikan jumlahnya masuk akal (antara 1 dan 1000).

2. Penyimpanan Berat Kelinci

- Program membuat sebuah array untuk menyimpan berat anak-anak kelinci.
- Pengguna diminta untuk memasukkan berat masing-masing anak kelinci satu per satu melalui input.

3. Inisialisasi Nilai Terkecil dan Terbesar

- Dua variabel diinisialisasi untuk menyimpan berat kelinci terkecil dan terbesar. Pada awalnya, kedua variabel ini diatur dengan nilai berat kelinci pertama.

4. Pencarian Nilai Ekstrim

- Program kemudian melakukan iterasi melalui berat semua anak kelinci yang diinput oleh pengguna.
- Selama iterasi, berat setiap kelinci dibandingkan dengan nilai terkecil dan terbesar yang sudah ditemukan sebelumnya. Jika berat kelinci saat ini lebih kecil dari nilai terkecil yang ada, maka nilai terkecil diperbarui. Sebaliknya, jika berat kelinci saat ini lebih besar dari nilai terbesar yang ada, maka nilai terbesar diperbarui.

5. Menampilkan Hasil

- Setelah iterasi selesai, program menampilkan berat kelinci paling ringan dan berat kelinci paling berat berdasarkan data yang telah diinput oleh pengguna.

2. Soal Studi Case

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    var jumlahIkan, jumlahWadah int
    fmt.Print("Masukkan jumlah ikan dan wadah: ")
    fmt.Scan(&jumlahIkan, &jumlahWadah)

    if jumlahIkan <= 0 || jumlahIkan > 1000 ||
    jumlahWadah <= 0 || jumlahWadah > 1000 {
        fmt.Println("Input tidak valid")
        return
    }

    beratIkan := make([]float64, jumlahIkan)
    fmt.Printf("Masukkan berat dari %d ikan:\n",
    jumlahIkan)
    for i := 0; i < jumlahIkan; i++ {
        fmt.Scan(&beratIkan[i])
    }
}
```

```
beratTotalWadah := make([]float64, jumlahWadah)
jumlahIkanPerWadah := make([]int, jumlahWadah)

for i := 0; i < jumlahIkan; i++ {
    indeksWadah := i % jumlahWadah
    beratTotalWadah[indeksWadah] += beratIkan[i]
    jumlahIkanPerWadah[indeksWadah]++
}

rataRataWadah := make([]float64, jumlahWadah)
for i := 0; i < jumlahWadah; i++ {
    if jumlahIkanPerWadah[i] > 0 {
        rataRataWadah[i] = beratTotalWadah[i] /
float64(jumlahIkanPerWadah[i])
    }
}

fmt.Println("Berat total pada masing-masing wadah:")
for i := 0; i < jumlahWadah; i++ {
    fmt.Printf("Wadah %d: %.2f kg\n", i+1,
beratTotalWadah[i])
}

fmt.Println("Rata-rata berat ikan di masing-masing
wadah:")
for i := 0; i < jumlahWadah; i++ {
    fmt.Printf("Wadah %d: %.2f kg\n", i+1,
rataRataWadah[i])
}
}
```

Screenshoot Output

```
akim\Kuliah\Semester 3\prak alpro 2> go run "c:\Aby
Hakim\Kuliah\Semester 3\prak alpro 2\modul11\unguide
d2\unguided2.go"
Masukkan jumlah ikan dan wadah: 2 3
Masukkan berat dari 2 ikan:
21 34
Berat total pada masing-masing wadah:
Wadah 1: 21.00 kg
Wadah 2: 34.00 kg
Wadah 3: 0.00 kg
Rata-rata berat ikan di masing-masing wadah:
Wadah 1: 21.00 kg
Wadah 2: 34.00 kg
Wadah 3: 0.00 kg
PS C:\Aby Hakim\Kuliah\Semester 3\prak alpro 2> |
```

Deskripsi Program

1. Deklarasi dan Inisialisasi

- Program dimulai dengan meminta input dari pengguna mengenai jumlah ikan dan jumlah wadah. Input ini digunakan untuk menentukan berapa banyak ikan dan wadah yang akan dipertimbangkan dalam program.
- Validasi dilakukan untuk memastikan jumlah ikan dan jumlah wadah adalah valid, yakni tidak kurang dari 1 dan tidak lebih dari 1000. Jika input tidak valid, program akan berhenti dan menampilkan pesan kesalahan.

2. Memasukkan Berat Ikan

- Array beratIkan dibuat untuk menyimpan berat masing-masing ikan yang diinput oleh pengguna.
- Pengguna diminta untuk memasukkan berat ikan satu per satu.

3. Menghitung Berat Total dan Jumlah Ikan per Wadah

- Array beratTotalWadah dibuat untuk menyimpan total berat ikan di setiap wadah.
- Array jumlahIkanPerWadah dibuat untuk menyimpan jumlah ikan di setiap wadah.

- Program kemudian melakukan iterasi melalui semua ikan dan menambahkannya ke wadah yang sesuai berdasarkan indeks. Indeks wadah ditentukan dengan menggunakan operasi modulus (%).

4. Menghitung Rata-Rata Berat Ikan per Wadah

- Array rataRataWadah dibuat untuk menyimpan rata-rata berat ikan di setiap wadah.
- Program menghitung rata-rata berat ikan di setiap wadah dengan membagi total berat ikan dalam wadah tersebut dengan jumlah ikan yang ada dalam wadah tersebut. Rata-rata hanya dihitung jika ada ikan di dalam wadah tersebut.

5. Menampilkan Hasil

- Program menampilkan berat total ikan di setiap wadah.
- Program juga menampilkan rata-rata berat ikan di setiap wadah.

3. Soal Studi Case

Sourcecode

```
package main

import (
    "fmt"
)

type BeratBalita [100]float64

func cariEkstrim(data BeratBalita, jumlah int, terkecil,
    terbesar *float64) {
    *terkecil = data[0]
    *terbesar = data[0]

    for index := 1; index < jumlah; index++ {
        if data[index] < *terkecil {
            *terkecil = data[index]
        }
        if data[index] > *terbesar {
            *terbesar = data[index]
        }
    }
}

func hitungRataRata(data BeratBalita, jumlah int)
float64 {
    totalBerat := 0.0
```

```
        for i := 0; i < jumlah; i++ {
            totalBerat += data[i]
        }
        return totalBerat / float64(jumlah)
    }

func main() {
    var jumlahData int
    var dataBerat BeratBalita

    fmt.Print("Masukkan jumlah balita: ")
    fmt.Scan(&jumlahData)

    if jumlahData < 1 || jumlahData > 100 {
        fmt.Println("Jumlah balita harus angka")
        return
    }

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

    var beratTerkecil, beratTerbesar float64

    cariEkstrim(dataBerat, jumlahData, &beratTerkecil,
    &beratTerbesar)

    rataRata := hitungRataRata(dataBerat, jumlahData)

    fmt.Printf("Berat balita terkecil: %.2f kg\n",
    beratTerkecil)
    fmt.Printf("Berat balita terbesar: %.2f kg\n",
    beratTerbesar)
    fmt.Printf("Rata-rata berat balita: %.2f kg\n",
    rataRata)
}
```

Screenshoot Output

```
akim\Kuliah\Semester 3\prak alpro 2> go run "c:\Aby
Hakim\Kuliah\Semester 3\prak alpro 2\modul11\unguide
d3\tempCodeRunnerFile.go"
Masukkan jumlah balita: 2
Masukkan berat balita ke-1: 10
Masukkan berat balita ke-2: 13
Berat balita terkecil: 10.00 kg
Berat balita terbesar: 13.00 kg
Rata-rata berat balita: 11.50 kg
PS C:\Aby Hakim\Kuliah\Semester 3\prak alpro 2> |
```

Deskripsi Program

1. Definisi Tipe Data Berat Balita

- Program mendeklarasikan tipe data array BeratBalita yang dapat menampung hingga 100 nilai berat balita. Tipe data ini akan digunakan untuk menyimpan berat badan balita yang dimasukkan oleh pengguna.

2. Fungsi Mencari Nilai Ekstrim

- Fungsi cariEkstrim digunakan untuk mencari berat balita terkecil dan terbesar dalam array. Fungsi ini menerima array berat balita, jumlah balita, serta referensi untuk menyimpan nilai berat terkecil dan terbesar.
- Pada awalnya, nilai berat terkecil dan terbesar diinisialisasi dengan nilai berat balita pertama dalam array.
- Fungsi kemudian membandingkan setiap berat balita dengan nilai terkecil dan terbesar yang sudah ditemukan sebelumnya. Jika ditemukan berat yang lebih kecil atau lebih besar, nilai terkecil atau terbesar akan diperbarui.

3. Fungsi Menghitung Rata-Rata Berat Balita

- Fungsi hitungRataRata digunakan untuk menghitung rata-rata berat balita dalam array. Fungsi ini menerima array berat balita dan jumlah balita.
- Fungsi menjumlahkan semua berat balita dalam array dan membaginya dengan jumlah balita untuk mendapatkan rata-rata berat.