

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL VIII

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Disusun Oleh :

Naya Putwi Setiasih / 2311102155

S1 11 IF - 05

Dosen Pengampu :

Arif Amrulloh, 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 dalam rak buku, dan contoh lainnya.

Ide algoritma itu sederhana, 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 :

- a. Jadikan data pertama sebagai nilai ekstrim
- b. Lakukan validasi nilai ekstrim dari data kedua hingga data terakhir.
 - ➔ Apabila nilai ekstrim tidak valid, maka update nilai ekstrim tersebut dengan data yang dicek.
- c. 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	$\text{max} \leftarrow 1$	$\text{max} = 0$
2	$i \leftarrow 2$	$i = 1$
3	while $i \leq n$ do	for $i < n$ {
4	if $a[i] > a[\text{max}]$ then	if $a[i] > a[\text{max}]$ {
5	$\text{max} \leftarrow i$	$\text{max} = i$
6	endif	}
7	$i \leftarrow i + 1$	$i = i + 1$
8	endwhile	}

- Pencarian Nilai Ekstrim pada Array Bertipe Data Besar

Misal terdefinisi sebuah array of integer dengan kapasitas 2023, dan array terisi sejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut.

```

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 diatas sedikit berbeda dari sebelumnya karena penggunaan indeks array pada bahasa Go dimulai dari nol atau 0 seperti penjelasan sebelumnya. Di penjelasan sebelumnya juga dijelaskan bahwa pada pencarian yang terpenting adalah posisi atau indeks dari nilai yang dicari dalam kumpulan data atau array. Oleh karena itu modifikasi pada program diatas 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 juga dapat 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 }

```

Maka kita akan memperoleh IPK tertinggi, tetapi kita tidak memperoleh identitas mahasiswa dengan IPK tertinggi tersebut. Berikut ini adalah modifikasi program untuk mengembalikan indeks 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 }

```

Sehingga melalui program diatas, identitas mahasiswa dapat diperoleh, misalnya T[idx].nama, T[idx].nim, T[idx].kelas, hingga T[idx].jurusan.

II. UNGUIDED

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 rill berikutnya adalah berat dari anak kelinci yang akan dijual.
Keluaran : terdiri dari dua buah bilangan rill yang menyatakan berat kelinci terkecil dan terbesar.

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    // Variabel untuk menyimpan jumlah kelinci
    var n int

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

    // Validasi input
    if n <= 0 || n > 1000 {
        fmt.Println("Jumlah anak kelinci harus di antara 1 hingga 1000.")
        return
    }

    // Array untuk menyimpan berat kelinci
    berat := make([]float64, n)

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

    // Cari berat terkecil dan terbesar
    minimalBerat, maksimalBerat := berat[0], berat[0]
    for i := 1; i < n; i++ {
        if berat[i] < minimalBerat {
            minimalBerat = berat[i]
        }
    }
}
```

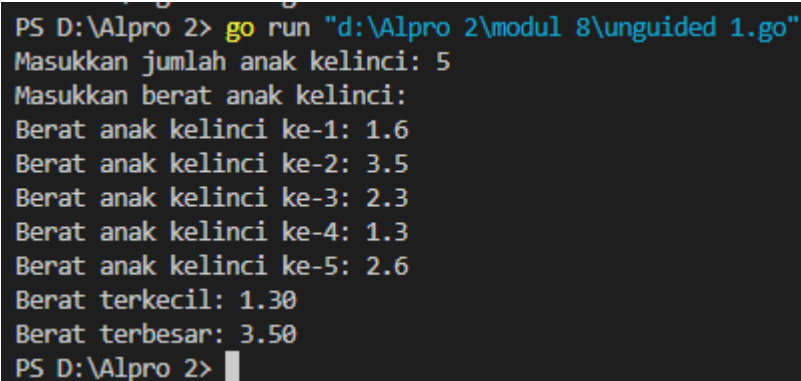
```

        if berat[i] > maksimalBerat {
            maksimalBerat = berat[i]
        }
    }

    // Output berat terkecil dan terbesar
    fmt.Printf("Berat terkecil: %.2f\n", minimalBerat)
    fmt.Printf("Berat terbesar: %.2f\n", maksimalBerat)
}

```

Screenshoot Output



```

PS D:\Alpro 2> go run "d:\Alpro 2\modul 8\unguided 1.go"
Masukkan jumlah anak kelinci: 5
Masukkan berat anak kelinci:
Berat anak kelinci ke-1: 1.6
Berat anak kelinci ke-2: 3.5
Berat anak kelinci ke-3: 2.3
Berat anak kelinci ke-4: 1.3
Berat anak kelinci ke-5: 2.6
Berat terkecil: 1.30
Berat terbesar: 3.50
PS D:\Alpro 2>

```

Deskripsi Program

Program ini ditulis dalam Bahasa Go dan dirancang untuk menghitung dan menampilkan berat terkecil dan berat dari sejumlah anak kelinci yang dimasukkan oleh pengguna. Pertama, program meminta pengguna untuk memasukkan jumlah anak kelinci, dengan validasi untuk memastikan bahwa jumlah tersebut berada dalam rentang 1 hingga 1000. Setelah jumlah kelinci diterima, program kemudian meminta pengguna untuk memasukkan berat masing – masing kelinci satu per satu. Berat ini disimpan dalam sebuah array bertipe float64. Selanjutnya, program melakukan iterasi melalui array untuk menemukan berat terkecil dan terbesar dengan membandingkan setiap elemen. Akhirnya, program mencetak hasilnya, menampilkan berat terkecil dan terbesar dengan format dua decimal. Program ini memberikan cara yang efektif untuk mengelola dan menganalisis data berat kelinci secara interaktif.

2. Sebuah program digunakan untuk menemukan 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 dimasukkan ke

dalam wadah. Baris kedua terdiri dari sejumlah x bilangan rill yang menyatakan banyaknya ikan yang akan dijual.

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

Sourcecode

```
package main

import (
    "fmt"
)

func main() {
    // Variabel untuk input x dan y
    var x, y int

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

    // Validasi input
    if x <= 0 || x > 1000 || y <= 0 {
        fmt.Println("Input tidak valid. Pastikan x dan y
adalah bilangan positif dan x <= 1000.")
        return
    }

    // Array untuk menyimpan berat ikan
    weights := make([]float64, x)

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

    // Variabel untuk menghitung total wadah
    totalWadah := (x + y - 1) / y // Menggunakan
pembulatan ke atas untuk jumlah wadah
    totalBerat := make([]float64, totalWadah) // Berat
total per wadah
```

```

rataBerat := make([]float64, totalWadah) // Berat
rata-rata per wadah

// Hitung berat total dan rata-rata per wadah
for i := 0; i < x; i++ {
    wadahKe := i / y
    totalBerat[wadahKe] += weights[i]
}

for i := 0; i < totalWadah; i++ {
    // Hitung rata-rata berdasarkan jumlah ikan
    dalam wadah
    ikanDalamWadah := y
    if i == totalWadah-16
    { // Wadah terakhir mungkin berisi lebih
    sedikit ikan
        ikanDalamWadah = x % y
        if ikanDalamWadah == 0 {
            ikanDalamWadah = y
        }
    }
    rataBerat[i] = totalBerat[i] /
float64(ikanDalamWadah)
}

// Output hasil
fmt.Println("\nBerat total di setiap wadah:")
for i, berat := range totalBerat {
    fmt.Printf("Wadah %d: %.2f\n", i+1, berat)
}

fmt.Println("\nBerat rata-rata di setiap wadah:")
for i, rata := range rataBerat {
    fmt.Printf("Wadah %d: %.2f\n", i+1, rata)
}
}

```

Screenshoot Output


```
PS D:\Alpro 2> go run "d:\Alpro 2\modul 8\unguided 2.go"
Masukkan jumlah ikan yang akan dijual (x): 6
Masukkan kapasitas maksimal ikan per wadah (y): 2
Masukkan berat ikan:
Berat ikan ke-1: 2
Berat ikan ke-2: 3
Berat ikan ke-3: 4
Berat ikan ke-4: 1
Berat ikan ke-5: 5
Berat ikan ke-6: 6

Berat total di setiap wadah:
Wadah 1: 5.00
Wadah 2: 5.00
Wadah 3: 11.00

Berat rata-rata di setiap wadah:
Wadah 1: 2.50
Wadah 2: 2.50
Wadah 3: 5.50
```

Deskripsi Program

Program yang ditulis dalam Bahasa Go ini bertujuan untuk menghitung berat total dan rata – rata ikan yang akan dijual berdasarkan input dari pengguna. Program dimulai dengan meminta pengguna untuk memasukkan jumlah ikan yang akan dijual (x) dan kapasitas maksimal ikan per wadah (y). Setelah itu, program melakukan validasi untuk memastikan bahwa nilai x dan y adalah bilangan positif, dengan x tidak lebih 1000.

Setelah validasi, pengguna diminta untuk memasukkan berat masing – masing ikan, yang disimpan dalam sebuah array. Program kemudian menghitung jumlah wadah yang diperlukan dengan membagi jumlah ikan dengan kapasitas wadah dan menggunakan pembulatan ke atas. Selanjutnya, program menghitung berat total setiap wadah dengan menjumlahkan berat ikan yang dimasukkan berdasarkan urutan. Untuk menghitung rata – rata berat per wadah, program memperhatikan jumlah ikan yang ada di setiap wadah, termasuk penanganan khusus untuk wadah terakhir jika jumlah ikan tidak habis dibagi kapasitas wadah.

Pada akhirnya, program menampilkan hasil berupa berat total dan rata – rata di setiap wadah. Pengguna dapat dengan mudah memahami distribusi berat ikan dalam wadah – wadah yang telah ditentukan.

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 yang 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"
)

// Tipe array untuk menyimpan data berat balita
type arrBalita [100]float64
```

```

// Fungsi untuk menghitung nilai minimum dan maksimum
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]
        }
    }
}

// Fungsi untuk menghitung rata-rata
func rerata(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 berat balita
    fmt.Print("Masukkan banyak data berat balita: ")
    fmt.Scanln(&n)

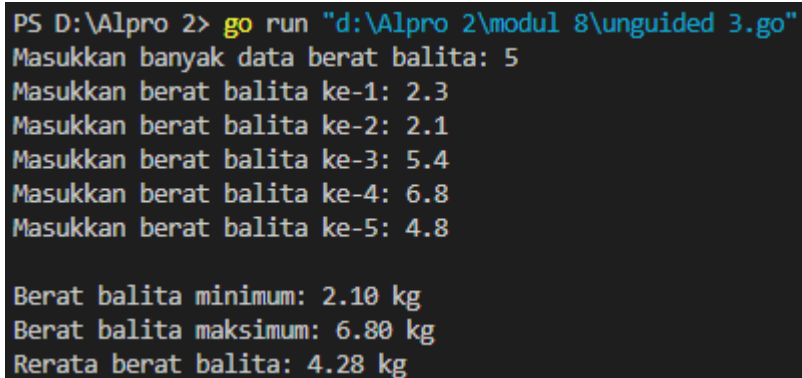
    // Input data berat balita
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
        fmt.Scanln(&arr[i])
    }

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

    // Output hasil
    fmt.Printf("\nBerat balita minimum: %.2f kg\n",
bMin)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)
    fmt.Printf("Rerata berat balita: %.2f kg\n", rata)
}

```

Screenshoot Output



```
PS D:\Alpro 2> go run "d:\Alpro 2\modul 8\unguided 3.go"
Masukkan banyak data berat balita: 5
Masukkan berat balita ke-1: 2.3
Masukkan berat balita ke-2: 2.1
Masukkan berat balita ke-3: 5.4
Masukkan berat balita ke-4: 6.8
Masukkan berat balita ke-5: 4.8

Berat balita minimum: 2.10 kg
Berat balita maksimum: 6.80 kg
Rerata berat balita: 4.28 kg
```

Deskripsi Program

Program di atas adalah aplikasi yang ditulis dalam bahasa pemrograman Go, yang bertujuan untuk menghitung dan menganalisis data berat balita. Program dimulai dengan mendefinisikan tipe array khusus bernama “arrBalita”, yang dapat menyimpan hingga 100 nilai berat balita dalam format float64. Pengguna diminta untuk memasukkan jumlah data berat balita yang ingin diinput. Setelah itu, program akan meminta pengguna untuk memasukkan berat masing – masing balita satu per satu.

Setelah semua data berat dimasukkan, program menggunakan fungsi “hitungMinMax” untuk menentukan nilai minimum dan maksimum dari data berat tersebut. Fungsi ini mengiterasi melalui array dan membandingkan setiap nilai dengan nilai minimum dan maksimum yang sudah ada, memperbarui keduanya sesuai kebutuhan. Selain itu, program juga menghitung rata – rata berat balita menggunakan fungsi “rerata”, yang menjumlahkan semua berat dan membaginya dengan jumlah data yang ada. Pada akhirnya, program menampilkan hasil analisis berupa berat balita minimum, maksimum, dan rata – rata dalam format yang mudah dibaca.

DAFTAR PUSTAKA

Asisten Praktikum, “Modul 8 Pencarian Nilai Ekstrim pada Himpunan Data”,
Learning Management System, 2024.