

LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL 10
PENCARIAN NILAI EXSTRIM PADA HIMPUNAN DATA



Oleh:
HAIKAL SATRIATAMA
2311102066

IF - 11 – 02
S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

I. DASAR TEORI

1. Ide Pencarian Nilai Max/Min

Pencarian merupakan kegiatan yang sering kita lakukan dalam kehidupan sehari-hari. Berbagai contoh penerapannya dapat ditemukan, seperti pencarian file dalam direktori komputer, mencari teks dalam dokumen, atau mencari buku di rak buku. Pada modul ini, kita akan mempelajari salah satu algoritma pencarian, yaitu pencarian nilai ekstrim (terkecil atau terbesar) dalam sekumpulan data.

Konsep dasar algoritma ini cukup sederhana. Karena data harus diproses secara berurutan, maka nilai atau indeks dari nilai maksimum yang telah diproses akan disimpan untuk dibandingkan dengan data berikutnya. Nilai yang disimpan hingga akhir proses adalah nilai maksimum yang dicari.

Algoritma secara umum dapat dijelaskan sebagai berikut:

- I. Anggap data pertama sebagai nilai ekstrim.
- II. Validasi nilai ekstrim dengan membandingkan data kedua hingga data terakhir.
 - Jika nilai ekstrim tidak valid, perbarui nilai ekstrim dengan data yang sedang diperiksa.
- III. Setelah semua data diperiksa, nilai ekstrim yang tersisa adalah yang 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 \leftarrow 1	max = 0
2	i \leftarrow 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 \leftarrow i	max = i
6	endif	}
7	i \leftarrow i + 1	i = i + 1
8	endwhile	}

2. Pencarian Nilal Ekstrim pada Array Bertipe Data Dasar

```

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 tersebut memiliki perbedaan kecil dibandingkan dengan yang sebelumnya, karena dalam bahasa Go, indeks array dimulai dari nol (0), seperti yang telah dijelaskan pada modul 9. Selain itu, seperti yang dijelaskan di awal bab 3, dalam pencarian, hal yang paling penting adalah mengetahui posisi atau indeks dari nilai yang dicari dalam kumpulan data atau array. Modifikasi program tersebut adalah 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 }

```

3. Pencarian Nilai Ekstrim Pada Array Bertipe Data Terstruktur.

Dalam situasi yang lebih rumit, pencarian ekstrim dapat diterapkan, seperti mencari mahasiswa dengan nilai tertinggi, lagu dengan durasi terpanjang, atau pembalap dengan waktu balap tercepat, dan sebagainya. Sebagai contoh, sebuah array dapat digunakan untuk menyimpan data mahasiswa, sementara fungsi IPK digunakan untuk menemukan 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 }

```

Sebagai contoh, sebuah array digunakan untuk menyimpan data mahasiswa, dan diikuti dengan fungsi IPK yang berfungsi untuk mencari mahasiswa dengan IPK tertinggi.

```
...
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 }
```

II. GUIDED

No.1

```
package main

import (
    "fmt"
)

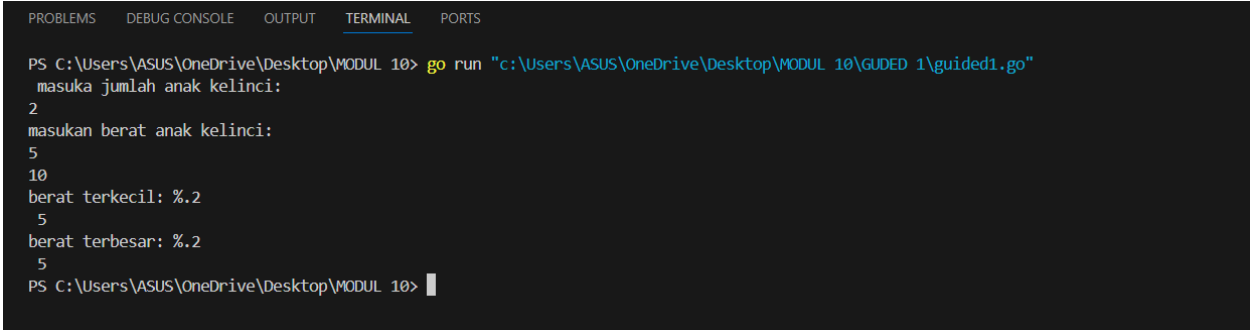
func main() {
    var N int
    var berat [1000]float64
    fmt.Println("masuka jumlah anak kelinci:")
    fmt.Scan(&N)
    fmt.Println("masukan berat anak kelinci:")
    for i := 0; i < N; i++ {
        fmt.Scan(&berat[i])
    }
    min := berat[0]
    max := berat[0]
```

```

    for i := 1; i < N; i++ {
        if berat[i] < min {
            min = berat[i]
        }
        if berat[i] > max {
            max = berat[i]
        }
    }
    fmt.Println("berat terkecil: %.2\n", min)
    fmt.Println("berat terbesar: %.2\n", max)
}

```

Output :



```

PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL  PORTS

PS C:\Users\ASUS\OneDrive\Desktop\MODUL 10> go run "c:\Users\ASUS\OneDrive\Desktop\MODUL 10\GUDED 1\guided1.go"
masuka jumlah anak kelinci:
2
masukan berat anak kelinci:
5
10
berat terkecil: %.2
5
berat terbesar: %.2
5
PS C:\Users\ASUS\OneDrive\Desktop\MODUL 10>

```

Penjelasan :

Program ini memungkinkan pengguna untuk memasukkan data berat anak kelinci dan menghitung berat terkecil dan terbesar di antara mereka. Program perlu diperbaiki sedikit pada logika dan penggunaan fungsi pencetakan agar berfungsi dengan benar. Program tersebut bertujuan untuk menerima masukan berupa jumlah anak kelinci dan berat masing-masing, lalu menentukan berat terkecil dan terbesar dari anak-anak kelinci tersebut.

No 2.

```
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()
}

```

Output :

```
PROBLEMS 2 DEBUG CONSOLE OUTPUT TERMINAL PORTS
PS C:\Users\ASUS\OneDrive\Desktop\MODUL 10> go run "c:\Users\ASUS\OneDrive\Desktop\MODUL 10\GUIDED 1\guided2.go"
masukkan jumlah ikan dan kapasitas wadah: 10 10
masukkan berat tiap ikan:
2
2
3
4
3
6
5
2
4
5
total berat tiap wadah:
36.00
rata rata berat tiap wadah:
3.60
PS C:\Users\ASUS\OneDrive\Desktop\MODUL 10> |
```

Penjelasan :

Program dirancang untuk membantu menghitung distribusi berat ikan ke dalam wadah dengan kapasitas tertentu. Program menerima input jumlah ikan, kapasitas wadah, serta berat masing-masing ikan. Selanjutnya, program menghitung total berat di setiap wadah dan rata-rata berat per wadah.

III. UNGUIDED

Soal Modul 10

Source code no 3

```
package main

import "fmt"

const maxEntries = 100

type weights [maxEntries]float64

func findMinMax(data weights, count int) {
    minVal := data[0]
```



```

maxVal := data[0]

for i := 1; i < count; i++ {
    if data[i] < minVal {
        minVal = data[i]
    }
    if data[i] > maxVal {
        maxVal = data[i]
    }
}

fmt.Printf("Berat minimum balita: %.2f\n", minVal)
fmt.Printf("Berat maksimum balita: %.2f\n", maxVal)
}

func calculateAverage(data weights, count int) float64 {
    var total float64
    for i := 0; i < count; i++ {
        total += data[i]
    }
    return total / float64(count)
}

func main() {
    var data weights
    var count int

    fmt.Print("Masukkan jumlah data berat balita: ")
    fmt.Scan(&count)

    fmt.Println("Masukkan berat masing-masing balita:")
    for i := 0; i < count; i++ {
        fmt.Scan(&data[i])
    }
}

```

```
findMinMax(data, count)
fmt.Printf("Rata-rata berat balita: %.2f\n",
    calculateAverage(data, count))
}
```

Output :

```
PS C:\Users\ASUS\OneDrive\Desktop\MODUL 10> go run "c:\Users\ASUS\OneDrive\Desktop\MODUL 10\UNGUIDED\unguided1.go"
Masukkan jumlah data berat balita: 4
Masukkan berat masing-masing balita:
5.3
6.2
4.1
9.9
Berat minimum balita: 4.10
Berat maksimum balita: 9.90
Rata-rata berat balita: 6.38
PS C:\Users\ASUS\OneDrive\Desktop\MODUL 10> █
```

Penjelasan :

Program ini berfungsi membaca data berat balita, lalu menghitung dan menampilkan berat minimum, berat maksimum, serta rata-rata dari data tersebut. Kemudian menghitung berat minimum, berat maksimum, dan rata-rata dari data yang dimasukkan. Program menggunakan fungsi untuk memisahkan logika perhitungan minimum/maksimum dan rata-rata.