

LAPORAN PRAKTIKUM
ALGORITMA PROGRAM 2
MODUL 11
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Oleh:

ADITHANA DHARMA PUTRA

2311102207

IF – 11- 02

S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024

I. DASAR TEORI

1. 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.
- 3) Apabila nilai ekstrim tidak valid, maka update nilai ekstrims tersebut dengan data yang dicek.
- 4) 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	}

2. 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 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 "0" 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  }

```

3. Pencarian Nilai Ekstrem pada Array Bertipe Data Terstruktur

Pada kasus yang lebih kompleks pencarian ekstrem 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 }
```

1. Guided 1

Source Code

```
package main
import "fmt"

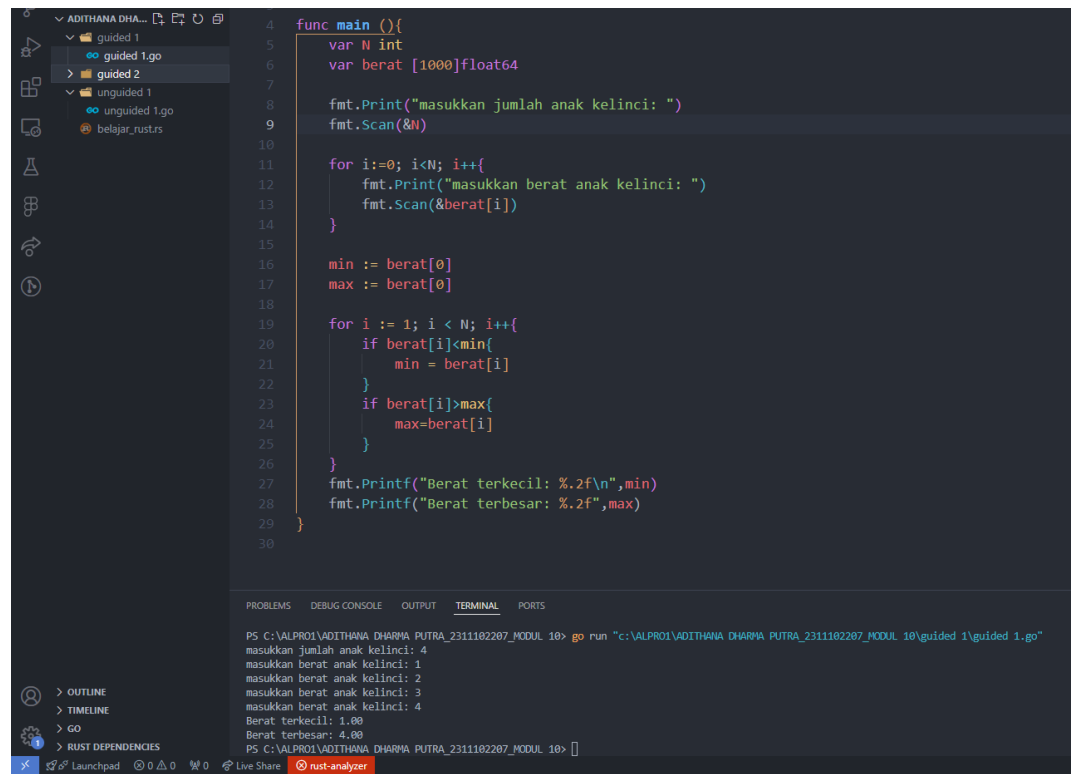
func main (){
    var N int
    var berat [1000]float64

    fmt.Print("masukkan jumlah anak kelinci: ")
    fmt.Scan(&N)

    for i:=0; i<N; i++){
        fmt.Print("masukkan berat anak kelinci: ")
        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.Printf("Berat terkecil: %.2f\n",min)
    fmt.Printf("Berat terbesar: %.2f",max)
}
```

Screenshot



```
4 func main(){
5     var N int
6     var berat [1000]float64
7
8     fmt.Print("masukkan jumlah anak kelinci: ")
9     fmt.Scan(&N)
10
11     for i:=0; i<N; i++){
12         fmt.Print("masukkan berat anak kelinci: ")
13         fmt.Scan(&berat[i])
14     }
15
16     min := berat[0]
17     max := berat[0]
18
19     for i := 1; i < N; i++){
20         if berat[i]<min{
21             min = berat[i]
22         }
23         if berat[i]>max{
24             max=berat[i]
25         }
26     }
27     fmt.Printf("Berat terkecil: %.2f\n",min)
28     fmt.Printf("Berat terbesar: %.2f",max)
29 }
30
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
PS C:\ALPRO1\ADITHANA DHARMA PUTRA_2311102207_MODUL 10> go run "c:\ALPRO1\ADITHANA DHARMA PUTRA_2311102207_MODUL 10\guided 1\guided 1.go"
masukkan jumlah anak kelinci: 4
masukkan berat anak kelinci: 1
masukkan berat anak kelinci: 2
masukkan berat anak kelinci: 3
masukkan berat anak kelinci: 4
Berat terkecil: 1.00
Berat terbesar: 4.00
PS C:\ALPRO1\ADITHANA DHARMA PUTRA_2311102207_MODUL 10>
```

Deskripsi:

Kode di atas adalah program yang meminta pengguna untuk memasukkan jumlah anak kelinci dan berat masing-masing anak kelinci. Program ini kemudian menentukan dan menampilkan berat terkecil dan terbesar dari anak-anak kelinci tersebut. Di dalam fungsi main, variabel N dideklarasikan untuk menyimpan jumlah anak kelinci, dan array berat dengan kapasitas 1000 elemen dideklarasikan untuk menyimpan berat masing-masing anak kelinci. Program meminta pengguna untuk memasukkan jumlah anak kelinci dan kemudian menggunakan loop for untuk meminta berat setiap anak kelinci, yang disimpan dalam array berat. Setelah semua berat dimasukkan, program menginisialisasi variabel min dan max dengan nilai berat anak kelinci pertama. Kemudian, program menggunakan loop for untuk memeriksa setiap berat anak kelinci dan memperbarui min jika berat lebih kecil dari min, serta memperbarui max jika berat lebih besar dari max. Akhirnya, program

menampilkan berat terkecil dan terbesar menggunakan `fmt.Printf` dengan format dua angka desimal.

2. Guided 2

Source Code

```
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 // pembulatan ke atas jika x tidak habis
    dibagi y
    totalBeratWadah := make([]float64, jumlahWadah)

    for i := 0; i < x; i++ {
        indeksWadah := i / y
        totalBeratWadah[indeksWadah] += berat[i]
    }

    // Output total berat tiap wadah
    fmt.Println("Total berat tiap wadah:")
    for _, total := range totalBeratWadah {
```

```

        fmt.Printf("%.2f ", total)
    }

    fmt.Println()

    // Output rata-rata berat tiap wadah
    fmt.Println("Rata-rata berat tiap wadah:")

    for _, total := range totalBeratWadah {
        rataRata := total / float64(y)
        fmt.Printf("%.2f ", rataRata)
    }

    fmt.Println()
}

```

Screenshot

The screenshot shows a Rust IDE with a project named 'ADITHIANA.DHA...'. The file explorer on the left shows a directory structure with files like 'guided 1', 'guided 2', 'guided 2.go', 'unguided 1', and 'belajar_rust.rs'. The main editor displays a Rust program that calculates the average weight of fish in different containers. The code includes comments in Indonesian and uses standard Rust syntax for arrays, loops, and formatting. The terminal at the bottom shows the command to run the program and its output, which matches the expected results from the code.

```

4
5 func main() {
6     var x, y int
7     fmt.Print("Masukkan jumlah ikan dan kapasitas wadah: ")
8     fmt.Scan(&x, &y)
9
10    berat := make([]float64, x)
11    fmt.Println("Masukkan berat tiap ikan:")
12    for i := 0; i < x; i++ {
13        fmt.Scan(&berat[i])
14    }
15
16    jumlahWadah := (x + y - 1) / y
17    totalBeratWadah := make([]float64, jumlahWadah)
18    jumlahIkanDiWadah := make([]int, jumlahWadah)
19
20    for i := 0; i < x; i++ {
21        indeksWadah := i / y
22        totalBeratWadah[indeksWadah] += berat[i]
23        jumlahIkanDiWadah[indeksWadah]++
24    }
25
26    // Output total berat tiap wadah
27    fmt.Println("Total berat tiap wadah:")
28    for _, total := range totalBeratWadah {
29        fmt.Printf("%.2f ", total)
30    }
31    fmt.Println()
32
33

```

PROBLEMS DEBUG CONSOLE OUTPUT **TERMINAL** PORTS

```

PS C:\VALPROI\ADITHIANA DHARMA PUTRA_2311102207_MODUL 10> go run "c:\valproi\adithana dharm putra_2311102207_modul 10\guided 2\guided 2.go"
Masukkan jumlah ikan dan kapasitas wadah: 5 7
Masukkan berat tiap ikan:
1 2 3 4 5
Total berat tiap wadah:
15.00
Rata-rata berat tiap wadah:
3.00
PS C:\VALPROI\ADITHIANA DHARMA PUTRA_2311102207_MODUL 10>

```

Ln 26, Col 37

Deskripsi:

Kode ini adalah program yang meminta pengguna untuk memasukkan jumlah ikan dan kapasitas wadah, kemudian menghitung total berat dan rata-rata berat tiap wadah. Pertama, program meminta pengguna untuk memasukkan jumlah ikan dan kapasitas wadah, serta berat masing-masing ikan. Setelah itu, program menghitung jumlah wadah yang diperlukan dengan membulatkan ke atas jika jumlah ikan tidak habis dibagi kapasitas wadah. Program kemudian mengisi array `totalBeratWadah` dengan menambahkan berat ikan ke wadah yang sesuai berdasarkan indeks. Setelah semua berat ikan dimasukkan ke wadah yang tepat, program menampilkan total berat tiap wadah dan rata-rata berat tiap wadah dengan format dua angka desimal.

II. UNGUIDED

1. Unguided 1

Source Code

```
package main
import "fmt"

func hitungMinMax(arrBerat207[100]float64, bMin float64, bMax float64,N int){
    for i := 0; i < N; i++){
        if arrBerat207[i]<bMin{
            bMin = arrBerat207[i]
        }
        if arrBerat207[i]>bMax{
            bMax=arrBerat207[i]
        }
    }
    fmt.Printf("Berat Minumum: %.2f\n",bMin)
    fmt.Printf("Berat Maksimum: %.2f\n",bMax)
    fmt.Printf("Rerata berat balita: %.2f",rerata(arrBerat207,N))
}

func rerata(arrBerat207[100]float64,N int) float64{
    var jumlah float64
    for i:=0;i<N;i++){
        jumlah += arrBerat207[i]
    }
    BanyakData := float64(N)
    return jumlah/BanyakData
}

func main (){
    var N int
```

```

var arrBerat207 [100]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(&arrBerat207[i])
}

bMin := arrBerat207[0]
bMax := arrBerat207[0]
hitungMinMax(arrBerat207,bMin,bMax,N)
}

```

Screenshot

```

4 func hitungMinMax(arrBerat207[100]float64, bMin float64, bMax float64, N int){
5     for i := 0; i < N; i++){
6         if arrBerat207[i]<bMin{
7             bMin = arrBerat207[i]
8         }
9         if arrBerat207[i]>bMax{
10            bMax=arrBerat207[i]
11        }
12    }
13    fmt.Printf("Berat Minumum: %.2f\n",bMin)
14    fmt.Printf("Berat Maksimum: %.2f\n",bMax)
15    fmt.Printf("Rerata berat balita: %.2f",rerata(arrBerat207,N))
16 }
17
18 func rerata(arrBerat207[100]float64,N int) float64{
19     var jumlah float64
20     for i:=0;i<N;i++){
21         jumlah += arrBerat207[i]
22     }
23     BanyakData := float64(N)
24     return jumlah/BanyakData
25 }
26 func main (){
27     var N int
28     var arrBerat207 [100]float64
29
30     fmt.Print("Masukkan Banyak data berat Balita: ")
31     fmt.Scan(&N)

```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```

PS C:\VALPROJ\ADITHANA_DHARWA_PUTRA_231102207_PCCOL 10> go run "c:\VALPROJ\ADITHANA_DHARWA_PUTRA_231102207_PCCOL 10\unguided 1\unguided 1.go"
Masukkan Banyak data berat Balita: 4
Masukkan berat Balita ke-1: 5.3
Masukkan berat Balita ke-2: 6.2
Masukkan berat Balita ke-3: 4.1
Masukkan berat Balita ke-4: 9.9
Berat Minumum: 4.10
Berat Maksimum: 9.90
Rerata berat balita: 6.38
PS C:\VALPROJ\ADITHANA_DHARWA_PUTRA_231102207_PCCOL 10>

```

OUTLINE TIMELINE GO

Launchpad 0 0 0 0 0 Live Share Ln 3, Col 1

Deskripsi

Kode ini adalah program yang mencari berat maksimum, minimum, dan rerata dari data balita. Seperti yang diminta dalam spesifikasi pseudocode yang ada di modul, pertama kita buat fungsi untuk mencari nilai min dan max, pada `func hitungMinMax`. Kita akan membuat iterasi untuk selanjutnya di dalamnya kita membuat pengkondisian untuk mencari berat minimal dan berat maksimal dengan membandingkan nilainya untuk setiap indeks di array. Ketika sudah ditemukan selanjutnya kita memanggil fungsi kedua yaitu `func rerata`. Pada fungsi ini kita akan menjumlahkan seluruh isi array lalu membaginya dengan jumlah data. Perlu diperhatikan karena variabel `N` adalah integer maka kita perlu casting agar berubah menjadi `float64` selanjutnya fungsi akan mereturn value agar bisa mengoutputkan nilai rerata tadi