LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 10

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Oleh:

MUHAMMAD RUSDIYANTO

2311102053

S1IF-11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Nilai ekstrem merupakan konsep penting dalam pemrograman yang merujuk pada nilai terkecil (minimum) dan terbesar (maksimum) dalam suatu koleksi data, seperti array atau daftar. Menemukan nilai ekstrem seringkali diperlukan dalam berbagai konteks, mulai dari analisis data hingga optimasi algoritma. Untuk mencari nilai ekstrem, metode yang umum digunakan adalah iterasi (looping) melalui elemen-elemen dalam koleksi data tersebut. Setiap iterasi bertujuan membandingkan elemen saat ini dengan nilai ekstrem yang telah ditemukan sejauh ini; jika elemen tersebut lebih kecil dari nilai minimum yang ada atau lebih besar dari nilai maksimum, maka nilai ekstrem tersebut akan diperbarui. Proses iterasi ini berlanjut hingga seluruh elemen dalam koleksi data selesai dibandingkan, yang memastikan bahwa nilai ekstrem yang ditemukan adalah nilai yang valid dan akurat.

Selain dengan iterasi, pencarian nilai ekstrem juga dapat dilakukan dengan metode lain, tergantung pada kebutuhan dan ukuran data. Untuk koleksi data berukuran besar atau yang terstruktur secara khusus, metode pencarian yang lebih efisien, seperti algoritma divide and conquer, dapat dipertimbangkan guna mengurangi kompleksitas waktu pencarian. Dalam bahasa pemrograman tertentu, beberapa pustaka atau fungsi bawaan juga menyediakan metode untuk menemukan nilai minimum dan maksimum secara langsung, yang dapat mempermudah pengembang dalam menulis kode yang efisien. Pemahaman tentang pencarian nilai ekstrem tidak hanya penting untuk operasi dasar pada data, tetapi juga menjadi fondasi dalam pengembangan algoritma yang lebih kompleks, seperti sorting dan searching, yang sering membutuhkan identifikasi nilai minimum atau maksimum sebagai bagian dari proses komputasi..

II. GUIDED

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)

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

    // Inisialisasi nilai min dan max dengan elemen pertama
    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\n", max)
}
```

Guided 1 | Output

```
Masukkan jumlah anak kelinci: 5
Masukkan berat anak kelinci:
2.3
1.4
1.6
1.9
2.0
Berat terkecil: 1.40
Berat terbesar: 2.30
```

Guided 1 | Penjelasan

Program di atas adalah program sederhana untuk menentukan berat terkecil dan terbesar dari sejumlah anak kelinci berdasarkan data berat badan yang dimasukkan oleh pengguna. Program ini menggunakan array untuk menyimpan berat badan dan perulangan untuk mencari nilai minimum dan maksimum.

Proses dimulai dengan meminta pengguna memasukkan jumlah anak kelinci (N). Setelah itu, pengguna diminta memasukkan berat masing-masing anak kelinci, yang disimpan ke dalam array berat. Nilai berat badan pertama dari array digunakan untuk menginisialisasi nilai minimum (min) dan maksimum (max).

Selanjutnya, program menggunakan perulangan untuk memeriksa setiap elemen array mulai dari indeks ke-1 hingga indeks ke-N-1. Jika berat badan yang sedang diperiksa lebih kecil dari nilai min, maka min diperbarui dengan nilai tersebut. Sebaliknya, jika berat badan lebih besar dari nilai max, maka max diperbarui.

Setelah perulangan selesai, program mencetak hasil berupa berat badan terkecil dan terbesar dengan format dua desimal. Program ini berguna untuk memproses data sederhana seperti mencari rentang berat badan dalam populasi tertentu.

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()
}
```

Guided 2 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day6\Muhammad Rusdiyanto_2311102053> go run guided2.go
Masukkan jumlah ikan dan kapasitas wadah: 4 8
Masukkan berat tiap ikan:
4.3
5.2
2.9
3.0
Total berat tiap wadah:
15.40
Rata - rata berat tiap wadah:
1.93
```

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day6\Muhammad Rusdiyanto_2311102053> go run guided2.go
Masukkan jumlah ikan dan kapasitas wadah: 4 4
Masukkan berat tiap ikan:
4.3
5.2
2.9
3.0
Total berat tiap wadah:
15.40
Rata - rata berat tiap wadah:
3.85
```

Guided 2 | Penjelasan

Program di atas adalah program untuk menghitung total berat dan rata-rata berat ikan dalam beberapa wadah, di mana setiap wadah memiliki kapasitas maksimum sejumlah ikan tertentu. Program ini berguna untuk mendistribusikan ikan ke wadah secara efisien berdasarkan berat mereka.

Proses dimulai dengan meminta pengguna memasukkan jumlah total ikan (x) dan kapasitas maksimum wadah dalam jumlah ikan (y). Selanjutnya, pengguna diminta untuk memasukkan berat masing-masing ikan, yang disimpan dalam slice berat.

Program menghitung jumlah wadah yang dibutuhkan menggunakan formula (x + y - 1) / y, yang memastikan pembulatan ke atas jika jumlah ikan tidak habis dibagi kapasitas wadah. Slice totalBeratWadah kemudian digunakan untuk menyimpan total berat ikan dalam masing-masing wadah.

Dalam perulangan, setiap berat ikan ditambahkan ke wadah yang sesuai, di mana indeks wadah dihitung menggunakan operasi pembagian integer (i / y). Setelah semua berat ikan didistribusikan, program mencetak total berat tiap wadah. Kemudian, untuk menghitung rata-rata berat ikan di tiap wadah, program membagi total berat wadah dengan kapasitas maksimum y. Hasil perhitungan tadi nantinya akan ditampilkan ke pengguna.

III. UNGUIDED

Unguided 1 | Source Code

```
package main

import "fmt"

type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita, bMin *float64, bMax *float64, n
int) {
   for i := 1; i < n; i++ {
      if arrBerat[i] < *bMin {
         fmt.Print(arrBerat[i])
         *bMin = arrBerat[i]
      }
      if arrBerat[i] > *bMax {
         *bMax = arrBerat[i]
      }
   }
}

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 arrBerat arrBalita
   fmt.Print("Masukkan banyak data berat balita : ")
   fmt.Scan(&n)

   if n > 100 {
      fmt.Print("Ukuran terlalu besar. Harus kurang dari atau sama
dengan 100.")
   }

   for i := 1; i <= n; i++ {
      fmt.Printf("Masukkan berat balita ke-%v: ", i)
      fmt.Scan(&arrBerat[i-1])
   }
```

```
    var min, max float64 = arrBerat[0], arrBerat[0]
    hitungMinMax(arrBerat, &min, &max, n)
    fmt.Printf("Berat balita minimum: %.2f KG\n", min)
    fmt.Printf("Berat balita maksimum: %.2f KG\n", max)
    fmt.Printf("Rerata berat balita: %.2f KG\n", rerata(arrBerat, n))
}
```

Unguided 1 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day6\Muhammad Rusdiyanto_2311102053> go run unguided1.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
4.1Berat balita minimum: 4.1
Berat balita maksimum: 9.9
Rerata berat balita: 6.38
```

Unguided 1 | Penjelasan

Program di atas adalah program untuk menganalisis data berat badan balita, meliputi perhitungan berat minimum, maksimum, dan rata-rata dari sejumlah data yang dimasukkan oleh pengguna. Program ini menggunakan tipe data array arrBalita untuk menyimpan data berat balita dengan maksimum 100 data.

Proses dimulai dengan meminta pengguna memasukkan jumlah data berat balita (n). Jika jumlah data melebihi 100, program akan menampilkan pesan kesalahan. Selanjutnya, pengguna diminta untuk memasukkan berat badan setiap balita yang kemudian disimpan dalam array arrBerat.

Fungsi hitungMinMax digunakan untuk menghitung berat minimum dan maksimum. Fungsi ini memeriksa setiap elemen array dan memperbarui nilai minimum (bMin) jika ditemukan berat yang lebih kecil, atau maksimum (bMax) jika ditemukan berat yang lebih besar. Fungsi rerata digunakan untuk menghitung rata-rata berat badan dengan menjumlahkan seluruh nilai dalam array dan membaginya dengan jumlah data.

Setelah semua perhitungan selesai, program mencetak hasil berupa berat minimum, maksimum, dan rata-rata ke layar dengan format dua desimal. Program ini berguna untuk menganalisis distribusi berat badan balita secara sederhana.