# LAPORAN PRAKTIKUM
# ALGORITMA DAN PEMROGRAMAN 2

## MODUL XI

## PENCARIAN NILAI EXTREME PADA HIMPUNAN DATA


**Disusun Oleh :**

**Fahrial Aufa Ramadhan / 2311102241**

**IF-11-6**

**Dosen Pengampu :**

**ABEDNEGO DWI SEPTIADI**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**


**2024**

# I. DASAR TEORI

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

Dalam pencarian nilai ekstrim, terdapat beberapa metode kunci yang dapat diimplementasikan menggunakan Golang, masing-masing dengan karakteristik dan use case yang berbeda. Linear Search merupakan metode paling sederhana dengan kompleksitas O(n) yang melakukan iterasi langsung pada seluruh data untuk menemukan nilai maksimum atau minimum, sangat cocok untuk dataset kecil atau ketika data tidak terurut; Sorted Array Method membutuhkan langkah pengurutan terlebih dahulu dengan kompleksitas O(n log n) namun memberikan keuntungan untuk operasi pencarian berulang karena data sudah terurut; sedangkan metode Divide and Conquer menerapkan strategi pembagian data menjadi bagian-bagian lebih kecil dengan kompleksitas O(log n), sangat efisien untuk dataset besar yang terurut dan dapat diimplementasikan secara rekursif - pemilihan metode yang tepat akan sangat bergantung pada karakteristik data, kebutuhan performa, dan konteks penggunaan dalam aplikas

# I. GUIDED

**Soal Studi Case**

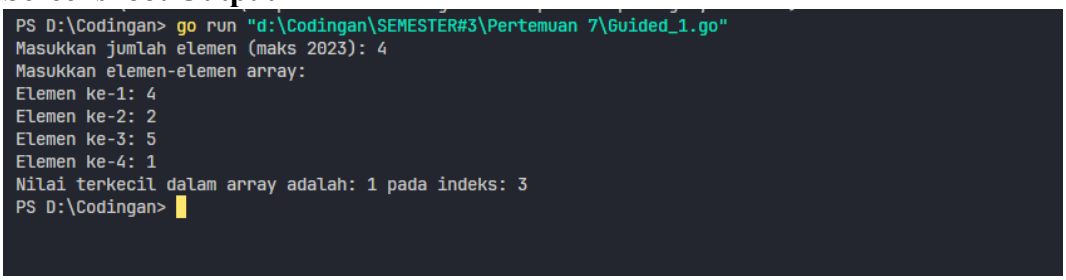
Misalnya terdefinisi sebuah array of integer dengan kapasitas 2023, dan array terisi sejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut. Perhatikan potongan program dalam bahasa Go berikut ini

**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**




**Deskripsi Program**

Program di atas dibuat untuk membantu menemukan angka terkecil dalam sebuah daftar angka (array) yang bisa menampung hingga 2023 elemen. Setelah pengguna memasukkan jumlah elemen dan angka-angkanya, program akan memeriksa setiap angka untuk mencari yang terkecil. Hasil akhirnya adalah angka terkecil tersebut beserta posisi atau urutannya dalam daftar, yang kemudian ditampilkan dengan cara yang mudah dimengerti. Program ini juga memastikan data yang dimasukkan sesuai, sehingga aman digunaka

# II. GUIDED

**Soal Studi Case**

Pada kasus yang lebih kompleks pencarian ekstrim 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**.**

**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**

```
PS D:\Codingan> go run "d:\Codingan\SEMESTER#3\Pertemuan 7\Guided_2.go"
Masukkan jumlah mahasiswa (maks 2023): 2

Masukkan data mahasiswa ke-1
Nama: Fahrial
NIM: 2311102241
Kelas: IF-11-06
Jurusan: Informatic
IPK: 4.0

Masukkan data mahasiswa ke-2
Nama: Aufa
NIM: 2311102242
Kelas: IF-11-34
Jurusan: Informatic
IPK: 3.9

IPK tertinggi dari 2 mahasiswa adalah: 4.00
PS D:\Codingan>
```

**Deskripsi Program**

Program di atas adalah aplikasi sederhana dalam golang yang dirancang untuk mengolah data mahasiswa dan mencari IPK tertinggi dari sekumpulan mahasiswa.Program ini menggunakan struktur data struct untuk menyimpan informasi mahasiswa

# III. UNGUIDED

**Soal Studi Case**

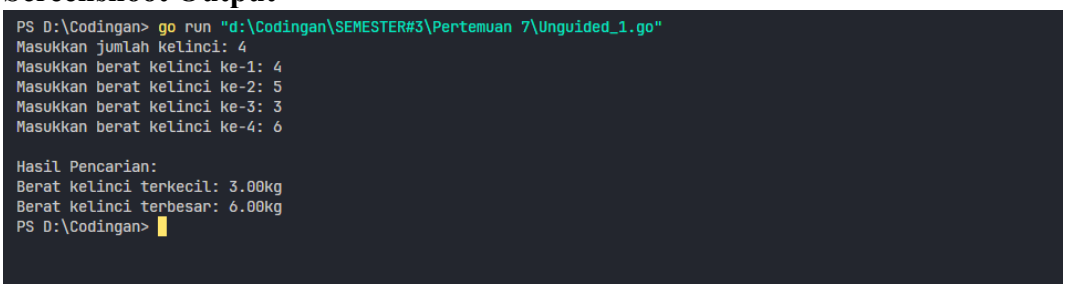
Sebuah program digunakan untuk mendata berat anak kelinci yang akan dijual he pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat anak belinci yang akan dijual.

**Sourcecode**

```
package main

import "fmt"

func main() {
	var jumKelinci_2311102241 int
	var beratKelinci = make([]float64, 0, 1000)

	fmt.Print("Masukkan jumlah kelinci: ")
	fmt.Scan(&jumKelinci_2311102241)

	if jumKelinci_2311102241 > 1000 {
		fmt.Println("Maaf, jumlah kelinci tidak
boleh lebih dari 1000")
		return
	}

	for i := 1; i <= jumKelinci_2311102241; i++ {
		var berat float64
		fmt.Printf("Masukkan berat kelinci ke-
%d: ", i)
		fmt.Scan(&berat)
		beratKelinci = append(beratKelinci,
berat)
	}

	var beratTerkecil = beratKelinci[0]
	var beratTerbesar = beratKelinci[0]

	for i := 0; i < jumKelinci_2311102241; i++ {
		if beratKelinci[i] < beratTerkecil {
			beratTerkecil = beratKelinci[i]
		}
		if beratKelinci[i] > beratTerbesar {
			beratTerbesar = beratKelinci[i]
		}
	}

	fmt.Println("\nHasil Pencarian:")
	fmt.Printf("Berat kelinci terkecil: %.2fkg\n",
beratTerkecil)
	fmt.Printf("Berat kelinci terbesar: %.2fkg\n",
beratTerbesar)
```

```
}
```

**Screenshoot Output**




**Deskripsi Program**

Program ini dibuat untuk mempermudah pengguna menemukan berat kelinci terkecil dan terbesar dari data yang dimasukkan. Setelah diminta memasukkan jumlah kelinci (maksimal 1000), pengguna dapat mengisi berat masing-masing kelinci satu per satu. Program secara otomatis memeriksa data yang diinput untuk memastikan tidak melebihi batas, lalu menghitung berat terkecil dan terbesar dari semua kelinci tersebut. Hasilnya akan ditampilkan dengan jelas dalam format angka dua desimal agar mudah dipahami.

# IV. UNGUIDED

**Soal Studi Case**

Sebuah program digunakan untuk menentukan tarif ikan yang akan dijual be pasar. Program Ini menggunakan array dengan kapasitas 1000 untuk menampung data berat ikan yang ahan dijual.

**Sourcecode**

```
package main

import "fmt"

func main() {
	var jumlahIkan int
	var jumlahWadah int

	fmt.Print("Berapa jumlah ikan(x): ")
	fmt.Scan(&jumlahIkan)

	if jumlahIkan > 1000 {
		fmt.Println("Maaf, jumlah ikan tidak
boleh lebih dari 1000")
		return
	}

	fmt.Print("Berapa jumlah wadah(y): ")
	fmt.Scan(&jumlahWadah)

	var beratIkan = make([]float64, 0, 1000)

	for i := 0; i < jumlahIkan; i++ {
		var berat float64
		fmt.Printf("Masukkan berat ikan ke-%d:
", i+1)
		fmt.Scan(&berat)
		beratIkan = append(beratIkan, berat)
	}

	var totalPerWadah = make([]float64,
jumlahWadah)

	var ikanPerWadah = jumlahIkan / jumlahWadah
	var nomorIkan = 0

	for wadah := 0; wadah < jumlahWadah; wadah++ {
		var totalBerat float64 = 0

		for ikan := 0; ikan < ikanPerWadah;
ikan++ {
			if nomorIkan < jumlahIkan {
```

```
                              totalBerat = totalBerat +
      beratIkan[nomorIkan]
                              nomorIkan = nomorIkan + 1
                        }
                  }

                  totalPerWadah[wadah] = totalBerat
            }

            fmt.Println("\nTotal berat tiap wadah:")
            for i := 0; i < jumlahWadah; i++ {
                  fmt.Printf("Wadah %d: %.1f kg\n", i+1,
      totalPerWadah[i])
            }

            var totalSemuaWadah float64 = 0
            for i := 0; i < jumlahWadah; i++ {
                  totalSemuaWadah = totalSemuaWadah +
      totalPerWadah[i]
            }
            var rataRata = totalSemuaWadah /
      float64(jumlahWadah)

            fmt.Printf("\nRata-rata berat ikan per wadah:
      %.1f kg\n", rataRata)
}
```

**Screenshoot Output**

```
PS D:\Codingan> go run "d:\Codingan\SEMESTER#3\Pertemuan 7\Unguided_2.go"
Berapa jumlah ikan(x): 4
Berapa jumlah wadah(y): 2
Masukkan berat ikan ke-1: 1.4
Masukkan berat ikan ke-2: 2.1
Masukkan berat ikan ke-3: 3.2
Masukkan berat ikan ke-4: 1.2

Total berat tiap wadah:
Wadah 1: 3.5 kg
Wadah 2: 4.4 kg

Rata-rata berat ikan per wadah: 4.0 kg
PS D:\Codingan>
```

**DeskripsiProgram**

Program ini dirancang untuk membantu mendistribusikan ikan ke beberapa wadah secara merata dan menghitung berat total serta rata-rata ikan per wadah. Pengguna diminta memasukkan jumlah ikan (maksimal 1000), jumlah wadah, dan berat masing-masing ikan.

# V. UNGUIDED

**Soal Studi Case**

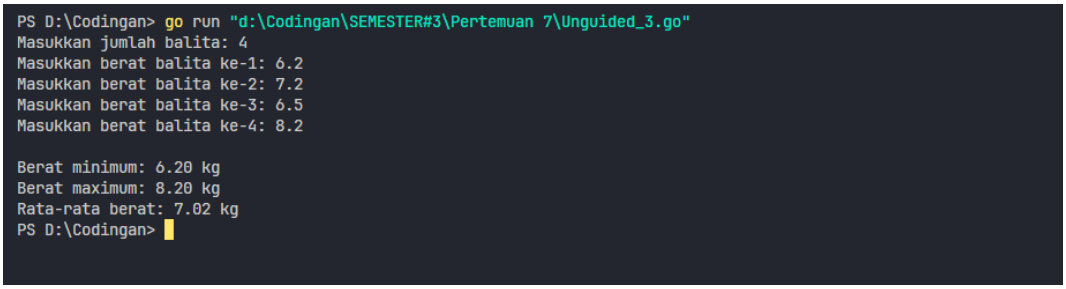
Pos Pelayanan Terpadu (posyandu) sebagai tempat pelayanan kesehatan perlu mencatat data berat balita (dalam kg). Petugas akan memasukkan data tersebut ke dalam array. Dari data yang diperoleh akan dicari berat balita terkecil, terbesar, dan reratanya.

**Sourcecode**

```
package main

import "fmt"

type arrBalita [100]float64

func main() {
	var dataBalita arrBalita
	var jumlahBalita int
	var min, max, total float64

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

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

	min = dataBalita[0]
	max = dataBalita[0]

	for i := 0; i < jumlahBalita; i++ {
		if dataBalita[i] < min {
			min = dataBalita[i]
		}

		if dataBalita[i] > max {
			max = dataBalita[i]
		}

		total = total + dataBalita[i]
	}

	rataRata := total / float64(jumlahBalita)

	fmt.Printf("\nBerat minimum: %.2f kg\n", min)
	fmt.Printf("Berat maximum: %.2f kg\n", max)
	fmt.Printf("Rata-rata berat: %.2f kg\n",
rataRata)
```

```
}
```

## Screenshoot Output




## Deskripsi Program

Program di atas bertujuan untuk mengolah data berat badan balita dan memberikan informasi tentang berat minimum, berat maksimum, serta rata-rata berat dari data yang dimasukkan. Pengguna diminta untuk memasukkan jumlah balita terlebih dahulu, lalu menginput berat masing-masing balita. Program kemudian memproses data untuk menemukan berat terendah, berat tertinggi, serta menghitung total berat balita untuk mendapatkan rata-ratanya