# LAPORAN PRAKTIKUM
# ALGORITMA DAN PEMROGRAMAN 2

## MODUL XII & XIII

## PENGURUTAN 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**

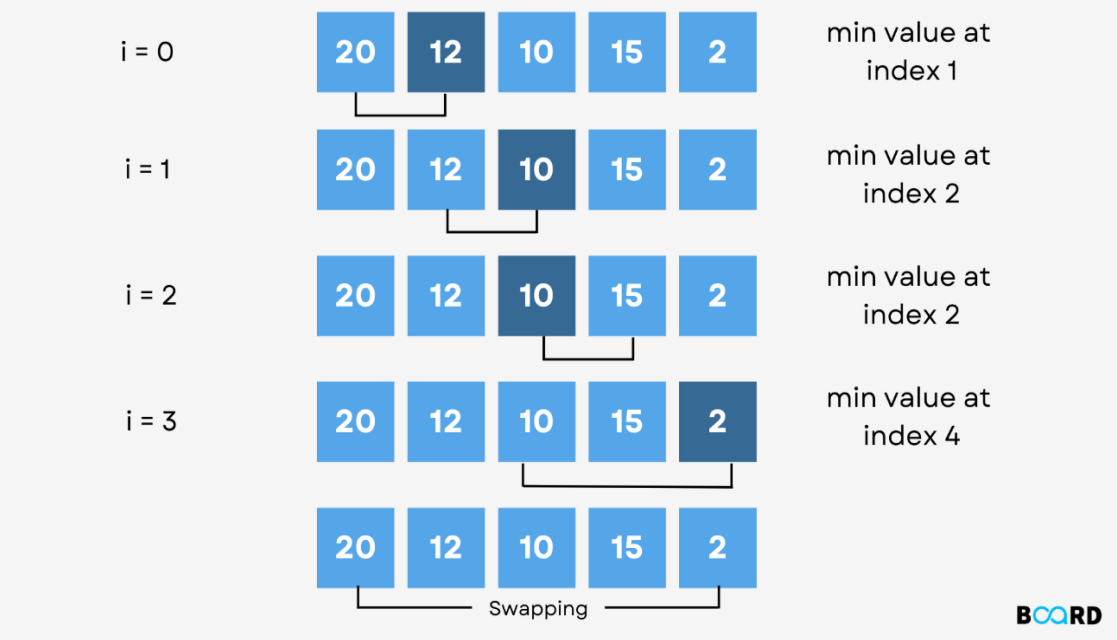
# I. DASAR TEORI

Pengurutan secara seleksi ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian meletakkan pada posisi yang seharusnya. Pada penjelasan berikut ini data akan diurut membesar (ascending), dan data dengan Indeks kecil ada di "kiri" dan Indeks besar ada di "kanan".

1) Cari nilai terkecil di dalam rentang data tersisa
2) Pindahkan/tukar tempat dengan data yang berada pada posisi paling kiri pada rentang data tersisa tersebut.
3) Ulangi proses ini sampai tersisa hanya satu data saja.

pemilihan algoritma pengurutan yang tepat merupakan faktor kritis yang dapat memengaruhi kinerja keseluruhan aplikasi. Setiap algoritma pengurutan memiliki karakteristik unik dalam hal kompleksitas waktu dan ruang, sehingga pengembang perlu mempertimbangkan dengan cermat struktur data, ukuran kumpulan data, dan kebutuhan spesifik sistem. Misalnya, untuk kumpulan data kecil, algoritma sederhana seperti Bubble Sort atau Insertion Sort mungkin cukup efisien, namun untuk dataset besar, algoritma seperti Merge Sort atau Quick Sort dengan kompleksitas O(n log n) jauh lebih disarankan karena kemampuannya menangani volume data yang besar dengan kinerja yang optimal.

## Cara Kerja Selection Short



# I. GUIDED

**Soal Studi Case**

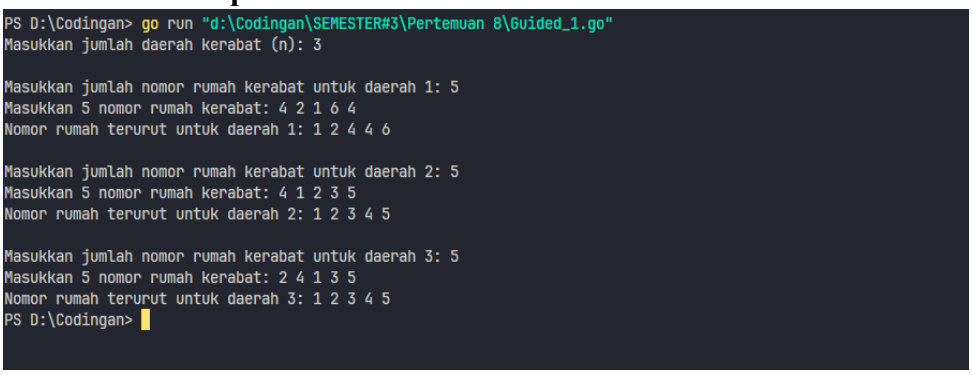
Hercules, preman terkenal seantero ibukota, memiliki kerabat di banyak daerah. Tentunya Hercules sangat suka mengunjungi semua kerabatnya itu. Diberikan masukan nomor rumah dari semua kerabatnya di suatu daerah, buatlah program rumahkerabat yang akan menyusun nomor-nomor rumah kerabatnya secara terurut membesar menggunakan algoritma selection sort..

**Sourcecode**

```
package main

import (
	"fmt"
)

// Fungsi untuk mengurutkan array menggunakan
Selection Sort
func selectionSort(arr []int, n int) {
	for i := 0; i < n-1; i++ {
		idxMin := i
		for j := i + 1; j < n; j++ {
			// Cari elemen terkecil
			if arr[j] < arr[idxMin] {
				idxMin = j
			}
		}
		// Tukar elemen terkecil dengan elemen
di posisi i
		arr[i], arr[idxMin] = arr[idxMin],
arr[i]
	}
}

func main() {
	var n int
	fmt.Print("Masukkan jumlah daerah kerabat (n):
")
	fmt.Scan(&n)

	// Proses tiap daerah
	for daerah := 1; daerah <= n; daerah++ {
		var m int
		fmt.Printf("\nMasukkan jumlah nomor
rumah kerabat untuk daerah %d: ", daerah)
		fmt.Scan(&m)

		// Membaca nomor rumah untuk daerah ini
		arr := make([]int, m)
		fmt.Printf("Masukkan %d nomor rumah
kerabat: ", m)
		for i := 0; i < m; i++ {
			fmt.Scan(&arr[i])
		}
```

```
                // Urutkan array dari terkecil ke
        terbesar
                selectionSort(arr, m)

                // Tampilkan hasil
                fmt.Printf("Nomor rumah terurut untuk
        daerah %d: ", daerah)
                for _, num := range arr {
                        fmt.Printf("%d ", num)
                }
                fmt.Println()
            }
}
```

**Screenshoot Output**




**Deskripsi Program**

Program ini bertujuan untuk mengurutkan nomor rumah kerabat di beberapa daerah menggunakan algoritma Selection Sort. Pengguna dapat memasukkan jumlah daerah dan nomor rumah untuk setiap daerah secara dinamis, lalu program akan mengurutkan nomor-nomor tersebut dari yang terkecil hingga terbesar.

# II. GUIDED

## Soal Studi Case

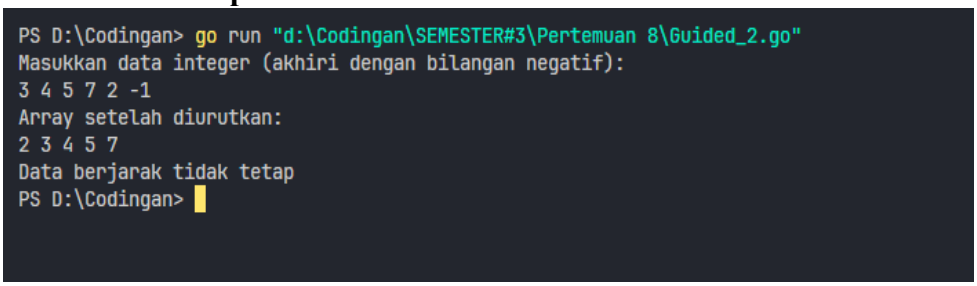
Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metoda insertion sort), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.

## Sourcecode

```
package main

import (
	"fmt"
)

// Fungsi untuk mengurutkan array menggunakan
Insertion Sort
func insertionSort(arr []int, n int) {
	for i := 1; i < n; i++ {
		key := arr[i]
		j := i - 1

		// Geser elemen yang lebih besar dari
key ke kanan
		for j >= 0 && arr[j] > key {
			arr[j+1] = arr[j]
			j--
		}
		arr[j+1] = key
	}
}

// Fungsi untuk memeriksa apakah selisih elemen
array tetap
func isConstantDifference(arr []int, n int) (bool,
int) {
	if n < 2 {
		return true, 0
	}

	difference := arr[1] - arr[0]
	for i := 1; i < n-1; i++ {
		if arr[i+1]-arr[i] != difference {
			return false, 0
		}
	}
	return true, difference
}

func main() {
	var arr []int
```

```
	var num int

	// Input data hingga bilangan negatif
	ditemukan
	fmt.Println("Masukkan data integer (akhiri
	dengan bilangan negatif):")
	for {
		fmt.Scan(&num)
		if num < 0 {
			break
		}
		arr = append(arr, num)
	}

	n := len(arr)

	// Urutkan array menggunakan Insertion Sort
	insertionSort(arr, n)

	// Periksa apakah selisih elemen tetap
	isConstant, difference :=
	isConstantDifference(arr, n)

	// Tampilkan hasil pengurutan
	fmt.Println("Array setelah diurutkan:")
	for _, val := range arr {
		fmt.Printf("%d ", val)
	}
	fmt.Println()

	// Tampilkan status jarak
	if isConstant {
		fmt.Printf("Data berjarak %d\n",
	difference)
	} else {
		fmt.Println("Data berjarak tidak tetap")
	}
}
```

**Screenshoot Output**



**Deskripsi Program**

Program ini dirancang untuk membaca sekumpulan bilangan bulat, mengurutkannya menggunakan algoritma **Insertion Sort**, lalu memeriksa apakah elemen-elemen dalam array memiliki pola dengan selisih tetap. Pengguna dapat memasukkan data secara dinamis hingga bilangan negatif ditemukan sebagai tanda akhir input. Setelah data diurutkan, program akan menentukan dan menampilkan hasil pengurutan serta status pola aritmatika, lengkap dengan nilai selisihnya jika berlaku.

# III. UNGUIDED

**Soal Studi Case**

Belakangan diketahui ternyata Hercules itu tidak berani menyeberang Jalan, maka selalu diusahakan agar hanya menyeberang jalan sesedikit mungkin, hanya diujung jalan. Karena nomor rumah sisi kiri jalan selalu ganjil dan sisi kanan jalan selalu genap, maka buatlah program kerabat dekat yang akan menampilkan nomor rumah mulai dari nomor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

**Sourcecode**

```
package main

import (
	"fmt"
	"sort"
)

func main() {
	var n int
	fmt.Print("Masukkan jumlah: ")
	fmt.Scan(&n)

	for daerah := 1; daerah <= n; daerah++ {
		var m int
		fmt.Printf("Masukkan jumlah nomor %d: ",
daerah)
		fmt.Scan(&m)

		arr := make([]int, m)
		fmt.Printf("Masukkan %d nomor rumah: ",
m)
		for i := 0; i < m; i++ {
			fmt.Scan(&arr[i])
		}

		ganjil := []int{}
		genap := []int{}
		for _, num := range arr {
			if num%2 == 1 {
				ganjil = append(ganjil, num)
			} else {
				genap = append(genap, num)
			}
		}

		sort.Ints(ganjil)

	sort.Sort(sort.Reverse(sort.IntSlice(genap)))
```

```
                fmt.Printf("Nomor rumah terurut untuk
      daerah %d: ", daerah)
                for _, num := range ganjil {
                      fmt.Printf("%d ", num)
                }
                for _, num := range genap {
                      fmt.Printf("%d ", num)
                }
                fmt.Println()
          }
}
```

**Screenshoot Output**

```
Masukkan jumlah: 3
Masukkan jumlah nomor 1: 5 2 1 7 9 13
Masukkan 5 nomor rumah: Nomor rumah terurut untuk daerah 1: 1 7 9 13 2
Masukkan jumlah nomor 2: 6 189 15 27 39 75 133
Masukkan 6 nomor rumah: Nomor rumah terurut untuk daerah 2: 15 27 39 75 133 189
Masukkan jumlah nomor 3: 3 4 9 1
Masukkan 3 nomor rumah: Nomor rumah terurut untuk daerah 3: 1 9 4
PS D:\Codingan>
```

**Deskripsi Program**

Program ini memungkinkan pengguna untuk memasukkan sejumlah nomor rumah dari beberapa daerah dan kemudian mengurutkan nomor rumah berdasarkan dua kategori: bilangan ganjil dan bilangan genap. Setiap kali pengguna memasukkan nomor rumah untuk sebuah daerah, program akan memisahkan nomor rumah menjadi dua kelompok: satu untuk bilangan ganjil dan satu lagi untuk bilangan genap

# IV. UNGUIDED

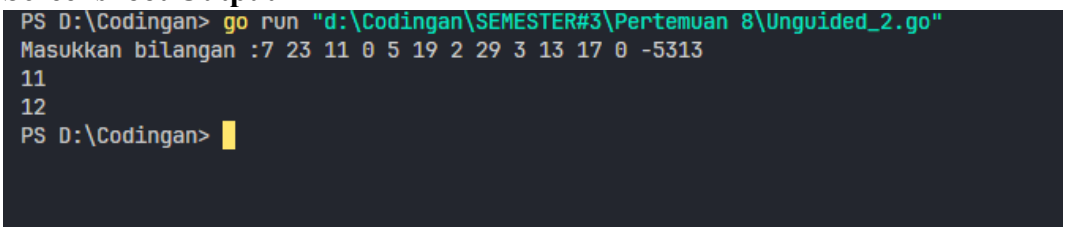
**Soal Studi Case**

Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi ternama. Dalam kompetisi tersebut, setiap tim berlomba untuk menyelesaikan sebanyak mungkin problem yang diberikan. Dari 13 problem yang diberikan, ada satu problem yang menarik. Problem tersebut mudah dipahami, hampir semua tim mencoba untuk menyelesaikannya, tetapi hanya 3 tim yang berhasil. Apa sih problemnya?

**Sourcecode**

```
	package main

	import (
		"fmt"
		"sort"
	)

	func hitungMedian(daftarBilangan []int) int {
		jumlahBilangan := len(daftarBilangan)

		if jumlahBilangan%2 == 1 {
			return daftarBilangan[jumlahBilangan/2]
		}

		return (daftarBilangan[(jumlahBilangan/2)-1] +
	daftarBilangan[jumlahBilangan/2]) / 2
	}

	func main() {
		var inputData int
		var dataBill []int

		fmt.Print("Masukkan bilangan :")
		for {
			fmt.Scan(&inputData)
			if inputData == 0 {
				if len(dataBill) > 0 {
					sort.Ints(dataBill)
					median :=
	hitungMedian(dataBill)
					fmt.Printf("%d\n", median)
				}
			} else {
				dataBill = append(dataBill,
	inputData)
			}
		}
}
```

**Screenshoot Output**




**DeskripsiProgram**

Program ini dirancang untuk membantu menghitung median dari sejumlah bilangan yang dimasukkan oleh pengguna secara interaktif. Pengguna dapat memasukkan bilangan satu per satu, dan setiap kali angka 0 dimasukkan, program akan langsung menghitung dan menampilkan median dari bilangan yang telah dimasukkan sebelumnya

# V. UNGUIDED

**Soal Studi Case**

Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Misalnya terdefinisi struct dan array seperti berikut ini

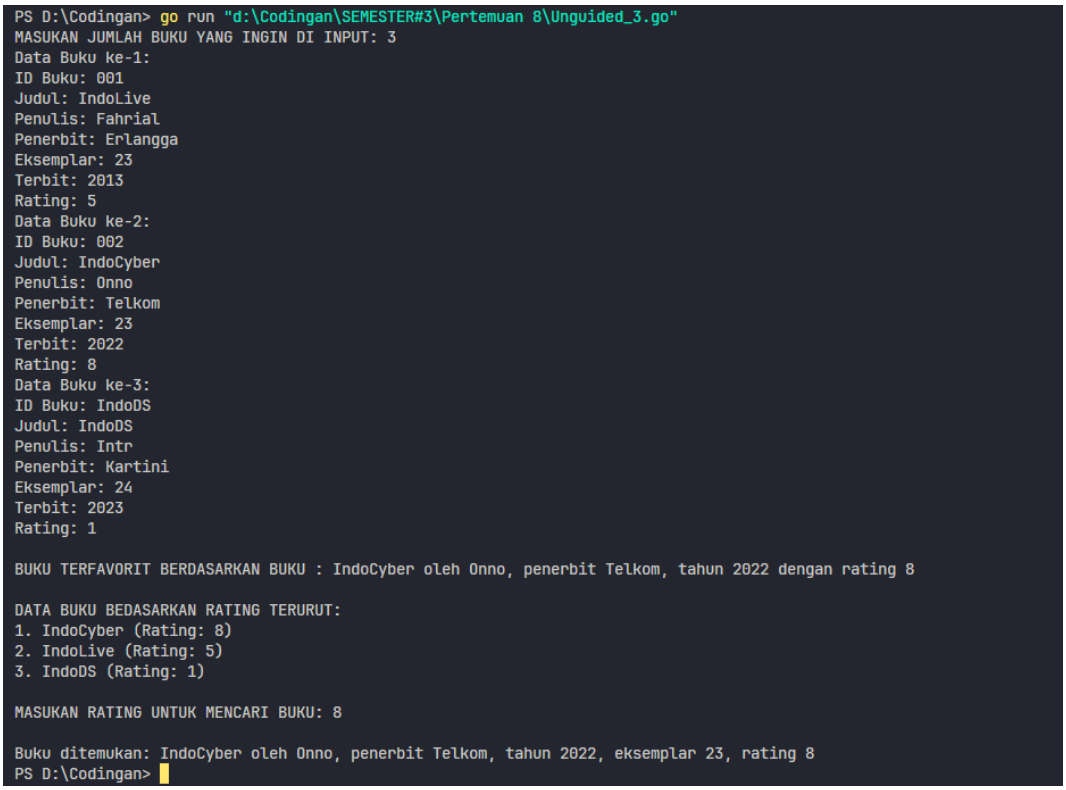
**Sourcecode**

```
package main

import (
	"fmt"
)

const NMAX int = 7919

type Buku struct {
	id, judul, penulis, penerbit string
	eksemplar, tahun, rating     int
}

func main() {
	var n int
	var pustaka [NMAX]Buku

	fmt.Print("MASUKAN JUMLAH BUKU YANG INGIN DI
INPUT: ")
	fmt.Scanln(&n)

	if n > NMAX {
		fmt.Println("Jumlah buku tidak boleh
lebih dari", NMAX)
		return
	}

	for i := 0; i < n; i++ {
		fmt.Printf("Data Buku ke-%d:\n", i+1)
		fmt.Print("ID Buku: ")
		fmt.Scanln(&pustaka[i].id)
		fmt.Print("Judul: ")
		fmt.Scanln(&pustaka[i].judul)
		fmt.Print("Penulis: ")
		fmt.Scanln(&pustaka[i].penulis)
		fmt.Print("Penerbit: ")
		fmt.Scanln(&pustaka[i].penerbit)
		fmt.Print("Eksemplar: ")
		fmt.Scanln(&pustaka[i].eksemplar)
		fmt.Print("Terbit: ")
		fmt.Scanln(&pustaka[i].tahun)
		fmt.Print("Rating: ")
		fmt.Scanln(&pustaka[i].rating)
```

```
	}

	var bukuFav Buku
	bukuFav = pustaka[0]
	for i := 1; i < n; i++ {
		if pustaka[i].rating > bukuFav.rating {
			bukuFav = pustaka[i]
		}
	}

	fmt.Printf("\nBUKU TERFAVORIT BERDASARKAN BUKU
: %s oleh %s, penerbit %s, tahun %d dengan rating
%d\n",
		bukuFav.judul, bukuFav.penulis,
bukuFav.penerbit, bukuFav.tahun, bukuFav.rating)

	for i := 0; i < n-1; i++ {
		for j := i + 1; j < n; j++ {
			if pustaka[i].rating <
pustaka[j].rating {
				pustaka[i], pustaka[j] =
pustaka[j], pustaka[i]
			}
		}
	}
	fmt.Println("\nDATA BUKU BEDASARKAN RATING
TERURUT:")
	limit := n
	if n > 5 {
		limit = 5
	}
	for i := 0; i < limit; i++ {
		fmt.Printf("%d. %s (Rating: %d)\n", i+1,
pustaka[i].judul, pustaka[i].rating)
	}
	var rating int
	fmt.Print("\nMASUKAN RATING UNTUK MENCARI
BUKU: ")
	fmt.Scanln(&rating)

	found := false
	for i := 0; i < n; i++ {
		if pustaka[i].rating == rating {
			fmt.Printf("\nBuku ditemukan: %s
oleh %s, penerbit %s, tahun %d, eksemplar %d, rating
%d\n",
				pustaka[i].judul,
pustaka[i].penulis, pustaka[i].penerbit,
pustaka[i].tahun, pustaka[i].eksemplar,
pustaka[i].rating)
			found = true
			break
```

```
                    }
                }
                if !found {
                    fmt.Println("\nTidak ada buku dengan
        rating tersebut.")
                }
}
```

## Screenshoot Output




## Deskripsi Program

Program ini memungkinkan pengguna untuk mengelola data buku dengan cara yang sederhana dan terstruktur. Pengguna dapat memasukkan sejumlah buku beserta detailnya seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Setelah itu, program akan menampilkan buku dengan rating tertinggi sebagai buku terfavorit, mengurutkan seluruh buku berdasarkan rating dari yang tertinggi hingga terendah, dan menampilkan lima buku teratas.