# LAPORAN PRAKTIKUM
# ALGORITMA DAN PEMROGRAMAN 2

## MODUL 12

## PENGURUTAN DATA


Oleh:

MUHAMMAD RUSDIYANTO

2311102053

S1IF-11-02


**S1 TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2024**

# I. DASAR TEORI

Sorting adalah proses pengaturan elemen-elemen dalam suatu kumpulan data secara berurutan berdasarkan kriteria tertentu, seperti dari yang terkecil ke terbesar (ascending) atau sebaliknya (descending). Berbagai algoritma sorting telah dikembangkan untuk menyelesaikan masalah ini dengan efisiensi yang berbeda-beda, bergantung pada ukuran data dan kebutuhan aplikasi.

Selection sort adalah algoritma sorting sederhana yang bekerja dengan cara menemukan elemen terkecil (atau terbesar, tergantung kebutuhan) dalam array dan menempatkannya di posisi yang sesuai secara berulang hingga seluruh array terurut. Prosesnya melibatkan dua bagian dalam array: bagian yang sudah diurutkan dan bagian yang belum. Pada setiap iterasi, algoritma mencari elemen minimum dari bagian yang belum diurutkan, kemudian menukarnya dengan elemen di awal bagian tersebut. Meskipun mudah dipahami dan diimplementasikan, selection sort kurang efisien untuk data dalam jumlah besar karena memiliki kompleksitas waktu sebesar $O(n^2)$.

Insertion sort adalah algoritma sorting yang bekerja dengan membangun array terurut secara bertahap. Setiap elemen dari array dimasukkan ke dalam posisi yang benar dalam bagian array yang sudah diurutkan. Algoritma ini sangat efisien untuk data kecil atau data yang sebagian besar sudah terurut, karena hanya membutuhkan sedikit pergeseran elemen dalam skenario tersebut. Prosesnya melibatkan pengambilan elemen dari bagian yang belum diurutkan, membandingkannya dengan elemen di bagian terurut, dan menyisipkannya di posisi yang tepat. Dengan kompleksitas waktu $O(n^2)$ dalam kasus rata-rata dan terburuk, insertion sort lebih cepat dibanding selection sort untuk data kecil atau hampir terurut karena jumlah perbandingan dan perpindahan elemen yang lebih sedikit.

## II. GUIDED

Guided 1 | Source Code

```
package main

import "fmt"

// Fungsi untuk mengurutkan array menggunakan selection sort
func selectionSort(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        maxIdx := i
        for j := i + 1; j < n; j++ {
            if arr[j] > arr[maxIdx] { // Cari elemen terbesar
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx], arr[i] // Tukar elemen
    }
}

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

    if n <= 0 || n >= 1000 {
        fmt.Println("n harus lebih besar dari 0 dan kurang dari 1000.")
        return
    }

    for i := 0; i < n; i++ {
        var m int
        fmt.Printf("Masukkan jumlah rumah kerabat untuk daerah ke-%d:
", i+1)
        fmt.Scan(&m)

        if m <= 0 || m >= 1000000 {
            fmt.Println("m harus lebih besar dari 0 dan kurang dari
1000000.")
            return
        }

        // Masukkan nomor rumah
        houses := make([]int, m)
```

```
        fmt.Printf("Masukkan nomor rumah kerabat untuk daerah ke-%d:
", i+1)
        for j := 0; j < m; j++ {
            fmt.Scan(&houses[j])
        }

        // Urutkan dengan selection sort
        selectionSort(houses)

        // Cetak hasil
        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ", i+1)
        for _, house := range houses {
            fmt.Printf("%d ", house)
        }
        fmt.Println()
    }
}
```

Guided 1 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day7> go run guided1.go
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 3
Masukkan nomor rumah kerabat untuk daerah ke-1: 3
4
5
Hasil urutan rumah untuk daerah ke-1: 5 4 3
Masukkan jumlah rumah kerabat untuk daerah ke-2: 2
Masukkan nomor rumah kerabat untuk daerah ke-2: 4
5
Hasil urutan rumah untuk daerah ke-2: 5 4
Masukkan jumlah rumah kerabat untuk daerah ke-3: 1
Masukkan nomor rumah kerabat untuk daerah ke-3: 1
Hasil urutan rumah untuk daerah ke-3: 1
```

Guided 1 | Penjelasan

Program ini berfungsi untuk mengurutkan nomor rumah di beberapa daerah menggunakan algoritma selection sort. Pengguna diminta untuk memasukkan jumlah total daerah, yang harus berada dalam rentang tertentu untuk validasi. Selanjutnya, untuk setiap daerah, pengguna diminta memasukkan jumlah rumah kerabat yang ada, juga dengan validasi jumlah dalam rentang tertentu.

Setelah jumlah rumah dimasukkan, program meminta pengguna memasukkan nomor rumah untuk daerah tersebut. Nomor-nomor rumah disimpan dalam array dinamis, dan kemudian diurutkan secara menurun menggunakan algoritma selection sort. Algoritma ini bekerja dengan mencari elemen terbesar di bagian yang belum diurutkan, lalu menukarnya dengan elemen di awal bagian tersebut. Proses ini diulangi hingga seluruh array terurut. Setelah proses pengurutan selesai, program mencetak nomor rumah yang sudah terurut untuk masing-masing daerah.

Guided 2 | Source Code

```
package main

import (
    "fmt"
    "math"
)

// Fungsi insertion sort untuk mengurutkan array
func insertionSort(arr []int) {
    n := len(arr)
    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 data berjarak tetap
func isDataConsistentlySpaced(arr []int) (bool, int) {
    if len(arr) < 2 {
        return true, 0 // Array dengan kurang dari 2 elemen dianggap
berjarak tetap
    }

    // Hitung selisih awal
    diff := int(math.Abs(float64(arr[1] - arr[0])))

    for i := 1; i < len(arr)-1; i++ {
        currentDiff := int(math.Abs(float64(arr[i+1] - arr[i])))
        if currentDiff != diff {
            return false, 0 // Jika ada selisih yang berbeda, tidak berjarak
tetap
        }
    }

    return true, diff
}
```

```
func main() {
    var data []int
    var input int

    fmt.Println("Masukkan data (akhiri dengan bilangan negatif):")
    for {
        fmt.Scan(&input)
        if input < 0 {
            break
        }
        data = append(data, input)
    }

    // Urutkan data menggunakan insertion sort
    insertionSort(data)

    // Periksa apakah data berjarak tetap
    isConsistent, diff := isDataConsistentlySpaced(data)

    // Cetak hasil
    fmt.Println("Hasil pengurutan:", data)
    if isConsistent {
        fmt.Printf("Data berjarak %d\n", diff)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}
```

Guided 2 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day7> go run guided2.go
Masukkan data (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6
```

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day7> go run guided2.go
Masukkan data (akhiri dengan bilangan negatif):
4 40 14 8 26 1 38 2 32 -31
Hasil pengurutan: [1 2 4 8 14 26 32 38 40]
Data berjarak tidak tetap
```

Guided 2 | Penjelasan

Program ini mengimplementasikan algoritma **insertion sort** untuk mengurutkan array angka yang dimasukkan pengguna. Selain itu, program memeriksa apakah elemen-elemen array memiliki jarak yang tetap.

Pengguna diminta untuk memasukkan serangkaian bilangan yang akan dimasukkan ke dalam array. Proses input terus berlanjut hingga pengguna memasukkan bilangan negatif, yang menandakan akhir dari data input. Data yang dimasukkan akan disimpan dalam array dinamis.

Array tersebut kemudian diurutkan menggunakan algoritma insertion sort. Algoritma ini bekerja dengan memindahkan elemen ke posisi yang sesuai dalam bagian array yang sudah diurutkan. Elemen-elemen yang lebih besar dari elemen yang sedang diproses digeser ke kanan untuk memberikan tempat pada elemen tersebut.

Setelah data diurutkan, program memeriksa apakah elemen-elemen array memiliki jarak tetap. Untuk melakukan ini, program menghitung selisih (difference) antara elemen-elemen yang berurutan. Selisih awal dihitung sebagai patokan, kemudian dibandingkan dengan selisih setiap pasangan elemen lainnya. Jika semua selisih sama, data dianggap berjarak tetap; jika tidak, data dianggap tidak memiliki jarak tetap.

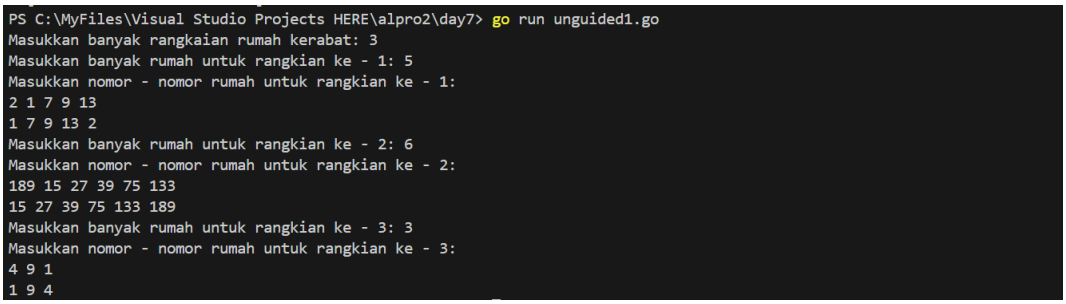
## III. UNGUIDED

Unguided 1 | Source Code

```
package main

import "fmt"

func printArray(arr []int) {
    n := len(arr)
    if n == 0 {
        return
    }
    for i := 0; i < n; i++ {
        fmt.Printf("%v ", arr[i])
    }
}

func sortArray(arr *[]int, n int) {
    if n == 0 {
        return
    }

    if (*arr)[0]%2 == 0 {
        for i := 0; i < n-1; i++ {
            maxIdx := i
            for j := i + 1; j < n; j++ {
                if (*arr)[j] > (*arr)[maxIdx] { // Cari elemen terbesar
                    maxIdx = j
                }
            }
            (*arr)[i], (*arr)[maxIdx] = (*arr)[maxIdx], (*arr)[i] // Tukar
elemen
        }
    } else {
        for i := 0; i < n-1; i++ {
            minIdx := i
            for j := i + 1; j < n; j++ {
                if (*arr)[j] < (*arr)[minIdx] { // Cari elemen terkecil
                    minIdx = j
                }
            }
            (*arr)[i], (*arr)[minIdx] = (*arr)[minIdx], (*arr)[i] // Tukar
elemen
        }
    }
}
```

```
func splitEvenOdd(arr []int, n int) ([]int, []int) {
    var even, odd []int
    for i := 0; i < n; i++ {
        if arr[i]%2 == 0 {
            even = append(even, arr[i])
        } else {
            odd = append(odd, arr[i])
        }
    }
    return even, odd
}

func main() {
    var nIteration int

    fmt.Print("Masukkan banyak rangkaian rumah kerabat: ")
    fmt.Scan(&nIteration)

    for i := 0; i < nIteration; i++ {
        var nHouses int
        fmt.Printf("Masukkan banyak rumah untuk rangkian ke - %v: ",
i+1)
        fmt.Scan(&nHouses)

        arrHouses := make([]int, nHouses)
        fmt.Printf("Masukkan nomor - nomor rumah untuk rangkian ke
- %v:\n", i+1)
        for j := 0; j < nHouses; j++ {
            fmt.Scan(&arrHouses[j])
        }

        evenHouses, oddHouses := splitEvenOdd(arrHouses, nHouses)

        sortArray(&evenHouses, len(evenHouses))
        sortArray(&oddHouses, len(oddHouses))

        printArray(oddHouses)
        printArray(evenHouses)
        fmt.Println()
    }
}
```

Unguided 1 | Output




Unguided 1 | Penjelasan

Kode di atas adalah program Go yang digunakan untuk mengolah data nomor rumah dalam beberapa rangkaian. Program dimulai dengan meminta input berupa jumlah rangkaian nomor rumah dari pengguna. Untuk setiap rangkaian, pengguna diminta memasukkan jumlah rumah dan nomor-nomor rumah tersebut. Program memisahkan nomor-nomor rumah ke dalam dua kelompok yaitu bilangan genap dan bilangan ganjil. Proses ini dilakukan oleh fungsi splitEvenOdd, yang memeriksa setiap nomor rumah dan menambahkan nomor tersebut ke array genap atau ganjil.

Setelah dipisahkan, masing-masing kelompok genap dan ganjil diurutkan dengan aturan yang berbeda menggunakan fungsi sortArray. Jika kelompok pertama (genap) dimulai dengan elemen genap, elemen-elemen diurutkan dalam urutan menurun (descending). Sebaliknya, jika kelompok pertama adalah ganjil, elemen-elemen diurutkan dalam urutan menaik (ascending). Algoritma pengurutan yang digunakan adalah selection sort, yang memindahkan elemen terkecil atau terbesar ke posisi yang sesuai dalam array pada setiap iterasi.

Hasil akhir berupa nomor rumah yang telah diurutkan, yang dicetak ke layar dalam dua kelompok (ganjil terlebih dahulu, lalu genap) untuk setiap rangkaian. Outputnya memberikan gambaran nomor rumah yang diolah berdasarkan pola genap dan ganjil serta aturan urutan yang sesuai.

Unguided 2 | Source Code

```
package main

import "fmt"

func sortArr(arr *[]int, n int) {
    for i := 0; i < n-1; i++ {
```

```
        minIdx := i
        for j := i + 1; j < n; j++ {
            if (*arr)[j] < (*arr)[minIdx] { // Cari elemen terkecil
                minIdx = j
            }
        }
        (*arr)[i], (*arr)[minIdx] = (*arr)[minIdx], (*arr)[i] // Tukar elemen
    }
}

func main() {
    const ARR_MAX int = 1000000
    var arrNum []int
    var elemCount int

    for i := 0; i < ARR_MAX; i++ {
        var temp int
        fmt.Scan(&temp)

        if temp < 0 {
            break
        } else {
            arrNum = append(arrNum, temp)
            elemCount++
        }
    }

    var arrFiltered []int
    for i := 0; i < elemCount; i++ {
        if arrNum[i] == 0 {
            n := len(arrFiltered)

            if n == 0 {
                continue // do nothing
            } else {
                sortArr(&arrFiltered, n)
                var index int = (n - 1) / 2
                if n%2 == 0 {
                    fmt.Print((arrFiltered[index] + arrFiltered[index+1]) / 2)
                } else {
                    fmt.Print(arrFiltered[index])
                }
            }
            fmt.Println(arrFiltered)
        } else {
            arrFiltered = append(arrFiltered, arrNum[i])
```

```
        }
    }
}
```

Unguided 2 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day7> go run unguided2.go
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
```

Unguided 2 | Penjelasan

Program di atas adalah program yang menghitung median dari suatu sub-array yang dipisah oleh angka nol. Program dimulai dengan membaca input bilangan secara terus-menerus ke dalam array arrNum hingga ditemukan bilangan negatif, yang bertindak sebagai penanda akhir input. Jumlah elemen yang dimasukkan dicatat dalam variabel elemCount. Setiap bilangan yang dibaca kemudian diproses. Jika bilangan tersebut nol, program melakukan operasi khusus pada subset bilangan yang telah dikumpulkan sebelumnya dalam array arrFiltered. Subset ini hanya berisi bilangan yang bukan nol.

Setiap kali bilangan nol terdeteksi, array arrFiltered diurutkan menggunakan algoritma selection sort yang diimplementasikan dalam fungsi sortArr. Setelah diurutkan, median dari array dihitung. Median adalah elemen tengah jika jumlah elemen ganjil, atau rata-rata dua elemen tengah jika jumlah elemen genap. Nilai median ini dicetak sebagai hasil untuk subset tersebut, dan proses dilanjutkan hingga elemen terakhir dari arrNum.

Unguided 3 | Source Code

```
package main

import "fmt"

const NMAX int = 9999

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

type DaftarBuku = [NMAX]Buku
```

```
func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    for i := 0; i < n; i++ {
        fmt.Printf("[Buku %v]\n", i+1)
        fmt.Print("Masukkan id buku: ")
        fmt.Scan(&pustaka[i].id)
        fmt.Print("Masukkan judul buku: ")
        fmt.Scan(&pustaka[i].judul)
        fmt.Print("Masukkan penulis buku: ")
        fmt.Scan(&pustaka[i].penulis)
        fmt.Print("Masukkan penerbit buku: ")
        fmt.Scan(&pustaka[i].penerbit)
        fmt.Print("Masukkan eksemplar buku: ")
        fmt.Scan(&pustaka[i].eksemplar)
        fmt.Print("Masukkan tahun buku: ")
        fmt.Scan(&pustaka[i].tahun)
        fmt.Print("Masukkan rating buku: ")
        fmt.Scan(&pustaka[i].rating)
        fmt.Println()
    }
}

func CetakTerFavorit(pustaka DaftarBuku, n int) {
    max := pustaka[0]
    for i := 1; i < n; i++ {
        if pustaka[i].rating > max.rating {
            max = pustaka[i]
        }
    }

    fmt.Println("[Buku Favorit]")
    fmt.Printf("- Judul: %v\n", max.judul)
    fmt.Printf("- Penulis: %v\n", max.penulis)
    fmt.Printf("- Penerbit: %v\n", max.penerbit)
    fmt.Printf("- Tahun: %v\n", max.tahun)
    fmt.Printf("- Rating: %v\n", max.rating)
}

func UrutBuku(pustaka *DaftarBuku, n int) {
    var temp Buku
    for i := 1; i < n; i++ {
        temp = pustaka[i]
        j := i - 1
        for j >= 0 && pustaka[j].rating > temp.rating {
            pustaka[j+1] = pustaka[j]
            j--
        }
```

```
        pustaka[j+1] = temp
    }
}

func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    fmt.Printf("[%v Buku Favorit]\n", n)
    for i := 0; i < n && i < 5; i++ {
        fmt.Printf("%v. %v\n", i+1, pustaka[i].judul)
    }
}

func CariBuku(pustaka DaftarBuku, n int, r int) {
    low, high := 0, n
    for low <= high {
        mid := (low + high) / 2
        if r == pustaka[mid].rating {
            fmt.Printf("[Buku dengan Rating %v]\n", r)
            fmt.Printf("- Judul: %v\n", pustaka[mid].judul)
            fmt.Printf("- Penulis: %v\n", pustaka[mid].penulis)
            fmt.Printf("- Penerbit: %v\n", pustaka[mid].penerbit)
            fmt.Printf("- Tahun: %v\n", pustaka[mid].tahun)
            fmt.Printf("- Eksemplar: %v\n", pustaka[mid].eksemplar)
            fmt.Printf("- Rating: %v\n", pustaka[mid].rating)
            return
        } else if r > pustaka[mid].rating {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }
    fmt.Println("Tidak ada buku dengan rating seperti itu.")
}

func main() {
    var pustaka DaftarBuku
    var nPustaka, rBuku int

    fmt.Print("Masukkan jumlah buku: ")
    fmt.Scan(&nPustaka)

    DaftarkanBuku(&pustaka, nPustaka)
    CetakTerFavorit(pustaka, nPustaka)
    UrutBuku(&pustaka, nPustaka)
    Cetak5Terbaru(pustaka, nPustaka)

    fmt.Print("Masukkan rating buku yang ingin dicari: ")
```

```
    fmt.Scan(&rBuku)

    CariBuku(pustaka, nPustaka, rBuku)
}
```

Unguided 3 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day7> go run unguided1.go
Masukkan jumlah buku: 3
[Buku 1]
Masukkan id buku: 1
Masukkan judul buku: Abadar
Masukkan penulis buku: Abidin
Masukkan penerbit buku: Alaidin
Masukkan eksemplar buku: 3
Masukkan tahun buku: 2004
Masukkan rating buku: 6
```

```
[Buku 2]
Masukkan id buku: 2
Masukkan judul buku: Modernlingga
Masukkan penulis buku: Oh
Masukkan penerbit buku: Owh
Masukkan eksemplar buku: 12
Masukkan tahun buku: 2005
Masukkan rating buku: 10
```

```
[Buku 3]
Masukkan id buku: 3
Masukkan judul buku: Kotamuda
Masukkan penulis buku: Ptoh
Masukkan penerbit buku: PisangPublish
Masukkan eksemplar buku: 6
Masukkan tahun buku: 2023
Masukkan rating buku: 6
```

```
[Buku Favorit]
- Judul: Modernlingga
- Penulis: Oh
- Penerbit: Owh
- Tahun: 2005
- Rating: 10
[3 Buku Favorit]
1. Abadar
2. Kotamuda
3. Modernlingga
Masukkan rating buku yang ingin dicari: 3
Tidak ada buku dengan rating seperti itu.
```

Unguided 3 | Penjelasan

Program ini mengelola daftar buku dalam sebuah pustaka dengan berbagai fitur, seperti pendaftaran buku, pencarian berdasarkan rating, pengurutan buku, dan pencetakan buku terfavorit maupun terbaru. Berikut penjelasan rinci cara kerja setiap bagian:

Pengguna diminta untuk memasukkan jumlah buku yang ingin didaftarkan. Kemudian, untuk setiap buku, pengguna menginput detail seperti ID buku, judul, penulis, penerbit, jumlah eksemplar, tahun, dan rating. Data ini disimpan dalam array DaftarBuku.

Setelah semua data dimasukkan, program menentukan buku dengan rating tertinggi menggunakan fungsi CetakTerFavorit. Fungsi ini mencari buku dengan nilai rating tertinggi dalam array dan mencetak detail buku tersebut.

Program juga mengurutkan daftar buku berdasarkan rating menggunakan insertion sort dalam fungsi UrutBuku. Dalam algoritma ini, setiap elemen dibandingkan dan ditempatkan pada posisi yang tepat dalam bagian array yang sudah diurutkan. Hasilnya adalah daftar buku terurut dari rating terendah ke tertinggi.

Setelah diurutkan, fungsi Cetak5Terbaru mencetak hingga lima buku pertama dari daftar terurut, yang merupakan buku dengan rating terendah (karena array diurutkan menaik berdasarkan rating).

Pengguna juga dapat mencari buku berdasarkan rating tertentu menggunakan fungsi CariBuku. Fungsi ini memanfaatkan algoritma binary search untuk menemukan buku dengan rating yang dimasukkan pengguna. Karena array sudah diurutkan, binary search dapat menemukan data dengan cepat menggunakan strategi pembagian.