

**LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2**

**MODUL 11
PENGURUTAN DATA**



Oleh:

ANANDA BASKORO PUTRA

2311102187

IF 11 02

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO 202**

I. DASAR TEORI

Pengurutan data (sorting) adalah salah satu konsep fundamental dalam ilmu komputer dan pemrograman, yang bertujuan untuk mengatur elemen dalam suatu struktur data berdasarkan suatu kriteria tertentu, seperti urutan menaik (*ascending*) atau menurun (*descending*). Pengurutan mempermudah berbagai operasi pada data, seperti pencarian, penghapusan duplikat, atau pengelompokan data berdasarkan kategori. Teknik ini banyak digunakan di berbagai aplikasi seperti sistem manajemen basis data, pengolahan dokumen, dan sistem pencarian.

Berbagai algoritma pengurutan telah dikembangkan, masing-masing dengan karakteristik, keunggulan, dan kelemahannya. Algoritma seperti *selection sort*, *insertion sort*, dan *bubble sort* tergolong sederhana dan cocok untuk dataset kecil karena mudah diimplementasikan. Namun, algoritma ini memiliki kompleksitas waktu rata-rata, yang kurang efisien untuk dataset besar. Di sisi lain, algoritma seperti *merge sort*, *quick sort*, dan *heap sort* menawarkan kompleksitas waktu lebih baik, yang membuatnya lebih cocok untuk pengurutan dataset skala besar.

Pemilihan algoritma pengurutan bergantung pada kebutuhan spesifik, seperti ukuran data, struktur data yang digunakan, atau batasan waktu dan ruang. Misalnya, *quick sort* cenderung lebih cepat dalam kasus rata-rata tetapi memiliki risiko kinerja buruk pada kasus tertentu, sementara *merge sort* lebih stabil dalam hal performa tetapi membutuhkan ruang tambahan untuk proses rekursif. Oleh karena itu, memahami algoritma pengurutan dan karakteristiknya menjadi kunci untuk memilih metode yang paling efisien sesuai konteks aplikasi yang sedang dikembangkan.

I. GUIDED

Source Code + Screenshot hasil program beserta penjelasan

Guided 1

```
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()  
  
}  
  
}
```

Screenshoot Output

```
PS E:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11> go run "e:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11\Guided\Guided1.go"  
Masukkan jumlah daerah (n): 2  
Masukkan jumlah rumah kerabat untuk daerah ke-1: 3  
Masukkan nomor rumah kerabat untuk daerah ke-1: 1 5 12  
Hasil urutan rumah untuk daerah ke-1: 12 5 1  
Masukkan jumlah rumah kerabat untuk daerah ke-2: 5  
Masukkan nomor rumah kerabat untuk daerah ke-2: 15 25 35 45 555  
Hasil urutan rumah untuk daerah ke-2: 555 45 35 25 15  
PS E:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11> █
```

Deskripsi Program

Program ini merupakan implementasi sorting dengan metode selection sort untuk mengurutkan nomor rumah kerabat dalam setiap daerah berdasarkan urutan menurun. Pengguna diminta untuk memasukkan jumlah daerah (n) dan jumlah rumah kerabat (m) di setiap daerah, dengan batasan nilai n (1-999) dan m (1-999999). Nomor rumah kerabat diinput sebagai array, kemudian diurutkan menggunakan selection sort. Setelah proses pengurutan selesai, program akan mencetak daftar nomor rumah yang telah diurutkan untuk setiap daerah.

Guided 2

```
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)

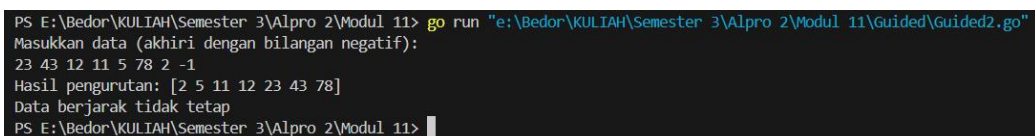
} else {

    fmt.Println("Data berjarak tidak tetap")

}

}
```

Screenshoot Output



```
PS E:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11> go run "e:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11\Guided\guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
23 43 12 11 5 78 2 -1
Hasil pengurutan: [2 5 11 12 23 43 78]
Data berjarak tidak tetap
PS E:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11> █
```

Deskripsi Program

Program ini mengurutkan data bilangan bulat menggunakan metode insertion sort dan memeriksa apakah data tersebut memiliki jarak yang konsisten antara elemen-elemennya. Pengguna dapat memasukkan bilangan secara berurutan hingga memasukkan bilangan negatif untuk menghentikan input. Setelah data diurutkan, program menghitung selisih antar elemen secara berurutan untuk memeriksa apakah semua selisih sama. Jika data memiliki jarak konsisten, program menampilkan jarak tersebut; jika tidak, program menyatakan bahwa data tidak memiliki jarak tetap.

II. UNGUIDED

Source Code + Screenshot hasil program beserta penjelasan

Unguided 1

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array secara ascending menggunakan selection sort
func selectionSortAsc(arr []int) {
    n := len(arr)
    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
    }
}

// Fungsi untuk mengurutkan array secara descending menggunakan selection sort
func selectionSortDesc(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("\nMasukkan jumlah rumah untuk daerah ke-%d: ", i+1)

        fmt.Scan(&m)

```

```
if m <= 0 {  
  
    fmt.Println("Jumlah rumah harus lebih besar dari 0.")  
  
    return  
  
}  
  
// Membaca array angka sebagai input  
  
fmt.Printf("Masukkan nomor rumah untuk daerah ke-%d:\n", i+1)  
  
houses := make([]int, m)  
  
for j := 0; j < m; j++ {  
  
    fmt.Scan(&houses[j])  
  
}  
  
// Pisahkan bilangan ganjil dan genap  
  
var ganjil []int  
  
var genap []int  
  
for _, num := range houses {  
  
    if num%2 == 0 {  
  
        genap = append(genap, num)  
  
    } else {  
  
        ganjil = append(ganjil, num)  
  
    }  
  
}  
  
// Urutkan ganjil (ascending) dan genap (descending)  
  
selectionSortAsc(ganjil)  
  
selectionSortDesc(genap)
```

```
// Cetak hasil

fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ", i+1)

for _, num := range ganjil {

    fmt.Printf("%d ", num)

}

for _, num := range genap {

    fmt.Printf("%d ", num)

}

fmt.Println()

}

}
```

Screenshoot Output

```
PS E:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11> go run "e:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11\Unguided\Unguided1.go"
Masukkan jumlah daerah (n): 3

Masukkan jumlah rumah untuk daerah ke-1: 6
Masukkan nomor rumah untuk daerah ke-1:
1 3 5 8 12 16
Hasil urutan rumah untuk daerah ke-1: 1 3 5 16 12 8

Masukkan jumlah rumah untuk daerah ke-2: 5
Masukkan nomor rumah untuk daerah ke-2:
32 21 46 57 78
Hasil urutan rumah untuk daerah ke-2: 21 57 78 46 32

Masukkan jumlah rumah untuk daerah ke-3: 3
Masukkan nomor rumah untuk daerah ke-3:
100 93 96
Hasil urutan rumah untuk daerah ke-3: 93 100 96
PS E:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11> |
```

Deskripsi Program

Program ini mengurutkan nomor rumah berdasarkan bilangan ganjil dan genap untuk setiap daerah yang dimasukkan oleh pengguna. Nomor rumah ganjil diurutkan secara ascending (menaik), sedangkan nomor rumah genap diurutkan secara descending (menurun). Pengguna diminta memasukkan jumlah daerah dan nomor rumah untuk masing-masing daerah. Setelah itu, program memisahkan nomor rumah menjadi bilangan ganjil dan genap, mengurutkannya sesuai aturan, lalu mencetak hasil pengurutan dalam satu baris, dimulai dari ganjil diikuti genap.

Unguided 2

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan insertion sort
func insertionSort(arr []int) {
    n := len(arr)
    for i := 1; i < n; i++ {
        key := arr[i]
        j := i - 1
        // Pindahkan elemen yang lebih besar dari key ke satu posisi di depan
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
    }
}
```

```

arr[j+1] = key

}

}

// Fungsi untuk menghitung median dari array yang sudah terurut

func getMedian(arr []int) int {

n := len(arr)

if n%2 == 1 {

// Jika jumlah elemen ganjil, median adalah elemen tengah

return arr[n/2]

}

// Jika jumlah elemen genap, median adalah rata-rata dua elemen tengah

return (arr[n/2-1] + arr[n/2]) / 2

}

func main() {

var input int

data := []int{} // Array untuk menyimpan data yang valid

fmt.Println("Masukkan bilangan bulat (akhiri dengan -5313):")

for {

fmt.Scan(&input)

if input == -5313 {

// Marker untuk mengakhiri program

break

} else if input == 0 {

```

```
// Jika menemukan angka 0, urutkan array dan cetak median

insertionSort(data)

if len(data) > 0 {

median := getMedian(data)

fmt.Println("Median saat ini:", median)

}

} else if input > 0 {

// Hanya tambahkan bilangan bulat positif ke array

data = append(data, input)

}

}

}
```

Screenshoot Output

```
PS E:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11> go run "e:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11\Unguided\Unguided2.go"
Masukkan bilangan bulat (akhiri dengan -5313):
1 4 0 15 23 39 42 89 95 46 73 10 -5313
Median saat ini: 2
PS E:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11> █
```

Deskripsi Program

Program ini menerima input bilangan bulat positif dari pengguna, menghentikan input ketika bilangan khusus -5313 dimasukkan. Jika pengguna memasukkan 0, program akan mengurutkan bilangan yang telah dimasukkan menggunakan insertion sort dan menghitung mediannya. Median dihitung berdasarkan array yang telah terurut, dengan mempertimbangkan jumlah elemen ganjil atau genap. Program hanya menyimpan bilangan positif dan mengabaikan input lainnya. Median diperbarui dan ditampilkan setiap kali 0 dimasukkan.

Unguided 3

```
package main

import (
    "fmt"
)

// Definisi struct untuk Buku

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

// Fungsi untuk menambahkan data buku ke pustaka

func DaftarkanBuku(pustaka []*Buku, n int) {
    for i := 0; i < n; i++ {
        var buku Buku

        fmt.Printf("Masukkan data untuk buku ke-%d (id, judul, penulis, penerbit, eksemplar, tahun, rating):\n", i+1)

        fmt.Scan(&buku.id, &buku.judul, &buku.penulis, &buku.penerbit, &buku.eksemplar, &buku.tahun, &buku.rating)
```



```

*pustaka = append(*pustaka, buku)

}

}

// Fungsi untuk mencetak buku dengan rating tertinggi

func CetakFavorit(pustaka []Buku, n int) {

if len(pustaka) == 0 {

fmt.Println("Pustaka kosong.")

return

}

terfavorit := pustaka[0]

for _, buku := range pustaka {

if buku.rating > terfavorit.rating {

terfavorit = buku

}

}

fmt.Println("Buku dengan rating tertinggi:")

fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Eksemplar: %d,
Tahun: %d, Rating: %d\n",

terfavorit.id, terfavorit.judul, terfavorit.penulis, terfavorit.penerbit,
terfavorit.eksemplar, terfavorit.tahun, terfavorit.rating)

}

// Fungsi untuk mengurutkan array buku berdasarkan rating secara descending

func UrutkanBuku(pustaka *[]Buku, n int) {

for i := 1; i < len(*pustaka); i++ {

```

```

key := (*pustaka)[i]

j := i - 1

for j >= 0 && (*pustaka)[j].rating < key.rating {
    (*pustaka)[j+1] = (*pustaka)[j]
    j--
}

(*pustaka)[j+1] = key
}
}

// Fungsi untuk mencetak lima buku dengan rating tertinggi

func Cetak5Terbaik(pustaka []Buku, n int) {

    fmt.Println("Lima buku dengan rating tertinggi:")

    for i := 0; i < 5 && i < len(pustaka); i++ {

        buku := pustaka[i]

        fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Eksemplar: %d,
        Tahun: %d, Rating: %d\n",

        buku.id, buku.judul, buku.penulis, buku.penerbit, buku.eksemplar, buku.tahun,
        buku.rating)

    }

}

// Fungsi untuk mencari buku dengan rating tertentu

func CariBuku(pustaka []Buku, n int, r int) {

    ditemukan := false

    for _, buku := range pustaka {

```

```
if buku.rating == r {  
  
    ditemukan = true  
  
    fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s, Eksemplar: %d,  
    Tahun: %d, Rating: %d\n",  
  
    buku.id, buku.judul, buku.penulis, buku.penerbit, buku.eksemplar, buku.tahun,  
    buku.rating)  
  
}  
  
}  
  
if !ditemukan {  
  
    fmt.Println("Tidak ada buku dengan rating tersebut.")  
  
}  
  
}  
  
func main() {  
  
    var n int  
  
    fmt.Print("Masukkan jumlah buku di perpustakaan: ")  
  
    fmt.Scan(&n)  
  
    if n <= 0 || n > 7919 {  
  
        fmt.Println("Jumlah buku harus antara 1 hingga 7919.")  
  
        return  
  
    }  
  
    var pustaka []Buku  
  
    // Input data buku  
  
    DaftarkanBuku(&pustaka, n)  
  
    // Cetak buku dengan rating tertinggi
```

```

CetakFavorit(pustaka, n)

// Urutkan buku berdasarkan rating

UrutkanBuku(&pustaka, n)

// Cetak lima buku dengan rating tertinggi

Cetak5Terbaik(pustaka, n)

// Cari buku dengan rating tertentu

var rating int

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

fmt.Scan(&rating)

CariBuku(pustaka, n, rating)

}

```

Screenshoot Output

```

PS E:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11> go run "e:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11\Unguided\Unguided3.go"
Masukkan jumlah buku di perpustakaan: 4
Masukkan data untuk buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 bukuA penulisA penerbitA 200 2021 90
Masukkan data untuk buku ke-2 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 bukuB penulisB penerbitB 210 2021 100
Masukkan data untuk buku ke-3 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
3 bukuC penulisC penerbitC 220 2021 110
Masukkan data untuk buku ke-4 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
4 bukuD penulisC penerbitC 230 2021 120
Buku dengan rating tertinggi:
ID: 4, Judul: bukuD, Penulis: penulisC, Penerbit: penerbitC, Eksemplar: 230, Tahun: 2021, Rating: 120
Lima buku dengan rating tertinggi:
ID: 4, Judul: bukuD, Penulis: penulisC, Penerbit: penerbitC, Eksemplar: 230, Tahun: 2021, Rating: 120
ID: 3, Judul: bukuC, Penulis: penulisC, Penerbit: penerbitC, Eksemplar: 220, Tahun: 2021, Rating: 110
ID: 2, Judul: bukuB, Penulis: penulisB, Penerbit: penerbitB, Eksemplar: 210, Tahun: 2021, Rating: 100
ID: 1, Judul: bukuA, Penulis: penulisA, Penerbit: penerbitA, Eksemplar: 200, Tahun: 2021, Rating: 90
Masukkan rating buku yang ingin dicari: 110
ID: 3, Judul: bukuC, Penulis: penulisC, Penerbit: penerbitC, Eksemplar: 220, Tahun: 2021, Rating: 110
PS E:\Bedor\KULIAH\Semester 3\Alpro 2\Modul 11>

```

Deskripsi Program

Program diatas adalah sistem manajemen perpustakaan untuk mengelola data buku berdasarkan ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Pengguna dapat mendaftarkan sejumlah buku ke dalam pustaka, menampilkan buku dengan rating tertinggi, dan mengurutkan buku berdasarkan rating secara descending. Selain itu, program juga menampilkan lima buku dengan rating tertinggi dan memungkinkan pengguna mencari buku dengan rating tertentu. Dengan batas jumlah buku antara 1 hingga 7919, program membantu mengelola data buku secara efisien untuk keperluan perpustakaan.