

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII & XIII
PENGURUTAN DATA**



Disusun Oleh :

Rendi Widya Anggita/2311102278

S1IF-11-06

Dosen Pengampu :

ABEDNEGO DWI SEPTIADI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data (sorting) adalah proses menyusun elemen-elemen data dalam suatu urutan tertentu, seperti urutan menaik (ascending) atau menurun (descending). Tujuan utama dari pengurutan adalah mempermudah pencarian, analisis, atau manipulasi data dalam suatu struktur.

Insertion Sort

Insertion Sort adalah algoritma pengurutan yang bekerja dengan membagi data menjadi dua bagian: bagian yang terurut dan bagian yang belum terurut. Algoritma ini secara bertahap mengambil elemen dari bagian yang belum terurut, kemudian menyisipkannya ke posisi yang sesuai dalam bagian yang terurut. Proses ini diulang hingga seluruh elemen terurut.

Langkah-langkah:

1. Mulai dari elemen kedua (indeks 1) dalam array, anggap elemen tersebut sebagai elemen yang akan disisipkan.
2. Bandingkan elemen ini dengan elemen-elemen di bagian yang terurut (sebelumnya).
3. Geser elemen-elemen yang lebih besar ke kanan untuk memberi ruang bagi elemen yang akan disisipkan.
4. Tempatkan elemen di posisi yang sesuai.
5. Ulangi langkah-langkah ini hingga semua elemen terurut.

Selection Sort

Selection Sort adalah algoritma pengurutan yang bekerja dengan membagi array menjadi dua bagian: bagian yang terurut dan bagian yang belum terurut. Algoritma ini menemukan elemen terkecil (atau terbesar) dari bagian yang belum terurut, lalu menukarnya dengan elemen pertama dari bagian yang belum terurut. Proses ini diulang hingga seluruh elemen berada di bagian terurut.

Langkah-langkah:

1. Temukan elemen terkecil dalam array yang belum terurut.
2. Tukar elemen terkecil tersebut dengan elemen pertama dari bagian yang belum terurut.
3. Geser batas antara bagian terurut dan bagian belum terurut.
4. Ulangi hingga seluruh elemen berada di bagian terurut.

II. GUIDED

1. Soal Studi Case

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 no rumah kerabat untuk daerah %d: ", daerah)
        fmt.Scan(&m)

        // Membaca no rumah untuk daerah ini
        arr := make([]int, m)
        fmt.Printf("Masukkan %d no 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("No rumah terurut untuk daerah %d: ",
```

```

daerah)
    for _, num := range arr {
        fmt.Printf("%d ", num)
    }
    fmt.Println()
}
}

```

Screenshoot Output

```

PS C:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\U. Alpro 2\modul 12> go run "c:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\U. Alpro 2\modul 12\guided\guided.go"
Masukkan jumlah daerah kerabat (n): 3
Masukkan 3 no rumah kerabat: 9 12 69
No rumah terbesar untuk daerah 1: 9 12 69
PS C:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\U. Alpro 2\modul 12>

```

Deskripsi Program

Program ini dirancang untuk mengelompokkan dan mengurutkan nomor rumah kerabat di beberapa daerah. Yang pertama user akan memasukkan data nomor rumah untuk setiap daerah, lalu program ini akan mengurutkan nomor rumah tersebut dari yang terkecil ke terbesar menggunakan algoritma Selection Sort.

Langkah Kerja Program:

- Input Jumlah Daerah

Program meminta kita untuk memasukkan jumlah daerah yang memiliki kerabat. Misalnya, kita ingin mengurutkan nomor rumah di 3 daerah.
- Input Jumlah dan Nomor Rumah di Setiap Daerah

Untuk setiap daerah, program akan meminta:
- Berapa jumlah nomor rumah kerabat di daerah tersebut.

Nomor rumah kerabat di daerah itu (masukkan satu per satu).

Setelah semua nomor rumah diinput, program akan mengurutkannya dari yang terkecil ke yang terbesar. Misalnya, nomor rumah [42, 15, 78, 11, 34] akan menjadi [11, 15, 34, 42, 78].

2. Soal Studi Case

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

```

PS C:\Users\ASUS\OneDrive\Documents\KuliaH\Semester 3\P. Alpro 2\modul 12> go run "c:\Users\ASUS\OneDrive\Documents\KuliaH\Semester 3\P. Alpro 2\modul 12\guided\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
2 4 5 7 9 -1
Array setelah diurutkan:
2 4 5 7 9
Data berjarak tidak tetap
PS C:\Users\ASUS\OneDrive\Documents\KuliaH\Semester 3\P. Alpro 2\modul 12>

```

Deskripsi Program

Program ini dirancang untuk mengelola daftar bilangan bulat (integer) dengan beberapa fungsi utama, yaitu:

1. Mengurutkan elemen-elemen dalam daftar menggunakan algoritma Insertion Sort.
1. Memeriksa apakah selisih antara elemen-elemen yang telah diurutkan adalah tetap. Jika ya, program akan menampilkan nilai selisih tersebut; jika tidak, akan menampilkan pesan bahwa jaraknya tidak tetap.

Langkah-Langkah Kerja Program :

1. Input Data Angka

Program meminta pengguna untuk memasukkan angka-angka integer satu per satu. Data ini dimasukkan hingga pengguna mengetikkan angka negatif.

2. Pengurutan Data dengan Insertion Sort

Setelah semua angka dikumpulkan, program mengurutkan angka-angka tersebut dari yang terkecil hingga terbesar menggunakan algoritma Insertion Sort. Algoritma ini bekerja dengan cara memindahkan elemen-elemen ke posisi yang tepat dalam array. Contoh: jika pengguna memasukkan [10, 5, 15], algoritma akan mengurutkan angka menjadi [5, 10, 15].

3. Pemeriksaan Pola Selisih Antar Elemen

Setelah data diurutkan, program memeriksa apakah selisih antara elemen-elemen yang berurutan adalah tetap. Pemeriksaan ini dilakukan dengan membandingkan selisih antara dua elemen pertama, dua elemen berikutnya, dan seterusnya. Jika seluruh selisih sama, berarti data memiliki jarak yang tetap. Jika ada selisih yang berbeda, maka data tidak memiliki jarak tetap.

4. Output Hasil Pemrosesan

Program menampilkan:

- Array yang sudah diurutkan.
- Status pola selisih antar elemen:
- Jika jarak tetap, program menampilkan pesan "Data berjarak X", di mana X adalah nilai selisihnya.
- Jika jarak tidak tetap, program menampilkan pesan "Data berjarak tidak tetap".

III. UNGUIDED

1. Soal Studi Case

Sourcecode

```
package main

import "fmt"

func sortNumbers(arr []int, isAsc bool) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        extremeIdx := i
        for j := i + 1; j < n; j++ {
            if (isAsc && arr[j] < arr[extremeIdx]) || (!isAsc
&& arr[j] > arr[extremeIdx]) {
                extremeIdx = j
            }
        }
        arr[i], arr[extremeIdx] = arr[extremeIdx], arr[i]
    }
}

func collectHouseNumbers(areaIndex, numHouses int) ([]int,
[]int) {
    var houseNumbers = make([]int, numHouses)
    fmt.Printf("Masukkan %d no rumah untuk daerah ke-%d: ",
numHouses, areaIndex+1)
    for i := 0; i < numHouses; i++ {
        fmt.Scan(&houseNumbers[i])
    }

    oddNumbers, evenNumbers :=
categorizeNumbers(houseNumbers)
    return oddNumbers, evenNumbers
}

func categorizeNumbers(houseNumbers []int) (odd []int, even
[]int) {
    for _, num := range houseNumbers {
        if num%2 == 0 {
            even = append(even, num)
        } else {
            odd = append(odd, num)
        }
    }
}
```



```

    }
    return odd, even
}

func displaySortedResults(areaIndex int, odd, even []int) {
    sortNumbers(odd, true)
    sortNumbers(even, false)

    fmt.Printf("\nno rumah terurut untuk daerah ke-%d:\n",
areaIndex+1)
    printArray(odd)
    printArray(even)
}

func printArray(arr []int) {
    for _, num := range arr {
        fmt.Printf("%d ", num)
    }
    fmt.Println()
}

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

    for areaIndex := 0; areaIndex < totalAreas; areaIndex++ {
        var numHouses int
        fmt.Printf("Masukkan jumlah rumah kerabat di daerah
ke-%d: ", areaIndex+1)
        fmt.Scan(&numHouses)

        odd, even := collectHouseNumbers(areaIndex,
numHouses)

        displaySortedResults(areaIndex, odd, even)
    }
}

```

Screenshot Output

```

PS C:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\P. Alpro 2\modul 12> go run "c:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\P. Alpro 2\modul 12\unguided\unguided1.g
o"
Masukkan jumlah daerah kerabat: 1
Masukkan jumlah rumah kerabat di daerah ke-1: 3
Masukkan 3 no rumah untuk daerah ke-1: 12 9 69

no rumah terurut untuk daerah ke-1:
9 69
12
PS C:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\P. Alpro 2\modul 12>

```

Deskripsi Program

Program ini bertujuan untuk mengelompokkan dan mengurutkan nomor rumah kerabat di beberapa daerah. Pengguna diminta memasukkan jumlah daerah dan nomor rumah untuk setiap daerah. Program akan memisahkan nomor rumah menjadi dua kelompok: nomor ganjil dan nomor genap. Nomor ganjil diurutkan dalam urutan menaik (kecil ke besar), sedangkan nomor genap diurutkan dalam urutan menurun (besar ke kecil).

Setiap daerah diproses satu per satu. Setelah data nomor rumah diinput, program mengelompokkan nomor ganjil dan genap menggunakan fungsi khusus, lalu mengurutkannya dengan algoritma sederhana. Hasil pengurutan ini ditampilkan secara terpisah untuk masing-masing daerah sehingga data menjadi rapi dan mudah digunakan.

Dengan pendekatan modular, program ini memisahkan setiap proses seperti pengelompokan angka, pengurutan, dan penampilan hasil. Hal ini mempermudah pengelolaan kode dan memungkinkan modifikasi sesuai kebutuhan. Program ini berguna untuk mengatur data nomor rumah secara sistematis, seperti untuk membuat daftar kunjungan kerabat yang terstruktur.

2. Soal Studi Case

Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

func main() {
    var number int
    var numbers []int

    fmt.Println("Masukkan bilangan (akhiri dengan -5313):")
    for {
        fmt.Scan(&number)

        if number == -5313 {
            break
        }

        if number == 0 {
            if len(numbers) > 0 {
                sort.Ints(numbers)
                median := findMedian(numbers)
                fmt.Printf("Median: %d\n", median)
            }
        } else {
            numbers = append(numbers, number)
        }
    }
}

func findMedian(numbers []int) int {
    length := len(numbers)
    if length%2 == 1 {
        return numbers[length/2]
    }
    return (numbers[(length/2)-1] + numbers[length/2]) / 2
}
```

Screenshoot Output



```
PS C:\Users\ASUS\OneDrive\Documents\Uliab\Semester 3\P. Alpro 2\modul 12> go run "c:\Users\ASUS\OneDrive\Documents\Uliab\Semester 3\P. Alpro 2\modul 12\unguided\unguided2.g"
Masukkan bilangan (akhiri dengan -5313):
2 4 5 7 9 -5313
PS C:\Users\ASUS\OneDrive\Documents\Uliab\Semester 3\P. Alpro 2\modul 12>
```

Deskripsi Program

Program ini menerima bilangan dari pengguna untuk menghitung median, yaitu nilai tengah dari data yang telah diurutkan. Jika jumlah data ganjil, median adalah elemen tengah. Jika genap, median adalah rata-rata dari dua elemen tengah. Pengguna dapat menghentikan input dengan angka -5313 atau meminta median sementara dengan mengetik angka nol (0).

Saat angka nol dimasukkan, program akan mengurutkan bilangan yang telah diinput menggunakan pustaka `sort` dan menghitung median. Hasil median ini langsung ditampilkan, sementara input data dapat dilanjutkan hingga pengguna memutuskan untuk berhenti dengan -5313.

Program ini cocok untuk menganalisis data yang terus bertambah secara dinamis, seperti nilai pengukuran atau data survei real-time. Pendekatan ini sederhana namun efektif untuk pengolahan data dengan kebutuhan median secara berkala.

3. Soal Studi Case

Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

type DaftarBuku struct {
    nBuku int
    buku []DataBuku
}

type DataBuku struct {
    id        int
    judul     string
    pengarang string
    tahun     int
    rating    int
}

func DaftarkanBuku(n int) *DaftarBuku {
    return &DaftarBuku{
        nBuku: 0,
        buku:  make([]DataBuku, n),
    }
}

func CetakTerfavorit(pustaka *DaftarBuku, n int) {
    if n == 0 || pustaka.nBuku == 0 {
        return
    }

    // Create temporary slice for sorting
    temp := make([]DataBuku, pustaka.nBuku)
    copy(temp, pustaka.buku[:pustaka.nBuku])

    // Sort by rating (highest first)
    sort.Slice(temp, func(i, j int) bool {
        return temp[i].rating > temp[j].rating
    })

    fmt.Println("Buku Terfavorit:")
    fmt.Printf("Judul: %s, Pengarang: %s, Tahun: %d, Rating: %d\n",
```

```

        temp[0].judul, temp[0].pengarang, temp[0].tahun,
temp[0].rating)
    }

func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < pustaka.nBuku; i++ {
        key := pustaka.buku[i]
        j := i - 1

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

func Cetak5Terbaru(pustaka *DaftarBuku, n int) {
    fmt.Println("5 Buku Rating Tertinggi:")
    limit := 5
    if pustaka.nBuku < 5 {
        limit = pustaka.nBuku
    }

    for i := 0; i < limit; i++ {
        fmt.Printf("%d. Judul: %s, Rating: %d\n",
            i+1, pustaka.buku[i].judul,
pustaka.buku[i].rating)
    }
}

func CariBuku(pustaka *DaftarBuku, n int, r int) {
    left := 0
    right := pustaka.nBuku - 1
    found := false

    for left <= right {
        mid := (left + right) / 2
        if pustaka.buku[mid].rating == r {
            fmt.Printf("Buku ditemukan:\nJudul:
%s\nPengarang: %s\nTahun: %d\nRating: %d\n",
                pustaka.buku[mid].judul,
pustaka.buku[mid].pengarang,
                pustaka.buku[mid].tahun,
pustaka.buku[mid].rating)
            found = true
            break
        }
        if pustaka.buku[mid].rating < r {
            right = mid - 1
        } else {
            left = mid + 1
        }
    }
}

```

```

        if !found {
            fmt.Printf("Tidak ada buku dengan rating %d\n", r)
        }
    }
}

func main() {
    perpustakaan := DaftarkanBuku(10)

    perpustakaan.buku[0] = DataBuku{1, "Frontend Developer",
    "Umar", 2005, 5}
    perpustakaan.buku[1] = DataBuku{2, "Backend Developer",
    "Slamet", 2006, 2}
    perpustakaan.buku[2] = DataBuku{3, "Fullstack Developer",
    "Miskan", 2007, 4}
    perpustakaan.buku[3] = DataBuku{4, "Mobile Developer",
    "Yanto", 2008, 3}
    perpustakaan.nBuku = 4

    fmt.Println("Original books:")
    for i := 0; i < perpustakaan.nBuku; i++ {
        fmt.Printf("%+v\n", perpustakaan.buku[i])
    }

    fmt.Println("\nTesting CetakTerfavorit:")
    CetakTerfavorit(perpustakaan, perpustakaan.nBuku)

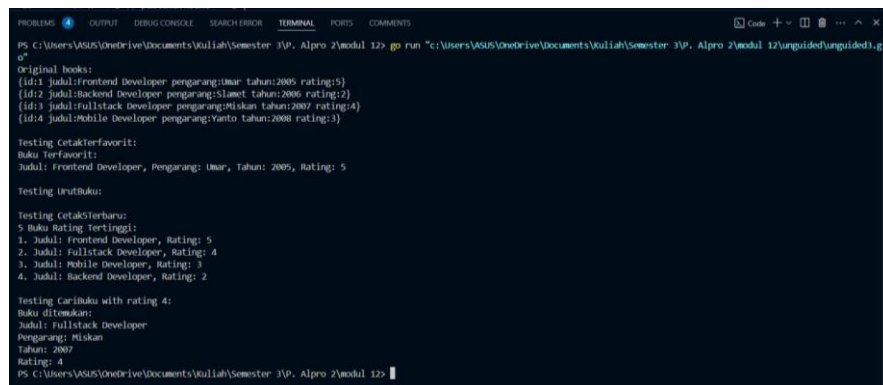
    fmt.Println("\nTesting UrutBuku:")
    UrutBuku(perpustakaan, perpustakaan.nBuku)

    fmt.Println("\nTesting Cetak5Terbaru:")
    Cetak5Terbaru(perpustakaan, perpustakaan.nBuku)

    fmt.Println("\nTesting CariBuku with rating 4:")
    CariBuku(perpustakaan, perpustakaan.nBuku, 4)
}

```

Screenshoot Output



```

PS C:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\P. Alpro 2\modul 12> go run "c:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\P. Alpro 2\modul 12\Unguided\unguided3.g
o"
Original books:
{id1 judul:frontend Developer pengarang:umar tahun:2005 rating:5}
{id2 judul:backend Developer pengarang:slamet tahun:2006 rating:2}
{id3 judul:fullstack Developer pengarang:miskan tahun:2007 rating:4}
{id4 judul:mobile Developer pengarang:yanto tahun:2008 rating:3}

Testing CetakTerfavorit:
Buku Terfavorit:
Judul: Frontend Developer, Pengarang: Umar, Tahun: 2005, Rating: 5

Testing UrutBuku:

Testing Cetak5Terbaru:
5 Buku rating Tertinggi:
1. Judul: Frontend Developer, Rating: 5
2. Judul: Fullstack Developer, Rating: 4
3. Judul: Mobile Developer, Rating: 3
4. Judul: Backend Developer, Rating: 2

Testing CariBuku with rating 4:
Buku ditemukan:
Judul: Fullstack Developer
Pengarang: Miskan
Tahun: 2007
Rating: 4
PS C:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\P. Alpro 2\modul 12>

```

Deskripsi Program

Program ini mengelola data buku di perpustakaan kecil. Setiap buku memiliki atribut seperti ID, judul, pengarang, tahun terbit, dan rating. Program menyediakan fitur untuk mencetak buku terfavorit, mengurutkan buku berdasarkan rating, menampilkan 5 buku terbaik, dan mencari buku berdasarkan rating tertentu.

Fitur Utama:

1. **Pendaftaran Buku:** Buku dimasukkan ke sistem dengan atribut lengkap seperti judul, tahun, dan rating.
2. **Cetak Buku Terfavorit:** Menampilkan buku dengan rating tertinggi menggunakan fungsi *CetakTerfavorit*.
3. **Pengurutan Buku:** Mengurutkan buku berdasarkan rating tertinggi menggunakan algoritma *Insertion Sort*.
4. **Cetak 5 Buku Terbaik:** Menampilkan hingga 5 buku dengan rating tertinggi.
5. **Pencarian Buku:** Mencari buku dengan rating tertentu menggunakan *Binary Search*.