LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII & XIII

PENGURUTAN DATA

Disusun Oleh :

M. Haidar Akhbiyani / 2311102276

S1-IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data adalah salah satu proses dasar dalam ilmu komputer yang bertujuan untuk mengatur elemen-elemen dalam suatu kumpulan data menjadi terurut berdasarkan kriteria tertentu, seperti nilai numerik, alfabetis, atau lainnya. Pengurutan memegang peranan penting dalam berbagai aplikasi seperti pencarian data, analisis statistik, dan pemrosesan data yang lebih efisien.

1. **Konsep Pengurutan**
   Pengurutan bertujuan untuk menata elemen-elemen data dalam urutan tertentu, baik secara **menaik (ascending)** maupun **menurun (descending)**. Dengan data yang terurut, operasi lain seperti pencarian (searching) dan analisis data menjadi lebih cepat dan efisien.

2. **Jenis-Jenis Algoritma Pengurutan**
   a. **Selection Sort**

   - Algoritma ini mencari elemen terkecil (atau terbesar) dalam daftar yang belum diurutkan, lalu menukarnya dengan elemen pertama (atau elemen terakhir).
   - Proses ini diulang hingga semua elemen terurut. Selection Sort sederhana namun tidak efisien untuk data dalam jumlah besar karena memiliki kompleksitas waktu O(n2)O(n^2).

   b. **Insertion Sort**

   - Algoritma ini bekerja dengan menyisipkan elemen pada posisi yang tepat dalam bagian daftar yang sudah terurut.
   - Algoritma ini efisien untuk data yang hampir terurut dan memiliki kompleksitas waktu O(n2)O(n^2) dalam kasus terburuk, namun lebih cepat dalam dataset kecil.

   c. **Bubble Sort**

   - Algoritma ini membandingkan elemen bersebelahan dan menukarnya jika tidak dalam urutan yang diinginkan. Proses ini diulang hingga daftar terurut.

- Bubble Sort sederhana tetapi memiliki performa buruk untuk data besar karena kompleksitas waktu O(n2)O(n^2).

d. **Quick Sort dan Merge Sort**

- Algoritma ini lebih kompleks dan efisien dibandingkan metode sebelumnya. Mereka menggunakan pendekatan divide-and-conquer dan memiliki kompleksitas waktu rata-rata O(nlogn)O(n \log n).

3. **Pentingnya Pengurutan Data**

   - **Efisiensi Proses Pencarian**: Data yang terurut memungkinkan algoritma pencarian seperti binary search bekerja lebih cepat.
   - **Kemudahan Analisis**: Data yang terurut mempermudah pengamatan pola, median, atau range.
   - **Manajemen Data**: Dalam sistem manajemen basis data, pengurutan digunakan untuk mengelola dan menyusun informasi.

4. **Aplikasi dalam Kehidupan Sehari-hari**
   Pengurutan digunakan dalam berbagai aplikasi seperti mengatur data buku perpustakaan, menyusun nilai siswa, pengelolaan inventaris, atau penjadwalan tugas. Dalam praktikum, pengurutan data membantu mahasiswa memahami logika dasar pemrograman algoritmik serta implementasinya pada kasus nyata.

   Praktikum pengurutan data bertujuan untuk memperkenalkan dan melatih mahasiswa dalam menggunakan berbagai algoritma pengurutan, memahami kelebihan dan kekurangan masing-masing metode, serta mengetahui konteks aplikasinya untuk berbagai jenis dataset.

II. GUIDED

1. Guided 1

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

```
PS C:\Users\Lenovo> go run "e:\2311102276_M.Haidar Akhbiyani Modul XII\GUIDED 1 MODUL 12.go"
Masukkan jumlah daerah kerabat (n): 2

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 12
Masukkan 12 nomor rumah kerabat: 3
2
3
6
76
12
21
32
12
22
33
44
Nomor rumah terurut untuk daerah 1: 2 3 3 6 12 12 21 22 32 33 44 76

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 3
Masukkan 3 nomor rumah kerabat: 1
1
2
Nomor rumah terurut untuk daerah 2: 1 1 2
```

Sort.Permintaan awal pengguna disuruh memasukan jumlah daerah yang diinginkan.Setiap daerah akan disuruh memasukan jumlah nomor rumah dan daftar nomor rumah tersebut.

2. Guided 2

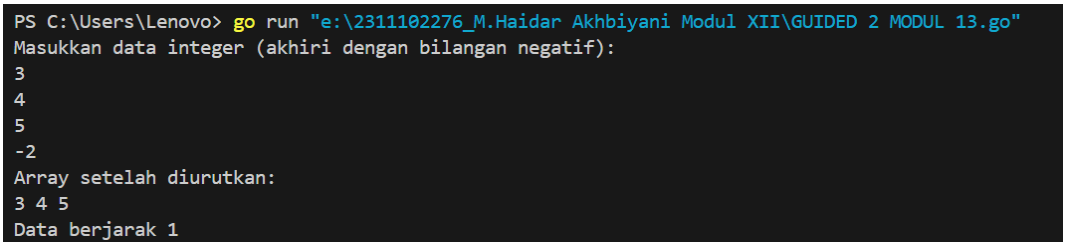
Sourcecode

```
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 mengurutkan daftar bilangan integer menggunakan algoritma *Insertion Sort* dan memeriksa apakah selisih antar elemen dalam array yang telah diurutkan memiliki jarak tetap. . Pengguna diminta memasukkan bilangan satu per satu hingga bilangan negatif dimasukkan sebagai tanda berhenti.

I. UNGUIDED

1. Unguided 1

Sourcecode

```
package main

import "fmt"

func printGanjilGenap(arr []int, n int)  {
    arrGanjil := make([]int, n)
    arrGenap := make([]int, n)
    idxGenap := 0
    idxGanjil := 0

    for i := 0; i < n; i++ {
        if arr[i]%2 == 0 {
            arrGenap[idxGenap] = arr[i]
```

```
            idxGenap++
        } else {
            arrGanjil[idxGanjil] = arr[i]
            idxGanjil++
        }

    }

    selectionSort(arrGenap, arrGanjil, idxGenap, idxGanjil)

    for i := 0; i < idxGanjil; i++ {
        fmt.Print(arrGanjil[i], " ")
    }
    for i := 0; i < idxGenap; i++ {
        fmt.Print(arrGenap[i], " ")
    }

}

func selectionSort(arrGenap, arrGanjil []int, idxGenap,
idxGanjil int) {
    var arrass int
    var arrdess int

    for i := 0; i < idxGanjil; i++ {
        for j := i+1; j < idxGanjil; j++ {
            if arrGanjil[i] > arrGanjil[j] {
                arrass = arrGanjil[j]
                arrGanjil[j] = arrGanjil[i]
                arrGanjil[i] = arrass
            }
        }
    }
    for i := 0; i < idxGenap; i++ {
        for j := i+1; j < idxGenap; j++ {
            if arrGenap[i] < arrGenap[j] {
                arrdess = arrGenap[j]
                arrGenap[j] = arrGenap[i]
                arrGenap[i] = arrdess
            }
        }
    }
}

func main() {
```

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

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

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

        fmt.Printf("Nomor rumah terurut untuk daerah %d: ",
daerah)

        printGanjilGenap(arr, m)

        fmt.Println()
    }
}
```

Screenshoot Output

```
PS C:\Users\Lenovo> go run "e:\2311102276_M.Haidar Akhbiyani Modul XII\UNGUIDED 1.go"
Masukkan jumlah daerah kerabat (n): 3

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 2
Masukkan 2 nomor rumah kerabat: 3
2
Nomor rumah terurut untuk daerah 1: 3 2

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 3
Masukkan 3 nomor rumah kerabat: 2
4
5
Nomor rumah terurut untuk daerah 2: 5 4 2

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 3
Masukkan 3 nomor rumah kerabat: 4
6
7
Nomor rumah terurut untuk daerah 3: 7 6 4
```

Deskripsi Program

Program ini adalah implementasi pengurutan nomor rumah berdasarkan kategori **ganjil** dan **genap** menggunakan bahasa Go. Nomor rumah genap diurutkan secara **menurun**, sedangkan nomor rumah ganjil diurutkan secara **menaik**. Pengguna diminta memasukkan jumlah daerah dan nomor rumah di masing-masing daerah. Proses pengelompokan dilakukan dengan memisahkan angka ganjil dan genap ke dalam array terpisah. Setelah itu, setiap array diurutkan menggunakan algoritma Selection Sort, di mana array ganjil diurutkan secara menaik dan array genap secara menurun. Hasil akhirnya ditampilkan dengan angka ganjil diikuti angka genap untuk setiap daerah. Program ini membantu pengguna untuk mengelola data nomor rumah dengan pengelompokan yang terstruktur.

2. Unguided 2

Sourcecode

```
package main

import (
    "fmt"
)

func selectionSort(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        idxMin := i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}

func median(arr []int, n int) int {
    if n%2 == 0 {
```

```
        return (arr[n/2-1] + arr[n/2]) / 2
    } else {
        return arr[(n-1)/2]
    }
}

func main() {
    var arr [1000000]int
    var arrMedian [1000000]int
    var n, nMed, datum int

    fmt.Printf("Masukkan data: ")
    for datum != -5313 && n < 1000000 {
        fmt.Scan(&datum)
        if datum == 0 {
            selectionSort(arr[:n], n)
            arrMedian[nMed] = median(arr[:n], n)
            nMed++
        } else {
            arr[n] = datum
            n++
        }
    }

    fmt.Println(" Hasil: ")
    for i := 0; i < nMed; i++ {
        fmt.Println(arrMedian[i], " ")
    }
}
```

Screenshoot Output

```
PS C:\Users\Lenovo> go run "e:\2311102276_M.Haidar Akhbiyani Modul XII\UNGUIDED 2.go"
Masukkan data: 29 11 3 12 13 14 32 10 0 -5313
 Hasil:
12
PS C:\Users\Lenovo>
```

Deskripsi Program

Program ini membaca data angka dari pengguna, mengelompokkan data berdasarkan pemisah angka 0, lalu menghitung **median** dari setiap kelompok. Setiap kali angka 0 ditemukan, data dalam kelompok sebelumnya diurutkan menggunakan **Selection Sort**, dan median dihitung berdasarkan elemen tengah dari data terurut; jika jumlah data genap, median dihitung sebagai rata-rata dua elemen tengah. Median dari setiap kelompok disimpan dan ditampilkan di akhir proses. Program berakhir ketika angka -5313 dimasukkan, yang menandakan penghentian input data.

3. Unguided 3

Sourcecode

```
 package main

import "fmt"

const nMax = 7919

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

type DaftarBuku struct {
    Pustaka  [nMax]Buku
    nPustaka int
}

func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    for i := 0; i < n; i++ {
```

```
        fmt.Println("Masukkan data buku (id, judul, penulis,
penerbit, eksemplar, tahun, rating):")
        fmt.Scan(&pustaka.Pustaka[i].id,
&pustaka.Pustaka[i].judul, &pustaka.Pustaka[i].penulis,
&pustaka.Pustaka[i].penerbit, &pustaka.Pustaka[i].eksemplar,
&pustaka.Pustaka[i].tahun, &pustaka.Pustaka[i].rating)
    }
    pustaka.nPustaka = n
}

func CetakTerfavorit(pustaka DaftarBuku) {
    if pustaka.nPustaka == 0 {
        fmt.Println("Tidak ada buku dalam pustaka.")
        return
    }

    terfavorit := pustaka.Pustaka[0]
    for i := 1; i < pustaka.nPustaka; i++ {
        if pustaka.Pustaka[i].rating > terfavorit.rating {
            terfavorit = pustaka.Pustaka[i]
        }
    }

    fmt.Println("Buku terfavorit:")
    fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun:
%d, Rating: %d\n",
        terfavorit.judul, terfavorit.penulis,
terfavorit.penerbit, terfavorit.tahun, terfavorit.rating)
}

func UrutBuku(pustaka *DaftarBuku) {
    for i := 1; i < pustaka.nPustaka; i++ {
        key := pustaka.Pustaka[i]
        j := i - 1
        for j >= 0 && pustaka.Pustaka[j].rating < key.rating
{
            pustaka.Pustaka[j+1] = pustaka.Pustaka[j]
            j--
        }
        pustaka.Pustaka[j+1] = key
    }
}

func Cetak5Terbaru(pustaka DaftarBuku) {
    fmt.Println("5 Buku dengan rating tertinggi:")
```

```
    for i := 0; i < 5 && i < pustaka.nPustaka; i++ {
        buku := pustaka.Pustaka[i]
        fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s,
Tahun: %d, Rating: %d\n",
            buku.judul, buku.penulis, buku.penerbit,
buku.tahun, buku.rating)
    }
}

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

func main() {
    var pustaka DaftarBuku
    var n, rating int

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

    DaftarkanBuku(&pustaka, n)

    CetakTerfavorit(pustaka)

    UrutBuku(&pustaka)

    Cetak5Terbaru(pustaka)

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

```
    fmt.Scan(&rating)
    CariBuku(pustaka, rating)
}
```

Screenshoot Output

```
PS C:\Users\Lenovo> go run "e:\2311102276_M.Haidar Akhbiyani Modul XII\UNGUI
DED 3.go"
Masukkan jumlah buku: 5
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 Jejeg Ucup J2g 120 2009 5
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 Cempaka Paki CMP 122 2010 4,5
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
3 Krikil Wasto KRK 230 2005 4
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
4 Sanjur Gombs SNJ 202 2018 3,5
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
Buku terfavorit:
Judul: Jejeg, Penulis: Ucup, Penerbit: J2g, Tahun: 2009, Rating: 5
5 Buku dengan rating tertinggi:
Judul: Jejeg, Penulis: Ucup, Penerbit: J2g, Tahun: 2009, Rating: 5
Judul: Cempaka, Penulis: Paki, Penerbit: CMP, Tahun: 2010, Rating: 4
Judul: 3, Penulis: Krikil, Penerbit: Wasto, Tahun: 0, Rating: 0
Judul: 230, Penulis: 2005, Penerbit: 4, Tahun: 0, Rating: 0
Judul: Gombs, Penulis: SNJ, Penerbit: 202, Tahun: 3, Rating: 0
Masukkan rating buku yang ingin dicari: Buku ditemukan: Judul: Jejeg, Penuli
s: Ucup, Penerbit: J2g, Tahun: 2009, Rating: 5
```

memungkinkan pengguna untuk mendaftarkan buku dengan informasi seperti ID, judul, penulis, penerbit, eksemplar, tahun, dan rating. Setelah data dimasukkan, program menampilkan buku terfavorit berdasarkan rating tertinggi dan mengurutkan buku secara menurun berdasarkan rating menggunakan algoritma Insertion Sort. Selain itu, lima buku dengan rating tertinggi ditampilkan sebagai rekomendasi, dan pengguna dapat mencari buku tertentu berdasarkan rating menggunakan metode pencarian biner. Program ini membantu mengelola pustaka dengan fokus pada pengurutan dan pencarian buku berdasarkan popularitasnya.