

XLAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL XII
PENGURUTAN DATA



Disusun Oleh :

Aji Tri Prasetyo / 2311102064

IF-11-02

Dosen Pengampu :

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data merupakan salah satu teknik manipulasi data yang bertujuan untuk menyusun elemen-elemen dalam sebuah kumpulan data (array, slice, atau struktur lainnya) berdasarkan urutan tertentu, seperti ascending (menaik) atau descending (menurun). Dalam Golang, pengurutan data dapat dilakukan dengan memanfaatkan package sort bawaan dari pustaka standar Go.

Pengertian Sorting

Sorting adalah proses menyusun elemen-elemen data dalam suatu urutan yang diinginkan, baik secara:

1. **Ascending:** Urutan dari kecil ke besar.
2. **Descending:** Urutan dari besar ke kecil.

Dalam implementasi algoritma sorting, terdapat berbagai metode, seperti:

- **Bubble Sort**
- **Selection Sort**
- **Insertion Sort**
- **Merge Sort**
- **Quick Sort**

Namun, Golang menyediakan implementasi sorting yang efisien melalui package sort untuk menangani berbagai jenis data.

Kesimpulan

Dalam Golang, pengurutan data menjadi lebih mudah dengan adanya package sort. Golang mendukung pengurutan pada tipe data bawaan seperti integer, string, dan float64, serta memberikan fleksibilitas untuk tipe data kustom dengan menggunakan interface sort.Interface. Penggunaan pengurutan yang efisien dan optimal ini sangat berguna untuk memanipulasi data dalam berbagai skenario pemrograman.

II. GUIDED

1. Guided 1

Sourcecode

```
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

```

Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5 3 12 3 4 6
Masukkan nomor rumah kerabat untuk daerah ke-1: Hasil urutan rumah untuk daerah ke-1: 12 6 4 3 3
Masukkan jumlah rumah kerabat untuk daerah ke-2: 12 34 22 13 11 21 45
Masukkan nomor rumah kerabat untuk daerah ke-2:
1
23

f
se
f
Hasil urutan rumah untuk daerah ke-2: 45 34 23 22 21 13 11 1 0 0 0 0

```

Deskripsi Program

Fungsi selectionSort Fungsi ini bertugas untuk mengurutkan array dalam urutan menurun menggunakan algoritma selection sort: Fungsi menerima array arr sebagai parameter, Dilakukan iterasi untuk mencari elemen terbesar di bagian array yang belum terurut, Elemen terbesar ditukar dengan elemen pertama di bagian tersebut, Proses diulangi hingga seluruh array terurut.

2. Guided 2

Sourcecode

```

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

```

Masukkan data (akhiri dengan bilangan negatif):
23 45 32 22 1 -1
Hasil pengurutan: [1 22 23 32 45]
Data berjarak tidak tetap

```

Deskripsi Program

Fungsi `isDataConsistentlySpaced` Tujuan: Memeriksa apakah selisih antar elemen dalam array yang telah diurutkan adalah konstan.

Langkah Kerja: Jika array memiliki kurang dari 2 elemen, dianggap berjarak tetap dengan selisih 0, Hitung selisih antara elemen pertama dan kedua (diff). Iterasi melalui array: Hitung selisih antara elemen saat ini dan elemen berikutnya, Jika selisih tidak sama dengan diff, array dianggap tidak berjarak tetap, Jika semua selisih sama, array dinyatakan berjarak tetap dengan nilai diff.

III. UNGUIDED

Unguided 1

Sourcecode

```
package main
import "fmt"

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] {
                minIdx = j
            }
        }
        arr[i], arr[minIdx] = arr[minIdx], arr[i]
    }
}

func main() {
    var n int
    fmt.Println("Input")
    fmt.Scan(&n)

    if n <= 0 || n >= 1000 {
        fmt.Println("Jumlah daerah harus di antara 1 dan 999.")
        return
    }
    masukan := make([][]int, n)

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

        if m <= 0 || m >= 1000000 {
            fmt.Println("Jumlah rumah harus di antara 1 dan 999999.")
            return
        }

        masukan[i] = make([]int, m)
        for j := 0; j < m; j++ {
            fmt.Scan(&masukan[i][j])
        }
    }

    fmt.Println("\nOutput:")
    for _, daerah := range masukan {
```

```

var ganjil []int
var genap []int

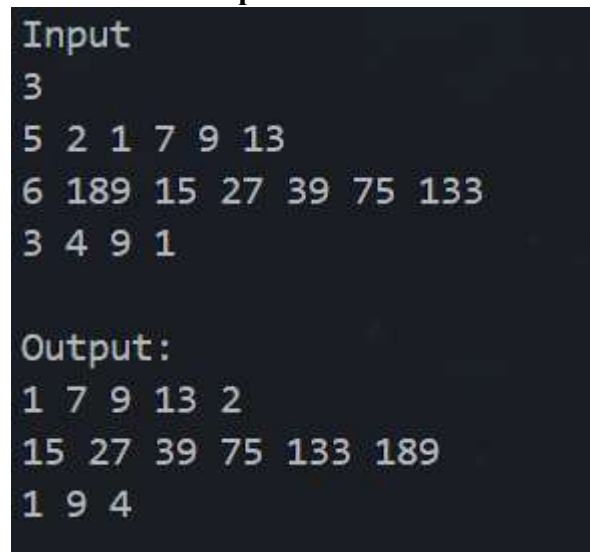
for _, num := range daerah {
    if num%2 == 0 {
        genap = append(genap, num)
    } else {
        ganjil = append(ganjil, num)
    }
}

selectionSortAsc(ganjil)
selectionSortAsc(genap)
hasil := append(ganjil, genap...)

for _, val := range hasil {
    fmt.Print(val, " ")
}
fmt.Println()
}

```

Screenshoot Output



The screenshot shows the input and output of the program. The input consists of three lines of numbers: '3', '5 2 1 7 9 13', and '6 189 15 27 39 75 133'. The output consists of three lines of sorted numbers: '1 7 9 13 2', '15 27 39 75 133 189', and '1 9 4'.

```

Input
3
5 2 1 7 9 13
6 189 15 27 39 75 133
3 4 9 1

Output:
1 7 9 13 2
15 27 39 75 133 189
1 9 4

```

Deskripsi Program

Fungsi selectionSortAsc Tujuan: Mengurutkan array dalam urutan menaik menggunakan algoritma selection sort. Langkah Kerja: Iterasi untuk menemukan elemen terkecil di bagian array yang belum terurut, Elemen terkecil dipindahkan ke posisi terdepan, Proses diulangi hingga seluruh array terurut.

Unguided 2

Sourcecode

```
package main

import "fmt"

func hitungMedian(arr []int) int {
    n := len(arr)
    if n%2 == 1 {
        return arr[n/2]
    }
    return (arr[n/2-1] + arr[n/2]) / 2
}

func insertionSort(arr []int) {
    for i := 1; i < len(arr); i++ {
        key := arr[i]
        j := i - 1
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j = j - 1
        }
        arr[j+1] = key
    }
}

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

    for {
        fmt.Scan(&input)
        if input == -5313 {
            break
        }

        if input == 0 {
            insertionSort(data)
            median := hitungMedian(data)
            fmt.Println(median)
        } else {
            data = append(data, input)
        }
    }
}
```

Screenshoot Output

```
7 23 11 0 5 19 2 29 3 13 17 0 -5313  
11  
12
```

Deskripsi Program

Fungsi hitungMedian Tujuan: Menghitung median dari array yang telah diurutkan. Jika jumlah elemen ganjil: Median adalah elemen di tengah array. Jika jumlah elemen genap: Median adalah rata-rata dari dua elemen tengah.

Unguided 3

Sourcecode

```
package main
import (
    "fmt"
)
type Buku struct {
    id        int
    judul     string
    penulis   string
    penerbit  string
    eksemplar int
    tahun     int
    rating    int
}

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)
    }
}

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)
}

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
    }
}

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)
    }
}

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

    DaftarkanBuku(&pustaka, n)
    CetakFavorit(pustaka, n)

```

```
        UrutkanBuku(&pustaka, n)

        Cetak5Terbaik(pustaka, n)
        var rating int
        fmt.Print("Masukkan rating buku yang ingin dicari: ")
        fmt.Scan(&rating)
        CariBuku(pustaka, n, rating)
    }
```

Screenshoot Output

```
Masukkan jumlah buku di perpustakaan: 2
Masukkan data untuk buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 Kisah Aji PT 6 2024 5
Masukkan data untuk buku ke-2 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 Kasih Aji PT 5 2024 4
Buku dengan rating tertinggi:
ID: 1, Judul: Kisah, Penulis: Aji, Penerbit: PT, Eksemplar: 6, Tahun: 2024, Rating: 5
Lima buku dengan rating tertinggi:
ID: 1, Judul: Kisah, Penulis: Aji, Penerbit: PT, Eksemplar: 6, Tahun: 2024, Rating: 5
ID: 2, Judul: Kasih, Penulis: Aji, Penerbit: PT, Eksemplar: 5, Tahun: 2024, Rating: 4
Masukkan rating buku yang ingin dicari: 4
ID: 2, Judul: Kasih, Penulis: Aji, Penerbit: PT, Eksemplar: 5, Tahun: 2024, Rating: 4
```

Deskripsi Program

Program di atas adalah sistem manajemen perpustakaan sederhana dalam bahasa Go. Program memungkinkan pengguna untuk mendaftarkan buku, mencetak buku dengan rating tertinggi, mengurutkan buku berdasarkan rating, menampilkan lima buku terbaik, dan mencari buku berdasarkan rating tertentu. Fungsi `UrutkanBuku` Mengurutkan buku berdasarkan rating secara menurun (descending).