

XLAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL XII
PENGURUTAN DATA



Disusun Oleh :

ARVAN MURBIYANTO

2311102074

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

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\MSI> go run "c:\Users\MSI\Documents\laprak modul 12\guided\Guided 1.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5
Masukkan nomor rumah kerabat untuk daerah ke-1: 21 33 42 10 32
Hasil urutan rumah untuk daerah ke-1: 42 33 32 21 10
Masukkan jumlah rumah kerabat untuk daerah ke-2: 5
Masukkan nomor rumah kerabat untuk daerah ke-2: 121 434 324 12 453
Hasil urutan rumah untuk daerah ke-2: 453 434 324 121 12
Masukkan jumlah rumah kerabat untuk daerah ke-3: 5
Masukkan nomor rumah kerabat untuk daerah ke-3: 1213 324 52 23 1
Hasil urutan rumah untuk daerah ke-3: 1213 324 52 23 1
PS C:\Users\MSI>

```

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
```

```

    "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)
}

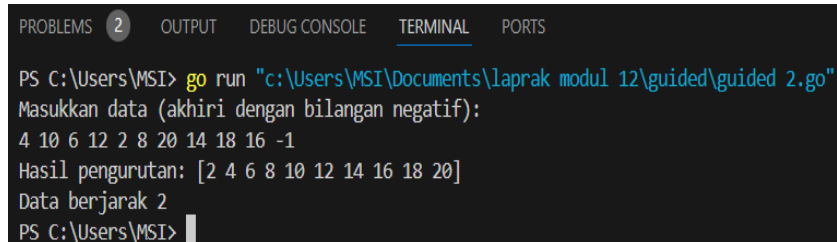
// Urutkan data menggunakan insertion sort
insertionSort(data)

// 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



```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\MSI> go run "c:\Users\MSI\Documents\laprak modul 12\guided\guided 2.go"
Masukkan data (akhiri dengan bilangan negatif):
4 10 6 12 2 8 20 14 18 16 -1
Hasil pengurutan: [2 4 6 8 10 12 14 16 18 20]
Data berjarak 2
PS C:\Users\MSI>

```

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])
```

```
    }
```

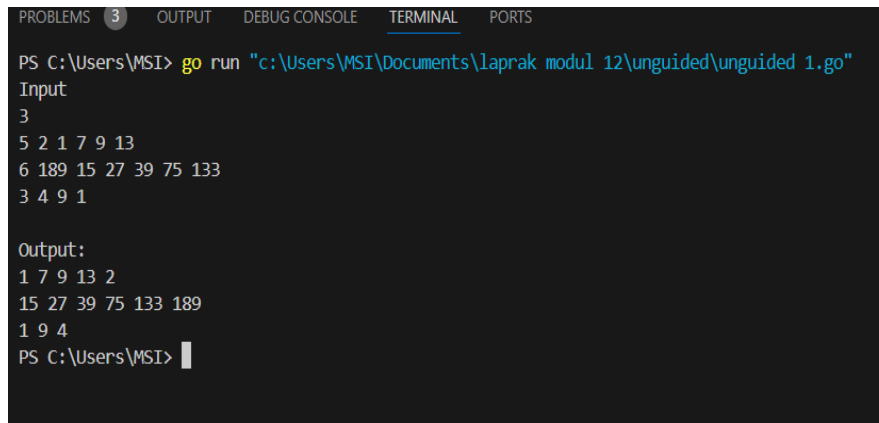
```
}
```

```
fmt.Println("\nOutput:")
```



```
for _, daerah := range masukan {  
  
    var ganjil, 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

A screenshot of a Go program's output in a terminal window. The terminal has tabs for PROBLEMS (3), OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The command executed is `PS C:\Users\MSI> go run "c:\Users\MSI\Documents\laprak modul 12\unguided\unguided 1.go"`. The program prompts for "Input" and receives three lines of numbers: `3`, `5 2 1 7 9 13`, and `6 189 15 27 39 75 133`. It then outputs the sorted arrays: `Output:`, `1 7 9 13 2`, `15 27 39 75 133 189`, and `1 9 4`. The prompt `PS C:\Users\MSI>` is visible at the bottom.

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\MSI> go run "c:\Users\MSI\Documents\laprak modul 12\unguided\unguided 1.go"
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
PS C:\Users\MSI>
```

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

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

    for {

        fmt.Scan(&inpu
```

```

t)
    if input
    == -5313 {
        break
    }

    if input
    == 0 {
        if len(data) == 0
        {
            continue
        }

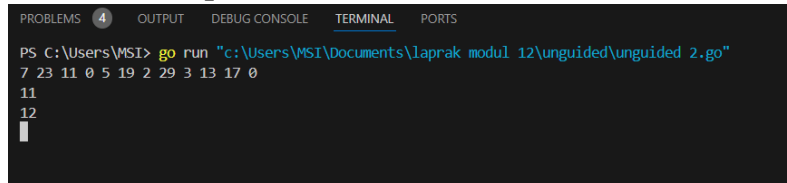
        selectionSort(da
ta) // Menggunakan
selectionSort

        median :=
calculateMedian(data)

        fmt.Println(med
ian)
    } else {
        data =
append(data, input)
    }
}

```

Screenshoot Output



```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\MSI> go run "c:\Users\MSI\Documents\laprak modul 12\unguided\unguided 2.go"
7 23 11 0 5 19 2 29 3 13 17 0
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"

const nMax = 7919
type Buku struct {
    id          int
    judul       string
    penulis     string
    penerbit    string
    eksemplar   int
    tahun       int
    rating      int
}

type DaftarBuku struct {
    pustaka []Buku
}

func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    for i := 0; i < n; i++ {
        var buku Buku
        fmt.Printf("Masukkan data untuk buku ke-%d\n", i+1)
        fmt.Scan(&buku.id, &buku.judul,
            &buku.penulis, &buku.penerbit, &buku.eksemplar,
            &buku.tahun, &buku.rating)
        if len(pustaka.pustaka) < nMax {
            pustaka.pustaka =
                append(pustaka.pustaka, buku)
        }
    }
}

func CetakFavorit(pustaka DaftarBuku) {
    if len(pustaka.pustaka) == 0 {
        fmt.Println("Pustaka kosong.")
        return
    }
    terfavorit := pustaka.pustaka[0]
    for _, buku := range pustaka.pustaka {
        if buku.rating > terfavorit.rating {
            terfavorit = buku
        }
    }
    fmt.Println("\nBuku 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
*DaftarBuku) {
    for i := 1; i <
        len(pustaka.pustaka);
        i++ {
            key :=
                pustaka.pustaka[i]
            j := i - 1
                for j >=
                    0 &&
                        pustaka.pustaka[j].ra
                            ting < key.rating {
                                pustaka.pustaka[j+1]
                                    = pustaka.pustaka[j]
                                        j--
                            }

                                pustaka.pustaka
                                    [j+1] = key
                        }
    }

func Cetak5Terbaik(pustaka
DaftarBuku) {
    fmt.Println("\nLima
buku dengan rating
tertinggi:")
    for i := 0; i < 5 &&
        i <
            len(pustaka.pustaka);
            i++ {
                buku :=
                    pustaka.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 DaftarBuku, r int) {
    ditemukan := false
    for _, buku := range pustaka.pustaka {
        if buku.rating == r {
            fmt.Printf("\nDitemukan buku dengan rating
%d:\n", r)
            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)
            ditemukan = true
        }
    }
    if !ditemukan {
        fmt.Println("\nTidak ada buku dengan rating
tersebut.")
    }
}

func main() {
    var pustaka DaftarBuku
    var n int

    fmt.Print("Masukkan jumlah buku di perpustakaan: ")
    fmt.Scan(&n)
    if n <= 0 || n > nMax {
        fmt.Println("Jumlah buku harus antara 1 hingga
7919.")
        return
    }

    DaftarkanBuku(&pustaka, n)
    CetakFavorit(pustaka)
    UrutkanBuku(&pustaka)
    Cetak5Terbaik(pustaka)

    var rating int
    fmt.Print("\nMasukkan rating buku yang ingin dicari:
")
    fmt.Scan(&rating)
    CariBuku(pustaka, rating)
}

```

Screenshoot Output

```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukkan data untuk buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 Kisahku ARVAN PTbukuku 220 2024 9
Masukkan data untuk buku ke-2 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 Kisahnya Arvan PTbukuku 186 2025 8

Buku dengan rating tertinggi:
ID: 1, Judul: Kisahku, Penulis: ARVAN, Penerbit: PTbukuku, Eksemplar: 220, Tahun: 2024, Rating: 9

Lima buku dengan rating tertinggi:
ID: 1, Judul: Kisahku, Penulis: ARVAN, Penerbit: PTbukuku, Eksemplar: 220, Tahun: 2024, Rating: 9
ID: 2, Judul: Kisahnya, Penulis: Arvan, Penerbit: PTbukuku, Eksemplar: 186, Tahun: 2025, Rating: 8

Masukkan rating buku yang ingin dicari: 9

Ditemukan buku dengan rating 9:
ID: 1, Judul: Kisahku, Penulis: ARVAN, Penerbit: PTbukuku, Eksemplar: 220, Tahun: 2024, Rating: 9
PS C:\Users\MSI>

```


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).