

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL 11
PENGURUTAN DATA**



Oleh:

DAMARA GALUH PEMBAYUN

2311102110

IF-11-02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

I. DASAR TEORI

Dua algoritma pengurutan sederhana yang sering diajarkan sebagai dasar adalah insert sort dan selection sort. Sementara Selection Sort mencari elemen terkecil dalam bagian array yang belum terurut dan menukarnya dengan elemen pertama pada bagian yang belum terurut, Insertion Sort menyisipkan setiap elemen ke posisi yang benar dalam bagian array yang sudah terurut. Meskipun keduanya memiliki kompleksitas waktu rata-rata $O(n^2)$, Sort Insertion biasanya lebih baik untuk data yang hampir terurut, sementara Sort Selection lebih mudah dilakukan tetapi tidak stabil, artinya urutan elemen yang sama nilainya dapat berubah setelah pengurutan. Karakteristik data dan kebutuhan aplikasi menentukan pemilihan algoritma yang tepat.

II. GUIDED

Guided 1

SourchCode

```
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()
    }
}

```

Hasil ScreenShoot

```

PS C:\golang\modul11> go run "c:\golang\modul11\guided1.go"
Masukkan jumlah daerah (n): 4
Masukkan jumlah rumah kerabat untuk daerah ke-1: 8
Masukkan nomor rumah kerabat untuk daerah ke-1: 3
4
6
7
8
9
9
9
Hasil urutan rumah untuk daerah ke-1: 9 9 9 8 7 6 4 3
Masukkan jumlah rumah kerabat untuk daerah ke-2: 4
Masukkan nomor rumah kerabat untuk daerah ke-2: 2
8
9
6
Hasil urutan rumah untuk daerah ke-2: 9 8 6 2
Masukkan jumlah rumah kerabat untuk daerah ke-3: 6
Masukkan nomor rumah kerabat untuk daerah ke-3: 3
6
7
8
7
6
Hasil urutan rumah untuk daerah ke-3: 8 7 7 6 6 3
Masukkan jumlah rumah kerabat untuk daerah ke-4: 5
Masukkan nomor rumah kerabat untuk daerah ke-4: 2
1
3
6
7
Hasil urutan rumah untuk daerah ke-4: 7 6 3 2 1
PS C:\golang\modul11>

```

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")
    }
}

```

Hasil ScreenShoot

```

PS C:\golang\modul11> go run "c:\golang\modul11\guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
5 7 8 4 6 3 -9
Hasil pengurutan: [3 4 5 6 7 8]
Data berjarak 1
PS C:\golang\modul11>

```

III. UNGUIDED

Unguided 1

SourchCode

```

package main

import (
    "fmt"
    "sort"
)

func findMedian(data []int) float64 {
    // Fungsi ini menghitung median dari sebuah slice
    of int
    n := len(data)
    if n == 0 {
        return 0 // Jika data kosong, kembalikan 0
    }
    sort.Ints(data) // Urutkan data secara ascending

```

```

        if n%2 == 0 {
            // Jika jumlah data genap, ambil rata-rata
            dari dua nilai tengah
            return float64(data[n/2-1]+data[n/2]) / 2
        } else {
            // Jika jumlah data ganjil, ambil nilai
            tengah
            return float64(data[n/2])
        }
    }

func main() {
    var data []int
    var input int
    for {
        fmt.Scan(&input)
        if input == -5313 {
            break
        }
        if input == 0 {
            median := findMedian(data)
            fmt.Println(median)
            data = []int{} // Kosongkan slice
            data untuk input berikutnya
        } else {
            data = append(data, input)
        }
    }
}

```

Hasil ScreenShoot

(tetiba vscode saya error kak)

Deskripsi

Kode inii dibuat untuk menghitung median dari sekumpulan bilangan bulat yang diberikan pengguna sebagai input. Sampai pengguna memasukkan angka -5313 sebagai tanda berhenti, program akan terus meminta angka. Kemudian, program akan menghitung median dari semua angka yang dimasukkan dan mencetak hasilnya. Data bilangan bulat diurutkan terlebih dahulu menggunakan fungsi sort oleh algoritma yang digunakan. Nilai tengah ditentukan oleh inti Go. Jika jumlah data genap, median adalah rata-rata dari dua nilai tengah, dan jika jumlah data ganjil, median adalah nilai tengahnya. Banyak kebutuhan analisis statistik, seperti menemukan nilai tengah distribusi data, dapat diselesaikan dengan program ini.

Unguided 2

SourchCode

```
package main

import (
    "fmt"
)

func insertionSort(arr []int) {
    // Fungsi untuk mengurutkan array menggunakan
    metode Insertion Sort
    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 checkDistance(arr []int) string {
    // Fungsi untuk memeriksa apakah jarak antar
    elemen dalam array konsisten
    distance := arr[1] - arr[0]
    for i := 2; i < len(arr); i++ {
        if arr[i]-arr[i-1] != distance {
            return "Data berjarak tidak tetap"
        }
    }
    return fmt.Sprintf("Data berjarak %d", distance)
}

func main() {
    // Contoh penggunaan
    var arr []int
    var input int

    fmt.Println("Masukkan bilangan bulat (akhiri
    dengan bilangan negatif):")
    for {
        _, err := fmt.Scan(&input)
        if err != nil || input < 0 {
            break
        }
        arr = append(arr, input)
    }
}
```



```

        insertionSort(arr)
        fmt.Println(arr)
        fmt.Println(checkDistance(arr))
    }

```

Hasil ScreenShoot

(tetiba vscode saya error kak)

```

Output
Masukkan bilangan bulat (akhiri dengan bilangan negatif):
3 4 5 6 7 8 -2
[3 4 5 6 7 8]
Data berjarak 1

=== Code Execution Successful ===
=== Session Ended. Please Run the code again ===

```

Deskripsi

Kode Go di atas dimaksudkan untuk menggunakan algoritma Sort Insertion untuk mengurutkan sekumpulan bilangan bulat yang dimasukkan oleh pengguna. Kemudian, algoritma ini mengevaluasi konsistensi jarak antara setiap elemen dalam array yang telah diurutkan. Algorithm Insertion Sort membandingkan setiap komponen dengan komponen sebelumnya dan memindahkan mereka ke posisi yang tepat untuk menghasilkan urutan yang terurut. Kemudian, fungsi pengecekan jarak melihat apakah selisih antara setiap elemen yang berdekatan selalu sama. Jika selisihnya selalu sama, fungsi akan mengembalikan pesan bahwa data jarak tetap bersama dengan nilai jarak. Jika tidak, fungsi akan mengembalikan pesan bahwa data jarak tidak tetap.

Unguided 3

SourchCode

```

package main

import "fmt"

const maxBuku = 7919

// Struktur data untuk mewakili sebuah buku
type Buku struct {
    id        string
    judul     string
    penulis   string
    penerbit  string
}

```

```

        eksemplar int
        tahun      int
        rating      int
    }

    // Array untuk menyimpan daftar buku
    type DaftarBuku [maxBuku]Buku

    func main() {
        var pustaka DaftarBuku
        var n int // Jumlah buku

        // Input jumlah buku dan data buku
        fmt.Print("Masukkan jumlah buku: ")
        fmt.Scanln(&n)
        for i := 0; i < n; i++ {
            fmt.Printf("Buku ke-%d\n", i+1)
            fmt.Print("ID: ")
            fmt.Scanln(&pustaka[i].id)
            // ... (input data lainnya)
        }

        // Mengurutkan buku berdasarkan rating
        (descending)
        UrutBuku(&pustaka, n)

        // Menampilkan buku dengan rating tertinggi
        fmt.Println("\nBuku Terfavorit:")
        CetakTerfavorit(pustaka[n-1])

        // Menampilkan 5 buku dengan rating tertinggi
        fmt.Println("\n5 Buku dengan Rating Tertinggi:")
        Cetak5Terbaru(pustaka, n)

        // Mencari buku berdasarkan rating
        var ratingCari int
        fmt.Print("Masukkan rating buku yang ingin dicari: ")
        fmt.Scanln(&ratingCari)
        CariBuku(pustaka, n, ratingCari)
    }

    // Mengurutkan buku berdasarkan rating menggunakan
    Insertion Sort (descending)
    func UrutBuku(pustaka *DaftarBuku, n int) {
        for i := 1; i < n; i++ {
            buku := (*pustaka)[i]
            j := i - 1

```

```

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

// Menampilkan data buku
func CetakBuku(buku Buku) {
    fmt.Printf("ID: %s, Judul: %s, Penulis: %s,
Penerbit: %s, Tahun: %d, Rating: %d\n",
        buku.id, buku.judul, buku.penulis,
buku.penerbit, buku.tahun, buku.rating)
}

// Menampilkan buku dengan rating tertinggi
func CetakTerfavorit(buku Buku) {
    CetakBuku(buku)
}

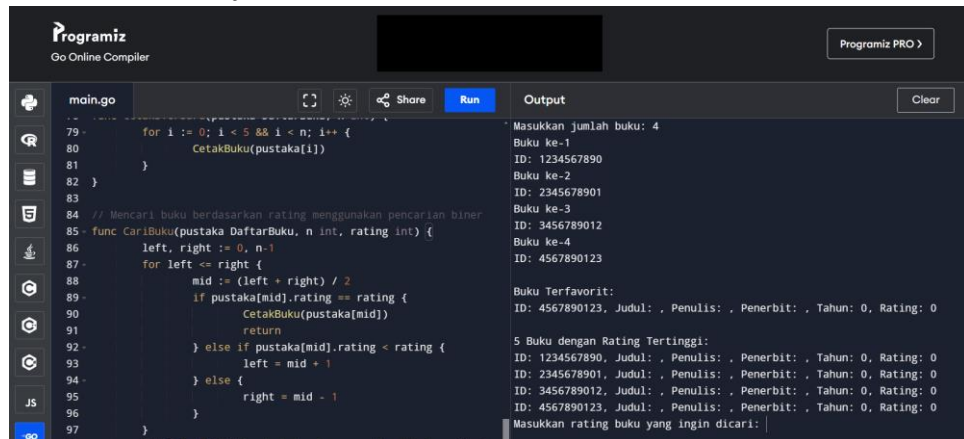
// Menampilkan 5 buku dengan rating tertinggi
func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    for i := 0; i < 5 && i < n; i++ {
        CetakBuku(pustaka[i])
    }
}

// Mencari buku berdasarkan rating menggunakan pencarian
biner
func CariBuku(pustaka DaftarBuku, n int, rating int) {
    left, right := 0, n-1
    for left <= right {
        mid := (left + right) / 2
        if pustaka[mid].rating == rating {
            CetakBuku(pustaka[mid])
            return
        } else if pustaka[mid].rating < rating {
            left = mid + 1
        } else {
            right = mid - 1
        }
    }
    fmt.Printf("Tidak ada buku dengan rating %d\n",
rating)
}

```

Hasil ScreenShoot

(tetiba vscode saya error kak)



The screenshot shows the Programiz Go Online Compiler interface. The editor on the left contains a Go program in `main.go` with line numbers 79 to 97. The program includes a loop to input book data, a function `CetakBuku` to print book details, and a binary search function `CariBuku` to find a book by rating. The output on the right shows the program's execution: it prompts for the number of books (4), lists the books with their IDs, and then displays the top 5 books by rating, with the highest-rated book being ID 4567890123.

```
79- // Go Online Compiler
80- for i := 0; i < 5 && i < n; i++ {
81-     CetakBuku(pustaka[i])
82- }
83-
84- // Mencari buku berdasarkan rating menggunakan pencarian biner
85- func CariBuku(pustaka DaftarBuku, n int, rating int) {
86-     left, right := 0, n-1
87-     for left <= right {
88-         mid := (left + right) / 2
89-         if pustaka[mid].rating == rating {
90-             CetakBuku(pustaka[mid])
91-             return
92-         } else if pustaka[mid].rating < rating {
93-             left = mid + 1
94-         } else {
95-             right = mid - 1
96-         }
97-     }
```

Output:

```
Masukkan jumlah buku: 4
Buku ke-1
ID: 1234567890
Buku ke-2
ID: 2345678901
Buku ke-3
ID: 3456789012
Buku ke-4
ID: 4567890123

Buku Terfavorit:
ID: 4567890123, Judul: , Penulis: , Penerbit: , Tahun: 0, Rating: 0

5 Buku dengan Rating Tertinggi:
ID: 1234567890, Judul: , Penulis: , Penerbit: , Tahun: 0, Rating: 0
ID: 2345678901, Judul: , Penulis: , Penerbit: , Tahun: 0, Rating: 0
ID: 3456789012, Judul: , Penulis: , Penerbit: , Tahun: 0, Rating: 0
ID: 4567890123, Judul: , Penulis: , Penerbit: , Tahun: 0, Rating: 0
Masukkan rating buku yang ingin dicari: |
```

Deskripsi

Kode Go ini adalah contoh sederhana dari sistem perpustakaan. Dengan program ini, pengguna dapat memasukkan data tentang buku, seperti ID, judul, penulis, penerbit, tahun terbit, dan rating. Setelah data dimasukkan, program dapat mengurutkan buku berdasarkan rating, menampilkan buku dengan rating tertinggi, menampilkan lima buku dengan rating tertinggi, atau mencari buku berdasarkan rating tertentu. Untuk mengurutkan buku berdasarkan rating, program menggunakan algoritma Insertion Sort, dan untuk mencari buku berdasarkan rating, program menggunakan teknik untuk mencari buku berdasarkan rating.