

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 12 & 13
SORTING**



Disusun Oleh :

ANDIKA NEVIANTORO / 2311102167

IF-11-05

Dosen Pengampu :

Arif Amrulloh,S.Kom.,M.Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Dalam bahasa pemrograman Go, algoritma pengurutan seperti Selection Sort dan Insertion Sort sangat penting untuk dipahami dalam mengatur data secara efisien. Berikut adalah penjelasan singkat mengenai kedua algoritma tersebut:

- Selection Sort: Selection Sort bekerja dengan cara memilih elemen terkecil (atau terbesar, tergantung urutan yang diinginkan) dari bagian data yang belum terurut dan menukarnya dengan elemen pertama yang belum terurut. Proses ini diulang hingga seluruh data terurut. Waktu kompleksitas dari Selection Sort adalah $O(n^2)$, yang menjadikannya tidak efisien untuk dataset besar, namun cukup berguna untuk kumpulan data kecil.

Di Go, algoritma ini dapat diimplementasikan dengan cara mengiterasi daftar, menemukan nilai terkecil dalam bagian yang belum terurut, dan menukarnya dengan elemen pertama yang belum terurut.

- Insertion Sort: Insertion Sort mirip dengan cara orang menyusun kartu di tangan mereka: algoritma ini memindahkan satu elemen pada suatu waktu dan menempatkannya di posisi yang tepat dalam bagian data yang sudah terurut. Algoritma ini efisien untuk data yang kecil atau hampir terurut. Sama seperti Selection Sort, kompleksitas waktu Insertion Sort adalah $O(n^2)$ dalam kasus terburuk, tetapi lebih cepat pada dataset yang lebih kecil atau hampir terurut.

Algoritma ini mengiterasi array, membandingkan elemen saat ini dengan elemen sebelumnya, dan memindahkan elemen-elemen yang lebih besar ke depan sebelum memasukkan elemen saat ini ke posisi yang benar.

II. GUIDED

1. Program pengurutan nomor rumah kerabat menggunakan algoritma *Selection Sort*.

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)

    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])
        }
        selectionSort(arr, m)

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

Screenshoot Output :

```
Masukkan jumlah nomor rumah kerabat untuk daerah 2: 6 189 15 27 39 75 133
Masukkan 6 nomor rumah kerabat : Nomor rumah terurut untuk daerah 2 : 15 27 39 75 133 189

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 3 4 9 1
Masukkan 3 nomor rumah kerabat : Nomor rumah terurut untuk daerah 3 : 1 4 9
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 12 & 13> █
```

Deskripsi Program :

Program ini meminta pengguna untuk memasukkan jumlah daerah kerabat (n). Untuk setiap daerah, pengguna diminta memasukkan jumlah nomor rumah (m) dan kemudian memasukkan nomor rumah tersebut ke dalam sebuah array. Array ini kemudian diurutkan menggunakan algoritma *Selection Sort*, yang bekerja dengan cara menemukan elemen terkecil dalam array dan menukarnya dengan elemen di posisi saat ini. Setelah diurutkan, program menampilkan nomor rumah yang telah diurutkan untuk setiap daerah. Program ini berfungsi sebagai alat untuk mengurutkan data numerik sederhana dari beberapa kelompok data secara berulang.

2. Program mengurutkan data integer menggunakan algoritma *Insertion Sort* dan memeriksa apakah selisih antar elemen dalam array yang sudah diurutkan bersifat konstan.

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 :

```
● Masukkan data integer (akhiri dengan bilangan negatif):  
31 13 25 43 1 7 19 37 -5  
Array setelah diurutkan:  
1 7 13 19 25 31 37 43  
Data berjarak 6  
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 12 & 13> go run  
Masukkan data integer (akhiri dengan bilangan negatif):  
4 40 14 8 26 1 38 2 32 -31  
Array setelah diurutkan:  
1 2 4 8 14 26 32 38 40  
Data berjarak tidak tetap  
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 12 & 13> █
```

Deskripsi Program :

Program meminta pengguna untuk memasukkan serangkaian bilangan bulat, yang dihentikan ketika bilangan negatif dimasukkan. Setelah data diurutkan, program mengevaluasi apakah perbedaan antara elemen-elemen berturut-turut sama, yang menunjukkan pola aritmetika. Hasil pengurutan dan status selisih antar elemen ditampilkan. Program ini berguna untuk memverifikasi apakah sekumpulan angka membentuk deret aritmetika setelah pengurutan.

2. UNGUIDED

1. Program untuk mengurutkan nomor rumah berdasarkan sifat bilangan (ganjil atau genap) dari beberapa daerah.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk melakukan selection sort ascending
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]
    }
}

// Fungsi untuk melakukan selection sort descending
func selectionSortDesc(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] {
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx], arr[i]
    }
}

func main() {
    var n int
    fmt.Println("Masukkan jumlah daerah: ")
    fmt.Scan(&n) // Membaca jumlah daerah

    // Loop untuk membaca setiap daerah
    for i := 0; i < n; i++ {
        var m int
        fmt.Printf("Masukkan jumlah rumah untuk daerah %d: \n", i+1)
        fmt.Scan(&m) // Membaca jumlah rumah di daerah ini

        // Membaca bilangan rumah
        odd := []int{} // Untuk bilangan ganjil
        even := []int{} // Untuk bilangan genap
```

```

        fmt.Println("Masukkan nomor rumah (spasi untuk
pisahkan setiap nomor):")
        for j := 0; j < m; j++ {
            var num int
            fmt.Scan(&num)
            if num%2 == 0 {
                even = append(even, num)
            } else {
                odd = append(odd, num)
            }
        }

        // Sort odd in ascending order
        selectionSortAsc(odd)

        // Sort even in descending order
        selectionSortDesc(even)

        // Cetak bilangan ganjil
        for _, v := range odd {
            fmt.Printf("%d ", v)
        }
        // Cetak bilangan genap
        for _, v := range even {
            fmt.Printf("%d ", v)
        }
        fmt.Println()
    }
}

```

Screenshoot Output :

```

● PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 12 & 13> go run
Masukkan jumlah daerah:
3
Masukkan jumlah rumah untuk daerah 1:
6
Masukkan nomor rumah (spasi untuk pisahkan setiap nomor):
5 2 1 7 9 13
1 5 7 9 13 2
Masukkan jumlah rumah untuk daerah 2:
5
Masukkan nomor rumah (spasi untuk pisahkan setiap nomor):
6 189 15 27 39 75 133
15 27 39 75 133 189 6
Masukkan jumlah rumah untuk daerah 3:
4
Masukkan nomor rumah (spasi untuk pisahkan setiap nomor):
3 4 9 1
1 3 9 4
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 12 & 13>

```


Deskripsi Program :

Program ini membaca sejumlah daerah, di mana masing-masing daerah memiliki daftar nomor rumah. Setiap daftar diproses dengan memisahkan nomor rumah menjadi bilangan ganjil dan genap. Bilangan ganjil diurutkan secara menaik (ascending) menggunakan algoritma selection sort, sedangkan bilangan genap diurutkan secara menurun (descending) dengan cara yang sama. Setelah proses pengurutan selesai, program mencetak bilangan ganjil terlebih dahulu, diikuti oleh bilangan genap, sesuai urutan yang telah ditentukan. Dengan pendekatan ini, keluaran menampilkan nomor rumah ganjil secara terurut dari kecil ke besar, diikuti dengan nomor rumah genap dari besar ke kecil, untuk setiap daerah yang diproses.

2. Program untuk menghitung dan mencetak median dari sebuah array setiap kali menemukan angka 0 dalam input.

Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

// Fungsi untuk menghitung median dari array yang telah
// terurut
func findMedian(arr []int) int {
    n := len(arr)
    if n%2 == 1 {
        // Jika jumlah elemen ganjil, ambil nilai tengah
        return arr[n/2]
    } else {
        // Jika jumlah elemen genap, rata-rata dua nilai
        // tengah, dibulatkan ke bawah
        return (arr[n/2-1] + arr[n/2]) / 2
    }
}

func main() {
    var num int
    data := []int{} // Array untuk menyimpan data

    for {
        fmt.Scan(&num)
        if num == -5313 {
            break // Akhiri saat menemukan marker -5313
        } else if num == 0 {
            // Urutkan array dengan insertion sort setiap
            // kali menemukan angka 0
            sort.Ints(data) // Menggunakan sort bawaan
            // Go untuk menyederhanakan proses
        } else {
            data = append(data, num)
        }
    }

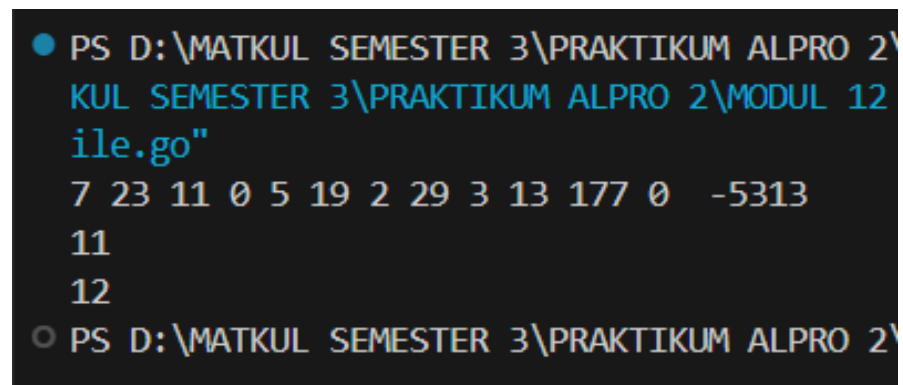
    // Cetak median
    fmt.Println("Median:", findMedian(data))
}
```

```

        median := findMedian(data)
        fmt.Println(median) // Cetak median
    } else {
        data = append(data, num) // Tambahkan bilangan
        ke dalam array
    }
}
}

```

Screenshoot Output :



```

PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\KUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 12> ile.go"
7 23 11 0 5 19 2 29 3 13 177 0 -5313
11
12
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\

```

Deskripsi Program :

Program ini membaca serangkaian bilangan bulat dan menghitung median dari data yang telah terbaca setiap kali mendeteksi angka 0. Bilangan yang dibaca disimpan dalam array, kecuali angka -5313, yang menandakan akhir input. Ketika menemukan angka 0, program mengurutkan array menggunakan fungsi bawaan `sort.Ints` dan kemudian menghitung median: jika jumlah elemen ganjil, median adalah nilai tengah, sedangkan jika genap, median adalah rata-rata dua nilai tengah yang dibulatkan ke bawah. Hasil median ini kemudian dicetak. Proses berlanjut hingga semua input selesai diproses.

- Sebuah program aplikasi pengelolaan data buku dalam perpustakaan.

Sourcecode :

```

package main

import (
    "fmt"
)

// Konstanta untuk jumlah maksimal buku
const nMax int = 7919

// Definisi struct untuk Buku

```

```

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

// DaftarBuku adalah array dari Buku
type DaftarBuku [nMax]Buku

// Fungsi untuk mendaftarkan buku
func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    for i := 0; i < n; i++ {
        fmt.Printf("Data buku ke-%d:\n", i+1)
        fmt.Scan(&pustaka[i].id, &pustaka[i].judul,
&pustaka[i].penulis, &pustaka[i].penerbit,
&pustaka[i].eksemplar, &pustaka[i].tahun,
&pustaka[i].rating)
    }
}

// Fungsi untuk mencetak buku dengan rating tertinggi
func CetakTerfavorit(pustaka DaftarBuku, n int) {
    maxRating := pustaka[0].rating
    index := 0
    for i := 1; i < n; i++ {
        if pustaka[i].rating > maxRating {
            maxRating = pustaka[i].rating
            index = i
        }
    }
    fmt.Println("Buku Terfavorit:")
    fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s,
Tahun: %d\n", pustaka[index].judul,
pustaka[index].penulis, pustaka[index].penerbit,
pustaka[index].tahun)
}

// Fungsi untuk mengurutkan buku berdasarkan rating
(insertion sort)
func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := pustaka[i]
        j := i - 1
        // Mengurutkan secara menurun
        for j >= 0 && pustaka[j].rating < key.rating {

```

```

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

// Fungsi untuk mencetak 5 buku dengan rating tertinggi
func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    fmt.Println("5 Buku dengan Rating Tertinggi:")
    for i := 0; i < n && i < 5; i++ {
        fmt.Printf("%d. %s\n", i+1, pustaka[i].judul)
    }
}

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

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

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

    // Mendaftarkan buku
    DaftarkanBuku(&pustaka, n)

```

```

// Cetak buku terfavorit sebelum pengurutan
CetakTerfavorit(pustaka, n)

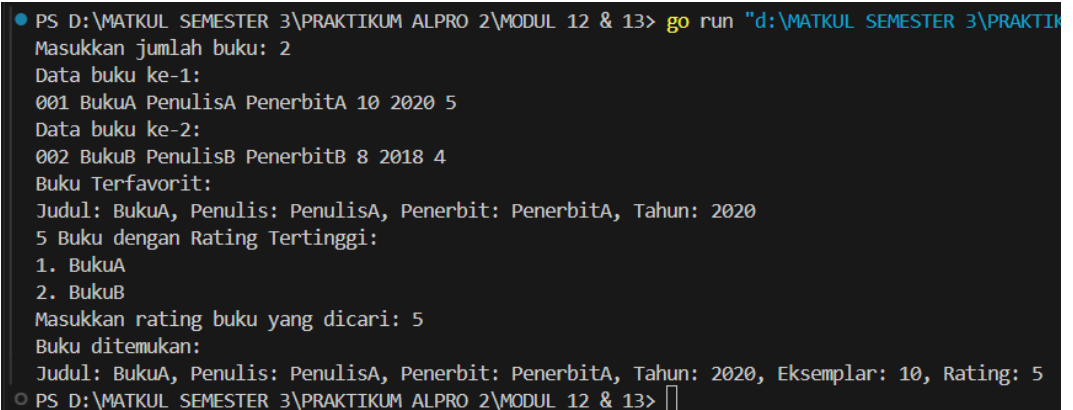
// Mengurutkan buku berdasarkan rating
UrutBuku(&pustaka, n)

// Cetak 5 buku dengan rating tertinggi
Cetak5Terbaru(pustaka, n)

// Mencari buku dengan rating tertentu
fmt.Print("Masukkan rating buku yang dicari: ")
fmt.Scan(&ratingCari)
CariBuku(pustaka, n, ratingCari)
}

```

Screenshoot Output :



```

PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 12 & 13> go run "d:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 12 & 13\main.go"
Masukkan jumlah buku: 2
Data buku ke-1:
001 BukuA PenulisA PenerbitA 10 2020 5
Data buku ke-2:
002 BukuB PenulisB PenerbitB 8 2018 4
Buku Terfavorit:
Judul: BukuA, Penulis: PenulisA, Penerbit: PenerbitA, Tahun: 2020
5 Buku dengan Rating Tertinggi:
1. BukuA
2. BukuB
Masukkan rating buku yang dicari: 5
Buku ditemukan:
Judul: BukuA, Penulis: PenulisA, Penerbit: PenerbitA, Tahun: 2020, Eksemplar: 10, Rating: 5
PS D:\MATKUL SEMESTER 3\PRAKTIKUM ALPRO 2\MODUL 12 & 13>

```

Deskripsi Program :

Program ini digunakan untuk mengelola data buku di sebuah perpustakaan dengan berbagai fitur utama. Pertama, pengguna memasukkan data buku ke dalam array yang disimpan dalam struktur Buku. Setelah data dimasukkan, program mencetak buku dengan rating tertinggi sebelum pengurutan. Selanjutnya, array buku diurutkan berdasarkan rating secara menurun menggunakan metode insertion sort. Setelah pengurutan, program mencetak lima buku dengan rating tertinggi. Terakhir, pengguna dapat mencari buku berdasarkan rating tertentu menggunakan metode pencarian biner. Jika ditemukan, informasi buku ditampilkan; jika tidak, muncul pesan bahwa buku dengan rating tersebut tidak ada.

KESIMPULAN

Kesimpulan mengenai praktikum sort adalah sebagai berikut:

Sorting adalah proses pengurutan data dalam urutan tertentu, baik itu secara menaik maupun menurun. Dalam bahasa Go, ada beberapa algoritma sorting yang dapat digunakan, seperti Selection Sort dan Insertion Sort. Selection Sort berfungsi dengan memilih elemen terkecil dari bagian yang belum terurut dan menukarnya dengan elemen pertama yang belum terurut. Meskipun algoritma ini sederhana, ia memiliki kompleksitas waktu $O(n^2)$, yang menjadikannya kurang efisien untuk data dalam jumlah besar. Insertion Sort, di sisi lain, bekerja dengan cara menyusun elemen satu per satu ke posisi yang benar dalam bagian yang terurut, dan meskipun kompleksitas waktunya juga $O(n^2)$ pada kasus terburuk, algoritma ini lebih efisien saat bekerja dengan data yang hampir terurut. Meskipun keduanya cocok untuk pengurutan data kecil atau yang hampir terurut, mereka tidak optimal untuk dataset besar jika dibandingkan dengan algoritma lain yang lebih canggih.

REFERENSI

[1] Dorin, Algorithms in Go: Insertion Sort. Dev.to. 2020.

<https://dev.to/dorin/algorithms-in-go-insertion-sort-2i0n>

[2] gostudent, Implementation of insertion sorting in go. Github. 2018.

<https://gostudent.github.io/Letsgo/Implementation-of-Insertion-Sort-in-Go>