

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 12&13
PENGURUTAN DATA**



Disusun Oleh:

Boutefhika Nuha Ziyadatul Khair/2311102316

IF-11-05

Dosen Pengampu:

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

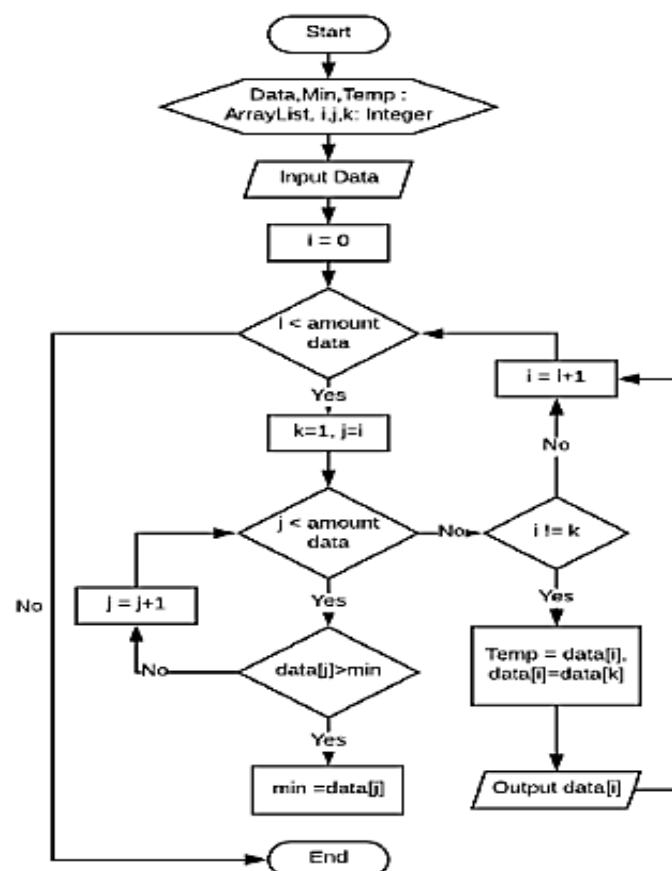
2024

I. DASAR TEORI

A. Selection Sort

Selection Sort Salah satu algoritma pengurutan yang paling sederhana. Algoritma yang lebih dikenal dengan algoritma pemilihan dimana melakukan pemilihan elemen minimum ataupun maksimum. Selection Sort adalah suatu Algoritma pengurutan yang membandingkan elemen yang sekarang dengan elemen berikutnya sampai ke elemen yang terakhir. Jika ditemukan elemen lain yang lebih kecil dari elemen sekarang maka dicatat posisinya dan langsung ditukar. Konsep Selection Sort adalah mencari atau memilih nilai terkecil ataupun nilai terbesar dan menukarnya dengan elemen paling awal pada paling kiri pada setiap tahap, proses akan terus berjalan hingga data akan menghasilkan data yang terurut. Algoritma ini melakukan banyak perbandingan namun memiliki jumlah perpindahan data yang sedikit.

Flowchart Algoritma Selection



Simulasi algoritma Selection Sort:

Data awal:

12	11	96	5	4
----	----	----	---	---

Iterasi 1:

12	11	96	5	4
----	----	----	---	---

Iterasi 2:

4	11	96	5	12
---	----	----	---	----

Iterasi 3:

4	5	96	11	12
---	---	----	----	----

Iterasi 4:

4	5	11	96	12
---	---	----	----	----

Iterasi 5:

4	5	11	12	96
---	---	----	----	----

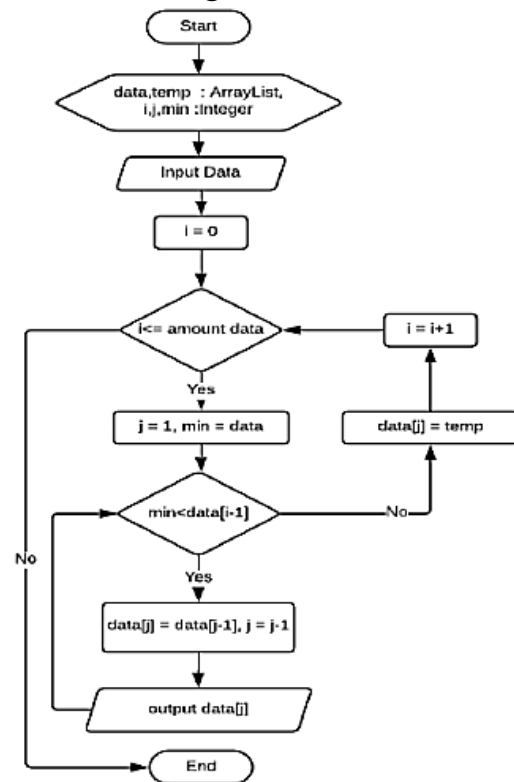
Kompleksitas waktu eksekusi Selection Sort pada Persamaan (2):

$$T(n) = n - 1 + n - 2 + n - 3 + \dots + 2 + 1 = \sum_{i=1}^{n-1} i = n - i \frac{n(n-1)}{2} = O(n^2)$$

B. Insertion Sort

Insertion Sort adalah sebuah algoritma pengurutan yang membandingkan dua elemen data pertama, mengurutkannya, kemudian mengecek elemen data berikutnya satu persatu dan membandingkannya dengan elemen data yang telah diurutkan. Algoritma pengurutan data insertion sort akan memeriksa setiap elemen, menemukan yang terkecil atau yang terbesar sesuai kebutuhan jenis pengurutan, dan menyisipkan dalam tempat yang tepat.

Flowchart Algoritma Insertion Sort



Simulasi algoritma Insertion Sort

Data awal:

12	11	96	5	4
----	----	----	---	---

Iterasi 1:

11	12	96	5	4
----	----	----	---	---

Iterasi 2:

11	12	96	5	4
----	----	----	---	---

Iterasi 3:

5	11	12	96	4
---	----	----	----	---

Iterasi 4:

4	5	11	12	96
---	---	----	----	----

Kompleksitas waktu eksekusi *Insertion Sort* pada Persamaan (1):

$$T(n) = 1 + 2 + 3 + \dots + n - 1 = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = O(n^2)$$

II. GUIDED

1. Hercules, preman terkenal seantero ibukota, memiliki kerabat di banyak daerah. Tentunya Hercules sangat suka mengunjungi semua kerabatnya itu.

Diberikan masukan nomor rumah dari semua kerabatnya di suatu daerah, buatlah program rumahkerabat yang akan menyusun nomor-nomor rumah kerabatnya secara terurut membesar menggunakan algoritma selection sort.

Masukan dimulai dengan sebuah integer n ($0 < n < 1000$), banyaknya daerah kerabat Hercules tinggal. Isi n baris berikutnya selalu dimulai dengan sebuah integer m ($0 < m < 1000000$) yang menyatakan banyaknya rumah kerabat di daerah tersebut, diikuti dengan rangkaian bilangan bulat positif, nomor rumah para kerabat.

Keluaran terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar di masing-masing daerah.

Sourcecode

```
package main

import (
    "fmt"
)

func selectionSort(arr []int, m int) {
    for i := 0; i < m-1; i++ {
        idxMin := i
        for j := i + 1; j < m; j++ { // Perbaikan,
membandingkan elemen setelah i
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
// Menukar elemen
    }
}

func main() {
    var n int
    fmt.Scan(&n) // Masukkan jumlah daerah kerabat

    for daerah := 1; daerah <= n; daerah++ {
        var m int
```

```

        fmt.Scan(&m) // Masukkan jumlah rumah
        kerabat di daerah ini

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

        // Output nomor rumah kerabat yang terurut
        for i, num := range arr {
            if i > 0 {
                fmt.Print(" ")
            }
            fmt.Print(num)
        }
        fmt.Println() // Pindah baris setelah
        selesai output nomor rumah
    }
}

```

Screenshoot Program

```

PS C:\Users\ASUS\Documents\Semester3> go run "c:\Users\ASUS\Documents\Semester3\Modul12&13\Guided\guided1.go"
3
5 2 1 7 9 13
1 2 7 9 13
6 189 15 27 39 75 133
15 27 39 75 133 189
3 4 9 1
1 4 9

```

Deskripsi Program

Program di atas menggunakan algoritma selection sort untuk mengurutkan nomor rumah kerabat di beberapa daerah. Pertama, program meminta input jumlah daerah (n). Untuk setiap daerah, program menerima jumlah rumah kerabat (m), kemudian menginputkan nomor rumah kerabat tersebut ke dalam array. Setelah itu, fungsi selectionSort digunakan untuk mengurutkan array nomor rumah tersebut dalam urutan menaik. Fungsi selectionSort bekerja dengan cara mencari elemen terkecil dalam array yang belum terurut dan menukarnya dengan elemen di posisi yang sesuai. Setelah array nomor rumah terurut, program mencetak hasilnya dalam satu baris, memisahkan setiap nomor rumah dengan spasi, dan melanjutkan ke daerah berikutnya. Setelah semua daerah diproses, program akan menampilkan nomor rumah kerabat yang terurut untuk masing-masing daerah.

2. Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metoda insertion sort), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.

Masukan terdiri dari sekumpulan bilangan bulat yang diakhiri oleh bilangan negatif. Hanya bilangan non negatif saja yang disimpan ke dalam array.

Keluaran terdiri dari dua baris. Baris pertama adalah isi dari array setelah dilakukan pengurutan, sedangkan baris kedua adalah status jarak setiap bilangan yang ada di dalam array. "Data berjarak x" atau "data berjarak tidak tetap".

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan
Insertion Sort
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 jarak antar elemen
tetap
func checkConstantDistance(arr []int) (bool, int)
{
    n := len(arr)
    if n < 2 {
        return false, 0
    }
}
```

```

        distance := arr[1] - arr[0]
        for i := 1; i < n-1; i++ {
            if arr[i+1]-arr[i] != distance {
                return false, 0
            }
        }
        return true, distance
    }

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)
    }

    // Urutkan array menggunakan Insertion Sort
    insertionSort(arr)

    // Periksa apakah jarak antar elemen tetap
    isConstant, distance :=
checkConstantDistance(arr)

    // Tampilkan array yang sudah diurutkan
    for i, val := range arr {
        if i > 0 {
            fmt.Print(" ")
        }
        fmt.Print(val)
    }
    fmt.Println()

    // Tampilkan status jarak
    if isConstant {
        fmt.Printf("Data berjarak %d\n", distance)
    }
}

```



```

    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}

```

Screenshoot Program

```

PS C:\Users\ASUS\Documents\Semester3> go run "c:\Users\ASUS\Documents\Semester3\Modul12&13\Guided\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
1 7 13 19 25 31 37 43
Data berjarak 6
PS C:\Users\ASUS\Documents\Semester3> go run "c:\Users\ASUS\Documents\Semester3\Modul12&13\Guided\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
4 40 14 8 26 1 38 2 32 -31
1 2 4 8 14 26 32 38 40
Data berjarak tidak tetap

```

Deskripsi Program

Program di atas menginput deretan angka integer hingga angka negatif ditemukan sebagai penanda akhir input. Setelah itu, program mengurutkan angka-angka tersebut menggunakan algoritma insertion sort, yang digunakan untuk membandingkan elemen saat ini (key) dengan elemen sebelumnya, dan memindahkan elemen yang lebih besar ke posisi kanan hingga posisi yang sesuai ditemukan. Setelah array terurut, program memeriksa apakah jarak antar elemen dalam array tetap konstan dengan fungsi `checkConstantDistance`. Fungsi ini membandingkan selisih antara elemen berturut-turut dan memastikan bahwa selisihnya sama untuk setiap pasangan elemen. Program kemudian mencetak array yang sudah terurut, diikuti dengan informasi yang menyatakan apakah jarak antar elemen tetap konstan atau tidak, beserta nilai jaraknya jika konstan.

III. UNGUIDED

1. Belakangan diketahui ternyata Hercules itu tidak berani menyeberang jalan, maka selalu diusahakan agar hanya menyeberang jalan sesedikit mungkin, hanya diujung jalan. Karena nomor rumah sisi kiri jalan selalu ganjil dan sisi kanan jalan selalu genap, maka buatlah program kerabat dekat yang akan menampilkan nomor rumah mulai dari nomor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

Format **Masukan** masih persis sama seperti sebelumnya.

Keluaran terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap, di masing-masing daerah.

Keterangan: Terdapat 3 daerah dalam contoh masukan. Baris kedua berisi campuran bilangan ganjil dan genap. Baris berikutnya hanya berisi bilangan ganjil, dan baris terakhir hanya berisi bilangan genap.

Petunjuk:

- Waktu pembacaan data, bilangan ganjil dan genap dipisahkan ke dalam dua array yang berbeda, untuk kemudian masing-masing diurutkan tersendiri.
- Atau, tetap disimpan dalam satu array, diurutkan secara keseluruhan. Tetapi pada waktu pencetakan, mulai dengan mencetak semua nilai ganjil lebih dulu, kemudian setelah selesai cetaklah semua nilai genapnya.

Sourcecode

```
package main

import "fmt"

func selectionSort(arr []int, naik bool) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        idx := i
        for j := i + 1; j < n; j++ {
            if naik && arr[j] < arr[idx] {
                idx = j
            } else if !naik && arr[j] > arr[idx] {
                idx = j
            }
        }
        arr[i], arr[idx] = arr[idx], arr[i]
    }
}

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

    if n <= 0 || n >= 1000 {
        return
    }

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

        if m <= 0 || m >= 1000000 {
            return
        }

        rumah := make([]int, m)
```

```

        for j := 0; j < m; j++ {
            fmt.Scan(&rumah[j])
        }

        ganjil := []int{}
        genap := []int{}

        // Pisahkan ganjil dan genap
        for _, num := range rumah {
            if num%2 == 1 {
                ganjil = append(ganjil, num)
            } else {
                genap = append(genap, num)
            }
        }

        // Urutkan ganjil naik dan genap turun
        selectionSort(ganjil, true)
        selectionSort(genap, false)

        // Cetak hasil
        for _, num := range ganjil {
            fmt.Printf("%d ", num)
        }
        for _, num := range genap {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}

```

Screenshoot Program

```

PS C:\Users\ASUS\Documents\Semester3> go run "c:\Users\ASUS\Documents\Semester3\Modul12&13\Unguided\unguided1.go"
3
5 2 1 7 9 13
1 7 9 13 2
6 189 15 27 39 75 133
15 27 39 75 133 189
3 4 9 1
1 9 4

```

Deskripsi Program

Program di atas menginput jumlah daerah (n), kemudian untuk setiap daerah, program menginput jumlah rumah (m) dan nomor rumah kerabat di daerah tersebut. Nomor rumah dipisahkan menjadi dua kelompok: angka ganjil dan genap. Setelah pemisahan, program menggunakan algoritma selection sort untuk mengurutkan nomor rumah ganjil dalam urutan menaik dan nomor rumah genap dalam urutan menurun. Algoritma selection sort bekerja dengan cara mencari elemen

terkecil atau terbesar dalam array yang belum terurut dan menukarnya dengan elemen di posisi yang sesuai. Setelah kedua kelompok (ganjil dan genap) terurut, program mencetak nomor rumah ganjil terlebih dahulu diikuti oleh nomor rumah genap, masing-masing dipisahkan dengan spasi. Jika jumlah rumah atau daerah melebihi batas yang ditentukan, program akan langsung berhenti tanpa melakukan pengolahan lebih lanjut.

2. Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi ternama. Dalam kompetisi tersebut, setiap tim berlomba untuk menyelesaikan sebanyak mungkin problem yang diberikan. Dari 13 problem yang diberikan, ada satu problem yang menarik. Problem tersebut mudah dipahami, hampir semua tim mencoba untuk menyelesaikannya, tetapi hanya 3 tim yang berhasil. Apa sih problemnya?

"Median adalah nilai tengah dari suatu koleksi data yang sudah terurut. Jika jumlah data genap, maka nilai median adalah rerata dari kedua nilai tengahnya. Pada problem ini, semua data merupakan bilangan bulat positif, dan karenanya rerata nilai tengah dibulatkan ke bawah."

Buatlah program median yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah 0.

Masukan berbentuk rangkaian bilangan bulat. Masukan tidak akan berisi lebih dari 1000000 data, tidak termasuk bilangan 0. Data 0 merupakan tanda bahwa median harus dicetak, tidak termasuk data yang dicari mediannya. Data masukan diakhiri dengan bilangan bulat -5313.

Keluaran adalah median yang diminta, satu data per baris.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Insertion Sort
func insertionSort(arr []int) {
    n := len(arr)
    for i := 1; i < n; i++ {
        key := arr[i]
        j := i - 1
        // Pindahkan elemen yang lebih besar dari
        key ke posisi berikutnya
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j = j - 1
        }
    }
}
```

```

        // Tempatkan key di posisi yang benar
        arr[j+1] = key
    }
}

// Fungsi untuk menghitung median dari data yang sudah
terurut
func hitungMedian(data []int) int {
    insertionSort(data) // Urutkan data menggunakan
    Insertion Sort

    n := len(data)
    if n%2 == 1 {
        // Jika jumlah data ganjil, ambil nilai
        tengah
        return data[n/2]
    } else {
        // Jika jumlah data genap, ambil rata-rata
        dua nilai tengah, dibulatkan ke bawah
        return (data[n/2-1] + data[n/2]) / 2
    }
}

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

    // Membaca input hingga menemukan -5313
    for {
        fmt.Scan(&input)

        if input == -5313 {
            break
        }

        if input == 0 {
            // Jika input adalah 0, hitung dan
            tampilkan median
            fmt.Println(hitungMedian(data))
        } else {
            // Jika input bukan 0, tambahkan data
            ke dalam list
            data = append(data, input)
        }
    }
}

```

Screenshoot Program

```

PS C:\Users\ASUS\Documents\Semester3> go run "c:\Users\ASUS\Documents\Semester3\Modul12&13\Unguided\unguided2.go"
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12

```

Deskripsi Program

Program di atas menginput angka satu per satu hingga ditemukan angka -5313, yang berfungsi sebagai penanda akhir input. Setiap angka selain -5313 dan 0 akan ditambahkan ke dalam list data. Ketika program menerima input 0, fungsi hitungMedian akan dipanggil untuk menghitung dan menampilkan median dari data yang telah dimasukkan. Fungsi hitungMedian mengurutkan data menggunakan algoritma insertion sort, yang bekerja dengan cara membandingkan elemen saat ini (key) dengan elemen sebelumnya dan memindahkan elemen yang lebih besar ke posisi yang sesuai. Setelah data terurut, median dihitung dengan cara memeriksa apakah jumlah elemen ganjil atau genap. Jika jumlah elemen ganjil, median adalah elemen tengah, sedangkan jika jumlah elemen genap, median dihitung sebagai rata-rata dua elemen tengah yang dibulatkan ke bawah. Program akan terus meminta input sampai angka -5313 dimasukkan, dan setiap kali input 0 diterima, median dari data yang telah dimasukkan akan ditampilkan.

3. Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan.

Masukan terdiri dari beberapa baris. Baris pertama adalah bilangan bulat N yang menyatakan banyaknya data buku yang ada di dalam perpustakaan. N baris berikutnya, masing-masingnya adalah data buku sesuai dengan atribut atau field pada struct. Baris terakhir adalah bilangan bulat yang menyatakan rating buku yang akan dicari.

Keluaran terdiri dari beberapa baris. Baris pertama adalah data buku terfavorit, baris kedua adalah lima judul buku dengan rating tertinggi, selanjutnya baris terakhir adalah data buku yang dicari sesuai rating yang diberikan pada masukan baris terakhir.

Sourcecode

```
package main

import (
    "fmt"
)

const nMax = 7919

type Buku struct {
    ID          string
    Judul       string
    Penulis     string
    Penerbit    string
    Eks         int
    Tahun       int
    Rating      int
}
```

```

}

type DaftarBuku [nMax]Buku

var pustaka DaftarBuku
var nPustaka int

// DaftarkanBuku mendaftarkan buku ke dalam pustaka
func DaftarkanBuku() {
    fmt.Scan(&nPustaka)
    for i := 0; i < nPustaka; i++ {
        fmt.Scan(&pustaka[i].ID)
        fmt.Scan(&pustaka[i].Judul)
        fmt.Scan(&pustaka[i].Penulis)
        fmt.Scan(&pustaka[i].Penerbit)
        fmt.Scan(&pustaka[i].Tahun)
        fmt.Scan(&pustaka[i].Eks)
        fmt.Scan(&pustaka[i].Rating)
    }
}

// CetakTerfavorit mencetak buku dengan rating tertinggi
func CetakTerfavorit() {
    maxRating := pustaka[0].Rating
    for i := 1; i < nPustaka; i++ {
        if pustaka[i].Rating > maxRating {
            maxRating = pustaka[i].Rating
        }
    }

    for i := 0; i < nPustaka; i++ {
        if pustaka[i].Rating == maxRating {
            fmt.Println("Data Buku Terfavorit:")
            fmt.Printf("Judul: %s\n", pustaka[i].Judul)
            fmt.Printf("Penulis: %s\n",
pustaka[i].Penulis)
            fmt.Printf("Penerbit: %s\n",
pustaka[i].Penerbit)
            fmt.Printf("Tahun: %d\n", pustaka[i].Tahun)
            break // hanya cetak satu buku terfavorit
        }
    }
}

// UrutBuku mengurutkan buku berdasarkan rating
menggunakan Insertion Sort
func UrutBuku() {
    for i := 1; i < nPustaka; 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
    }
}

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

// CariBuku mencari buku berdasarkan rating tertentu
func CariBuku(r int) {
    low, high := 0, nPustaka-1
    found := false

    for low <= high {
        mid := (low + high) / 2
        if pustaka[mid].Rating == r {
            found = true
            fmt.Printf("Buku dengan rating %d:\n", r)
            fmt.Printf("Judul: %s\n", pustaka[mid].Judul)
            fmt.Printf("Penulis: %s\n", pustaka[mid].Penulis)
            fmt.Printf("Penerbit: %s\n", pustaka[mid].Penerbit)
            fmt.Printf("Tahun: %d\n", pustaka[mid].Tahun)
            fmt.Printf("Eksemplar: %d\n", pustaka[mid].Eks)
            fmt.Printf("Rating: %d\n", pustaka[mid].Rating)
            break
        } else if pustaka[mid].Rating < r {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
}

```



```

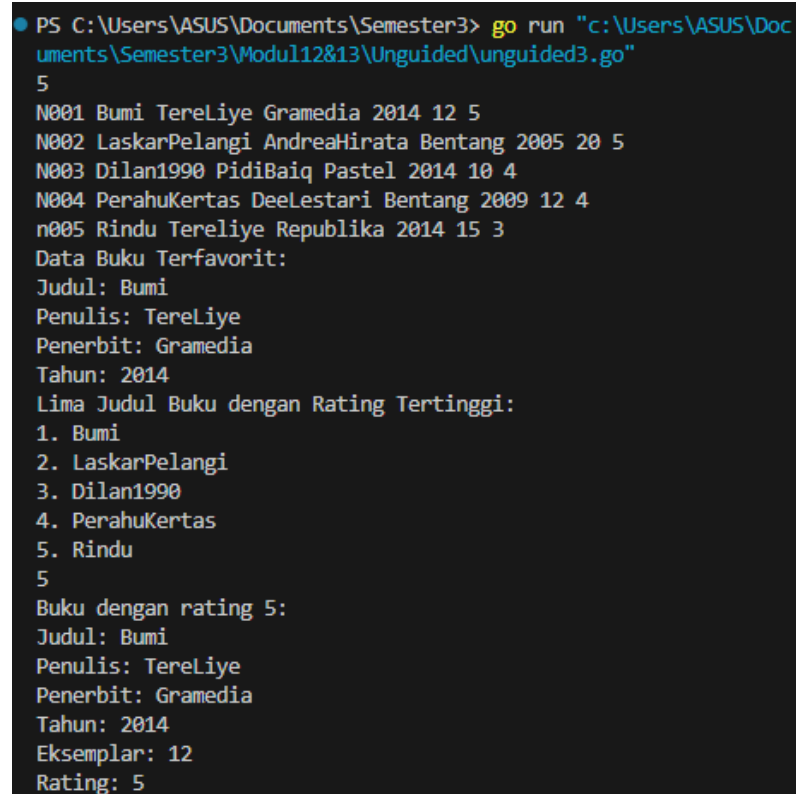
        if !found {
            fmt.Println("Tidak ada buku dengan rating seperti itu.")
        }
    }

    func main() {
        DaftarkanBuku()
        CetakTerfavorit()
        UrutBuku()
        Cetak5Terbaru()

        var ratingCari int
        fmt.Scan(&ratingCari)
        CariBuku(ratingCari)
    }

```

Screenshoot Program



```

PS C:\Users\ASUS\Documents\Semester3> go run "c:\Users\ASUS\Documents\Semester3\Modul12&13\Unguided\unguided3.go"
5
N001 Bumi TereLiye Gramedia 2014 12 5
N002 LaskarPelangi AndreaHirata Bentang 2005 20 5
N003 Dilan1990 PidiBaiq Pastel 2014 10 4
N004 PerahuKertas DeeLestari Bentang 2009 12 4
n005 Rindu TereLiye Republika 2014 15 3
Data Buku Terfavorit:
Judul: Bumi
Penulis: TereLiye
Penerbit: Gramedia
Tahun: 2014
Lima Judul Buku dengan Rating Tertinggi:
1. Bumi
2. LaskarPelangi
3. Dilan1990
4. PerahuKertas
5. Rindu
5
Buku dengan rating 5:
Judul: Bumi
Penulis: TereLiye
Penerbit: Gramedia
Tahun: 2014
Eksemplar: 12
Rating: 5

```

Deskripsi Program

Program di atas menginput jumlah buku yang akan didaftarkan, kemudian untuk setiap buku, data seperti ID, Judul, Penulis, Penerbit, Tahun, Eksemplar, dan Rating dimasukkan ke dalam array pustaka. Setelah itu, program menampilkan buku dengan rating tertinggi menggunakan fungsi CetakTerfavorit. Selanjutnya, buku diurutkan

berdasarkan rating tertinggi ke terendah dengan menggunakan algoritma insertion sort, dan lima buku dengan rating tertinggi ditampilkan menggunakan fungsi Cetak5Terbaru. Terakhir, program memungkinkan pengguna untuk mencari buku berdasarkan rating tertentu melalui fungsi CariBuku yang menggunakan metode pencarian biner. Jika buku dengan rating tersebut ditemukan, program akan menampilkan informasi lengkap tentang buku tersebut, jika tidak, program akan memberitahukan bahwa buku tersebut tidak ditemukan.

IV. KESIMPULAN

Selection Sort adalah algoritma pengurutan yang paling sederhana, yang memilih elemen terkecil atau terbesar dalam array dan menukarnya dengan elemen di posisi yang sesuai. Sedangkan Insertion Sort mengurutkan data dengan membandingkan elemen saat ini dengan elemen-elemen sebelumnya, dan memindahkan elemen yang lebih besar ke posisi yang sesuai. Contoh programnya seperti menyusun nomor rumah kerabat Hercules secara terurut menggunakan algoritma Selection Sort, serta memeriksa jarak antar elemen setelah diurutkan menggunakan Insertion Sort. Selain itu, ada program pengurutan nomor rumah berdasarkan ganjil dan genap dengan Selection Sort serta menghitung median dari data yang telah diurutkan. Program yang dihasilkan mengimplementasikan algoritma-algoritma tersebut untuk menyelesaikan permasalahan yang diberikan, termasuk pengurutan dan pemeriksaan kondisi tertentu pada data yang diberikan.

V. REFERENSI

- [1] Aditya, D., & Putra, M. F. (2020). *Perbandingan efisiensi algoritma sorting dalam penggunaan bandwidth*. ResearchGate.