LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN XII

MODUL 12 & 13

PENGURUTAN DATA

Disusun Oleh :

SHAFA ADILA SANTOSO / 2311102158

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

SORTING

Pengurutan data (sorting) adalah suatu proses pengurutan data yang tersusun secara acak pada suatu pola tertentu, sehingga tersusun secara teratur menurut aturan tertentu. pengurutan ini dapat dilakukan dengan cara Ascending dan descending serta digunakan juga untuk mengurutkan data yang bertipe angka atau karakter.

ALGOTITMA SELECTION SORT

Selection Sort adalah suatu metode 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. Metode selection sort adalah melakukan pemilihan dari suatu nilai yang terkecil dan kemudian menukarnya dengan elemen paling awal, lalu membandingkan dengan elemen yang sekarang dengan elemen berikutnya sampai dengan elemen terakhir, perbandingan dilakukan terus sampai tidak ada lagi pertukaran data. Langkah langkah Selesction Sort:

1. Cari nilai terkecil didalam rentang data tersisa
2. Kemudian tukar tempat dengan data yang berada pada posisi paling kiri pada rentang data tersisa.
3. Ulangi proses sampai tersisa satu data saja

Kemudian dibawah ini merupakan contoh algoritma Selection Sort pada pengurutan array bertipe data bilangan bulat secara ascending;

```
..  ...
5   type arrInt [4321]int
..  ...
15  func selectionSort1(T *arrInt, n int){
16  /* I.S. terdefinisi array T yang berisi n bilangan bulat
17     F.S. array T terurut secara asceding atau membesar dengan SELECTION SORT */
18      var t, i, j, idx_min int
19
```

```
20      i = 1
21      for i <= n-1 {
22          idx_min = i - 1
23          j = i
24          for j < n {
25              if T[idx_min] > T[j] {
26                  idx_min = j
27              }
28              j = j + 1
29          }
30          t = T[idx_min]
31          T[idx_min] = T[i-1]
32          T[i-1] = t
33          i = i + 1
34      }
35  }
```

ALGORITMA INSERTION SORT

Insertion Sort adalah sebuah algoritma sederhana yang cukup efisien untuk mengurutkan sebuah list yang hampir terurut. Cara kerja algoritma ini adlah dengan mengambil elemen list satu per satu dan masukannya di posisi yang benar seperti namanya. Pengurutan secara insertion ini idenya adalah menyisipkan suatu nilai pada posisi yang seharusnya. Berbeda dengan pengurutan seleksi, yang mana pada pengurutan ini tidak dilakukan pencarian nilai ekstrim terlebih dahulu, cukup memilih suatu nilai tertentu kemudian mencari posisinya secara sequential search. Pada penjelasan berikut ini data akan diurut mengecil (descending), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

1. Untuk satu data yang belum terurut dan sejumlah data yang sudah diurutkan: Geser data yang sudah terurut tersebut (ke kanan), sehingga ada satu ruang kosong untuk memasukkan data yang belum terurut ke dalam data yang sudah terurut dan tetap menjaga keterurutan.
2. Ulangi proses tersebut untuk setiap data yang belum terurut terhadap rangkaian data yang sudah terurut.
   Algoritma ini dikenal juga dengan nama Insertion Sort, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian sekuensial dan penyisipan.

	Notasi Algoritma	Notasi dalam bahasa Go
1	`i ← 1`	`i = 1`
2	`while i <= n-1`	`for i <= n-1 {`
3	`  j ← i`	`  j = i`
4	`  temp ← a[j]`	`  temp = a[j]`
5	`  while j > 0 and a[j] > a[j-1] do`	`  for j > 0 && a[j] > a[j-1] {`
6	`  a[j] ← a[j-1]`	`  a[j] = a[j-1]`
7	`  j ← j - 1`	`  j = j - 1`
8	`  endwhile`	`  }`
9	`  a[j] ← temp`	`  a[j] = temp`
10	`  i ← i + 1`	`  i = i + 1`
11	`endwhile`	`}`

Adapun algoritma insertion sort untuk mengurutkan array bertipe data bilangan bulat secara mengecil atau descending adalah sebagai berikut ini!

```
..  ...
5   type arrInt [4321]int
..  ...
15  func insertionSort1(T *arrInt, n int){
16  /* I.S. terdefinisi array T yang berisi n bilangan bulat
17     F.S. array T terurut secara mengecil atau descending dengan INSERTION SORT*/
18      var temp, i, j int
19      i = 1
20      for i <= n-1 {
21          j = i
22          temp = T[j]
23          for j > 0 && temp > T[j-1] {
24              T[j] = T[j-1]
25              j = j - 1
26          }
27          T[j] = temp
28          i = i + 1
29      }
30  }
```

II. GUIDED

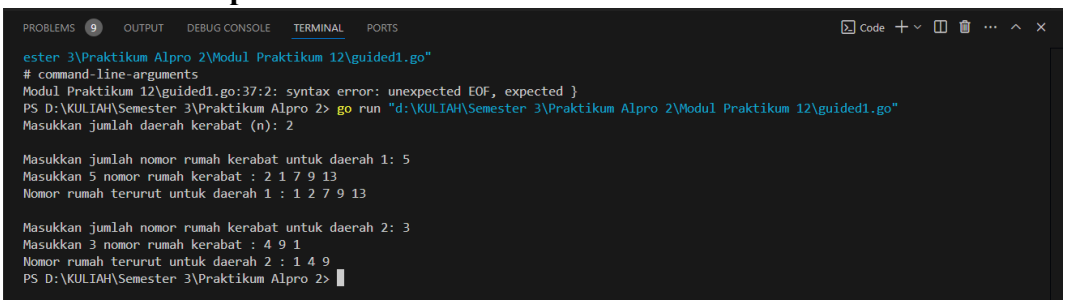
1. Program Hercules, preman terkenal seantero ibukota, memiliki kerabat di banyak daerah. Tentunya Hercules sangat suka mengunjungi semua kerabatnya itu. Diberikan masukan nomor rumah dari semua kerabatnya di suatu daerah, buatlah program rumahkerabat yang akan menyusun nomor-nomor rumah kerabatnya secara 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"
)

// 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



Deskripsi Program

Program ini merupakan program Go yang mengimplementasikan algoritma Sleestion Sort untuk mengurutkan nomor rumah kerabat di beberapa daerah. Program diawali dengan meminta input jumlah daerah kerabat (*n*), kemudian untuk setiap daerah, pengguna diminta memasukkan jumlah nomor rumah kerabat (*m*) dan daftar nomor rumah tersebut. Nomor-nomor rumah yang dimasukkan disimpan dalam array. Program menggunakan algoritma *selection sort* untuk mengurutkan array secara menaik, yaitu dengan mencari elemen terkecil dalam subarray yang belum diurutkan, lalu menukarnya dengan elemen di posisi awal subarray tersebut. Setelah proses pengurutan selesai, nomor-nomor rumah yang sudah terurut akan ditampilkan sebagai output untuk setiap daerah. Hasil akhirnya adalah daftar nomor rumah yang terurut untuk semua daerah yang dimasukkan pengguna.

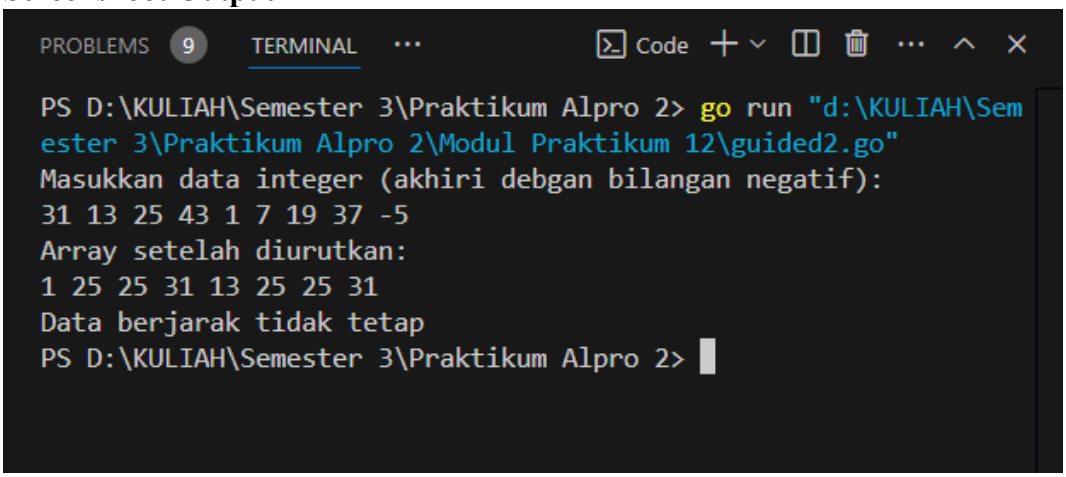
2. Buatlah sebuah program yang digunakan untuk membaca data Integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metoda insertion sort), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.
   Masukan terdiri dari sekumpulan bilangan bulat yang 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"

func insertionSort(arr []int, n int) {
    for i := 1; i < n; i++ {
        key := arr[i]
        j := i - 1

        for j >= 0 && arr[j] > key {
            arr[j+i] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

func isConstantDifference(arr []int, n int) (bool, int)
{
    if n < 2 {
        return true, 0
    }

    difference := arr [1] - arr[0]
    for i := 1; i < n-2; i++ {
        return false,  0
    }
    return true, difference
}

func main(){
    var arr []int
    var num int
    fmt.Println("Masukkan data integer (akhiri debgan
bilangan negatif): ")
    for {
        fmt.Scan(&num)
```

```
        if num < 0 {
            break
        }
        arr = append(arr, num)
    }

    n := len(arr)

    insertionSort(arr, n)

    isConstant, difference := isConstantDifference(arr,
n)

    fmt.Println("Array setelah diurutkan: ")
    for _, val := range arr {
        fmt.Printf("%d ", val)
    }
    fmt.Println()

    if isConstant {
        fmt.Printf("Data berjarak %d\n", difference)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}
```

Screenshoot Output



Deskripsi Program

Program ini merupakan progam Go yang mengimplementasikan dari algoritma Insertion Sort untuk mengurutkan sekumpulan bilangan bulat, kemudian memeriksa apakah perbedaan antara setiap elemen dalam array yang sudah diurutkan adalah konstan. Pengguna diminta untuk memasukkan bilangan bulat secara berurutan, dengan memasukkan

bilangan negatif sebagai tanda untuk mengakhiri input. Setelah data diterima, program mengurutkan bilangan tersebut dengan *insertion sort*, di mana setiap elemen array dimasukkan ke posisi yang tepat dalam subarray yang sudah diurutkan. Setelah array diurutkan, program memeriksa apakah selisih antara elemen-elemen bertetangga dalam array memiliki nilai yang sama. Jika ya, program mencetak perbedaan konstan tersebut; jika tidak, program menyatakan bahwa selisih tidak tetap. Sebagai output, program menampilkan array terurut dan hasil analisis jarak antar elemen.

III. UNGUIDED

1. Suatu lingkaran 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 harıya 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

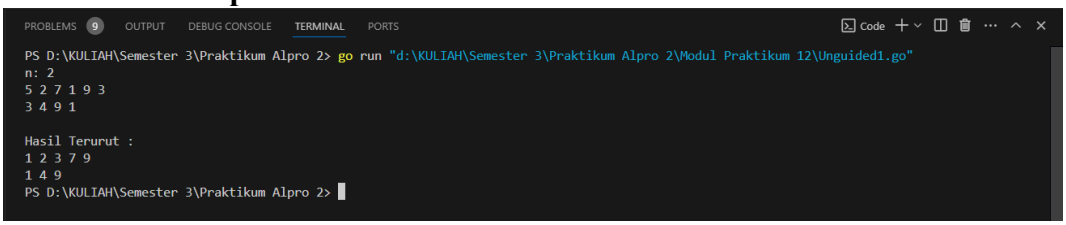
semua nilai ganjil lebih dulu, kemudian setelah selesal cetaklah semua nilai genapnya.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk melakukan selection sort menaik
func SelectionSortMembesar(arr []int, n_158 int) {
    for i := 0; i < n_158-1; i++ {
        idxMin := i
        for j := i + 1; j < n_158; j++ {
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}

// Fungsi untuk melakukan selection sort menurun
func SelectionSortMengecil(arr []int, n_158 int) {
    for i := 0; i < n_158-1; i++ {
        idxMax := i
        for j := i + 1; j < n_158; j++ {
            if arr[j] > arr[idxMax] {
                idxMax = j
            }
        }
        arr[i], arr[idxMax] = arr[idxMax], arr[i]
    }
}

func main() {
    var n_158 int
    fmt.Print("n: ")
    fmt.Scan(&n_158)

    // List untuk menyimpan semua hasil terurut
    var hasil [][]int

    for i := 0; i < n_158; i++ {
        var m int
        fmt.Scan(&m)

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

```
            if m%2 == 0 {
                SelectionSortMengecil(arr, m)
            } else {
                SelectionSortMembesar(arr, m)
            }

            hasil = append(hasil, arr)
        }

        // Cetak semua hasil terurut setelah input selesai
        fmt.Println("\nHasil Terurut :")
        for _, arr := range hasil {
            for _, num := range arr {
                fmt.Printf("%d ", num)
            }
            fmt.Println()
        }
    }
```

Screenshoot Output



Deskripsi Program

Program ini program Go yang mengimplementasikan algoritma Selection Sort untuk mengurutkan sekumpulan array bilangan bulat dalam dua mode: secara menaik (ascending) atau menurun (descending), tergantung pada panjang array. Pengguna diminta untuk memasukkan jumlah array yang akan diolah (n), lalu memasukkan jumlah elemen (m) dan nilai-nilai dari setiap array. Jika jumlah elemen array genap, array diurutkan secara menurun menggunakan fungsi SelectionSortMengecil. Sebaliknya, jika jumlah elemen ganjil, array diurutkan secara menaik menggunakan fungsi SelectionSortMembesar. Dalam kedua fungsi tersebut, algoritma selection sort mencari elemen maksimum atau minimum dalam subarray yang belum diurutkan, lalu menukarnya dengan elemen di posisi yang sesuai. Hasilnya, program menampilkan semua array yang telah diurutkan berdasarkan aturan tersebut. Output akhir berupa daftar array yang terurut untuk setiap input yang diberikan.

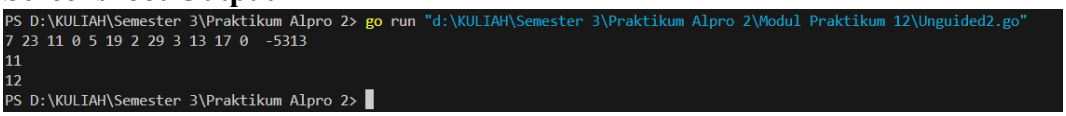
2. Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi temama. Dalam kompetisi tersebut, setiap tim berlomba untuk menyelesaikan sebanyak mungkin problem yang diberikan. Dari 13 problem yang diberikan, ada satu problem yang menarik. Problem tersebut mudah dipahami, 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 genop, 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.
   Keterangan:
   Sampai bilangan 0 yang pertama, data terbaca adalah 7 23 11, setelah tersusun: 7 11 23. maka median saat itu adalah 11.
   Sampai bilangan 0 yang bredua, data adalah 7 23 11 5 19 2 29 3 13. 17, setelah tersusun diperoleh. 2 3 5 7 11 13 17 19 23 29. Karena ada 10 data, genap, maka median adalah (11-13)/2=12.
   Petunjuk
   Untuk setiap data bukkan 0 (dan bukan marker -5313541) simpan ke dalam array, Dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metode insertion sort dan ambil mediannya.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk melakukan selection sort menaik
func SelectionSort(arr []int, n_158 int) {
    for i := 0; i < n_158-1; i++ {
        idxMin := i
        for j := i + 1; j < n_158; j++ {
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
```

```
            arr[i], arr[idxMin] = arr[idxMin], arr[i]
        }
    }

    // Fungsi untuk mencari median
    func findMedian(data []int) int {
        n_158 := len(data)
        if n_158%2 == 1 {
            return data[n_158/2]
        }
        return (data[n_158/2-1] + data[n_158/2]) / 2
    }

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

        for {
            fmt.Scan(&input)
            if input == -5313 {
                break
            }
            if input == 0 {
                if len(data) == 0 {
                    continue
                }
                SelectionSort(data, len(data))
                median := findMedian(data)
                fmt.Println(median)
            } else {
                data = append(data, input)
            }
        }
    }
```

Screenshoot Output



Deskripsi Program

Program ini merupakan program Go yang mengimplementasikan algoritma Selection Sort untuk menghitung nilai median dari data yang dimasukkan oleh pengguna secara dinamis. Pengguna dapat memasukkan bilangan bulat secara berurutan. Jika pengguna memasukkan angka 0, program akan mengurutkan data yang telah dimasukkan hingga saat itu secara menaik menggunakan selection sort, lalu menghitung median. Median dihitung sebagai elemen tengah untuk data dengan jumlah elemen ganjil, atau rata-rata dari dua elemen tengah untuk data dengan jumlah

elemen genap. Input dihentikan ketika pengguna memasukkan -5313. Program ini memproses setiap batch data yang diberikan sebelum angka 0, menampilkan median sebagai output, dan melanjutkan menerima data baru.

3. Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Misalnya terdefinisi struct dan array seperti berikut ini:

```
const nMax integer = 7919
type Buku = <
    id, judul, penulis, penerbit : string
    eksemplar, tahun, rating : integer >

type DaftarBuku array [1..nMax] of Buku
Pustaka : DaftarBuku
nPustaka: integer
```

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.
Lengkapi subprogram-subprogram dibawah ini, sesuai dengan I.S. dan F.S yang diberikan.

```
procedure DaftarkanBuku(in/out pustaka : DaftarBuku, n : integer)
{I.S. sejumlah n data buku telah siap para piranti masukan
F.S. n berisi sebuah nilai, dan pustaka berisi sejumlah n data buku}

procedure Cetak Terfavorit(in pustaka : DaftarBuku, in n : integer)
{I.S. array pustaka berisi n buah data buku dan belum terurut
F.S. Tampilan data buku (judul, penulis, penerbit, tahun) terfavorit, yaitu
memiliki rating tertinggi}

procedure UrutBuku(in/out pustaka : DaftarBuku, n : integer)
{I.S. Array pustaka berisi n data buku
F.S. Array pustaka terurut menurun/mengecil terhadap rating.
Catatan: Gunakan metoda Insertion sort}

procedure Cetak5Terbaru( in pustaka : DaftarBuku, n integer)
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
```

F.S. Laporan 5 judul buku dengan rating tertinggi
Catatan: Isi pustaka mungkin saja kurang dari 5}

procedure CariBuku (in pustaka : DaftarBuku, n : integer, r : integer)
{I.S. pustaka berisı n data buku yang sudah terurut menurut rating
F.S. Laporan salah satu buku (judul, penulis, penerbit, tahun, eksemplar, rating) dengan rating yang diberikan. Jika tidak ada buku dengan rating yang ditanyakan, cukup tuliskan "Tidak ada buku dengan rating seperti itu". Catatan: Gunakan pencarian biner/belah dua.}

Sourcecode

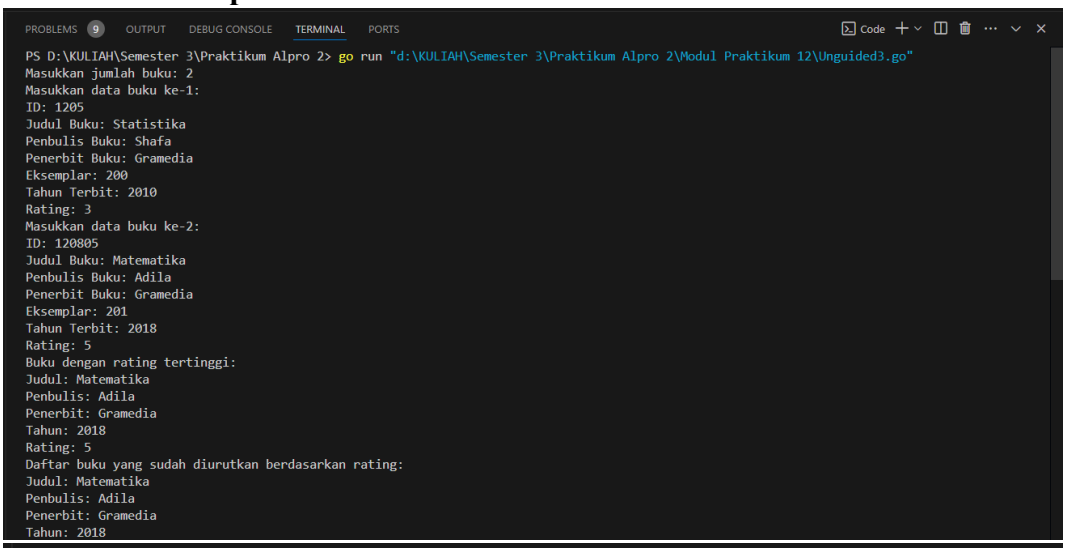
```
package main

import "fmt"

// Definisi konstanta dan struct
const nMax = 7919

type Buku struct {
    ID_158       int
    Judul_158    string
    Penbulis_158  string
    Penerbit_158 string
    Eksemplar_158 int
    Tahun_158    int
    Rating_158   int
}

type DaftarBuku [nMax]Buku

// Fungsi untuk memasukkan data buku
func DaftarKanBuku(pustaka *DaftarBuku, n *int) {
    var jumlahBuku int
    fmt.Print("Masukkan jumlah buku: ")
    fmt.Scan(&jumlahBuku)

    for i := 0; i < jumlahBuku; i++ {
        fmt.Printf("Masukkan data buku ke-%d:\n", i+1)
        fmt.Print("ID: ")
        fmt.Scan(&pustaka[i].ID_158)
        fmt.Print("Judul Buku: ")
        fmt.Scan(&pustaka[i].Judul_158)
        fmt.Print("Penbulis Buku: ")
        fmt.Scan(&pustaka[i].Penbulis_158)
        fmt.Print("Penerbit Buku: ")
        fmt.Scan(&pustaka[i].Penerbit_158)
        fmt.Print("Eksemplar: ")
        fmt.Scan(&pustaka[i].Eksemplar_158)
        fmt.Print("Tahun Terbit: ")
        fmt.Scan(&pustaka[i].Tahun_158)
        fmt.Print("Rating: ")
```

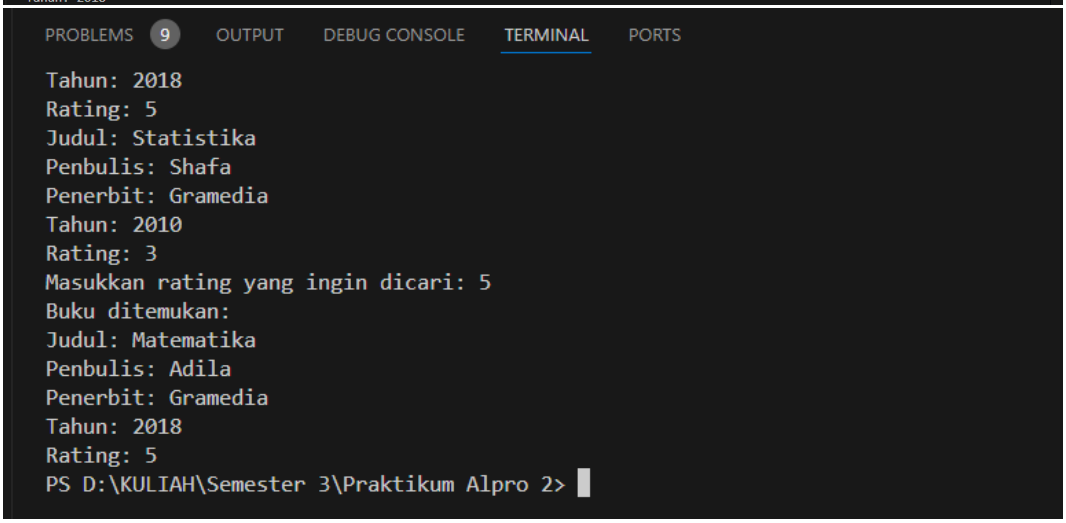
```
        fmt.Scan(&pustaka[i].Rating_158)
    }
    *n = jumlahBuku
}

// Fungsi untuk mencari buku favorit
func CetakFavorit(pustaka DaftarBuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada buku dalam pustaka.")
        return
    }

    tertinggi := pustaka[0]
    for i := 1; i < n; i++ {
        if pustaka[i].Rating_158 > tertinggi.Rating_158
{
            tertinggi = pustaka[i]
        }
    }

    fmt.Println("Buku dengan rating tertinggi:")
    fmt.Printf("Judul: %s\n", tertinggi.Judul_158)
    fmt.Printf("Penbulis: %s\n", tertinggi.Penbulis_158)
    fmt.Printf("Penerbit: %s\n", tertinggi.Penerbit_158)
    fmt.Printf("Tahun: %d\n", tertinggi.Tahun_158)
    fmt.Printf("Rating: %d\n", tertinggi.Rating_158)
}

// Fungsi untuk mengurutkan buku menggunakan algoritma
selection sort
func UrutkanBuku(pustaka *DaftarBuku, n int) {
    for i := 0; i < n-1; i++ {
        maxIdx := i
        for j := i + 1; j < n; j++ {
            if pustaka[j].Rating_158 >
pustaka[maxIdx].Rating_158 {
                maxIdx = j
            }
        }
        // Tukar buku dengan rating tertinggi ke posisi
i
        pustaka[i], pustaka[maxIdx] = pustaka[maxIdx],
pustaka[i]
    }
}

// Fungsi untuk mencetak buku yang sudah diurutkan
func CetakTerurut(pustaka DaftarBuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada buku dalam pustaka.")
        return
    }
```

```
    fmt.Println("Daftar buku yang sudah diurutkan
berdasarkan rating:")
    for i := 0; i < n; i++ {
        fmt.Printf("Judul: %s\n", pustaka[i].Judul_158)
        fmt.Printf("Penbulis: %s\n",
pustaka[i].Penbulis_158)
        fmt.Printf("Penerbit: %s\n",
pustaka[i].Penerbit_158)
        fmt.Printf("Tahun: %d\n", pustaka[i].Tahun_158)
        fmt.Printf("Rating: %d\n",
pustaka[i].Rating_158)
    }
}

// Fungsi untuk mencari buku berdasarkan rating tertentu
func CariBuku(pustaka DaftarBuku, n int, rating int) {
    found := false
    for i := 0; i < n; i++ {
        if pustaka[i].Rating_158 == rating {
            if !found {
                fmt.Println("Buku ditemukan:")
                found = true
            }
            fmt.Printf("Judul: %s\n",
pustaka[i].Judul_158)
            fmt.Printf("Penbulis: %s\n",
pustaka[i].Penbulis_158)
            fmt.Printf("Penerbit: %s\n",
pustaka[i].Penerbit_158)
            fmt.Printf("Tahun: %d\n",
pustaka[i].Tahun_158)
            fmt.Printf("Rating: %d\n",
pustaka[i].Rating_158)
        }
    }
    if !found {
        fmt.Println("Tidak ada buku dengan rating
tersebut.")
    }
}

// Fungsi utama
func main() {
    var pustaka DaftarBuku
    var n int

    DaftarKanBuku(&pustaka, &n)
    CetakFavorit(pustaka, n)
    UrutkanBuku(&pustaka, n)
    CetakTerurut(pustaka, n)
```

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

Screenshoot Output





Deskripsi Program

Program ini adalah program Go yang mengimplementasikan algoritma Selection Sort untuk memasukkan informasi buku, menemukan buku dengan rating tertinggi, mengurutkan buku berdasarkan rating, dan mencari buku berdasarkan rating tertentu. Data setiap buku mencakup ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Program menggunakan algoritma selection sort untuk mengurutkan buku berdasarkan rating dari yang tertinggi ke terendah. Setelah pengguna memasukkan data buku, program akan menampilkan buku dengan rating

tertinggi, mengurutkan dan mencetak daftar buku berdasarkan rating, serta memungkinkan pencarian buku berdasarkan rating tertentu. Outputnya mencakup detail buku yang dimasukkan, buku favorit, daftar buku terurut, dan hasil pencarian buku sesuai rating yang dimasukkan pengguna.

DAFTAR PUSTAKA

[1] Asisten Praktikum, “Modul 7 Struct dan Array”, Modul Praktikum, 2024.
[2] Rahayuningsih, P. A. (2016). Analisis Perbandingan Kompleksitas Algoritma Pengurutan Nilai (Sorting). EVOLUSI: Jurnal Sains dan Manajemen, 4(2).
[3] Lasriana, L., & Gunaryati, A. (2022). Sistem Informasi Apotek Berbasis Web Menggunakan Algoritma Sequential Search Dan Selection Sort. JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika), 7(2), 392-401.