LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII

PENGURUTAN DATA

Disusun Oleh :

Aji Noto Sutrisno (2311102262)

IF 11 05

Dosen Pengampu :

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

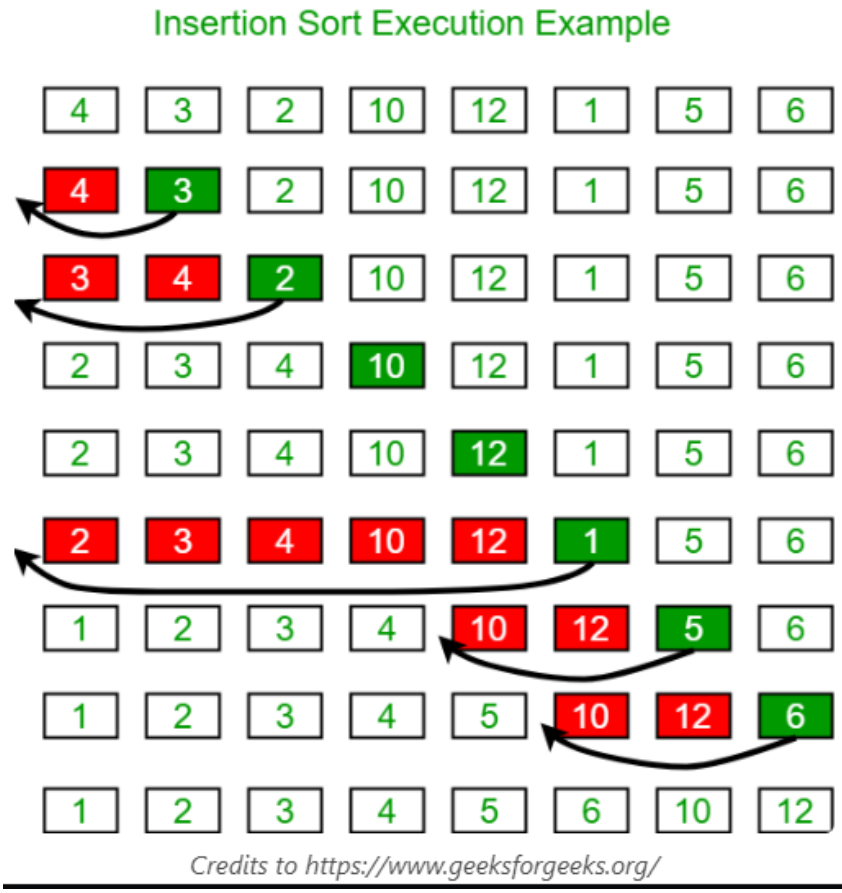
2024

I. DASAR TEORI

Pengurutan adalah proses mengatur data dalam format tertentu. Algoritme pengurutan menentukan cara untuk mengatur data dalam urutan tertentu yang dapat berupa urutan numerik atau leksikografis. Pentingnya pengurutan terletak pada kenyataan bahwa pencarian data dapat dioptimalkan ke tingkat yang sangat tinggi.

- Insertion Sort

  Insertion sort adalah algoritma pengurutan terkenal yang bekerja seperti mengurutkan setumpuk kartu di tangan. Saat Anda melanjutkan melalui elemen-elemen larik, Anda akan memindahkannya kembali sampai berada di tempat yang benar



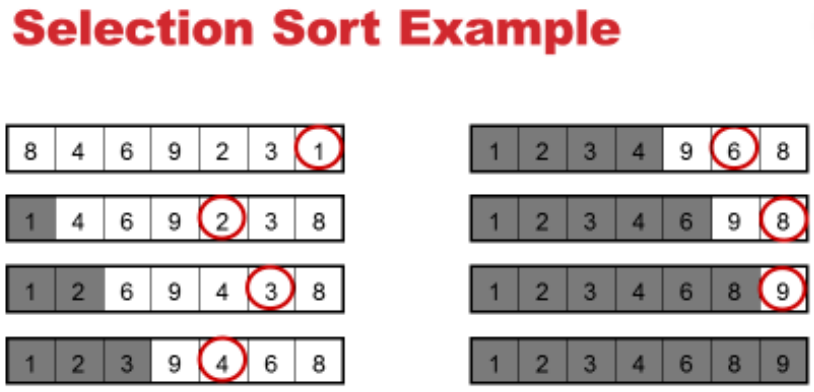
Contoh Program

```
package main


import "fmt"
```

```
func main() {
    var n = []int{1, 39, 2, 9, 7, 54, 11}

    var i = 1
    for i < len(n) {
        var j = i
        for j >= 1 && n[j] < n[j - 1] {
            n[j], n[j - 1] = n[j - 1], n[j]

            j--
        }

        i++
    }

    fmt.Println(n)
}
```

- Selection Sort

  Pengurutan seleksi cukup menarik dan sederhana. Yang perlu Anda lakukan hanyalah mengganti elemen saat ini dalam iterasi dengan nilai terendah di sebelah kanan. Ketika Anda melangkah lebih jauh, bagian kiri larik akan diurutkan, dan Anda tidak perlu memeriksa elemen terakhir.




```
package main
import "fmt"
```

```
func main() {
    var n = []int{1, 39, 2, 9, 7, 54, 11}
    var i = 1
    for i < len(n) - 1 {
        var j = i + 1
        var minIndex = i

        if j < len(n) {
            if n[j] < n[minIndex] {
                minIndex = j
            }
            j++
        }
        if minIndex != i {
            var temp = n[i]
            n[i] = n[minIndex]
            n[minIndex] = temp
        }
        i++
    }
    fmt.Println(n)
}
```

II. GUIDED

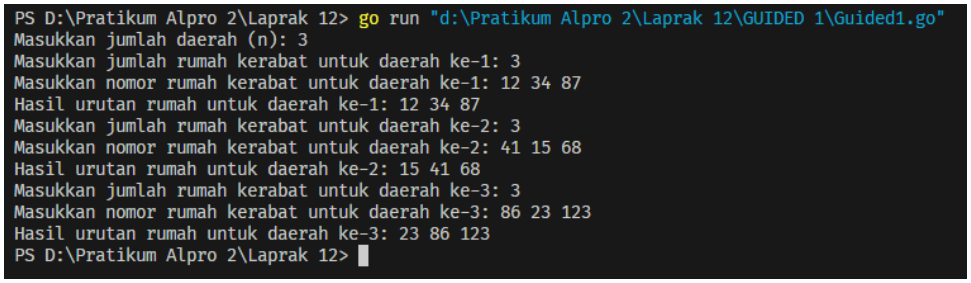
1. Soal Studi Case

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



Deskripsi Program

Program ini digunakan untuk mengurutkan data rumah disetiap daerah dan di daerah tersebut memiliki nomor rumah nomor rumah tersebut nantinya akan program urutkan. Cara kerja program ini adalah program akan meminta pengguna untuk menginputkan jumlah daerah (**n**) dengan sebuah kondisi `if n <= 0 || n >= 1000` jika memenuhi kondisi tersbut program akan berhenti atau tidak valid, setelah pengguna menginputkan (**n**) tidak memenuhi kondisi tersebut program kembali akan meminta pengguna untuk menginputkan jumlah rumah kerabat(**m**) yang juga memiliki kondisi `if m <= 0 || m >= 1000000` Jika kondisi ini tidak terpenuhi, program akan meminta pengguna untuk menginputkan nomor rumah menggunakan array yang panjangnya sesuai dengan (m). Setelah menginputkan data, program akan menjalankan selection sort, di mana elemen terkecil dicari di bagian yang belum diurutkan dan ditukar dengan elemen pertama dari bagian tersebut, berulang kali hingga seluruh array terurut. Setelah proses pengurutan selesai, program akan mencetak nomor rumah dari dari yang terkecil.

2. Soal Studi Case

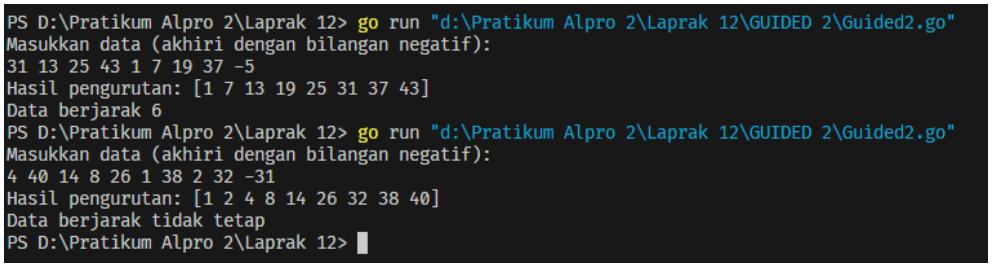
Sourcecode

```
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")
        }
}
```

Screenshoot Output



Deskripsi Program

Program ini dibuat untuk mengurutkan sebuah data dengan menggunakan insertion sort dari inputan pengguna secara berulang ulang hingga memenuhi kondisi berhenti, inputan pengguna kurang dari 0 atau negatif dan program ini juga menghitung jarak antar bilangan bulat apakah sama jika sama program akan menampilkan output data tersebut berjarak n. Cara kerja program ini adalah program akan meminta pengguna untuk mengiputkan data bilangan bulat kedalam sebuah array. Pengguna akan terus memasukkan data hingga memenuhi persyaratan untuk berhenti, yaitu inputan pengguna harus kurang dari 0. Kemudian program akan melakukan sorting insertion sort, insertion sort bekerja dengan mengambil elemen dari bagian yang belum terurut dan menyisipkannya ke posisi

yang sesuai dalam bagian array yang sudah terurut, dengan cara menggeser elemen-elemen yang lebih besar ke kanan. Setelah berhasil diurutkan program akan memeriksa setiap selisih antara elemen-elemen berurutan dalam array yang telah diurutkan, jika semuanya berurutan sama atau true, maka program akan menampilkan return hasil dari selisih antar bilangan, akan tetapi jika salah satu atau banyaknya data tidak memiliki selisih sama atau false, maka program akan menampilkan bahwa data tersebut tidak berjarak sama

III. UNGUIDED

1. Soal Studi Case

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.

No	Masukan	Keluaran
1	3 5 2 1 7 9 13 6 189 15 27 39 75 133 3 4 9 1	1 13 12 8 2 15 27 39 75 133 189 8 4 2

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.

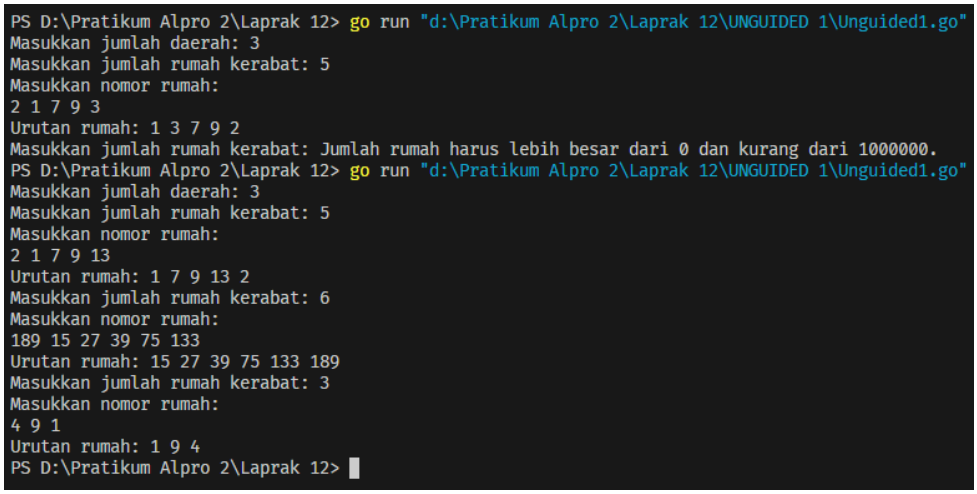
Sourcecode

```
package main

import "fmt"

func UrutNaik(angka []int) {
     for i := 0; i < len(angka)-1; i++ {
           for j := i + 1; j < len(angka); j++ {
                 if angka[j] < angka[i] {
                       angka[i], angka[j] = angka[j],
angka[i]
                 }
```

```
            }
      }
}

func UrutTurun(angka []int) {
      for i := 0; i < len(angka)-1; i++ {
            for j := i + 1; j < len(angka); j++ {
                  if angka[j] > angka[i] {
                        angka[i], angka[j] = angka[j],
angka[i]
                  }
            }
      }
}

func main() {
      var jumlahDaerah int
      fmt.Print("Masukkan jumlah daerah: ")
      fmt.Scan(&jumlahDaerah)

      if jumlahDaerah <= 0 || jumlahDaerah >= 1000 {
            fmt.Println("Jumlah daerah harus lebih besar
dari 0 dan kurang dari 1000.")
            return
      }

      for i := 0; i < jumlahDaerah; i++ {
            var jumlahRumah int
            fmt.Print("Masukkan jumlah rumah kerabat: ")
            fmt.Scan(&jumlahRumah)

            if jumlahRumah <= 0 || jumlahRumah >= 1000000
{
                  fmt.Println("Jumlah rumah harus lebih
besar dari 0 dan kurang dari 1000000.")
                  return
            }

            angkaRumah := make([]int, jumlahRumah)
            rumahGanjil := []int{}
            rumahGenap := []int{}

            fmt.Println("Masukkan nomor rumah:")
            for j := 0; j < jumlahRumah; j++ {
                  fmt.Scan(&angkaRumah[j])
                  if angkaRumah[j]%2 == 0 {
                        rumahGenap = append(rumahGenap,
angkaRumah[j])
                  } else {
                        rumahGanjil = append(rumahGanjil,
angkaRumah[j])
                  }
```

```
            }

            UrutNaik(rumahGanjil)
            UrutTurun(rumahGenap)

            fmt.Print("Urutan rumah: ")
            for _, nomor := range rumahGanjil {
                  fmt.Print(nomor, " ")
            }
            for _, nomor := range rumahGenap {
                  fmt.Print(nomor, " ")
            }
            fmt.Println()
      }
}
```

Screenshoot Output



Deskripsi Program

Program ini dibuat untuk mengurutkan data dari input pengguna, namun, outputnya akan dibagi menjadi dua, ganjil dan genap. Output ganjil akan diurutkan dari yang terkecil ke yang terbesar, dan output genap akan diurutkan dari yang terbesar ke yang terkecil. Program ini memiliki array sementara yang panjangnya sesuai dengan m, dan m memiliki kondisi `if m <= 0 || m >= 1000000`. Kemduian program akan memilah menjadi dua array, yaitu **rumahGenap, rumahGanjil**, dengan tipe data integer. Hal ini digunakan agar memudahkan saat pengembalian nilai Genap dan Ganjil pada setiap nomor rumah. Kemudian program akan mensorting dengan sorting yang berbeda, ganjil akan disorting *Ascending*

dan genap akan disorting *Descending*. Kemudian program akan mengeprint nilai array ganjil terlebih dahulu, kemudian array genap.

2. Soal Studi Case

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.

No	Masukan	Keluaran
1	7 23 11 0 5 19 2 29 3 13 17 0 -5313	11 12

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 kedua, 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.

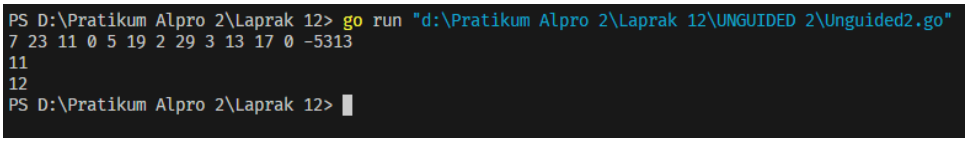
Petunjuk:

Untuk setiap data bukan 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"

//Insertion Sort
func Urutkan(data []int) {
	for i := 0; i < len(data); i++ {
		angkaSekarang := data[i]
		j := i - 1
		for j >= 0 && data[j] > angkaSekarang {
			data[j+1] = data[j]
			j--
		}
		data[j+1] = angkaSekarang
	}
}

func HitungMedian(data []int) float64 {
	Urutkan(data)
	jumlah := len(data)
	if jumlah%2 == 1 {
		return float64(data[jumlah/2])
	}
	return float64(data[jumlah/2-1]+data[jumlah/2]) / 2
}

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

	for { //infinite loop
		fmt.Scan(&angka)

		if angka == -5313 {
			break // kondisi keluar
		}

		if angka == 0 {
			fmt.Println(HitungMedian(data))
		} else {
			data = append(data, angka)
		}
	}
}
```

Screenshoot Output



Deskripsi Program

Program di atas dibuat untuk mencari nilai tengah dari inputan pengguna akan tetapi mencari nilai tengahnya setiap inputan pengguna yang berupa (0) akan menjalankan fungsi yang digunakan menghitung nilai tengah. Program diatas memiliki fungsi sorting `func Urutkan(data []int)` digunakan untuk mengurutkan data, karena untuk mencari sebuah median perlu mengurutkan data tersebut agar dapat menentukan nilai tengah nya. Cara kerja program ini adalah sebagai berikut: jika tidak memenuhi kondisi untuk keluar yaitu **n==0**, program dapat menerima inputan yang tidak terbatas. Pengguna dapat mengisikan bilangan bulat dan program dapat menjalankan `func HitungMedian(median [] int) float64`. Fungsi ini juga melakukan fungsi sorting, yang digunakan untuk mengurutkan data inputan pengguna dari awal hingga 0 (inputan). Program akan mencari data median atau nilai tengah dari data yang sudah disortir.

3. Soal Studi Case

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.

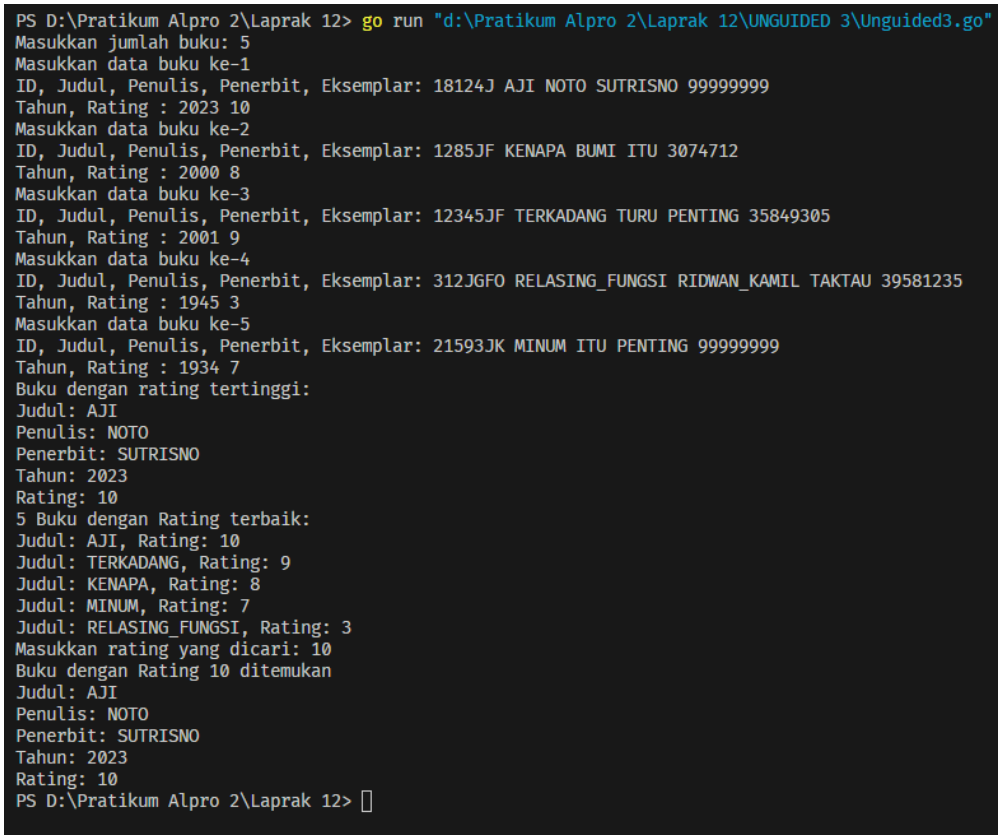
Sourcecode

```
package main

import "fmt"

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

type DaftarBuku []Buku

func DaftarkanBuku(Pustaka *DaftarBuku, n int) {
     for i := 0; i < n; i++ {
           var buku Buku
           fmt.Printf("Masukkan data buku ke-%d\n", i+1)
           fmt.Print("ID, Judul, Penulis, Penerbit,
Eksemplar: ")
           fmt.Scan(&buku.id, &buku.judul, &buku.penulis,
&buku.penerbit, &buku.eksemplar)
           fmt.Print("Tahun, Rating : ")
           fmt.Scan(&buku.tahun, &buku.rating)
           *Pustaka = append(*Pustaka, buku)
     }
}
```

```
func CetakTerfavorit(Pustaka DaftarBuku) {
     if len(Pustaka) == 0 {
           fmt.Println("Tidak ada buku dalam pustaka.")
           return
     }

     terfavorit := Pustaka[0]
     for _, buku := range Pustaka {
           if buku.rating > terfavorit.rating {
                 terfavorit = buku
           }
     }
     fmt.Println("Buku dengan rating tertinggi:")
     fmt.Printf("Judul: %s\n", terfavorit.judul)
     fmt.Printf("Penulis: %s\n", terfavorit.penulis)
     fmt.Printf("Penerbit: %s\n", terfavorit.penerbit)
     fmt.Printf("Tahun: %d\n", terfavorit.tahun)
     fmt.Printf("Rating: %d\n", terfavorit.rating)
}

func UrutBuku(Pustaka *DaftarBuku) {
     n := len(*Pustaka)
     for i := 1; i < n; i++ {
           cur := (*Pustaka)[i]
           j := i - 1
           for j >= 0 && (*Pustaka)[j].rating <
cur.rating {
                 (*Pustaka)[j+1] = (*Pustaka)[j]
                 j--
           }
           (*Pustaka)[j+1] = cur
     }
}

func Cetak5Terbaru(Pustaka DaftarBuku) {
     if len(Pustaka) < 5 {
           fmt.Println("Data pustaka kurang dari 5
buku.")
           return
     }

     fmt.Println("5 Buku dengan Rating terbaik:")
     for i := 0; i < 5; i++ {
           buku := Pustaka[i]
           fmt.Printf("Judul: %s, Rating: %d\n",
buku.judul, buku.rating)
     }
}

func CariBuku(Pustaka DaftarBuku, r int) {
     kiri, kanan := 0, len(Pustaka)-1
```

```
	for kiri <= kanan {
		tengah := (kiri + kanan) / 2
		if Pustaka[tengah].rating == r {
			fmt.Printf("Buku dengan Rating %d
 ditemukan\n", r)
			fmt.Printf("Judul: %s\n",
 Pustaka[tengah].judul)
			fmt.Printf("Penulis: %s\n",
 Pustaka[tengah].penulis)
			fmt.Printf("Penerbit: %s\n",
 Pustaka[tengah].penerbit)
			fmt.Printf("Tahun: %d\n",
 Pustaka[tengah].tahun)
			fmt.Printf("Rating: %d\n",
 Pustaka[tengah].rating)
			return
		} else if Pustaka[tengah].rating < r {
			kanan = tengah - 1
		} else {
			kiri = tengah + 1
		}
	}
	fmt.Printf("Buku dengan rating %d tidak
 ditemukan.\n", r)
 }

 func main() {
	var Pustaka DaftarBuku
	var n, key int

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

	DaftarkanBuku(&Pustaka, n)
	CetakTerfavorit(Pustaka)
	UrutBuku(&Pustaka)
	Cetak5Terbaru(Pustaka)

	fmt.Print("Masukkan rating yang dicari: ")
	fmt.Scan(&key)
	CariBuku(Pustaka, key)
}
```

Screenshoot Output



Deskripsi Program

Program ini dibuat untuk mencatat buku pengguna dan kemudian mengembalikan buku dengan rating terbaik dan rating 5 terbaik. Kemudian, program juga dapat menampilkan buku dengan mencari rating pengguna yang diingkan. Program ini memiliki struct yang berupa `type Buku struct` dan program ini juga memiliki sebuah fungsi yang memiliki kegunaannya masing masing sesuai dengan penamaan fungsinya. Berikut beberapa fungsi yang ada :

- `func DaftarkanBuku(Pustaka *DaftarBuku, n int)`
  Fungsi ini digunakan untuk menambah buku ke dalam daftar pustaka, fungsi ini akan meminta user untuk menginputkan **n** banyaknya buku, dan kemudian program meminta pengguna untuk menginputkan id, judul, penulis, penerbit, eksemplar, tahun, rating
- `func CetakTerfavorit(Pustaka DaftarBuku)`
  Fungsi ini digunakan untuk menampilkan buku dengan rating terbaik

- `func UrutBuku(Pustaka *DaftarBuku)`
  Fungsi ini digunakan untuk merapikan sebuah buku dengan cara mensorting semua buku yang ada dengan insertion sort
- `func Cetak5Terbaru(Pustaka DaftarBuku)`
  Fungsi ini digunakan untuk mencetak 5 rating terbaik dari daftar pustaka
- `func CariBuku(Pustaka DaftarBuku, r int)`
  Fungsi ini dibuat untuk mencari buku berdasarkan rating

IV. DAFTAR PUSTAKA

A. Babakan, "Sorting Algorithms in Go," DEV Community, 21 September 2020. https://dev.to/adnanbabakan/sorting-algorithms-in-go-725. diakses pada 30 Nopember 2024.