

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII & XIII
PENGURUTAN DATA**



Disusun Oleh :

RAKHA YUDHISTIRA / 2311102010

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi, S.Kom., M.Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

1. Definisi Sorting

Sorting adalah proses menyusun elemen data dalam urutan tertentu untuk mempermudah pencarian, pengolahan data, dan optimasi algoritma lainnya. Misalnya:

- Mengurutkan bilangan 3, 1, 4, 2 menjadi 1, 2, 3, 4 (ascending).
- Mengurutkan daftar nama secara alfabet.

2. Jenis Algoritma Sorting

a. Bubble Sort

- Prinsip: Bandingkan dua elemen yang bersebelahan, dan tukar jika tidak dalam urutan yang benar.
- Kelebihan: Implementasi sederhana.
- Kekurangan: Lambat untuk data besar (kompleksitas $O(n^2)$).

b. Selection Sort

- Prinsip: Cari elemen terkecil dari daftar yang belum terurut, dan tukar dengan elemen pertama.
- Kelebihan: Mudah diimplementasikan.
- Kekurangan: Tidak efisien untuk data besar (kompleksitas $O(n^2)$).

c. Insertion Sort

- Prinsip: Elemen dari data belum terurut diambil satu per satu, lalu ditempatkan pada posisi yang sesuai dalam data yang sudah terurut.
- Kelebihan: Baik untuk data kecil.
- Kekurangan: Kompleksitas waktu $O(n^2)$ untuk data besar.

d. Merge Sort

- Prinsip: Membagi data menjadi dua bagian, mengurutkan kedua bagian secara rekursif, lalu menggabungkan hasilnya.
- Kelebihan: Kompleksitas waktu $O(n \log n)$.
- Kekurangan: Membutuhkan memori tambahan.

e. Quick Sort

- Prinsip: Memilih elemen pivot, membagi elemen lebih kecil ke kiri dan lebih besar ke kanan, lalu mengurutkan secara rekursif.
- Kelebihan: Cepat untuk data besar ($O(n \log n)$).

- Kekurangan: Terburuk dalam data tertentu ($O(n^2)$).

II. GUIDED

1. Soal Studi Case

- 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.

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

Sourcecode

```
//2311102010_Rakha Yudhistira_IF-11-06

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)

    // Proses tiap daerah
    for daerah := 1; daerah <= n; daerah++ {
        var m int
        fmt.Printf("\nMasukkan jumlah nomor rumah
kerabat untuk daerah %d: ", daerah)
        fmt.Scan(&m)

        // Membaca nomor rumah untuk daerah ini
        arr := make([]int, m)
        fmt.Printf("Masukkan %d nomor rumah kerabat: ",
m)

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

        // Urutkan array dari terkecil ke terbesar
        selectionSort(arr, m)

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

```

Screenshoot Output

```
PS D:\ITTP\Semester 3\Alpro 2\2311102010_Rakha Yudhistira_Modul_12>
\guided1.go"
3
5 2 1 7 9 13
6 189 15 27 39 75 133
3 4 9 1
1 2 7 9 13
15 27 39 75 133 189
1 4 9
```

Deskripsi Program

Program di atas adalah program yang mengimplementasikan algoritma **Selection Sort** untuk mengurutkan elemen dalam beberapa himpunan data yang dimasukkan oleh pengguna. Program dimulai dengan membaca jumlah himpunan data n . Untuk setiap himpunan, program membaca jumlah elemen m , lalu elemen-elemen tersebut dimasukkan ke dalam array rumah. Selanjutnya, fungsi `selectionSort` digunakan untuk mengurutkan array rumah secara ascending. Setelah semua himpunan data diproses dan diurutkan, hasilnya disimpan dalam slice dua dimensi `results`. Terakhir, program mencetak setiap himpunan data yang telah diurutkan dalam format baris demi baris, dengan elemen dipisahkan oleh spasi. Program ini cocok untuk mengelola dan mengurutkan beberapa himpunan data secara independen.

2. Soal Studi Case

- 1) 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".

Contoh masukan dan keluaran

No	Masukan	Keluaran
1	31 13 25 43 1 7 19 37 -5	1 7 13 19 25 31 37 43 Data berjarak 6
2	4 40 14 8 26 1 38 2 32 -31	1 2 4 8 14 26 32 38 40 Data berjarak tidak tetap

Sourcecode

```
//2311102010_Rakha Yudhistira_IF-11-06
```

```

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)

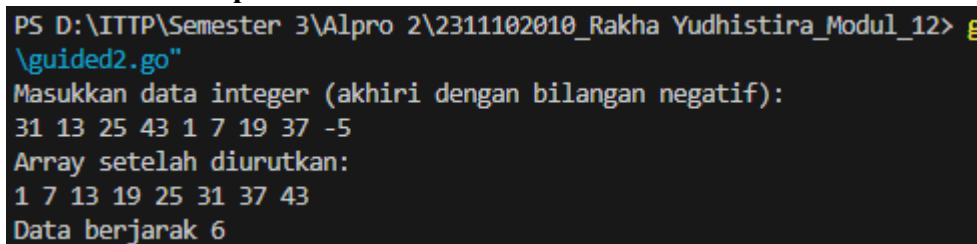
// Periksa apakah selisih elemen tetap
isConstant, difference := isConstantDifference(arr,
n)

// 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



```

PS D:\ITTP\Semester 3\Alpro 2\2311102010_Rakha Yudhistira_Modul_12> \guided2.go
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

```

Deskripsi Program

Program di atas adalah program yang mengimplementasikan algoritma **Insertion Sort** untuk mengurutkan array, kemudian memeriksa apakah selisih antara elemen-elemen yang telah diurutkan tetap konstan. Program dimulai dengan meminta pengguna untuk memasukkan bilangan bulat satu per satu, dengan proses input berakhir saat ditemukan bilangan negatif. Data yang dimasukkan disimpan dalam array `arr`. Setelah itu, array diurutkan menggunakan fungsi `insertionSort`. Kemudian, fungsi `isConstantDifference` digunakan untuk mengecek apakah selisih antara elemen-elemen yang berurutan dalam array memiliki nilai yang tetap (konstan). Program akan

mencetak array setelah diurutkan dan memberikan informasi apakah data memiliki selisih konstan serta nilai selisih tersebut. Jika selisih tidak konstan, program menampilkan pesan bahwa data tidak memiliki jarak tetap.

III. UNGUIDED

1. Soal Studi Case

- 2) 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

Sourcecode

```
//2311102010_Rakha Yudhistira_IF-11-06

package main
import "fmt"

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

func processDaerah(i, m int) ([]int, []int) {
    var arr = make([]int, m)
```



```

        fmt.Printf("Masukkan %d nomor rumah kerabat untuk
daerah ke-%d: ", m, i+1)
        for j := 0; j < m; j++ {
            fmt.Scan(&arr[j])
        }

        var odd, even []int
        for _, num := range arr {
            if num%2 == 0 {
                even = append(even, num)
            } else {
                odd = append(odd, num)
            }
        }
        return odd, even
    }

    func printSortedResults(i int, odd, even []int) {

        selectionSort(odd, true)
        selectionSort(even, false)

        fmt.Printf("\nNomor rumah terurut untuk daerah ke-
%d:\n", i+1)

        for _, num := range odd {
            fmt.Printf("%d ", num)
        }

        for _, num := range even {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }

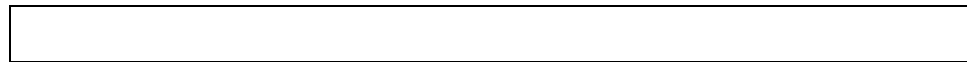
    func main() {
        var n int
        fmt.Print("Masukkan jumlah daerah kerabat (n): ")
        fmt.Scan(&n)

        for i := 0; i < n; i++ {
            var m int
            fmt.Printf("Masukkan jumlah rumah kerabat di
daerah ke-%d (m): ", i+1)
            fmt.Scan(&m)

            odd, even := processDaerah(i, m)

            printSortedResults(i, odd, even)
        }
    }

```



Screenshoot Output

```
Rakha Yudhistira_Modul_12> go run "d:\ITT
P\Semester 3\Alpro 2\2311102010_Rakha YudRakha Yudhistira_Modul_12>
histira_Modul_12\unguided1.go"
Masukkan jumlah daerah kerabat (n): 3
Masukkan jumlah rumah kerabat di daerah k
e-1 (m): 5 2 1 7 9 13
Masukkan 5 nomor rumah kerabat untuk daerah ke-1:
Nomor rumah terurut untuk daerah ke-1:
1 7 9 13 2
Masukkan jumlah rumah kerabat di daerah ke-2 (m): 6 189 15 27 39 75 133
Masukkan 6 nomor rumah kerabat untuk daerah ke-2:
Nomor rumah terurut untuk daerah ke-2:
15 27 39 75 133 189
Masukkan jumlah rumah kerabat di daerah ke-3 (m): 3 4 9 1
Masukkan 3 nomor rumah kerabat untuk daerah ke-3:
Nomor rumah terurut untuk daerah ke-3:
1 9 4
```

Deskripsi Program

Program di atas adalah program yang mengelola data nomor rumah kerabat di beberapa daerah dan mengurutkan nomor rumah berdasarkan bilangan ganjil (secara ascending) dan genap (secara descending) menggunakan algoritma **Selection Sort**. Program meminta input jumlah daerah (n) dan untuk setiap daerah meminta jumlah rumah (m) beserta nomor rumahnya. Fungsi `processDaerah` digunakan untuk memisahkan nomor rumah ke dalam bilangan ganjil dan genap. Nomor rumah ganjil diurutkan menaik (ascending) dan nomor rumah genap diurutkan menurun (descending) menggunakan fungsi `selectionSort`, yang mendukung pengurutan dalam kedua arah berdasarkan parameter boolean `asc`. Hasil pengurutan untuk setiap daerah kemudian ditampilkan secara berurutan, dengan bilangan ganjil tercetak lebih dahulu diikuti oleh bilangan genap. Program ini memungkinkan pengelolaan data terurut untuk masing-masing daerah dengan format hasil yang rapi.

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

Sourcecode

```
//2311102010_Rakha Yudhistira_IF-11-06

package main

import (
    "fmt"
)

func main() {
    var klubA, klubB string
    var skorA, skorB int
    var pemenang []string

    fmt.Print("Masukkan nama Klub A: ")
    fmt.Scanln(&klubA)
    fmt.Print("Masukkan nama Klub B: ")
    fmt.Scanln(&klubB)

    for i := 1; ; i++ {
```

```

        fmt.Printf("Pertandingan %d - Masukkan skor %s:
", i, klubA)
        fmt.Scan(&skorA)
        fmt.Printf("Pertandingan %d - Masukkan skor %s:
", i, klubB)
        fmt.Scan(&skorB)

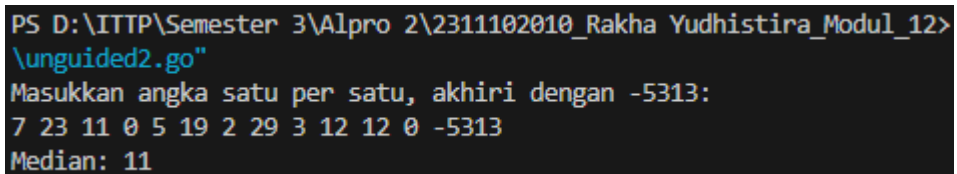
        if skorA < 0 || skorB < 0 {
            fmt.Println("Skor tidak valid. Pertandingan
selesai.")
            break
        }

        if skorA > skorB {
            pemenang = append(pemenang, klubA)
            fmt.Printf("Hasil %d: %s\n", i, klubA)
        } else if skorB > skorA {
            pemenang = append(pemenang, klubB)
            fmt.Printf("Hasil %d: %s\n", i, klubB)
        } else {
            fmt.Printf("Hasil %d: Draw\n", i)
        }
    }

    fmt.Println("\nDaftar klub yang memenangkan
pertandingan:")
    for _, klub := range pemenang {
        fmt.Println(klub)
    }
}

```

Screenshoot Output



```

PS D:\ITTP\Semester 3\Alpro 2\2311102010_Rakha Yudhistira_Modul_12>
\unguided2.go"
Masukkan angka satu per satu, akhiri dengan -5313:
7 23 11 0 5 19 2 29 3 12 12 0 -5313
Median: 11

```

Deskripsi Program

Program di atas adalah program yang membaca sekumpulan angka dari input pengguna, mengabaikan angka 0, dan menghentikan proses input saat angka -5313 dimasukkan. Program mengimplementasikan **Selection Sort** dalam fungsi `selectionSort` untuk mengurutkan angka, yang kemudian digunakan dalam fungsi `calculateMedian` untuk menghitung median dari data yang telah diurutkan. Jika jumlah elemen ganjil, median

diambil sebagai elemen tengah; jika genap, median dihitung sebagai rata-rata dari dua elemen tengah. Setelah input selesai, program memeriksa apakah terdapat angka yang valid untuk diproses, lalu mencetak nilai median dari data tersebut. Program ini berguna untuk menentukan median dari kumpulan angka sembarang dengan pengurutan manual menggunakan Selection Sort.

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

```
//2311102010_Rakha Yudhistira_IF-11-06

package main

import "fmt"

const NMAX int = 7919

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

type DaftarBuku [NMAX]buku

func daftarBuku(pustaka *DaftarBuku, N *int) {
    fmt.Print("Banyak Buku: ")
```

```

    fmt.Scanln(N)

    for i := 0; i < *N; i++ {
        fmt.Println("Data buku ke-", i+1)
        fmt.Scanln(&pustaka[i].id, &pustaka[i].judul,
&pustaka[i].penulis, &pustaka[i].penerbit,
&pustaka[i].eksemplar, &pustaka[i].tahun,
&pustaka[i].rating)
    }
}

func cetakFavorit(pustaka DaftarBuku, n int) {
    var buku_fav buku

    buku_fav = pustaka[0]
    for i := 1; i < n; i++ {
        if buku_fav.rating < pustaka[i].rating {
            buku_fav = pustaka[i]
        }
    }

    fmt.Printf("Buku terfavorit adalah %s oleh %s,
penerbit %s, tahun %d dengan rating %d\n",
buku_fav.judul, buku_fav.penulis, buku_fav.penerbit,
buku_fav.tahun, buku_fav.rating)
}

func urutanBuku(pustaka *DaftarBuku, n int) {
    var temp buku
    var j int

    for i := 1; i < n; i++ {
        j = i
        temp = pustaka[j]
        for j > 0 && temp.rating > pustaka[j-1].rating {
            pustaka[j] = pustaka[j-1]
            j--
        }
        pustaka[j] = temp
    }
}

func cetak5Terbaru(pustaka DaftarBuku, n int) {
    fmt.Println("Top Buku Terbaru:")
    if n < 5 {
        for i := 0; i < n; i++ {
            fmt.Println(i+1, ". ", pustaka[i].judul,
"Rating:", pustaka[i].rating)
        }
    } else {
        for i := 0; i < 5; i++ {

```

```

        fmt.Println(i+1, ". ", pustaka[i].judul,
"Rating:", pustaka[i].rating)
    }
}

func cariBuku(pustaka DaftarBuku, n int, r int) {
    low, high := 0, n-1
    found := false
    var idx int
    for low <= high {
        mid := (low + high) / 2
        if pustaka[mid].rating == r {
            found = true
            idx = mid
            break
        } else if pustaka[mid].rating < r {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }

    if found {
        fmt.Println("Buku ditemukan: ")
        fmt.Println(pustaka[idx].judul,
pustaka[idx].penulis, pustaka[idx].penerbit,
pustaka[idx].tahun, pustaka[idx].eksemplar,
pustaka[idx].rating)
    } else {
        fmt.Println("Tidak ada buku dengan rating
seperti itu")
    }
}

func main() {
    var Pustaka DaftarBuku
    var nPustaka int
    var rating int

    daftarBuku(&Pustaka, &nPustaka)
    cetakFavorit(Pustaka, nPustaka)
    urutanBuku(&Pustaka, nPustaka)
    cetak5Terbaru(Pustaka, nPustaka)
    fmt.Print("Buku dengan rating yang dicari: ")
    fmt.Scan(&rating)
    cariBuku(Pustaka, nPustaka, rating)
}

```

Screenshot Output

```
PS D:\ITTP\Semester 3\Alpro 2\2311102010_Rakha Yudhistira_Modul_12> go run "d:\ITTP\Semester 3\Alpro 2\2311102010_Rakha Yudhistira_Modul_12\tempCodeRunnerFile.go"
Banyak Buku: 2
Data buku ke- 1
b1 kancil agus gramedia 3 2008 5
Data buku ke- 2
b2 buaya wiwid yudhistira 4 2002 9
Buku terfavorit adalah buaya oleh wiwid, penerbit yudhistira, tahun 2002 dengan rating 9
Top Buku Terbaru:
1 . buaya Rating: 9
2 . kancil Rating: 5
Buku dengan rating yang dicari: 9
Buku ditemukan:
buaya wiwid yudhistira 2002 4 9
```

Deskripsi Program

Program di atas adalah program untuk mengelola data koleksi buku dengan berbagai fitur, seperti pencatatan, pencarian, pengurutan, dan penentuan buku favorit. Data buku disimpan dalam array statis DaftarBuku dengan elemen bertipe buku, yang mencakup atribut seperti id, judul, penulis, penerbit, eksemplar, tahun, dan rating. Program diawali dengan fungsi daftarBuku untuk menginput data buku, kemudian fungsi cetakFavorit menentukan buku dengan rating tertinggi sebagai buku terfavorit. Fungsi urutanBuku mengurutkan buku berdasarkan rating secara descending menggunakan **Insertion Sort**, sementara fungsi cetak5Terbaru menampilkan lima buku dengan rating tertinggi. Selanjutnya, fungsi cariBuku menggunakan **Binary Search** untuk mencari buku berdasarkan rating yang dimasukkan oleh pengguna. Semua fitur diintegrasikan dalam fungsi main untuk memproses data pustaka dengan input pengguna, menampilkan hasil, dan memberikan fleksibilitas dalam mengelola koleksi buku.