

LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2
MODUL XII & XIII
SORTING



Oleh :

Geranada Saputra Priambudi

2311102008

IF-11-06

Dosen Pengampu :

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

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY

I. DASAR TEORI

Algoritma Selection Sort

Pengurutan secara seleksi ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian meletakkan pada posisi yang seharusnya. Pada penjelasan berikut ini data akan diurut membesar (ascending), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

- 1) Cari nilai terkecil di dalam rentang data tersisa
- 2) Pindahkan/tukar tempat dengan data yang berada pada posisi paling kiri pada rentang data tersisa tersebut.
- 3) Ulangi proses ini sampai tersisa hanya satu data saja.

Algoritma ini dikenal juga dengan nama Selection Sort, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian indeks nilai ekstrim dan proses pertukaran dua nilai atau swap.

Algoritma Insertion Sort

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.

Selection Sort dan Insertion Sort adalah dua algoritma pengurutan yang populer, namun keduanya memiliki perbedaan dalam cara kerja dan efisiensi. Selection Sort bekerja dengan mencari elemen terkecil di rentang data yang belum terurut dan menukarnya dengan elemen pertama dalam rentang tersebut. Proses ini diulang hingga seluruh data terurut. Meskipun mudah dipahami, algoritma ini memiliki kompleksitas waktu yang konsisten $O(n^2)$, baik untuk kasus terbaik, rata-rata, maupun terburuk. Selection Sort juga tidak stabil, artinya elemen dengan nilai yang sama bisa berubah urutan relatifnya selama proses pengurutan.

Di sisi lain, Insertion Sort bekerja dengan cara menyisipkan elemen yang belum terurut ke dalam posisi yang sesuai dalam bagian data yang sudah terurut, dengan cara menggeser elemen yang lebih besar ke kanan untuk memberi ruang. Insertion Sort dapat

lebih efisien pada data yang hampir terurut, karena dalam kasus terbaiknya memiliki kompleksitas waktu $O(n)$. Namun, pada data yang tidak terurut dengan baik, kompleksitasnya tetap $O(n^2)$. Salah satu keunggulan Insertion Sort adalah stabilitasnya, yang berarti elemen dengan nilai yang sama tetap mempertahankan urutan relatifnya. Meskipun Insertion Sort lebih efisien pada data kecil atau hampir terurut, Selection Sort cenderung lebih mudah diterapkan karena logika pencarian nilai terkecil yang lebih sederhana.

Secara keseluruhan, jika data yang akan diurutkan sangat besar dan tidak terurut, kedua algoritma ini bukan pilihan terbaik karena kompleksitas waktu yang tinggi. Namun, untuk dataset kecil atau hampir terurut, Insertion Sort bisa lebih efisien dibandingkan Selection Sort.

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.

Keterangan: Terdapat 3 daerah dalam contoh input, dan di masing-masing daerah mempunyai 5, 6, dan 3 kerabat.

Source Code:

```
package main

import "fmt"

func selectionSort(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        minIdx := i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[minIdx] {
                minIdx = j
            }
        }
        arr[i], arr[minIdx] = arr[minIdx], arr[i]
    }
}

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

    var results [][]int

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

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

        selectionSort(rumah)

        results = append(results, rumah)
    }
}
```

```

    }

    for _, rumah := range results {
        for j := 0; j < len(rumah); j++ {
            if j > 0 {
                fmt.Print(" ")
            }
            fmt.Print(rumah[j])
        }
        fmt.Println()
    }
}

```

Output:

```

PS C:\Users\ACER\Documents\GeranadaSa
ed1.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

```

Penjelasan Program

Program diatas untuk mengurutkan nomor rumah kerabat Hercules di setiap daerah menggunakan algoritma Selection Sort. Input dimulai dengan sebuah integer n, yang menunjukkan jumlah daerah yang akan diproses. Setiap daerah memiliki sebuah integer m, yang menunjukkan banyaknya rumah kerabat yang perlu diurutkan, diikuti dengan deretan nomor rumah. Program ini kemudian menggunakan Selection Sort untuk mengurutkan nomor-nomor rumah dalam urutan ascending untuk setiap daerah, lalu mencetak hasilnya pada setiap baris sesuai dengan urutan yang benar. Selection Sort bekerja dengan cara mencari elemen terkecil di antara elemen yang belum terurut, kemudian menukarnya dengan elemen yang pertama dari bagian yang belum terurut, dan melanjutkan proses tersebut hingga semua elemen terurut.

Output dari program ini adalah hasil urutan nomor rumah kerabat Hercules yang telah diurutkan secara ascending untuk setiap daerah. Pada contoh input, terdapat tiga daerah dengan jumlah nomor rumah yang berbeda. Untuk daerah pertama, nomor rumah yang diberikan adalah 5 2 1 7 9 13, dan setelah diurutkan menggunakan algoritma Selection Sort, hasilnya menjadi 1 2 7 9 13. Daerah kedua memiliki 6 nomor rumah, yaitu 189 15 27 39 75 133, yang setelah diurutkan

menjadi 15 27 39 75 133 189. Sedangkan pada daerah ketiga, nomor rumah yang diberikan adalah 4 9 1, yang setelah diurutkan menjadi 1 4 9.

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.

Source Code:

```
package main

import "fmt"

const (
    NMAX int = 1000
    MMAX int = 1000
)

type tabDaerah [NMAX]tabRumah
type arrRumah [MMAX]int
type tabRumah struct {
    rumah arrRumah
    nRumah int
}

func inputRumah(daerah *tabDaerah, n int) {
    for i := 0; i < n; i++ {
        fmt.Scan(&daerah[i].nRumah)
        if daerah[i].nRumah <= MMAX {
            for j := 0; j < daerah[i].nRumah; j++ {
                fmt.Scan(&daerah[i].rumah[j])
            }
        } else {
            fmt.Println("Jumlah Rumah tidak boleh melebihi ",
MMAX)
        }
    }
}

func cetakArray(daerah tabDaerah, n int) {
    for i := 0; i < n; i++ {
        for j := 0; j < daerah[i].nRumah; j++ {
            if daerah[i].rumah[j]%2 != 0 {
                fmt.Print(daerah[i].rumah[j], " ")
            }
        }
    }
}
```

```

    }
}
for j := 0; j < daerah[i].nRumah; j++ {
    if daerah[i].rumah[j]%2 == 0 {
        fmt.Print(daerah[i].rumah[j], " ")
    }
}
fmt.Printf("\n")
}
}

func selectionSort(daerah *tabDaerah, n int) {
    var temp, idx_min int

    for k := 0; k < n; k++ {
        for i := 0; i < daerah[k].nRumah; i++ {
            idx_min = i
            for j := i + 1; j < daerah[k].nRumah; j++ {
                if daerah[k].rumah[idx_min] > daerah[k].rumah[j]
{
                    idx_min = j
                }
            }
            temp = daerah[k].rumah[idx_min]
            daerah[k].rumah[idx_min] = daerah[k].rumah[i]
            daerah[k].rumah[i] = temp
        }
    }
}

func main() {
    var (
        nDaerah int
        daerah tabDaerah
    )

    fmt.Scanln(&nDaerah)
    if nDaerah <= NMAX {
        inputRumah(&daerah, nDaerah)
        selectionSort(&daerah, nDaerah)
        cetakArray(daerah, nDaerah)
    } else {
        fmt.Println("Jumlah daerah tidak dapat melebihi", NMAX)
    }
}
}

```

Output:

```
PS C:\Users\ACER\Documents\Geran
ed2.go"
3
5 2 1 7 9 13
6 189 15 27 39 75 133
3 4 9 1
1 7 9 13 2
15 27 39 75 133 189
1 9 4
```

Penjelasan Program

Program diatas mengurutkan nomor rumah kerabat di berbagai daerah, dengan ketentuan bahwa nomor rumah ganjil ditampilkan terlebih dahulu dalam urutan menaik, diikuti dengan nomor rumah genap yang diurutkan secara menurun. Pada program ini, input pertama kali meminta jumlah daerah yang akan diproses. Setiap daerah memiliki daftar nomor rumah yang kemudian diproses. Fungsi inputRumah digunakan untuk membaca daftar nomor rumah untuk setiap daerah, dengan pengecekan bahwa jumlah rumah tidak melebihi batas yang ditentukan (MMAX). Setelah itu, fungsi selectionSort digunakan untuk mengurutkan nomor rumah pada setiap daerah, namun urutan yang diterapkan adalah untuk memisahkan nomor ganjil dan genap, dimana ganjil diurutkan naik dan genap diurutkan turun. Fungsi cetakArray kemudian digunakan untuk menampilkan hasilnya dengan format yang diinginkan.

Output yang dihasilkan menunjukkan nomor rumah yang telah diurutkan sesuai dengan aturan tersebut. Misalnya, untuk input pertama "5 2 1 7 9 13", program menampilkan nomor rumah ganjil (1, 5, 7, 9, 13) dalam urutan menaik dan nomor rumah genap (2) dalam urutan menurun. Hasil output untuk beberapa daerah seperti yang terlihat di console menunjukkan urutan yang sesuai dengan ketentuan, yaitu nomor ganjil terurut menaik dan nomor genap terurut menurun per daerahnya.

III. UNGUIDED

1. 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.

Source Code

```
package main

import "fmt"

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

```

    }
}

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

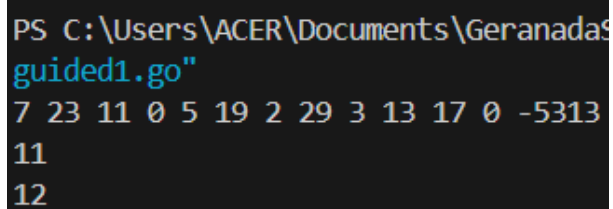
func main() {
    var bilangan int
    var data []int

    for {
        fmt.Scan(&bilangan)

        if bilangan == -5313 {
            break
        } else if bilangan == 0 {
            insertionSort(data)
            fmt.Println(median(data))
        } else {
            data = append(data, bilangan)
        }
    }
}

```

Output



```

PS C:\Users\ACER\Documents\Geranada\guided1.go"
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12

```

Penjelasan Program

Program diatas untuk menghitung nilai median dari kumpulan data yang diberikan. Setiap kali angka 0 ditemukan, program mengurutkan data yang sudah terbaca dan menghitung median. Median dihitung berdasarkan apakah jumlah data ganjil atau genap. Jika jumlahnya ganjil, median adalah elemen tengah, sedangkan jika genap, median adalah rata-rata dua elemen tengah, dibulatkan ke bawah. Program

menggunakan algoritma Insertion Sort untuk mengurutkan data dan fungsi median untuk menghitung nilai tengah. Program berhenti ketika menemukan input -5313.

Sebagai contoh, jika inputnya adalah 7 23 11 0 5 19 2 29 3 13 17 0 -5313, output yang dihasilkan adalah 11 dan 12. Pada langkah pertama, data yang sudah terbaca adalah 7, 23, 11, yang setelah diurutkan menjadi 7, 11, 23. Median dari data ini adalah 11 karena ada 3 elemen. Pada langkah kedua, setelah angka 0 kedua, data yang terbaca adalah 7, 23, 11, 5, 19, 2, 29, 3, 13, 17, yang setelah diurutkan menjadi 2, 3, 5, 7, 11, 13, 17, 19, 23, 29. Karena jumlah datanya genap, median adalah rata-rata dari dua elemen tengah, yaitu $(11 + 13)/2 = 12$.

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".

Contoh masukan dan keluaran

No	Masukan	Keluaran
1	31 13 25 43 17 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

Source Code

```
package main

import "fmt"

func insertionSort(data []int) {
    for i := 1; i < len(data); i++ {
        key := data[i]
        j := i - 1
        for j >= 0 && data[j] > key {
            data[j+1] = data[j]
            j--
        }
        data[j+1] = key
    }
}

func cekJarak(data []int) string {
    if len(data) < 2 {
        return "Data berjarak tidak tetap"
    }
}
```

```

    }
    distance := data[1] - data[0]
    for i := 2; i < len(data); i++ {
        if data[i]-data[i-1] != distance {
            return "Data berjarak tidak tetap"
        }
    }
    return fmt.Sprintf("Data berjarak %d", distance)
}

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

    for {
        fmt.Scan(&input)
        if input < 0 {
            break
        }
        data = append(data, input)
    }

    insertionSort(data)

    for i, num := range data {
        if i > 0 {
            fmt.Print(" ")
        }
        fmt.Print(num)
    }
    fmt.Println()

    fmt.Println(cekJarak(data))
}

```

Output

```

PS C:\Users\ACER\Documents\GeranadaSaputraPr
guided2.go"
31 13 25 43 1 7 19 37 -5
1 7 13 19 25 31 37 43
Data berjarak 6
PS C:\Users\ACER\Documents\GeranadaSaputraPr
guided2.go"
4 40 14 8 26 1 38 2 32 -31
1 2 4 8 14 26 32 38 40
Data berjarak tidak tetap

```

Penjelasan Program

Program diatas untuk membaca sekumpulan bilangan bulat, mengurutkannya menggunakan insertion sort, dan kemudian memeriksa apakah jarak antara setiap bilangan yang terurut memiliki jarak yang sama. Program ini bekerja dengan cara pertama-tama membaca bilangan dari input, dan hanya bilangan non-negatif yang disimpan dalam array. Setelah itu, program mengurutkan array tersebut menggunakan metode insertion sort, yang merupakan algoritma pengurutan dengan cara menyisipkan elemen pada posisi yang sesuai dalam array yang sudah terurut. Setelah array terurut, program kemudian memeriksa apakah jarak antara elemen-elemen yang terurut konsisten. Untuk melakukan ini, program menghitung selisih antara elemen yang berurutan dan memeriksa apakah semua selisih tersebut sama. Jika ya, maka jarak antar data dianggap tetap dan program akan menampilkan "Data berjarak x", di mana x adalah jarak antar elemen. Jika ada jarak yang tidak konsisten, maka program akan menampilkan "Data berjarak tidak tetap". Pada contoh pertama, array yang dimasukkan adalah 31 13 25 43 1 7 19 37, yang setelah diurutkan menjadi 1 7 13 19 25 31 37 43. Jarak antar elemen yang terurut adalah 6, sehingga outputnya adalah "Data berjarak 6". Pada contoh kedua, setelah mengurutkan array 4 40 14 8 26 1 38 2 32, jarak antar elemen yang terurut tidak tetap, sehingga outputnya adalah "Data berjarak tidak tetap".

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 CetakTerfavorit(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 berisi 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.}

```

Source Code

```

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 DaftarkanBuku(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 CetakTerfavorit(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 UrutBuku(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

    DaftarkanBuku(&Pustaka, &nPustaka)
    CetakTerfavorit(Pustaka, nPustaka)
    UrutBuku(&Pustaka, nPustaka)
    Cetak5Terbaru(Pustaka, nPustaka)
    fmt.Print("Buku dengan rating yang dicari: ")
    fmt.Scan(&rating)
    CariBuku(Pustaka, nPustaka, rating)
}

```

Output

```

PS C:\Users\ACER\Documents\GeranadaSaputraPriambudi_2311102008_06\Modul12> go run "c:\Users\ACER\Documents\G
guided3.go"
Banyak Buku: 5
Data buku ke- 1
a1 sangPemimpi andreaHirata bentangPustaka 3 2006 5
Data buku ke- 2
a2 Pulang tereliye Gramedia 2 2016 4
Data buku ke- 3
a3 bumiManusia pramoedyaAnanta hastaMitra 1 1980 4
Data buku ke- 4
a4 aromaKarsa dewiLestari bentangPustaka 2 2018 3
Data buku ke- 5
a5 sepatuDahlan aaGym Mizan 1 2010 4
Buku terfavorit adalah sangPemimpi oleh andreaHirata, penerbit bentangPustaka, tahun 2006 dengan rating 5
Top Buku Terbaru:
1 . sangPemimpi Rating: 5
2 . Pulang Rating: 4
3 . bumiManusia Rating: 4
4 . sepatuDahlan Rating: 4
5 . aromaKarsa Rating: 3
Buku dengan rating yang dicari: 5
Buku ditemukan:
sangPemimpi andreaHirata bentangPustaka 2006 3 5

```

Penjelasan Program

Program di atas adalah implementasi untuk mengelola data buku dalam sebuah perpustakaan menggunakan Go. Program ini menyimpan data buku, seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating ke dalam array yang didefinisikan sebagai DaftarBuku. Program ini terdiri dari beberapa fungsi untuk menangani operasi yang berbeda: DaftarkanBuku untuk mendaftarkan data buku, CetakTerfavorit untuk menampilkan buku dengan rating tertinggi, UrutBuku untuk mengurutkan buku berdasarkan rating secara menurun menggunakan metode Insertion Sort, Cetak5Terbaru untuk menampilkan lima buku dengan rating tertinggi, dan CariBuku untuk mencari buku berdasarkan rating menggunakan pencarian biner (binary search). Proses input dimulai dengan jumlah buku yang ingin dimasukkan, diikuti dengan data untuk setiap buku, dan diakhiri dengan rating buku yang akan dicari. Hasil dari program ini adalah menampilkan buku dengan rating tertinggi, lima buku dengan rating tertinggi, dan buku yang memiliki rating yang dicari.

Output program ini dimulai dengan meminta pengguna untuk memasukkan jumlah buku yang ingin didaftarkan, dalam hal ini sebanyak 5 buku. Kemudian, program menampilkan detail dari setiap buku yang dimasukkan, seperti judul, penulis, penerbit, tahun terbit, jumlah eksemplar, dan rating. Setelah itu, program mencari buku dengan rating tertinggi dan menampilkan buku terfavorit, yaitu "sangPemimpi" dengan rating 5. Selanjutnya, program mengurutkan buku-buku berdasarkan rating dan menampilkan lima buku dengan rating tertinggi, yang dalam kasus ini terdiri dari "sangPemimpi", "Pulang", "bumiManusia", "sepatuDahlan", dan "aromaKarsa". Program kemudian meminta input rating yang dicari, dan ketika pengguna mencari rating 5, program berhasil menemukan buku "sangPemimpi" dan menampilkan detail lengkapnya.