LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 12 & 13

SORTING

Disusun Oleh :

Natasya Intan Sukma Jiwanti / 2311102279

S1-IF-11-05

Dosen Pengampu :

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

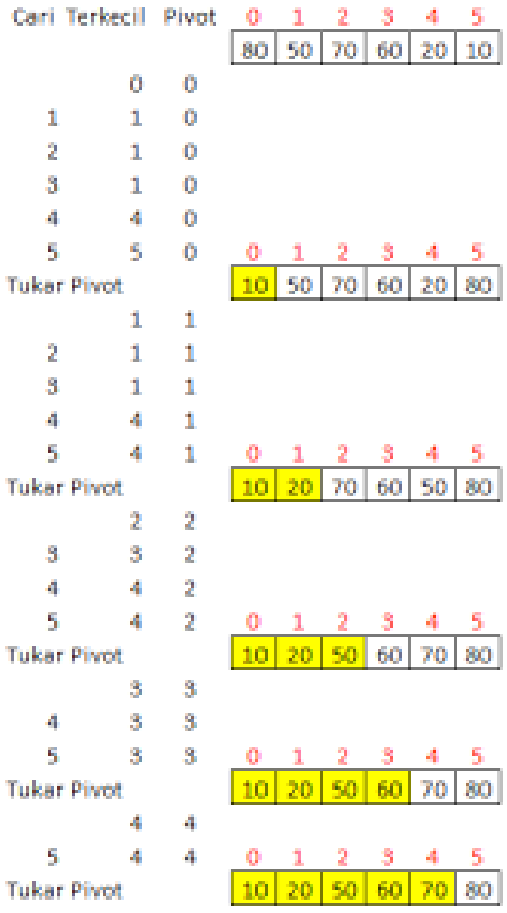
TELKOM UNIVERSITY PURWOKERTO

2024

DASAR TEORI

Selection Sort adalah algoritma pengurutan sederhana yang bekerja dengan cara membagi array menjadi dua bagian: bagian terurut dan tidak terurut. Kemudian, pada setiap iterasi, algoritma harus mencari dan menemukan elemen terkecil atau terbesar dari bagian yang tidak terurut Elemen terkecil atau terbesar tersebut kemudian ditukar dengan elemen pertama di bagian tidak terurut, Berikut cara kerja dari selection sort:

1. Membagi array menjadi dua bagian:
    a. Bagian terurut (di sebelah kiri)
    b. Bagian tidak terurut (di sebelah kanan)
2. Pada setiap iterasi, mencari elemen bernilai ekstrim (maximal atau minimal) dari bagian tidak terurut
3. Menukar elemen terkecil dengan elemen pertama di bagian tidak terurut



Sedangkan cara kerja Insertion Sort mirip dengan cara kita mengurutkan kartu remi di tangan yakni dengan membandingkannya secara bertahap. Pada Inserton sort program akan mengambil satu elemen yang akan menjadi key atau kunci lalu akan dilakukan perbandingan key dengan index atau elemen setelahnya. Setelah itu barulah key akan diletakkan dengan cara menyisipkannya ke posisi yang tepat.

GUIDED

1. Guided 1

Study Case

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.

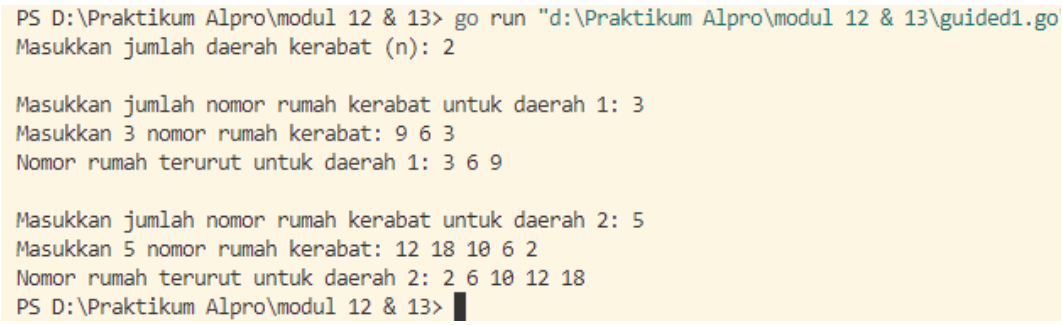
Keluaran terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar di masing-masing daerah.

Source Code

```go
package main
import "fmt"

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

        // Sort the house numbers
        selectionsort(arr, m)

        // Print sorted house numbers for the current daerah
        fmt.Printf("Nomor rumah terurut untuk daerah %d: ",
daerah)
        for _, num := range arr {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}
```

Screenshots Program



Deksripsi Program

Program diatas memiliki fungsi untuk mengurutkan atau sorting menggunakan selection sort yang cara kerjanya akan mencar nilai terkecil terlebih dahulu untuk ditaruh berurutan. Pada program diatas user akan dimintai 3 kali inputan. Inputan yang pertama adalah inputan untuk menunjukan jumlah persebaran daerah saudara herculas. Inputan yang

kedua akan berisi tentang jumlah rumah saudara Hercules di tiap daerah. Lalu inputan yang terakhir merupakan kumpulan nomor rumah saudara Hercules. Inputan ketiga inilah yang akan kita simpan di dalam slice untuk diurutkan menggunakan selection sort.

2. Guided 2

   Study Case

   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".

Source Code

```go
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
    }
}
```

```
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

    fmt.Println("masukan data interger (akhiri dengan 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)

    fmr.Print("array setelah diurutkan:")
    for_, val := range arr{
        fmt.Printf("%d", val)
    }
    fmt.Println()
}
```

Screenshots Program

```
PS D:\Praktikum Alpro\modul 12 & 13> go run "d:\Praktikum Alpro\modul 12 & 13\guided1.go"
Masukkan data integer (Akhiri dengan bilangan negatif) : 8 12 4 16 24 20 -1
Array setelah diurutkan : 4 8 12 16 20 24
Data berjarak  4
PS D:\Praktikum Alpro\modul 12 & 13> go run "d:\Praktikum Alpro\modul 12 & 13\guided1.go"
Masukkan data integer (Akhiri dengan bilangan negatif) : 23 17 38 15 16 98 -11
Array setelah diurutkan : 15 16 17 23 38 98
Data berjarak tidak tetap
PS D:\Praktikum Alpro\modul 12 & 13>
```

Deksripsi Program

Program diatas merupakan program yang digunakan untuk mengurutkan dan mencari tahu apakah perbedaan antara satu elemen dengan elemen lain yang berurutan pada array memiliki perbedaan yang bernilai tetap atau tidak. Untuk itu, kita akan menggunakan fungsi insertion sort yang memiliki cara kerja dengan menjadikan salah satu elemen menjadi kunci yang akan dibandingkan terlebih dahulu dengan elemen – elemen setelahnya atau sebelumnya untuk mencari posisi yang tepat bagi key. Setelah mendapatkan posisi yang tepat maka key akan disisipkan ke posisi tersebut. Begitu terus hingga semua elemen array sudah terurut. Maka kita akan mencari pembeda dari setiap elemen array yang berurutan lalu membanndingkan apakah nilainya sama atau tidak.

UNGUIDED

1. Unguided 1

Study 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.

Sourcecode

```
package main

import "fmt"

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

func urutturun(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        temp := i
            for j := i + 1; j < n; j++ {
            if arr[j] > arr[temp] {
                temp = j
            }
        }
        arr[i], arr[temp] = arr[temp], 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
        var nomorganjil, nomorgenap []int

        fmt.Printf("Masukkan 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])
        }

        for _, nomorrumah := range arr {
            if nomorrumah%2 != 0 {
                nomorganjil = append(nomorganjil,
nomorrumah)
            } else {
                nomorgenap = append(nomorgenap, nomorrumah)
            }
        }

        urutnaik(nomorganjil, len(nomorganjil))
        urutturun(nomorgenap, len(nomorgenap))
        ruterumah := append(nomorganjil, nomorgenap...)

        fmt.Printf("Nomor rumah terurut untuk daerah %d: ",
daerah)
        for _, num := range ruterumah {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}
```

Screenshoot Output

```
PS D:\Praktikum Alpro\modul 12 & 13> go run "d:\Praktikum Alpro\modul 12 & 13\unguided1.go"
Masukkan jumlah daerah kerabat (n): 2
Masukkan jumlah nomor rumah kerabat untuk daerah 1:
5
Masukkan 5 nomor rumah kerabat: 12 22 7 3 9
Nomor rumah terurut untuk daerah 1: 3 7 9 22 12
Masukkan jumlah nomor rumah kerabat untuk daerah 2: 6
Masukkan 6 nomor rumah kerabat: 79 8 17 19 25 12
Nomor rumah terurut untuk daerah 2: 17 19 25 79 12 8
PS D:\Praktikum Alpro\modul 12 & 13>
```

Deskripsi Program

Program diatas menggunakan selection sorting untuk mengurutkan array secara asceding (naik atau dari yang terkecil) dan desceding (turun atau dari yang terbesar). Namun, sebelum diurutkan pada program diatas inputan data nomor rumah di setiap daerahnya dari user akan dipecah terlebih dahulu yang mana nomor rumah ganjil dan yang mana nomor rumah genap. Pemecahan ini berfungsi untuk memetakan mana rumah sebelah kanan jalan dan sebelah kiri jalan sehingga hercules dapat meminimalkan menyebrang jalan. Setelah dipecah maka masing – masing array (baik itu nomor ganjil maupun genap) akan diurutkan menggunakan selection sort yang akan mencari nilai ekstrim nya terlebih dahulu. Untuk nomor ganjil akan dicari nilai minimumnya terlebih dahulu dan mengurutkaan, sedangkan nomor genap akan dicari dari nilai maksimalnya terlebih dahulu dan diurutkan hingga akhir. Setelah kedua array diurutkan, program akan mengumpulkannya di dalam satu array dengan perintah append, yang akan memasukkan array ganjil dulu baru genap. Setelah dikumpulkan maka array urutan rumah akan dicetak.

2. Unguided 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

```
package main
import "fmt"

const MAX int = 1000000

type data []int
func isidata(bilangan *data) {
    var input int
    fmt.Print("Masukkan Data: ")
    for {
        fmt.Scan(&input)
        if (input < 0 || input > 1000000) {
            break
        }
        if input == 0 {
            if len(*bilangan) > 0 {
```

```
                median := hitungmedian(*bilangan)
                fmt.Printf("Median: %d\n", median)
            } else {
                fmt.Println("No data entered.")
            }
        } else {
            *bilangan = append(*bilangan, input)
        }
    }
}

func pengurutandata(bilangan data) {
    n := len(bilangan)
    for i := 0; i < n-1; i++ {
        temp := i
        for j := i + 1; j < n; j++ {
            if bilangan[j] < bilangan[temp] {
                temp = j
            }
        }
        bilangan[i], bilangan[temp] = bilangan[temp],
bilangan[i]
    }
}

func hitungmedian(bilangan data) int {
    pengurutandata(bilangan)
    n := len(bilangan)
    var median int
    if n%2 != 0 {
        median = bilangan[n/2]
    } else {
        median = (bilangan[n/2-1] + bilangan[n/2]) / 2
    }
    return median
}

func main() {
    var bilangan data
    isidata(&bilangan)
}
```

Screenshoot Output

```
PS D:\Praktikum Alpro\modul 12 & 13> go run "d:\Praktikum Alpro\modul 12 & 13\unguided2.go"
Masukkan Data: 7 23 11 0 5 19 2 29 3 13 17 0 -2
Median: 11
Median: 12
PS D:\Praktikum Alpro\modul 12 & 13> go run "d:\Praktikum Alpro\modul 12 & 13\unguided2.go"
Masukkan Data: 2 3 6 5 7 9 17 27 0 -1
Median: 6
PS D:\Praktikum Alpro\modul 12 & 13>
```

Deskripsi Program

Program diatas digunakan untuk mencari median dari data yang berada di sebelah kiri 0. Untuk mencari data tersebut kita akan mengurutkannya menggunakan selection sort terlebih dahulu. Yakni, dengan cara mencari nilai minimumnya terlebih dahulu baru setelah itu diurutkan. Kemudian, kita akan mencari nilai median dengan cara mencari nilai indeks tengah dengan membagi 2 apabila jumlah data nya ganjil dan apabila jumlah datanya genap kita akan mencari rata – rata dengan cara menjumlahkan nilai indeks tengah (n//2) dan nilai indeks tengah +1 baru setelah itu dibagi 2. Untuk mencari median saat nilai 0 diinputkan kita akan menggunakan percabangan yang apabila nilai 0 diinputkan maka program akan memanggil fungsi cari median, sedangkan bila tidak ada nilai 0 yang diinputkan maka program akan memasukkan data tersebut ke array. Untuk berhenti kita akan menggunakan percabangan apabila user menginputkan bilangan negatif atau bilangan yang lebih besar dari nilai maksimum array yakni 1.000.000.

3. Unguided 3

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

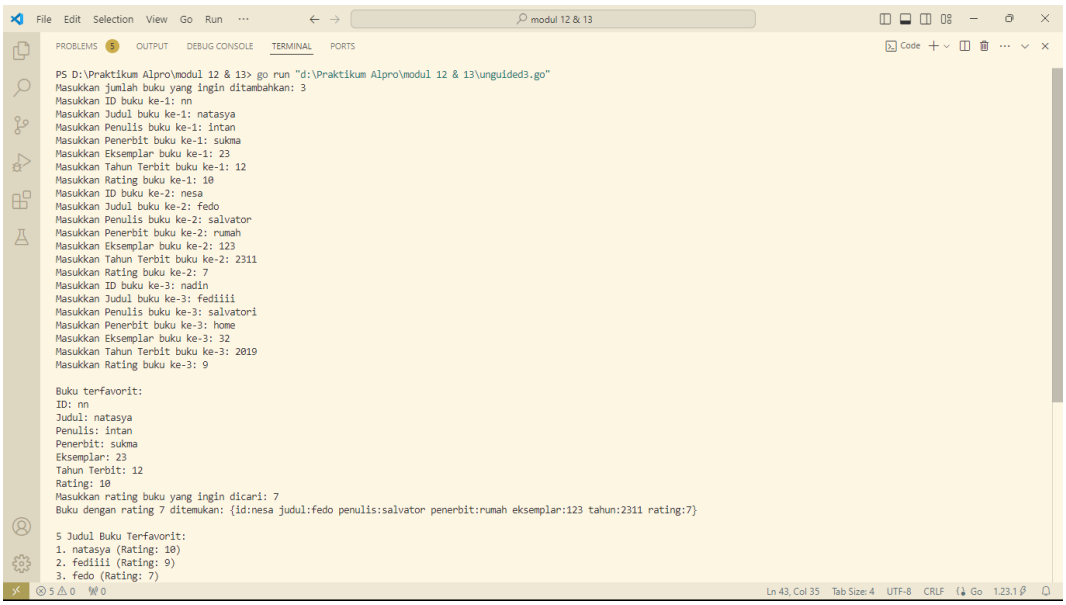
```
package main

import (
    "fmt"
)

const nMax int = 7919

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

type daftarbuku []Buku

func daftarkanbuku(pustaka *daftarbuku, n int) {
    var databuku Buku
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan ID buku ke-%d: ", i+1)
        fmt.Scan(&databuku.id)
        fmt.Printf("Masukkan Judul buku ke-%d: ", i+1)
        fmt.Scan(&databuku.judul)
        fmt.Printf("Masukkan Penulis buku ke-%d: ", i+1)
        fmt.Scan(&databuku.penulis)
        fmt.Printf("Masukkan Penerbit buku ke-%d: ", i+1)
        fmt.Scan(&databuku.penerbit)
        fmt.Printf("Masukkan Eksemplar buku ke-%d: ", i+1)
        fmt.Scan(&databuku.eksemplar)
        fmt.Printf("Masukkan Tahun Terbit buku ke-%d: ",
i+1)
        fmt.Scan(&databuku.tahun)
        fmt.Printf("Masukkan Rating buku ke-%d: ", i+1)
```

```
        fmt.Scan(&databuku.rating)
        *pustaka = append(*pustaka, databuku) // Append to
the slice
    }
}

func cetakFavorit(pustaka daftarbuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada buku dalam daftar.")
        return
    }
    terfav := pustaka[0]
    for _, buku := range pustaka {
        if buku.rating > terfav.rating {
            terfav = buku
        }
    }
    fmt.Printf("\nBuku terfavorit:\nID: %s\nJudul:
%s\nPenulis: %s\nPenerbit: %s\nEksemplar: %d\nTahun Terbit:
%d\nRating: %d\n", terfav.id, terfav.judul, terfav.penulis,
terfav.penerbit, terfav.eksemplar, terfav.tahun,
terfav.rating)
}

func urutbuku(pustaka daftarbuku, n int) {
    for i := 1; i < n; i++ {
        key := pustaka[i]
        j := i - 1
        for j >= 0 && pustaka[j].rating < key.rating {
            pustaka[j+1] = pustaka[j]
            j--
        }
        pustaka[j+1] = key
    }
}

func cetakTop5(pustaka daftarbuku) {
    fmt.Println("\n5 Judul Buku Terfavorit:")
    i := 0
    for _, buku := range pustaka {
        if i < 5 {
            fmt.Printf("%d. %s (Rating: %d)\n", i+1,
buku.judul, buku.rating)
            i++
        } else {
```

```
            break
        }
    }
}

func caribuku(pustaka daftarbuku, n int, r int) {
    left, right := 0, n-1
    found := false

    for left <= right {
        mid := left + (right-left)/2
        if pustaka[mid].rating == r {
            fmt.Printf("Buku dengan rating %d ditemukan:
%+v\n", r, pustaka[mid])
            found = true
            break
        } else if pustaka[mid].rating < r {
            right = mid - 1
        } else {
            left = mid + 1
        }
    }

    if !found {
        fmt.Printf("Tidak ada buku dengan rating %d\n", r)
    }
}

func main() {
    var pustaka daftarbuku
    var n int
    var r int

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

    if n <= nMax {
        daftarkanbuku(&pustaka, n)
        urutbuku(pustaka, n)
        cetakFavorit(pustaka, n)
        fmt.Print("Masukkan rating buku yang ingin dicari:
")
        fmt.Scan(&r)
        caribuku(pustaka, n, r)
```

```
        cetakTop5(pustaka)
    } else {
        fmt.Println("Inputan anda melebihi kapasitas
maksimal")
    }
}
```

Screenshoot Output



Deskripsi Program

Program diatas merupakan program yang digunakan untuk mendaftarkan buku dengan cara mengisi data – data buku dengan berbagai tipe data. Sehingga pada program diatas digunakan struct untuk menyimpan satu data buku dengan berbagai rincian dan array daftarbuku untuk menyimpan struct ke dalam program. Setelah buku – buku di daftarkan maka, akan ada pencaria untuk mencari buku mana yang memiliki rating tertinggi alias menjadi buku terfavorit menggunakan selection searching. Setelah dicari dan ditemukan buku dengan rating tertinggi maka data buku tersebut akan dicetak.

Pada program diatas, data – data buku yang didaftarkan juga akan diurutkan sesuai ratingnya dari rating tertinggi hingga rating terendah menggunakan Insertion Sort. Pada insertion sort kita akan menjadikan indeks ke – 0 pada array pustaka sebagai kunci awal perbandingan untuk mencari posisi yang

sesuai dan menyisipkannya pada posisi tersebut. Begitu terus hingga semua elemennya terurut. Setelah diurutkan program akan menjalankan fungsi untuk mencetak 5 judul buku dengan rating tertinggi. Selain itu, pada program diatas juga menggunakan binary search untuk mencari buku dengan rating tertentu. Pada binary search sendiri, kita akan mencari apakah rating yang diinputkan oleh user sama dengan nilai tengah pada array, jika kurang dari nilai tengah maka kita akan mencarinya dengan jangkauan maksimalnya = nilai tengah. Apabila nilai rating yang dicari melebihi nilai tengah maka kita akan mencari dengan jangkauan minimumnya = nilai tengah.