LAPORAN PRAKTIKUM
ALGORITMA DAN PEMPROGRAMAN 2

MODUL 12 & 13

PENGURUTAN DATA

Oleh:

GALIH TRISNA

2311102050

IF-11-02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Insertion Sort adalah algoritma pengurutan sederhana yang bekerja seperti cara seseorang menyortir kartu dalam permainan. Elemen-elemen dalam daftar diproses satu per satu, dengan memasukkan elemen yang belum terurut ke posisi yang sesuai dalam bagian yang sudah terurut. Proses ini dilakukan dengan membandingkan elemen tersebut dengan elemen sebelumnya dan menukar posisinya jika diperlukan.

Proses:

1. Mulai dari elemen kedua dalam daftar.
2. Bandingkan elemen ini dengan elemen sebelumnya.
3. Geser elemen-elemen yang lebih besar ke kanan untuk memberi ruang bagi elemen yang sedang diproses.
4. Masukkan elemen ke posisi yang sesuai.
5. Ulangi proses hingga semua elemen terurut.

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

Selection Sort adalah algoritma pengurutan yang secara berulang memilih elemen terkecil (atau terbesar, tergantung kebutuhan) dari daftar yang belum terurut dan memindahkannya ke posisi yang benar dalam daftar terurut. Algoritma ini bekerja dengan dua bagian utama: satu bagian terurut dan satu bagian belum terurut.

Proses:

1. Cari elemen terkecil dari bagian yang belum terurut.
2. Tukar elemen tersebut dengan elemen pertama dari bagian yang belum terurut.
3. Ulangi hingga seluruh daftar terurut.

	Notasi Algoritma	Notasi dalam bahasa Go
1	`i ← 1`	`i = 1`
2	`while i <= n-1 do`	`for i <= n-1 {`
3	`    idx_min ← i - 1`	`    idx_min = i - 1`
4	`    j ← i`	`    j = i`
5	`    while j < n do`	`    for j < n {`
6	`        if a[idx_min] > a[j] then`	`        if a[idx_min] > a[j] {`
7	`            idx_min ← j`	`            idx_min = j`
8	`        endif`	`        }`
9	`        j ← j + 1`	`        j = j + 1`
10	`    endwhile`	`    }`
11	`    t ← a[idx_min]`	`    t = a[idx_min]`
12	`    a[idx_min] ← a[i-1]`	`    a[idx_min] = a[i-1]`
13	`    a[i-1] ← t`	`    a[i-1] = t`
14	`    i ← i + 1`	`    i = i + 1`
15	`endwhile`	`}`

II. GUIDED

1. Source Code

```go
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()
    }
}
```

Screenshot

```
PS C:\Users\galih\Documents\belajar\Algoritma Pemprogramman 2> g
\guided\1\main.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5
Masukkan nomor rumah kerabat untuk daerah ke-1:  4 6 2 5 1
Hasil urutan rumah untuk daerah ke-1: 6 5 4 2 1
Masukkan jumlah rumah kerabat untuk daerah ke-2: 4
Masukkan nomor rumah kerabat untuk daerah ke-2: 2 4 8 1
Hasil urutan rumah untuk daerah ke-2: 8 4 2 1
Masukkan jumlah rumah kerabat untuk daerah ke-3: 3
Masukkan nomor rumah kerabat untuk daerah ke-3: 6 2 9
Hasil urutan rumah untuk daerah ke-3: 9 6 2
PS C:\Users\galih\Documents\belajar\Algoritma Pemprogramman 2>
```

Penjelasan

Program diatas merupakan program untuk mengurutkan nomor rumah menggunakan algoritma selection sort. Fungsi selectionSort digunakan untuk mengurutkan elemen-elemen dalam array secara descending (dari terbesar ke terkecil). Program dimulai dengan meminta pengguna memasukkan jumlah daerah n, dan setiap daerah memiliki sejumlah rumah kerabat dengan nomor rumah tertentu. Untuk setiap daerah, program menerima input jumlah rumah kerabat m dan nomor rumah tersebut, yang disimpan dalam array. Setelah itu, nomor rumah diurutkan menggunakan selectionSort, dan hasil pengurutan ditampilkan. Program juga memvalidasi input n dan m agar berada dalam rentang yang sesuai yaitu jumlah daerah tidak boleh kurang dari 1 atau lebih dari 999, sedangkan jumlah rumah kerabat per daerah tidak boleh kurang dari 1 atau lebih dari 999.999.

2. Source code

```go
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

```
PS C:\Users\galih\Documents\belajar\Algoritma Pemprogramman 2>
\guided\2\main.go"
Masukkan data (akhiri dengan bilangan negatif):
13 31 25 43 1 7 19 37 -3
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6
PS C:\Users\galih\Documents\belajar\Algoritma Pemprogramman 2>
\guided\2\main.go"
Masukkan data (akhiri dengan bilangan negatif):
23 12 4 32 12 4 51 1 -12
Hasil pengurutan: [1 4 4 12 12 23 32 51]
Data berjarak tidak tetap
PS C:\Users\galih\Documents\belajar\Algoritma Pemprogramman 2>
```

Penjelasan :

Program diatas merupakan program untuk mengurutkan array menggunakan insertion sort sekaligus memeriksa apakah data dalam array memiliki jarak yang konsisten antara elemen-elemen berurutan. Fungsi insertionSort berfungsi mengurutkan array dalam urutan ascending dengan cara menyisipkan elemen ke posisi yang sesuai, elemen-elemen yang lebih besar dari key akan digeser ke kanan. Setelah data selesai diurutkan, fungsi isDataConsistentlySpaced digunakan untuk mengecek apakah selisih antara setiap elemen berurutan dalam array memiliki nilai yang sama.

Program dimulai dengan menerima inputan berupa data integer yang dimasukkan oleh pengguna sampai pengguna memasukan bilangan negatif sebagai tanda akhir input. Data yang telah diterima kemudian diurutkan menggunakan fungsi insertionSort. Setelah proses pengurutan selesai, data tersebut diperiksa menggunakan fungsi isDataConsistentlySpaced untuk menentukan apakah elemen elemen dalam array memiliki selisih yang sama di seluruh pasangan elemen. Jika data memiliki jarak sama, program mencetak jarak tersebut. jika tidak, program menyatakan bahwa data tidak memiliki jarak tetap.

III. UNGUIDED

1. Source Code

```go
package main

import "fmt"

func selectionSortAsc(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 selectionSortDesc(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] {
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx], arr[i]
    }
}

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
        }

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

        var oddNumbers, evenNumbers []int
        for _, house := range rumah {
            if house%2 == 1 {
                oddNumbers = append(oddNumbers, house)
            } else {
                evenNumbers = append(evenNumbers, house)
            }
        }

        selectionSortAsc(oddNumbers)
        selectionSortDesc(evenNumbers)

        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ", i+1)

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

        for _, house := range evenNumbers {
            fmt.Printf("%d ", house)
        }
        fmt.Println()
    }
}
```

Screenshot

```
PS C:\Users\galih\Documents\belajar\Algoritma Pemprogramman 2>
\unguided\1\main.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 4
Masukkan nomor rumah kerabat untuk daerah ke-1: 2 4 7 1
Hasil urutan rumah untuk daerah ke-1: 1 7 4 2
Masukkan jumlah rumah kerabat untuk daerah ke-2: 5
Masukkan nomor rumah kerabat untuk daerah ke-2: 2 5 9 1 3
Hasil urutan rumah untuk daerah ke-2: 1 3 5 9 2
Masukkan jumlah rumah kerabat untuk daerah ke-3: 3
Masukkan nomor rumah kerabat untuk daerah ke-3: 5 2 7
Hasil urutan rumah untuk daerah ke-3: 5 7 2
PS C:\Users\galih\Documents\belajar\Algoritma Pemprogramman 2>
```

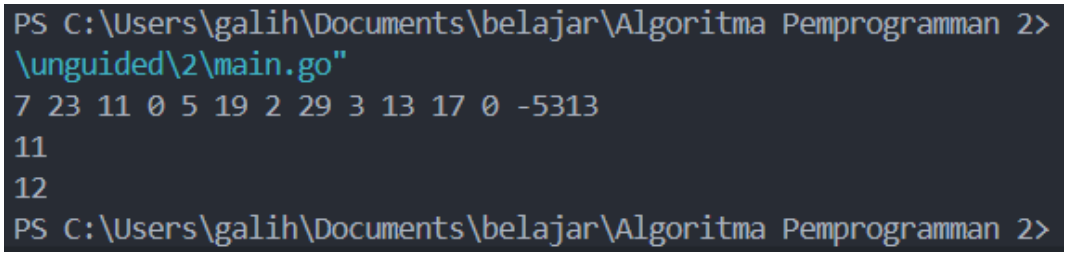
Penjelasan

Program diatas merupakan program untuk mengurutkan nomor rumah kerabat berdasarkan kriteria ganjil dan genap menggunakan selection sort. Fungsi selectionSortAsc digunakan untuk mengurutkan nomor rumah ganjil secara ascending, sementara fungsi selectionSortDesc digunakan untuk mengurutkan nomor rumah genap secara descending. Program dimulai dengan meminta pengguna untuk memasukkan jumlah daerah n yang memiliki rumah kerabat, kemudian untuk setiap daerah, jumlah rumah kerabat m dan nomor rumah yang akan dimasukkan. Nomor rumah kerabat yang ganjil dan genap dipisahkan terlebih dahulu, kemudian diurutkan sesuai dengan kriteria yang telah ditentukan. Setelah proses pengurutan selesai, hasilnya ditampilkan dengan nomor rumah ganjil yang terurut secara ascending dan nomor rumah genap yang terurut secara descending. Program ini juga melakukan validasi input agar jumlah daerah n berada dalam rentang yaitu jumlah daerah tidak boleh kurang dari 1 atau lebih dari 999, sedangkan jumlah rumah kerabat per daerah tidak boleh kurang dari 1 atau lebih dari 999.999.

2. Source Code

```go
package main

import "fmt"

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

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

func calculateMedian(arr []int) int {
    n := len(arr)
    if n%2 == 0 {
        return (arr[(n/2)-1] + arr[(n/2)]) / 2
    } else {
        return arr[(n / 2)]
    }
}

func main() {
    var inputData int
    var numberCollection = make([]int, 0)
    var currentNumbers = make([]int, 0)

    for inputData != -5313 {
        fmt.Scan(&inputData)
        if inputData != -5313 {
            numberCollection = append(numberCollection, inputData)
        }
    }

    for _, number := range numberCollection {
        if number == 0 {
            insertionSort(currentNumbers)
            fmt.Println(calculateMedian(currentNumbers))
        } else {
            currentNumbers = append(currentNumbers, number)
        }
    }
}
```

Screenshot



Penjelasan

Program diatas merupakan program mengurutkan data angka yang dimasukkan oleh pengguna dan menghitung median dari data yang telah diurutkan menggunakan insertion sort. Fungsi insertionSort digunakan untuk mengurutkan array angka secara ascending, di mana elemen-elemen yang lebih besar dari key akan digeser ke kanan hingga elemen tersebut berada pada posisi yang tepat. Fungsi calculateMedian digunakan untuk menghitung median dari array yang sudah diurutkan. Jika jumlah elemen ganjil, median adalah elemen tengah. jika genap, median adalah rata-rata dari dua elemen tengah.

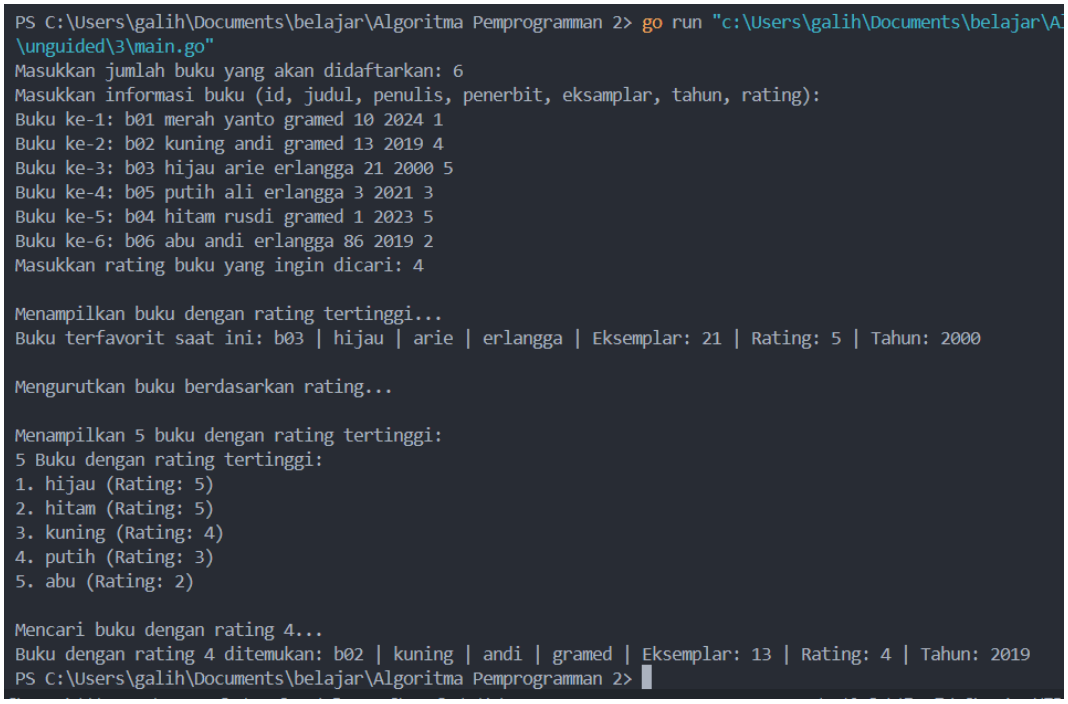
Program dimulai dengan meminta pengguna untuk memasukkan angka satu per satu hingga memasukkan angka -5313 sebagai tanda berakhirnya input. Setiap angka yang dimasukkan akan disimpan di dalam array currentNumbers. Ketika angka 0 dimasukkan, program akan mengurutkan data yang ada menggunakan insertionSort, kemudian menghitung dan mencetak median dari array yang telah terurut. Setelah itu, program akan melanjutkan untuk menerima input angka berikutnya.

3. Source Code

```go
package main

import "fmt"

const nMax = 7919

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

type DaftarBuku [nMax]Buku

func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    fmt.Println("Masukkan informasi buku (id, judul, penulis, penerbit, eksamplar, tahun, rating):")
    for i := 0; i < n; i++ {
        fmt.Printf("Buku ke-%d: ", i+1)
        fmt.Scan(&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 indexFav = 0
    for i := 1; i < n; i++ {
        if pustaka[i].rating > pustaka[indexFav].rating {
            indexFav = i
        }
    }

    fmt.Printf("Buku terfavorit saat ini: %s | %s | %s | %s | Eksemplar: %d | Rating: %d | Tahun:
%d\n",
        pustaka[indexFav].id, pustaka[indexFav].judul, pustaka[indexFav].penulis,
pustaka[indexFav].penerbit,
        pustaka[indexFav].eksemplar, pustaka[indexFav].rating, pustaka[indexFav].tahun)
}

func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        var key Buku = pustaka[i]
        var j int = i - 1

        for j >= 0 && pustaka[j].rating < key.rating {
            pustaka[j+1] = pustaka[j]
            j--
        }
        pustaka[j+1] = key
    }
}

func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    if n > 5 {
        n = 5
    }
    fmt.Printf("5 Buku dengan rating tertinggi:\n")
    for i := 0; i < n; i++ {
        fmt.Printf("%d. %s (Rating: %d)\n", i+1, pustaka[i].judul, pustaka[i].rating)
    }
}

func CariBuku(pustaka DaftarBuku, n, r int) {
    kr := 0
    kn := n - 1
    var med int
    var found bool = false

    for kr <= kn && !found {
        med = (kr + kn) / 2

        if pustaka[med].rating > r {
            kr = med + 1
        } else if pustaka[med].rating < r {
            kn = med - 1
        } else {
            found = true
        }
    }
    if found {
        fmt.Printf("Buku dengan rating %d ditemukan: %s | %s | %s | %s | Eksemplar: %d | Rating: %d |
Tahun: %d\n",
            r, pustaka[med].id, pustaka[med].judul, pustaka[med].penulis, pustaka[med].penerbit,
            pustaka[med].eksemplar, pustaka[med].rating, pustaka[med].tahun)
    } else {
        fmt.Printf("Tidak ada buku dengan rating %d.\n", r)
    }
}
```

```go
func main() {
    var pustaka DaftarBuku
    var nPustaka, chooseRating int
    fmt.Print("Masukkan jumlah buku yang akan didaftarkan: ")
    fmt.Scan(&nPustaka)
    DaftarkanBuku(&pustaka, nPustaka)
    fmt.Print("Masukkan rating buku yang ingin dicari: ")
    fmt.Scan(&chooseRating)
    fmt.Println("\nMenampilkan buku dengan rating tertinggi...")
    CetakTerfavorit(pustaka, nPustaka)
    fmt.Println("\nMengurutkan buku berdasarkan rating...")
    UrutBuku(&pustaka, nPustaka)
    fmt.Println("\nMenampilkan 5 buku dengan rating tertinggi:")
    Cetak5Terbaru(pustaka, nPustaka)
    fmt.Printf("\nMencari buku dengan rating %d...\n", chooseRating)
    CariBuku(pustaka, nPustaka, chooseRating)
}
```

Screenshot



Penjelasan

Program diatas merupakan program untuk mengelola data buku dalam sebuah pustaka dengan berbagai fitur seperti pendaftaran buku, pengurutan berdasarkan rating, pencarian buku berdasarkan rating, dan 5 buku dengan rating tertinggi. Program ini dimulai dengan meminta pengguna untuk memasukkan jumlah buku yang akan didaftarkan. Setiap buku memiliki atribut seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun penerbitan, dan rating.

Fungsi DaftarkanBuku digunakan untuk meminta input data buku dari pengguna dan menyimpannya dalam array pustaka. Setelah data buku dimasukkan, fungsi CetakTerfavorit digunakan untuk menampilkan buku dengan rating tertinggi. Selanjutnya, fungsi UrutBuku mengurutkan buku berdasarkan rating descending menggunakan algoritma insertion sort. Setelah pengurutan, fungsi Cetak5Terbaru

menampilkan 5 buku dengan rating tertinggi, atau sebanyak data buku yang ada jika kurang dari 5. Fungsi CariBuku digunakan untuk mencari buku berdasarkan rating tertentu dengan menggunakan metode binary search.

Pengguna dapat memasukkan rating yang ingin dicari, dan jika ditemukan, informasi buku dengan rating tersebut akan ditampilkan. Jika tidak ditemukan, program akan memberitahukan pengguna bahwa tidak ada buku dengan rating tersebut.