

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII & XIII
PENGURUTAN DATA**



Disusun Oleh:
Bayu Kuncoro Adi / 2311102031
S1 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

A. Algoritma Selection Sort

1. Ide 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) Ulang 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.

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

2. Algoritma Selection Sort

Adapun algoritma selection sort pada untuk mengurutkan array bertipe data bilangan bulat secara membesar atau ascending adalah sebagai berikut ini!

```
..    ...
5   type arrInt [4321]int
..    ...
15  func selectionSort1(T *arrInt, n int){
16  /* I.S. terdefinisi array T yang berisi n bilangan bulat
17     F.S. array T terurut secara ascending atau membesar dengan SELECTION SORT */
18     var t, i, j, idx_min int
19
```

```

20     i = 1
21     for i <= n-1 {
22         idx_min = i - 1
23         j = i
24         for j < n {
25             if T[idx_min] > T[j] {
26                 idx_min = j
27             }
28             j = j + 1
29         }
30         t = T[idx_min]
31         T[idx_min] = T[i-1]
32         T[i-1] = t
33         i = i + 1
34     }
35 }

```

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada saat proses perbandingan nilai ekstrim, kemudian tipe data dari variabel t sama dengan struct dari arraynya.

```

..     ...
5     type mahasiswa struct {
..         nama, nim, kelas, jurusan string
..         ipk float64
..     }
..     type arrMhs [2023]mahasiswa
..     ...
15     func selectionSort2(T * arrMhs, n int){
16         /* I.S. terdefinisi array T yang berisi n data mahasiswa
17            F.S. array T terurut secara ascending atau membesar berdasarkan ipk dengan
18            menggunakan algoritma SELECTION SORT */
19         var i, j, idx_min int
20         var t mahasiswa
21         i = 1
22         for i <= n-1 {
23             idx_min = i - 1
24             j = i
25             for j < n {
26                 if T[idx_min].ipk > T[j].ipk {
27                     idx_min = j
28                 }
29                 j = j + 1
30             }
31             t = T[idx_min]
32             T[idx_min] = T[i-1]
33             T[i-1] = t
34             i = i + 1
35         }
36     }

```

B. Algoritma Insertion Sort

1. Ide 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".

- 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 urutan keterurutan. informatika lab
- 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.

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

2. Algoritma Insertion Sort

Adapun algoritma insertion sort pada untuk mengurutkan array bertipe data bilangan bulat secara mengecil atau descending adalah sebagai berikut ini!

```
..    ...
5   type arrInt [4321]int
..    ...
15  func insertionSort1(T *arrInt, n int){
16  /* I.S. terdefinisi array T yang berisi n bilangan bulat
17     F.S. array T terurut secara mengecil atau descending dengan INSERTION SORT*/
18     var temp, i, j int
19     i = 1
20     for i <= n-1 {
21         j = i
22         temp = T[j]
23         for j > 0 && temp > T[j-1] {
24             T[j] = T[j-1]
25             j = j - 1
26         }
27         T[j] = temp
28         i = i + 1
29     }
30 }
```

```

..  ...
5  type mahasiswa struct {
..      nama, nim, kelas, jurusan string
..      ipk float64
..  }
..  type arrMhs [2023]mahasiswa
..  ...
15 func insertionSort2(T * arrMhs, n int){
16     /* I.S. terdefinisi array T yang berisi n data mahasiswa
17        F.S. array T terurut secara mengecil atau descending berdasarkan nama dengan
18        menggunakan algoritma INSERTION SORT */
19     var temp i, j int
20     var temp mahasiswa
21     i = 1
22     for i <= n-1 {
23         j = i
24         temp = T[j]
25         for j > 0 && temp.nama > T[j-1].nama {
26             T[j] = T[j-1]
27             j = j - 1
28         }
29         T[j] = temp
30         i = i + 1
31     }
32 }

```

A. 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.

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

```
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)

    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])
        }
        selectionSort(arr, m)

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

```

Screenshoot Program

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12\guided1.go"
Masukkan jumlah daerah kerabat (n): 3

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 1
Masukkan 1 nomor rumah kerabat : 1
Nomor rumah terurut untuk daerah 1 : 1

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 2
Masukkan 2 nomor rumah kerabat : 5
3
Nomor rumah terurut untuk daerah 2 : 3 5

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 4
Masukkan 4 nomor rumah kerabat : 4 6 7 8
Nomor rumah terurut untuk daerah 3 : 4 6 7 8
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12>

```

Deskripsi dan Cara Kerja Program

Program di atas adalah aplikasi berbasis bahasa Go untuk mengurutkan nomor rumah kerabat di beberapa daerah menggunakan algoritma selection sort. Pengguna diminta memasukkan jumlah daerah yang akan diproses, lalu untuk setiap daerah, mereka diminta memasukkan jumlah nomor rumah serta daftar nomor rumah yang akan diurutkan. Program menggunakan fungsi selectionSort untuk mengurutkan nomor rumah dalam setiap daerah secara menaik. Dalam proses selection sort, elemen terkecil dicari dan ditukar secara iteratif hingga seluruh daftar terurut. Setelah selesai, program menampilkan daftar nomor rumah yang telah diurutkan untuk masing-masing daerah.

Algoritma selection sort yang digunakan dalam program ini bekerja dengan kompleksitas waktu $O(n^2)$, di mana n adalah jumlah elemen dalam array yang akan diurutkan. Meski tidak seefisien algoritma lain seperti quick sort untuk dataset besar, selection sort cocok untuk aplikasi sederhana seperti ini karena implementasinya mudah dipahami. Program ini memanfaatkan struktur array dinamis dengan fungsi make untuk menyesuaikan jumlah elemen yang diinputkan pengguna, sehingga fleksibel untuk berbagai ukuran dataset. Hasil akhir berupa daftar nomor rumah yang terurut memberikan kemudahan bagi pengguna untuk mengelola data dengan lebih rapi dan terstruktur.

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 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

```
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)

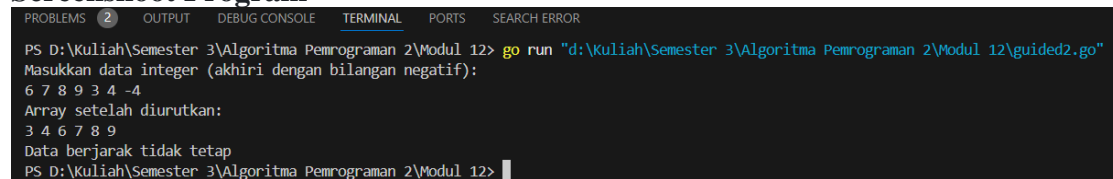
    // Periksa apakah selisih elemen tetap
    isConstant, difference := isConstantDifference(arr, n)

    // 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 Program



```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
6 7 8 9 3 4 -4
Array setelah diurutkan:
3 4 6 7 8 9
Data berjarak tidak tetap
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12>

```

Deskripsi dan Cara Kerja Program

Program di atas adalah aplikasi berbasis bahasa Go yang menggabungkan algoritma *insertion sort* dan logika pemeriksaan selisih elemen array untuk menentukan apakah array memiliki selisih tetap. Pengguna diminta untuk memasukkan sejumlah bilangan bulat secara berurutan, dan input akan dihentikan jika pengguna memasukkan bilangan negatif. Program kemudian mengurutkan array menggunakan *insertion sort*, di mana setiap elemen baru ditempatkan pada posisi yang tepat dengan cara membandingkan dan menggeser elemen sebelumnya. *Insertion sort* memiliki kompleksitas waktu $O(n^2)$ dalam kasus terburuk, tetapi cukup efisien untuk dataset kecil karena memanfaatkan pergeseran elemen.

Setelah array diurutkan, program memeriksa apakah selisih antara elemen-elemen bertetangga dalam array tetap konstan. Hal ini dilakukan dengan menghitung selisih elemen pertama dan kedua sebagai acuan, lalu membandingkan selisih tersebut dengan elemen-elemen berikutnya. Jika seluruh selisih sama, maka array dianggap memiliki "selisih tetap," dan nilai selisih tersebut ditampilkan. Namun, jika ada perbedaan dalam selisih, program menyatakan bahwa data tidak memiliki selisih tetap. Selain memberikan informasi mengenai jarak elemen, program juga membantu pengguna dalam melihat data yang telah diurutkan, memberikan manfaat untuk pengelolaan data secara lebih sistematis.

B. UNGUIDED

1. 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 normor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

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.

```
package main

import "fmt"

// Fungsi untuk mengurutkan array menggunakan selection sort dengan
parameter urutan naik atau turun
func urutkanDenganSelection(arr []int, urutNaik bool) {
    panjang := len(arr)
    for i := 0; i < panjang-1; i++ {
        indeksTerpilih := i
        for j := i + 1; j < panjang; j++ {
```

```

        if (urutNaik && arr[j] < arr[indeksTerpilih]) || (!urutNaik
&& arr[j] > arr[indeksTerpilih]) {
            indeksTerpilih = j
        }
    }
    arr[i], arr[indeksTerpilih] = arr[indeksTerpilih], arr[i]
}
}

func main() {
    var totalDaerah int
    // Meminta input jumlah daerah kerabat
    fmt.Print("Masukkan jumlah daerah kerabat: ")
    fmt.Scan(&totalDaerah)

    // Proses input dan pengurutan untuk setiap daerah
    for i := 0; i < totalDaerah; i++ {
        var totalRumah int
        // Input jumlah rumah untuk daerah tertentu
        fmt.Printf("\nMasukkan jumlah rumah di daerah %d: ", i+1)
        fmt.Scan(&totalRumah)

        nomorRumah := make([]int, totalRumah)
        // Input nomor rumah untuk daerah tersebut
        fmt.Printf("Masukkan %d nomor rumah: ", totalRumah)
        for j := 0; j < totalRumah; j++ {
            fmt.Scan(&nomorRumah[j])
        }

        // Pisahkan nomor rumah ganjil dan genap
        var rumahGanjil, rumahGenap []int
        for _, nomor := range nomorRumah {

```

```
        if nomor%2 == 0 {
            rumahGenap = append(rumahGenap, nomor)
        } else {
            rumahGanjil = append(rumahGanjil, nomor)
        }
    }

    // Urutkan rumah ganjil secara menaik
    urutkanDenganSelection(rumahGanjil, true)

    // Urutkan rumah genap secara menurun
    urutkanDenganSelection(rumahGenap, false)

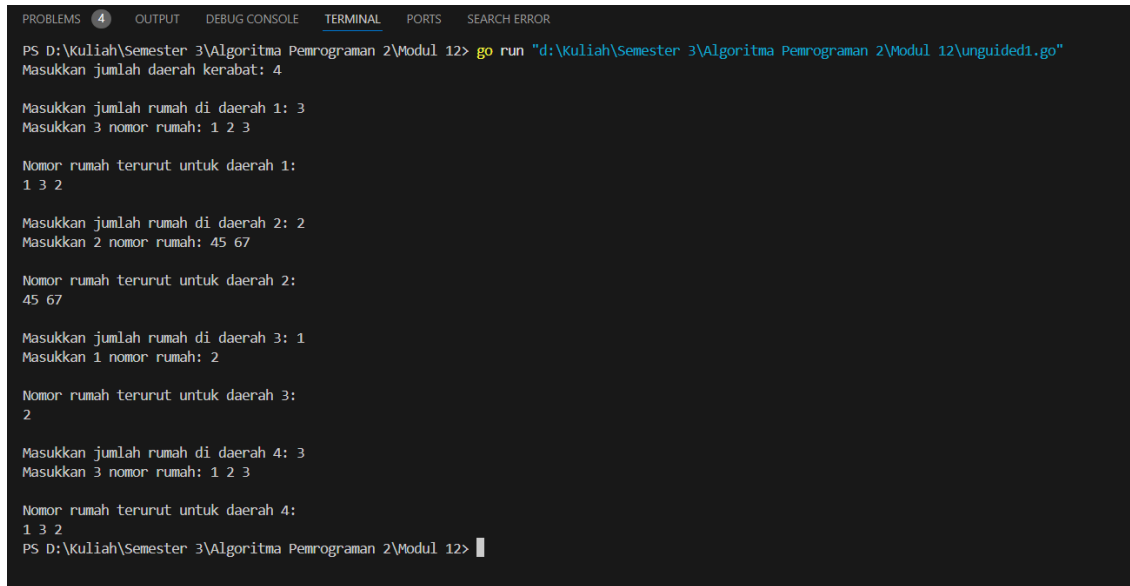
    // Tampilkan hasil nomor rumah yang terurut
    fmt.Printf("\nNomor rumah terurut untuk daerah %d:\n", i+1)

    // Tampilkan nomor rumah ganjil yang sudah terurut
    for _, nomor := range rumahGanjil {
        fmt.Printf("%d ", nomor)
    }

    // Tampilkan nomor rumah genap yang sudah terurut
    for _, nomor := range rumahGenap {
        fmt.Printf("%d ", nomor)
    }

    fmt.Println()
}
}
```

Screenshoot Program



```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12\unguided1.go"
Masukkan jumlah daerah kerabat: 4

Masukkan jumlah rumah di daerah 1: 3
Masukkan 3 nomor rumah: 1 2 3

Nomor rumah terurut untuk daerah 1:
1 3 2

Masukkan jumlah rumah di daerah 2: 2
Masukkan 2 nomor rumah: 45 67

Nomor rumah terurut untuk daerah 2:
45 67

Masukkan jumlah rumah di daerah 3: 1
Masukkan 1 nomor rumah: 2

Nomor rumah terurut untuk daerah 3:
2

Masukkan jumlah rumah di daerah 4: 3
Masukkan 3 nomor rumah: 1 2 3

Nomor rumah terurut untuk daerah 4:
1 3 2
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12> |
```

Deskripsi dan Cara Kerja Program

Program di atas adalah aplikasi berbasis bahasa Go yang dirancang untuk mengurutkan nomor rumah dari beberapa daerah kerabat berdasarkan kriteria tertentu. Pengguna diminta untuk memasukkan jumlah daerah dan jumlah rumah di setiap daerah. Untuk setiap daerah, program menerima input nomor rumah, lalu memisahkan nomor rumah menjadi dua kategori: ganjil dan genap. Nomor rumah ganjil diurutkan secara menaik (ascending), sedangkan nomor rumah genap diurutkan secara menurun (descending). Proses pengurutan menggunakan algoritma selection sort yang dimodifikasi agar dapat menentukan urutan berdasarkan parameter `urutNaik`.

Setelah selesai mengurutkan, program menampilkan daftar nomor rumah yang telah terpisah dan terurut untuk setiap daerah. Nomor rumah ganjil ditampilkan terlebih dahulu dalam urutan menaik, diikuti oleh nomor rumah genap dalam urutan menurun. Dengan memisahkan nomor ganjil dan genap serta memberikan urutan yang berbeda, program ini memberikan fleksibilitas dalam pengelolaan data nomor rumah. Aplikasi ini dapat membantu pengguna dalam menyusun nomor rumah dengan cara yang terorganisir dan sesuai dengan kebutuhan spesifik.

2. 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. Apakah 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.

```

package main

import "fmt"

// Fungsi untuk mengurutkan angka dalam array menggunakan metode
selection sort
func urutkanArray(arr []int) {
    panjang := len(arr)
    for i := 0; i < panjang-1; i++ {
        // Menentukan elemen terkecil di subarray yang belum terurut
        indeksTerkecil := i
        for j := i + 1; j < panjang; j++ {
            if arr[j] < arr[indeksTerkecil] {
                indeksTerkecil = j
            }
        }
        // Tukar posisi elemen terkecil dengan elemen pertama yang
        belum terurut
        arr[i], arr[indeksTerkecil] = arr[indeksTerkecil], arr[i]
    }
}

// Fungsi untuk menghitung dan mengembalikan nilai median dari array
yang sudah terurut
func hitungMedian(arr []int) int {
    totalElemen := len(arr)
    if totalElemen%2 == 1 {
        // Jika jumlah elemen ganjil, ambil elemen tengah
        return arr[totalElemen/2]
    }
    // Jika jumlah elemen genap, rata-rata dua elemen tengah
    return (arr[(totalElemen/2)-1] + arr[totalElemen/2]) / 2
}

func main() {
    var angkaList []int
    fmt.Print("Masukkan angka-angka (akhir dengan -5313): ")

    // Loop untuk menerima input angka
    for {
        var angka int
        _, err := fmt.Scan(&angka)

        // Cek jika angka yang dimasukkan adalah -5313 untuk keluar
        if angka == -5313 {
            break
        }

        // Jika terjadi error dalam input, keluar dari loop
        if err != nil {
            fmt.Println("Terjadi kesalahan input.")
            break
        }
    }

    // Jika angka adalah 0, urutkan dan tampilkan median
    if angka == 0 {
        urutkanArray(angkaList)
    }
}

```



```

        fmt.Printf("Median: %d\n", hitungMedian(angkaList))
    } else {
        // Menambahkan angka ke list
        angkaList = append(angkaList, angka)
    }
}
}

```

Screenshot Program

```

17     arr[i], arr[indeksTerkecil] = arr[indeksTerkecil], arr[i]
18 }
19 }
20 }
21 }
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

Deskripsi dan Cara Kerja Program

Program di atas adalah aplikasi berbasis bahasa Go yang dirancang untuk menerima serangkaian angka dari pengguna, mengurutkan angka-angka tersebut, dan menghitung nilai median dari angka yang telah diurutkan. Pengguna dapat memasukkan angka-angka secara berulang hingga mereka memasukkan angka khusus -5313, yang digunakan sebagai tanda akhir input. Setiap kali pengguna memasukkan angka 0, program secara otomatis mengurutkan daftar angka yang telah dimasukkan menggunakan algoritma selection sort. Setelah pengurutan, program menghitung nilai median dari daftar angka, yang merupakan elemen tengah untuk daftar dengan jumlah elemen ganjil, atau rata-rata dua elemen tengah untuk daftar dengan jumlah elemen genap.

Fungsi `urutkanArray` digunakan untuk mengurutkan angka-angka dalam array secara menaik dengan kompleksitas waktu $O(n^2)$, sedangkan fungsi `hitungMedian` menghitung median berdasarkan panjang array. Program ini dirancang untuk memberikan hasil yang real-time, sehingga pengguna dapat mengetahui median dari angka yang telah dimasukkan kapan saja dengan hanya mengetikkan angka 0. Fitur ini bermanfaat untuk analisis data sederhana, memberikan pandangan langsung tentang distribusi tengah dari angka yang dimasukkan tanpa perlu memproses data secara manual.

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.

```
package main

import "fmt"

const maxBooks = 7919

// Struktur Buku untuk menyimpan data buku
type Buku struct {
    kode, judul, penulis, penerbit string
    eksemplar, tahun, rating      int
}

// Array DaftarBuku untuk menyimpan buku-buku
type KoleksiBuku [maxBooks]Buku

// Fungsi untuk mendaftarkan buku ke koleksi
func tambahBuku(koleksi *KoleksiBuku, jumlah *int) {
    fmt.Print("Masukkan jumlah buku yang ingin didaftarkan: ")
    fmt.Scan(jumlah)

    for i := 0; i < *jumlah; i++ {
        fmt.Printf("\nMasukkan data buku ke-%d (kode, judul, penulis, penerbit, eksemplar, tahun, rating):\n", i+1)
        fmt.Scan(&koleksi[i].kode, &koleksi[i].judul, &koleksi[i].penulis, &koleksi[i].penerbit, &koleksi[i].eksemplar, &koleksi[i].tahun, &koleksi[i].rating)
    }
}

// Fungsi untuk mencetak buku dengan rating tertinggi
func bukuFavorit(koleksi KoleksiBuku, jumlah int) {
    if jumlah == 0 {
        fmt.Println("Tidak ada buku yang terdaftar.")
    }
}
```

```

        return
    }

    favorit := koleksi[0]
    for i := 1; i < jumlah; i++ {
        if koleksi[i].rating > favorit.rating {
            favorit = koleksi[i]
        }
    }

    fmt.Println("\nBuku Favorit:")
    fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d,
Rating: %d\n",
        favorit.judul,    favorit.penulis,    favorit.penerbit,
        favorit.tahun, favorit.rating)
}

// Fungsi untuk mengurutkan buku berdasarkan rating secara menurun
func urutkanBuku(koleksi *KoleksiBuku, jumlah int) {
    for i := 1; i < jumlah; i++ {
        buku := koleksi[i]
        j := i - 1

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

// Fungsi untuk menampilkan 5 buku dengan rating tertinggi
func tampilkanTop5(koleksi KoleksiBuku, jumlah int) {
    fmt.Println("\n5 Buku Dengan Rating Tertinggi:")
    batas := 5
    if jumlah < 5 {
        batas = jumlah
    }

    for i := 0; i < batas; i++ {
        fmt.Printf("%d. %s (Rating: %d)\n", i+1, koleksi[i].judul,
koleksi[i].rating)
    }
}

// Fungsi untuk mencari buku berdasarkan rating
func cariBukuByRating(koleksi KoleksiBuku, jumlah, rating int) {
    ditemukan := false
    fmt.Printf("\nMencari Buku dengan Rating %d:\n", rating)

    for i := 0; i < jumlah; i++ {
        if koleksi[i].rating == rating {
            ditemukan = true
            fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun:
%d, Eksemplar: %d, Rating: %d\n",
                koleksi[i].judul,    koleksi[i].penulis,
koleksi[i].penerbit,

```

```

                                koleksi[i].tahun,    koleksi[i].eksemplar,
koleksi[i].rating)
        }
    }

    if !ditemukan {
        fmt.Println("Tidak ada buku dengan rating tersebut.")
    }
}

func main() {
    var koleksi KoleksiBuku
    var jumlahBuku, ratingPencarian int

    // Menambahkan buku ke dalam koleksi
    tambahBuku(&koleksi, &jumlahBuku)

    // Menampilkan buku dengan rating tertinggi
    bukuFavorit(koleksi, jumlahBuku)

    // Mengurutkan buku berdasarkan rating dan menampilkan hasilnya
    urutkanBuku(&koleksi, jumlahBuku)

    // Menampilkan 5 buku dengan rating tertinggi
    tampilkanTop5(koleksi, jumlahBuku)

    // Mencari buku berdasarkan rating
    fmt.Print("\nMasukkan rating yang ingin dicari: ")
    fmt.Scan(&ratingPencarian)
    cariBukuByRating(koleksi, jumlahBuku, ratingPencarian)
}

```

Screenshoot Program

```

PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12\unguided3.go"
Masukkan jumlah buku yang ingin didaftarkan: 4

Masukkan data buku ke-1 (kode, judul, penulis, penerbit, eksemplar, tahun, rating):
B001 Naruto Masashi Shounen 11 2027 9

Masukkan data buku ke-2 (kode, judul, penulis, penerbit, eksemplar, tahun, rating):
B002 Si Miskin Bayu Gforce 23 2024 9

Masukkan data buku ke-3 (kode, judul, penulis, penerbit, eksemplar, tahun, rating):
B003 TakdirTuhan Takashi JG 12 2024 8

Masukkan data buku ke-4 (kode, judul, penulis, penerbit, eksemplar, tahun, rating):

Buku Favorit:
Judul: Naruto, Penulis: Masashi, Penerbit: Shounen, Tahun: 2027, Rating: 9

5 Buku Dengan Rating Tertinggi:
1. Naruto (Rating: 9)
2. TakdirTuhan (Rating: 8)
3. Si (Rating: 0)
4. 23 (Rating: 0)

Masukkan rating yang ingin dicari: 9

Mencari Buku dengan Rating 9:
Judul: Naruto, Penulis: Masashi, Penerbit: Shounen, Tahun: 2027, Eksemplar: 11, Rating: 9
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 12> 

```

Deskripsi dan Cara Kerja Program

Program di atas adalah sistem manajemen koleksi buku berbasis bahasa Go yang memungkinkan pengguna untuk mengelola daftar buku, menampilkan buku dengan rating tertinggi, mengurutkan buku berdasarkan rating, dan mencari buku tertentu berdasarkan rating. Data setiap buku disimpan dalam struktur Buku, yang mencakup informasi seperti kode buku, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Buku-buku ini disimpan dalam array statis KoleksiBuku dengan kapasitas maksimum 7919 buku. Pengguna dapat memasukkan data buku ke dalam koleksi melalui fungsi tambahBuku, di mana setiap buku diinputkan secara berurutan. Setelah data dimasukkan, pengguna dapat menjalankan berbagai fungsi seperti menemukan buku favorit berdasarkan rating tertinggi (bukuFavorit) atau mengurutkan buku secara menurun berdasarkan rating (urutkanBuku).

Setelah pengurutan, program dapat menampilkan daftar 5 buku dengan rating tertinggi melalui fungsi tampilkanTop5. Pengguna juga dapat mencari buku berdasarkan rating tertentu menggunakan fungsi cariBukuByRating, yang akan menampilkan semua buku dengan rating yang sesuai atau memberikan pesan jika tidak ada buku yang ditemukan. Program ini dirancang untuk mempermudah pengguna dalam mengelola data buku secara efisien, memberikan wawasan tentang buku terbaik dalam koleksi, serta memfasilitasi pencarian berdasarkan kriteria tertentu. Selain itu, dengan pengurutan berdasarkan rating, pengguna dapat lebih cepat mengidentifikasi buku-buku unggulan yang ada di dalam koleksi mereka.

KESIMPULAN

Berdasarkan laporan praktikum Algoritma dan Pemrograman 2 yang membahas tentang algoritma pengurutan (sorting), terdapat dua metode utama yang dibahas yaitu Selection Sort dan Insertion Sort. Selection Sort bekerja dengan cara mencari nilai ekstrim pada sekumpulan data dan memindahkannya ke posisi yang sesuai, sedangkan Insertion Sort fokus pada menyisipkan suatu nilai pada posisi yang tepat melalui pencarian sekuensial. Kedua algoritma ini memiliki kompleksitas waktu $O(n^2)$, yang berarti tidak terlalu efisien untuk dataset besar, namun cukup sederhana untuk dipahami dan diimplementasikan.

Dalam praktikum ini, mahasiswa mengembangkan beberapa program aplikatif menggunakan algoritma sorting untuk menyelesaikan permasalahan praktis. Program-program tersebut mencakup berbagai kasus seperti mengurutkan nomor rumah kerabat, menghitung median dari serangkaian angka, dan mengelola data buku perpustakaan. Setiap program memanfaatkan algoritma sorting untuk mengorganisir data, memudahkan pencarian, dan memberikan wawasan yang lebih terstruktur, menunjukkan fleksibilitas dan kegunaan algoritma pengurutan dalam pemecahan masalah nyata.

Secara keseluruhan, praktikum ini tidak hanya mengajarkan konsep dasar algoritma sorting, tetapi juga memberikan pemahaman praktis tentang penerapannya dalam konteks pemrograman. Mahasiswa dilatih untuk berpikir algoritmis, memahami cara kerja algoritma, dan mengimplementasikannya dalam bahasa pemrograman Go. Meskipun algoritma Selection Sort dan Insertion Sort memiliki keterbatasan kinerja untuk dataset besar, mereka tetap menjadi fondasi penting dalam memahami konsep pengurutan dan menjadi titik awal bagi mahasiswa untuk mempelajari algoritma sorting yang lebih kompleks di kemudian hari.

DAFTAR PUSTAKA

Modul 12 & 13 Praktikum Algoritma dan Pemrograman 2 Modul Pengurutan Data